Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

O *Software* Livre como Alternativa para a Inclusão Digital do Deficiente Visual

Autor: Samer Eberlin
Orientador: Prof. Dr. Luiz César Martini

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos para obtenção do título de Mestre em **Engenharia Elétrica**.

Banca Examinadora

José Raimundo de Oliveira, Dr. ..... DCA/FEEC/Unicamp
Luiz César Martini, Dr. ..... DECOM/FEEC/Unicamp
Rita de Cassia Ietto Montilha, Dra. ..... CEPRE/FCM/Unicamp
Yuzo Iano, Dr. ..... DECOM/FEEC/Unicamp

Campinas, SP – Brasil
Abril/2006

FICHA CATALOGRÁFICA ELABORADA PELA

BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Eberlin, Samer

Eb37s O *software* livre como alternativa para a inclusão digital do deficiente visual / Samer Eberlin. −−Campinas, SP: [s.n.], 2006.

Orientador: Luiz César Martini.

Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Acessibilidade. 2. Tecnologia educacional. 3. Inclusão digital. 4. Software livre. 5. Software de comunicação. 6. Síntese da voz. I. Martini, Luiz César. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Titulo em Inglês: The free software as an alternative for digital cohesion of visually impaired people

Palavras-chave em Inglês: Accessibility, Assistive technology, Digital cohesion, Free software, Screen reader, Voice synthesizer

Área de concentração: Telecomunicações e Telemática

Titulação: Mestre em Engenharia Elétrica

Banca examinadora: José Raimundo de Oliveira, Rita de Cássia Ietto Montilha e Yuzo Iano

Data da defesa: 19/04/2006

Resumo

A acelerada difusão do *software* "livre", tanto no Brasil como no exterior, vem se mostrando cada vez mais evidente nos mais diversos âmbitos (governo, empresas, escolas, etc.). A principal motivação para a transição do *software* "proprietário" para o "livre" é a redução de custos, mas para efetivar essa migração é necessário que ferramentas compatíveis estejam disponíveis para a manutenção da usabilidade do sistema. Essa é ainda uma barreira para a migração do usuário deficiente visual brasileiro, pois até este momento, nenhuma das tecnologias assistivas desenvolvidas para sistemas operacionais "livres" encontram-se disponíveis no idioma português. Como solução para esse problema, esta dissertação apresenta uma alternativa que efetivará essa migração, habilitando usuários cegos para realização de tarefas como edição de texto, acesso à *internet*, gerenciamento de arquivos, entre outras. O trabalho baseia-se na implementação de um sintetizador de voz para o português do Brasil e na tradução de uma tecnologia assistiva desenvolvida para sistemas operacionais "livres". Como parte integrante estão documentados também o desenvolvimento de um modelo compacto de computador pessoal e os resultados de testes realizados com usuários voluntários.

Palavras-chave: Acessibilidade, Tecnologia Assistiva, Inclusão Digital, *Software* Livre, Leitor de Telas, Sintetizador de Voz.

Abstract

The accelerated diffusion of the "free" software, as much in Brazil as in the foreign, has beem shown more and more evident in the most diverse scopes (government, companies, schools, etc.). The main motivation to the transition from "proprietary" software to the "free" one is the costs reduction, but to accomplish this migration compatible tools need to be available for the maintenance of the system usability. This is still a barrier for the migration of the brazilian visually impaired user, because up to this moment, none of the assistive technologies developed to "free" operating systems are available in portuguese language. As solution for this problem, this dissertation presents an alternative that will accomplish this migration, enabling blind users to carrying out tasks like text edition, internet access, file management, among others. The work is based on the implementation of a voice synthesizer for the portuguese from Brazil and on the translation of an assistive technology developed to "free" operating systems. As integrated part are also documented the development of a compact model of personal computer and the results of tests carried out with voluntary users.

Keywords: Accessibility, Assistive Technology, Digital Cohesion, Free Software, Screen Reader, Voice Synthesizer.

Aos meus pais, irmãos e avós.

Agradecimentos

Ao meu orientador, Prof. Dr. Luiz César Martini, sou grato não só pela orientação, mas pela confiança em mim depositada.

Aos profissionais do Centro de Estudos e Pesquisas em Reabilitação, em especial às minhas supervisoras Profa. Dra. Rita de Cássia Ietto Montilha e Profa. Especializada Sílvia Helena Rodrigues de Carvalho, pela valiosíssima experiência que me foi passada para o delineamento desta pesquisa.

À minha família, pelo apoio durante esta jornada, em especial à minha mãe Ana Maria e à minha irmã Samara, pelo incentivo inicial que originou este trabalho. À minha noiva Vanessa, pelo incentivo e compreensão perante uma infinidade de finais de semanas integralmente consumidos pelos estudos.

Aos amigos deficientes visuais, que direta ou indiretamente incentivaram e contribuíram para o sucesso deste projeto. Aos colegas de pós-graduação, pelas valiosas críticas e sugestões.

E à CNPq, pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

CAP - Centros de Apoio Pedagógico

CEPRE - Centro de Estudos e Pesquisas em Reabilitação

CPU - Unidade Central de Processamento

DSP - *Digital Signal Processing*

DV - Deficiente Visual

FCM - Faculdade de Ciências Médicas

FEEC - Faculdade de Engenharia Elétrica e de Computação

FNDE - Fundo Nacional de Desenvolvimento da Educação

GPL - *General Public License*

IBC - Instituto Benjamin Constant

LAB - Laboratório de Acessibilidade da Biblioteca Central

MCT - Ministério da Ciência e Tecnologia

MEC - Ministério da Educação

NCE - Núcleo de Computação Eletrônica

NLP - *Natural Language Processing*

SL - *Software* Livre

TTS - *Text-to-Speech*

UFRGS - Universidade Federal do Rio Grande do Sul

UFRJ - Universidade Federal do Rio de Janeiro

UNICAMP - Universidade Estadual de Campinas

Capítulo 1

Introdução

1.1 Motivação

Acessibilidade é um termo genérico freqüentemente utilizado para descrever o quão fácil é o procedimento para que pessoas com deficiência (motora, cognitiva ou sensorial) possam entender ou realizar determinadas atividades, ou seja, o quão acessível está uma determinada atividade para os diferentes tipos de necessidades especiais (WIKIPEDIA, 2005). O símbolo apresentado na figura 1.1 é freqüentemente utilizado para indicação de acessibilidade, seja qual for a necessidade especial ali representada. Como exemplos de "provedores" de acessibilidade podemos citar:

Fig. 1.1: Acessibilidade

- **O *Braille*:** que é um sistema de escrita através de pontos em relevo, criado em 1829 pelo francês "Louis Braille". Esse recurso provê ao deficiente visual (DV) acessibilidade na leitura e na escrita de documentos, ex. textos, equações, partituras, etc.;

- **A rampa de acesso:** que é a utilização do "piso inclinado" em construções imobiliárias ao invés dos "degraus". Esse recurso provê ao deficiente físico acessibilidade na locomoção com cadeira de rodas aos diferentes níveis das edificações;

- **A linguagem de sinais:** que é um sistema de comunicação através de gestos, com estrutura gramatical própria. Esse recurso provê ao deficiente auditivo acessibilidade na comunicação pessoal.

Também podemos classificar como “provedores” de acessibilidade os diferentes dispositivos tecnológicos que vem sendo desenvolvidos com o intuito de prover ou melhorar a acessibilidade em uma atividade qualquer. A constante evolução desses dispositivos, também conhecidos como tecnologias assistivas, tem possibilitado aos deficientes cada vez mais o acesso à informação e conseqüentemente à inclusão social. Esses recursos promovem maior independência às pessoas com deficiências, habilitando-as na realização de tarefas que em condições normais não seriam possíveis.

No caso especial da deficiência visual, a utilização do computador em conjunto com *softwares* dotados de síntese de voz tem proporcionado independência na realização de atividades nunca antes cogitadas. Segundo BORGES (1996) "O microcomputador [...] amplia até um limite inimaginável as oportunidades do cego". Como simples exemplo dessa independência, podemos citar o acesso integral ao conteúdo de jornais diários ou livros em formato digital, que até então só era possível através da transcrição para o sistema *Braille* ou com o auxílio de um “ledor” (termo técnico utilizado para indicar pessoas com visão normal que se dispõe a ler para DVs).

Algumas das maiores empresas desenvolvedoras de *softwares* na área de acessibilidade vem periodicamente apresentando novidades e lançamentos que impressionam até mesmo especialistas na área de reabilitação. Entretanto, paralelo ao alto nível de sofisticação está também o alto preço praticado na comercialização dessas tecnologias. Com relação à disponibilização, essas tecnologias (*softwares*) e os sistemas operacionais (conjunto de *softwares* responsável pelo funcionamento do computador) necessários para utilização das mesmas, serão aqui classificados como “proprietários” ou “livres”:

- **Proprietário:** para indicar todos aqueles *softwares* e/ou sistemas operacionais que para sua implantação necessitem da aquisição de qualquer tipo de licença ou honorários de utilização. Como exemplos de *softwares* “proprietários” podemos citar o “Microsoft Office®”, o “Adobe Photoshop®”, o “Corel Draw®”, etc.; E como exemplos de sistemas operacionais “proprietários” podemos citar o “Microsoft Windows®”, o “Apple MacOS®”, o “IBM OS/2®”, etc.

- **Livre:** para indicar todos aqueles *softwares* e/ou sistemas operacionais que não necessitem da aquisição de licença ou qualquer outro tipo de ônus para sua utilização. Como exemplos de *softwares* “livres” podemos citar o “OpenOffice”, o “Mozilla”, o “Gimp”, o “Emacs”, etc.; E como exemplos de sistemas operacionais “livres” podemos citar o “Linux”, o “FreeBSD”, o “Darwin”, etc.

É notório que o objetivo das empresas desenvolvedoras de *software* na área de acessibilidade é a comercialização, portanto esses produtos são em sua maioria classificados como "proprietários". Pelo mesmo motivo, os investimentos em pesquisas por parte dessas empresas são restritos ao desenvolvimento de tecnologias para sistemas operacionais "proprietários", motivo que eleva ainda mais o custo de implantação, restringindo de maneira significativa a porcentagem de DVs com acesso a tais benefícios.

Existem também tecnologias assistivas de utilização "livre", tanto para sistemas operacionais "proprietários" quanto para "livres", mas o desenvolvimento dessas opções gratuitas fica restrito ao trabalho de programadores voluntários, pesquisadores, estudantes e uma minoria de empresas que visam obtenção de lucros apenas no treinamento para utilização dessas tecnologias e na comercialização de livros didáticos sobre o *software* em questão.

A nível internacional, a melhor opção para redução de custos tem sido a migração de sistemas operacionais "proprietários" para "livres". A adoção do *software* livre (SL) pelo Governo Federal vem sendo definida como política pública desde 2003 e poderá virar obrigação ainda este ano, como a edição de decreto 10007 (DECRETO nº 10007, 2003). O recente projeto lançado também pelo Governo Federal "PC Conectado - Computador para Todos" visa uma redução significativa no custo do computador pessoal básico através da isenção de impostos, mas uma da exigências para efetivação é a implantação de *softwares* de "livre" utilização (PC Conectado, 2005).

Segundo resultados da pesquisa (a maior individualizada por país já realizada em todo o mundo) realizada pela "Softex" e pela "Universidade Estadual de Campinas" (UNICAMP) com o apoio do "Ministério da Ciência e Tecnologia" (MCT), o crescimento na utilização de SL vem se mostrando cada vez mais evidente nos mais diversos âmbitos (governo, empresas, escolas, etc.) e existem perspectivas de crescimento acelerado para os próximos anos (SOFTEX; UNICAMP, 2005). Mas até este momento nenhuma das tecnologias assistivas desenvolvidas para sistemas operacionais "livres" encontram-se disponíveis no idioma português, impondo assim uma barreira para que essa migração seja realizada também por DVs brasileiros.

Frente à essa verdadeira "revolução" na utilização de SL, mostra-se clara a necessidade de desenvolvimento de novas tecnologias assistivas para sistemas operacionais "livres", ou adaptação das existentes, para o idioma português. Pois sem o surgimento desses recursos, paralelo à "revolução" teremos também uma nova fase de exclusão social de DVs tanto na idade infantil como na adulta (visto que a utilização de SL já é uma prática comum tanto em escolas como em instituições, sejam elas públicas ou privadas).

1.2 Objetivo

O objetivo desta dissertação é que ao final de seu desenvolvimento seja apresentada uma tecnologia assistiva para sistemas operacionais "livres" no idioma português do Brasil e que com a disponibilização desse recurso seja eliminada definitivamente a barreira para que DVs possam também migrar para o SL.

É claro que não se pretende com esse trabalho atingir o nível de sofisticação alcançado pelas empresas desenvolvedoras de *software* na área de acessibilidade, mesmo porque esses patamares são resultados de anos de pesquisas envolvendo um numeroso grupo de desenvolvedores. Mas certamente a alternativa para sistemas operacionais livres aqui apresentada possibilitará ao DV brasileiro atividades como:

- Edição de texto;
- Acesso à *internet*;
- Gerenciamento de cartas eletrônicas (*e-mails*);
- Gerenciamento de arquivos;
- Execução de músicas (rádio, MP3 ou CD);
- Exibição de vídeos (AVI, MPEG ou DVD);
- Produção de textos com formatação (LaTeX);
- Produção de músicas e partituras (MusiXTeX);
- Instalação de novos aplicativos e até mesmo o desenvolvimento de novos recursos.

Para possibilitar futuras implementações de melhorias e adaptações, proveniente do trabalho de programadores voluntários e até mesmo de usuários DVs, o produto deste trabalho terá o seu código fonte disponibilizado na íntegra (código fonte aberto) sob os termos da "*General Public License*" (GPL). GPL é um sistema de licenciatura para *softwares*, originalmente escrito por "Richard Stallman" (patrono do SL) em 1989, que garante liberdade para utilização, modificação e redistribuição do produto em questão (STALLMAN, 1989). Em outras palavras, a adesão à essa licença libera o *software* para domínio público e garante que nada será cobrado para sua utilização.

Como parte integrante do desenvolvimento dessa tecnologia, essa dissertação apresenta também o desenvolvomento de um modelo compacto de computador pessoal (construído visando redução de tamanho, peso e principalmente redução de custo) e os resultados de testes realizados com usuários voluntários na utilização do *software* apresentado. O objetivo desse modelo de utilidade (computador pessoal compacto) é possibilitar que essa tecnologia possa

ser facilmente transportada da casa para a escola ou da casa para o trabalho, dispensando assim altos investimentos na aquisição de *notebooks*.

1.3 Materiais e Métodos

Para atingir o objetivo deste trabalho, antes de iniciar o desenvolvimento do *software* que será apresentado como a primeira tecnologia assistiva para sistemas operacionais "livres" disponível no idioma português, o capítulo 2 apresenta um estudo realizado sobre as mais utilizadas tecnologias assistivas "proprietárias" e "livres" disponíveis para os diferentes sistemas operacionais ("proprietários" e "livres"). Ainda como fase preparatória, esse capítulo apresenta uma análise comparativa entre as tecnologias "livres" para sistemas operacionais "livres", para indicar qual oferece melhor viabilidade de adaptação para o idioma português do Brasil.

Iniciando a fase de desenvolvimento, o capítulo 3 apresenta a implementação de um sintetizador de voz para o idioma português do Brasil, que é o meio fundamental de comunicação entre o computador e o usuário DV, e a tradução e adaptação da tecnologia previamente selecionada na fase preparatória.

Para satisfazer os objetivos desta pesquisa, o capítulo 4 apresenta o desenvolvimento do computador pessoal projetado e construído com redução de tamanho, peso e custo.

E para finalização do projeto, o capítulo 5 apresenta as sugestões para trabalhos futuros, os resultados das experiências com a utilização do computador pessoal compacto, os resultados dos testes realizados com o *software*, e as conclusões.

Capítulo 2

Tecnologias Assistivas

Este capítulo apresenta o resultado de um estudo, realizado com o apoio do "Centro de Estudos e Pesquisas em Reabilitação" (CEPRE) da "Faculdade de Ciências Médicas" (FCM) e do "Laboratório de Acessibilidade da Biblioteca Central" (LAB), ambos pertencentes à UNICAMP, sobre as mais utilizadas tecnologias assistivas "proprietárias" e "livres", na área de deficiência visual, disponíveis para os diferentes sistemas operacionais ("proprietários" e "livres"). O objetivo dessa investigação inicial foi conhecer as principais características dessas tecnologias, aprender os diferentes modos de operação, entender a problemática envolvida e estabelecer bases sólidas sobre as reais necessidades do usuário DV.

Sob supervisão da Profa. Dra. Rita de Cássia Ietto Montilha e da Profa. Especializada Silvia Helena Rodrigues de Carvalho, o autor desta dissertação estagiou intensamente por um período inicial de seis meses nas dependências do CEPRE e do LAB, onde esteve em contato direto com usuários dessas tecnologias. Essa fase de experiência possibilitou que além do aprendizado sobre esses recursos fosse também realizado um mapeamento das atividades mais cotidianas.

Para encerrar essa fase preparatória, este capítulo apresenta também uma análise comparativa entre as tecnologias assistivas "livres" para sistemas operacionais "livres", para indicar qual oferece melhor viabilidade de adaptação para o idioma português do Brasil.

2.1 Tecnologias "Proprietárias" para Sistemas Operacionais "Proprietários"

Nesta seção estão apresentados os *softwares* "proprietários" desenvolvidos para sistemas operacionais "proprietários", ou seja, além da aquisição de licenças para utilização dos *softwa-*

res é necessário que também seja adquirida uma licença para prévia instalação do sistema operacional para o qual foram desenvolvidos.

2.1.1 Jaws

O *software* "Jaws", ou "Job Access With Speech", é desenvolvido e comercializado pela empresa "Freedom Scientific" a um preço de aproximadamente U$1300,00 (mil e trezentos dólares). Como requisitos mínimos para seu funcionamento são exigidos além do sistema operacional "Microsoft Windows®", um processador de velocidade igual ou superior a 300MHz, 200MB de espaço livre em disco rígido e 128MB de Memória RAM (FREEDOM SCIENTIFIC, 2005).

Foi originalmente lançado em 1989 para o sistema operacional "Microsoft DOS®", e teve sua versão 1.0 para o "Microsoft Windows®" lançada em 1993. Passou a contar com síntese de voz no idioma português a partir da versão 3.7 e teve sua interface também traduzida e adaptada para comercialização no Brasil pela fundação "Laramara" (LARAMARA, 2005). Atualizações são anunciadas aproximadamente duas vezes por ano e atualmente encontra-se na versão 6.2.

Funciona como um leitor de telas, identifica e interpreta as informações que estão sendo exibidas na tela do monitor e repassa esse contexto ao DV por meio de síntese de voz. Basicamente um *software* leitor de telas funciona da seguinte forma:

- **Inicialização:** por padrão é ativado automaticamente durante a inicialização do sistema operacional e quando finalizada a inicialização informa o usuário com uma mensagem do tipo "O sistema está ativado";

- **Menu "Iniciar":** quando a deficiência visual é total, o usuário não faz uso do *mouse*, portanto, o acesso ao menu "Iniciar", assim como à qualquer outra função, é feito pelo teclado através de teclas de atalho (seqüência ou combinação de teclas). No caso do menu "Iniciar" o acesso é feito pela tecla "Windows" ou com o pressionamento simultâneo das teclas "Control" e "Escape". Quando acionado esse menu o leitor informa o usuário com uma mensagem do tipo "Menu Iniciar, use as setas para cima ou para baixo";

- **Submenu "Programas":** Estando no menu "Iniciar", ao pressionar as setas "para cima" ou "para baixo" o leitor informa a opção imediatamente sob o cursor, por exemplo, ao pressionar uma única vez a seta "para cima" o leitor informa "Desligar", ao pressionar novamente a seta, informa "Executar" e assim por diante. Quando o cursor atinge o

submenu “Programas” (figura 2.1), ao invés de informar simplesmente “Programas” o leitor informa “Submenu Programas”, para indicar que dentro dessa existem outras opções. O mesmo acontece quando selecionado, por exemplo, o submenu “Documentos” ou “Configurações”. Para explorar o conteúdo de um submenu a seta “para direita” deve ser pressionada e para retroceder ao menu anterior a seta “para esquerda”;

Fig. 2.1: Leitor de Telas - Menu e Submenu

- **Aplicativos:** para executar um aplicativo qualquer, presente no menu iniciar ou em submenus, é necessário que seja pressionada a tecla “Enter” quando o cursor estiver posicionado sobre o mesmo. Após iniciado o aplicativo, o leitor informa o conteúdo da barra de título da janela recém iniciada, geralmente seguido por um breve texto de ajuda. Por exemplo, após aberto o editor de textos “WordPad” (figura 2.2), o leitor informa “Documento WordPad, digite o texto”;

- **Barra de Menu:** para que seja acessada a barra de menu é necessário que seja pressionada a tecla “Alt” e em seguida as setas direcionais devem ser utilizadas para explorar o conteúdo disponível. O comportamento do leitor nesse caso é idêntico ao observado no menu “Iniciar”, ou seja, informa a opção imediatamente sob o cursor. Por exemplo, estando na situação da figura 2.3, o leitor informa “Abrir... Control+A”;

- **Digitação:** por padrão, ao iniciar a digitação em um editor de textos qualquer, as teclas são imediatamente pronunciadas (uma a uma) e após o pressionamento da “barra de espaço”, a palavra toda é pronunciada. Por exemplo, ao digitar “teste”, a resposta do

Fig. 2.2: Leitor de Telas - Barra de Título

Fig. 2.3: Leitor de Telas - Barra de Menu

leitor é “ t ”, “ e ”, “ s ”, “ t ”, “ e ”, e após pressionar a “barra de espaço” o leitor informa “teste”.

- **Janela composta:** ao acessar uma janela qualquer que contenha mais de um elemento, o leitor informa o elemento no qual está posicionado o cursor e para alternar o posicionamento do cursor entre os demais elementos deve ser pressionada a tecla “Tab”. Por exemplo, estando na situação da figura 2.4, o leitor informa “Nome do arquivo:”, pressionando a tecla “Tab” uma vez o cursor avança até o próximo elemento e informa

“Arquivos do tipo: Formato Rich Text (*.rtf)”, pressionando a tecla “Tab” novamente o leitor informa “Botão Abrir” e assim por diante.

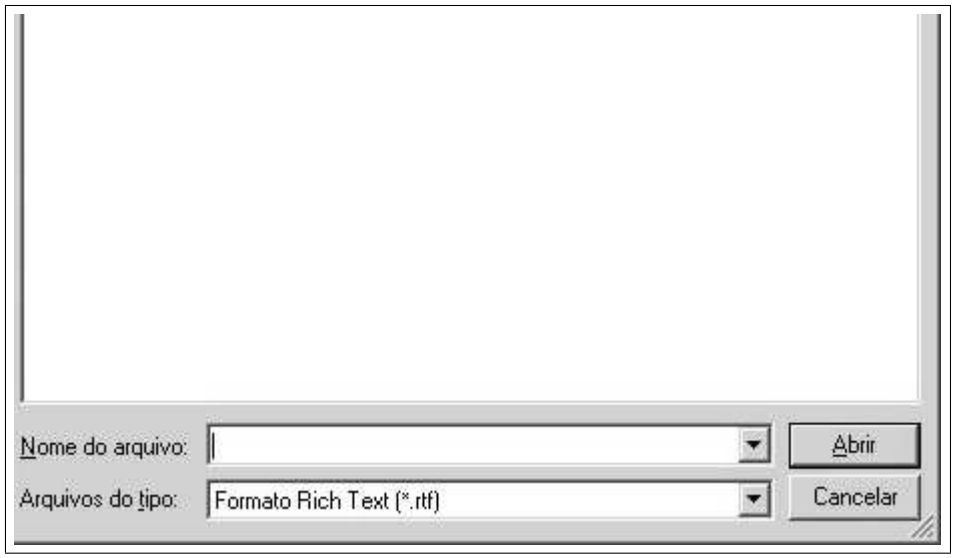


Fig. 2.4: Leitor de Telas - Janela Composta

Desse modo o DV pode realizar a maioria das atividades realizadas cotidianamente por um usuário com visão normal, como por exemplo edição de textos, acesso à *internet*, gerenciamento de arquivos, entre outras.

Disponível em diversos idiomas, essa ferramenta viabiliza o acesso integral à utilitários como “Outlook”, “Word”, “Excel”, “Internet Explorer”, “Windows Explorer”, etc., e ainda permite que os usuários criem regras customizadas para controlar a quantidade e o tipo de informação que é repassada em aplicações específicas. Outra grande vantagem é o fato de simular operações com o *mouse* através do teclado numérico, possibilitando acessos a determinados programas que até então eram impossíveis com outros leitores de tela (SONZA; SANTAROSA, 2003). Essas são algumas das razões responsáveis pela imensa popularidade dessa ferramenta dentre o universo de tecnologias assistivas para DVs a nível mundial.

2.1.2 ZoomText

O *software* “ZoomText”, é desenvolvido e comercializado pela empresa “Ai Squared” a um preço de aproximadamente U$800 (oitocentos dólares). Como requisitos mínimos para seu funcionamento são exigidos além do sistema operacional “Microsoft Windows®”, um processador de velocidade igual ou superior a 450MHz, 60MB de espaço livre em disco rígido e 256MB de Memória RAM (AI SQUARED, 2005).

Também funciona como um leitor de telas, seu funcionamento é semelhante ao do *software* "Jaws" descrito na subseção 2.1.1, mas a ênfase dessa tecnologia está nas diferentes e criativas maneiras disponíveis para ampliação da imagem da tela no monitor, provendo assim, melhor acessibilidade às pessoas com visão subnormal (pessoas com perda parcial da visão) estimulando-as na utilização da visão residual.

Seu funcionamento é basicamente controlado pelo *mouse*, que opera como uma espécie de "lupa eletrônica" ampliando uma determinada região sob o ponteiro. O tamanho da região a ser ampliada pode ser configurado de acordo com a necessidade, assim como também pode ser alterado o grau de ampliação. Possui também excelentes recursos para inversão de cores e ajuste de contraste que podem ser facilmente acessados por intermédio de teclas de atalho, minimizando consideravelmente o esforço do resíduo visual de pessoas com visão subnormal.

Foi originalmente lançado em 1988 para o sistema operacional "Microsoft DOS®", e teve sua versão 1.0 para o "Microsoft Windows®" lançada em 1991. Atualizações também são anunciadas com freqüência e atualmente encontra-se na versão 9.0.

2.1.3 Virtual Vision

O *software* "Virtual Vision", é desenvolvido e comercializado pela empresa "MicroPower" a um preço de aproximadamente U$750,00 (setecentos e cinqüenta dólares) para uso corporativo. Como requisitos mínimos para seu funcionamento são exigidos além do sistema operacional "Microsoft Windows®", um processador de velocidade igual ou superior a 300MHz, 30MB de espaço livre em disco rígido e 64MB de Memória RAM (MICROPOWER, 2005).

Funciona como um leitor de telas, exatamente como o *software* "Jaws" descrito na subseção 2.1.1. Possui também um sistema de mapeamento e adaptação a aplicativos que não oferecem acessibilidade a leitores de telas, utilizando sistemas de mapas de posicionamento e até mesmo reconhecimento de gráficos, que podem ser configurados pelo próprio usuário.

Foi desenvolvido a partir de solicitações de DVs que procuraram a empresa "MicroPower", logo após o lançamento do *software* sintetizador de voz "DeltaTalk" (considerado a nível mundial o melhor para o idioma português do Brasil). A primeira versão foi lançada em 1998 e atualmente encontra-se na versão 5.0.

2.1.4 Utilitários para Transcrição *Braille*

WinBraille: desenvolvido pela empresa "Index Braille" acompanha todos os modelos de impressoras *Braille* comercializadas pela mesma (INDEX BRAILLE, 2005). Funciona como um transcritor de texto do formato escrito (caracteres) para o sistema *Braille* (fi-

gura 2.5), possibilitando assim que qualquer informação disponível em meios eletrônicos (*internet*, disquetes, CDRoms, etc.) seja disponibilizada em *Braille*. No mesmo pacote existe também um utilitário para realização do processo inverso, com o auxílio de uma mesa digitalizadora, documentos em *Braille* podem ser transcritos para o formato de texto escrito, e então armazenados por meios eletrônicos.

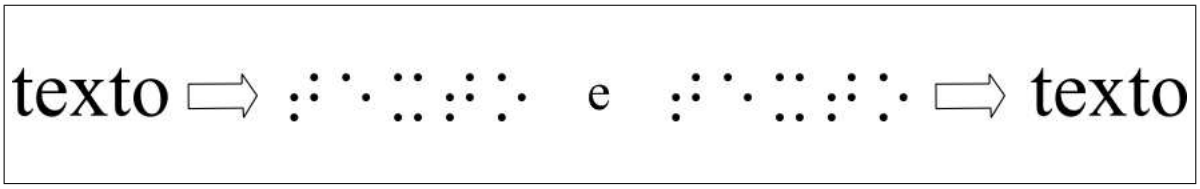


Fig. 2.5: WinBraille

TGD Pro: desenvolvido e comercializado pela empresa “Duxbury Systems” a um preço de aproximadamente U$500,00 (quinhentos dólares), este *software*, com o auxílio de uma impressora *Braille*, é capaz de transformar figuras do formato digital (desenhos ou fotografias digitalizadas) para figuras táteis (alto relevo) (figura 2.6). No mesmo pacote existe também um utilitário para confecção e impressão de figuras sem a necessidade de obtenção da mesma em formatos digitais (DUXBURY SYSTEMS, 2005).

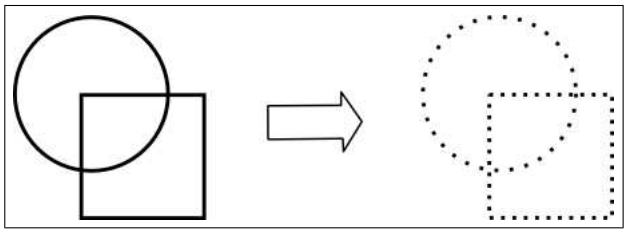


Fig. 2.6: TGD

Goodfeel: desenvolvido e comercializado pela empresa “Dancing Dots” a um preço de aproximadamente U$800,00 (oitocentos dólares), este *software* permite que, com o auxílio de uma mesa digitalizadora, partituras musicais sejam transcritas para o sistema *Braille* musical (DANCING DOTS, 2005). O processo envolve a execução de três aplicativos: o “SharpEye” que é o responsável pela digitalização e conversão da partitura para o formato “MIDI” (formato tipicamente utilizado para armazenamento digital de músicas e/ou partituras), o “Lime” que é o responsável por correções de possíveis falhas na conversão realizada pelo “SharpEye” e finalmente o “Goodfell” que transforma o arquivo do formato “MIDI” para o sistema *Braille* musical (figura 2.7).

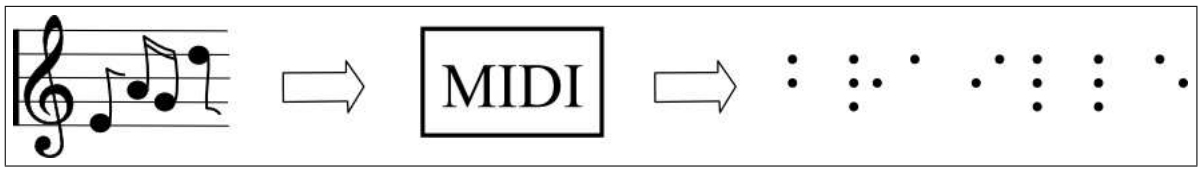


Fig. 2.7: Goodfeel

Braille Music Editor: desenvolvido e comercializado pela empresa "Dodiesis" a um preço de aproximadamente U$370,00 (trezentos e setenta dólares), este software é o mais completo programa de escrita musical para DVs atualmente disponível no mercado (DODIESIS, 2005). Essa tecnologia permite que a música seja editada diretamente através do teclado do computador em formatação *Braille* musical (figura 2.8), como se estivesse sendo editada numa "máquina *Perkins*" (dispositivo mecânico semelhante a uma máquina de escrever comum, utilizado para produção de material em *Braille*). Permite também ouvir e corrigir a música antes que seja enviada para uma impressora *Braille*. E em conjunto com o *software* de editoração musical "Finalle", permite que partituras musicais impressas sejam transcritas para o sistema *Braille* musical.

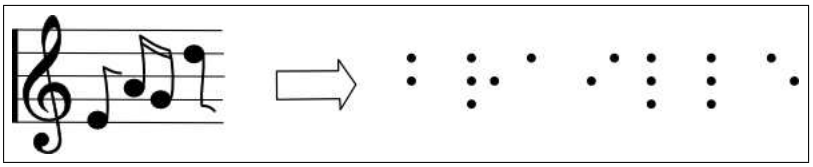


Fig. 2.8: Braille Music Editor

2.2 Tecnologias "Livres" para Sistemas Operacionais "Proprietários"

Nesta seção estão apresentados os *softwares* "livres" desenvolvidos para sistemas operacionais "proprietários", ou seja, não é necessária a aquisição de nenhum tipo de licença para utilização dos *softwares*, mas sim para prévia instalação do sistema operacional para o qual foram desenvolvidos.

2.2.1 Dosvox

O *software* "Dosvox", é desenvolvido e distribuído livremente pelo "Núcleo de Computação Eletrônica" (NCE) da "Universidade Federal do Rio de Janeiro" (UFRJ) sob orientação do

Professor José Antônio Borges (DOSVOX, 2005). Como requisitos mínimos para seu funcionamento são exigidos além do sistema operacional “Microsoft Windows®”, um processador de velocidade igual ou superior a 133MHz.

Foi originalmente lançado em 1993 a partir do trabalho de um aluno com deficiência visual, “Marcelo Pimentel”, que hoje é programador do NCE da UFRJ, onde trabalha sob a orientação do Professor “José Antônio Borges”. É atualizado freqüentemente e atualmente encontra-se na versão 3.2.

Este *software* também se comunica com o usuário através de síntese de voz, mas ao invés de identificar e pronunciar as informações que estão sendo exibidas na tela do monitor, oferece um pacote com mais de 70 aplicativos "falados", ou seja, ao invés de ler o que está escrito na tela, o “Dosvox” estabelece um diálogo amigável com o usuário através de aplicativos próprios. Grande parte desse diálogo é feito por voz humana gravada, o que resulta num baixo índice de estresse para o usuário, mesmo em situações de uso prolongado.

Sua utilização é bastante simples, pois uma das preocupações dos desenvolvedores foi reduzir ao máximo qualquer comprometimento técnico por parte dos usuários, portanto, dispensa conhecimentos prévios e estabelece uma “conversa” com o operador. Funciona da seguinte forma:

- **Inicialização:** pode ser configurado para que inicie automaticamente na inicialização do sistema operacional ou a qualquer momento através do pressionamento simultâneo das teclas “Control”, “Alt” e “D”. Após sua inicialização, a primeira frase do diálogo é “Dosvox - O que você deseja?” (figura 2.9). Essa frase será também pronunciada sempre que o sistema necessitar de uma nova informação.

- **Ajuda:** a tecla “F1” pode ser pressionada a qualquer momento para que seja informada uma lista de comandos disponíveis no aplicativo em execução. Opcionalmente as setas “para cima” e “para baixo” podem ser utilizadas para selecionar interativamente as opções e a tecla “Enter” para acessar a opção selecionada (figura 2.10).

- **Aplicativos:** os aplicativos podem ser iniciados através das teclas de atalho correspondentes ou pelo menu interativo que pode ser observado na figura 2.10. Geralmente, no menu interativo inicial selecionamos uma categoria de aplicativos e em seguida um aplicativo específico, por exemplo, a opção “J” seleciona a categoria “Jogos” e nesse ponto o novo menu (submenu) interativo que é apresentado com o auxílio das setas é o representado na figura 2.11.

- **Digitação:** o sistema de digitação segue o mesmo padrão dos leitores de telas, ao iniciar a digitação, por exemplo no editor de textos que é acionado pela tecla “E” (figura 2.10), as

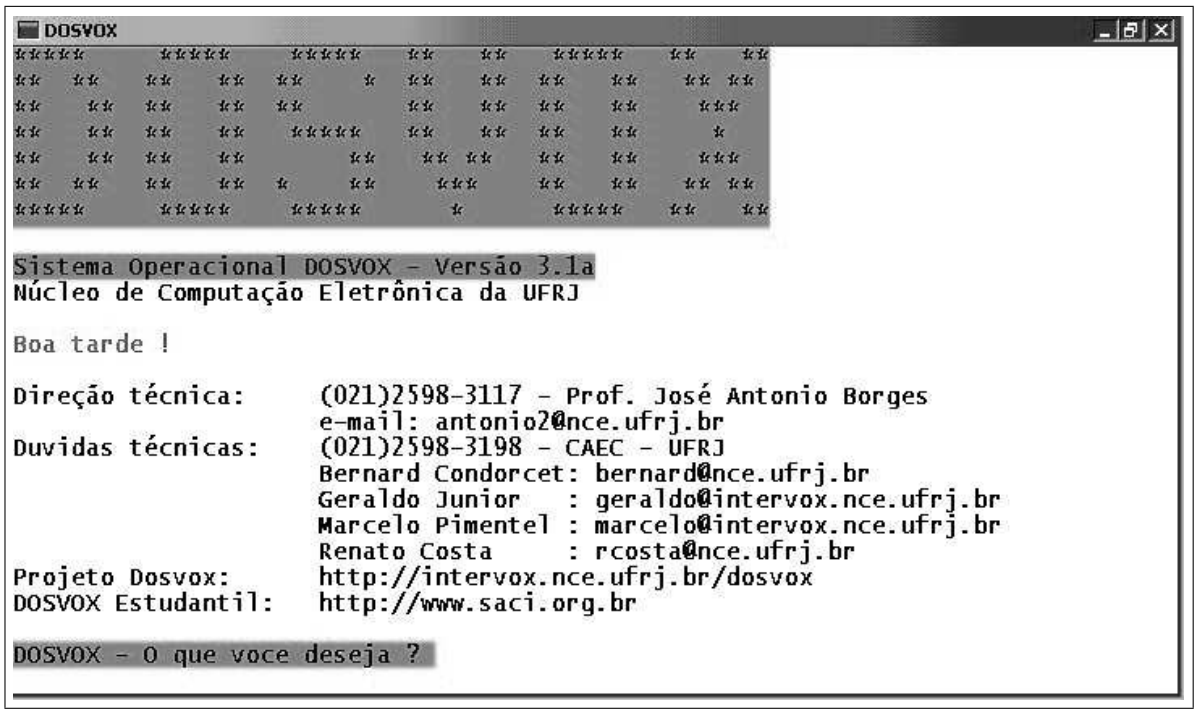


Fig. 2.9: Dosvox - Inicialização

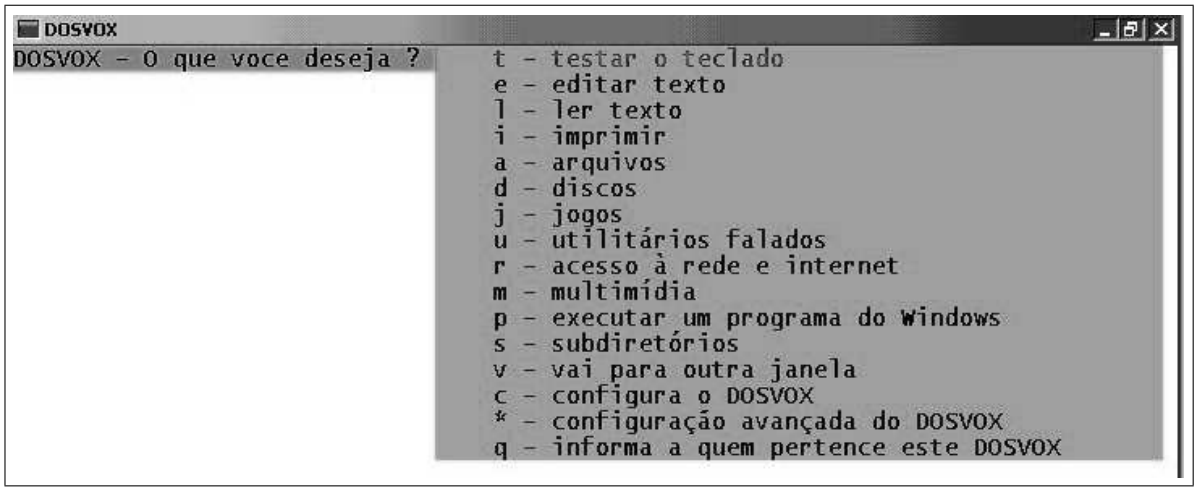


Fig. 2.10: Dosvox - Menu

teclas são imediatamente pronunciadas (uma a uma) e após o pressionamento da "barra de espaço", a palavra toda é então pronunciada.

- **Encerramento:** para encerrar qualquer aplicativo ou cancelar a execução de qualquer operação basta pressionar a tecla "Escape". Esta é também a tecla utilizada para encerrar o "Dosvox" quando nenhum outro aplicativo está em execução.

Fig. 2.11: Dosvox - Submenu

A síntese de voz do “Dosvox” foi reconhecida pela revista “PC-World” como sendo o primeiro compilador de voz para o idioma português criado e efetivamente usado em nosso país (PC-WORLD, 2001).

Além da edição de texto, acesso à *internet*, jogos, e uma extensa lista de utilitários falados, dentre os aplicativos do “Dosvox” devemos dar destaque ao “LentePro”, que funciona como uma lupa eletrônica para auxiliar pessoas com visão subnormal e ao “Monitvox”, que funciona como um leitor de telas simplificado.

2.2.2 Utilitários para Transcrição *Braille*

Braille Fácil: desenvolvido e distribuído livremente pelo NCE da UFRJ sob orientação dos Professores “José Antônio Borges” e “Geraldo José Chagas Júnior”, foi produzido com recursos provenientes do “Fundo Nacional de Desenvolvimento da Educação” (FNDE) para o projeto dos “Centros de Apoio Pedagógico” (CAP) do “Ministério da Educação” (MEC) e tem os direitos autorais em posse do “Instituto Benjamin Constant” (IBC) (BRAILLE FÁCIL, 2005). Além de funcionar como transcritor de texto do formato escrito (caracteres) para o sistema *Braille* (com a auxílio de uma impressora *Braille*), incorpora um simples, mas fantástico editor gráfico capaz de transcrever figuras do formato digital (desenhos ou fotografias digitalizadas) para figuras táteis (alto relevo) em poucos segundos (figura 2.12).

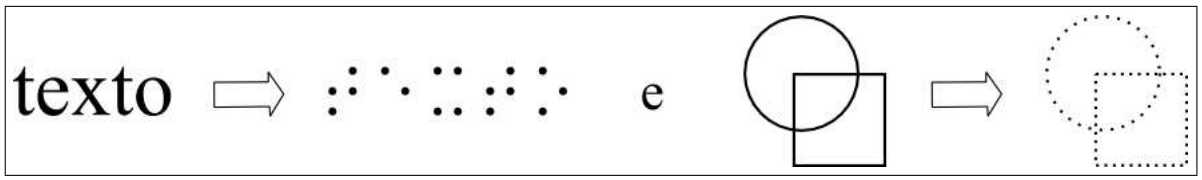


Fig. 2.12: Braille Fácil

BR Braille: foi desenvolvido e é distribuído livremente pela "Faculdade de Engenharia Elétrica e de Computação" (FEEC) da UNICAMP, fruto da dissertação de Mestrado de "Cláudia Maria Caixeta Bezerra", em conjunto com dois projetos de iniciação científica, de "Adriana Keiko Kawai" e "Rodrigo de Passos Barros", sob orientação da Professora "Vera Lúcia da Silveira Nantes Button" (BEZERRA et al., 2004). Este *software* permite que documentos em *Braille*, com o auxílio de uma mesa digitalizadora, sejam transcritos para o formato de texto escrito, e então armazenados por meios eletrônicos (figura 2.13). Esse recurso permite que professores possam corrigir tarefas escolares elaboradas por alunos cego, mesmo não conhecendo o sistema *Braille* (BR BRAILLE, 2005).

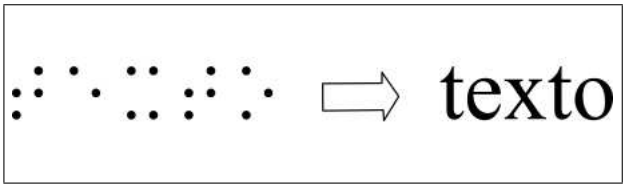


Fig. 2.13: BR Braille

2.3 Tecnologias "Livres" para Sistemas Operacionais "Livres"

Nesta seção estão apresentados os *softwares* "livres" desenvolvidos para sistemas operacionais "livres", ou seja, não é necessária a aquisição de nenhum tipo de licença para utilização dos *softwares* assim como do sistema operacional para o qual foram desenvolvidos.

2.3.1 Gnopernicus

O *software* "Gnopernicus", desenvolvido e distribuído livremente pela empresa "Baum Engineering SRL" (BAUM ENGINEERING, 2005), é o primeiro e único leitor de telas para sistemas operacionais "livres". Está ainda em fase experimental na versão 0.9, não atingiu

ainda uma versão estável, mas vem sendo bastante utilizado e bastante comentado nos fóruns e listas de discussões sobre acessibilidade em ambientes “livres”.

Seu funcionamento é semelhante ao do *software* “Jaws” descrito na subseção 2.1.1, mas além de leitura de telas, possui recursos de ampliação de telas semelhantes aos do *software* “ZoomText” descrito na subseção 2.1.2. Foi desenvolvido originalmente para o sistema operacional “Linux”, mas funciona em qualquer derivado do “Unix”, ex. “FreeBSD” e “Solaris”, desde que como gerenciador de janelas seja habilitado o “Gnome”. Os sistemas operacionais “livres” possuem em geral vários gerenciadores de janelas (“KDE”, “Gnome”, “IceWM”, “XFCE”, etc.) e uma das limitações do “Gnopernicus” é que funciona apenas no “Gnome” e interage apenas com as aplicações desenvolvidas para esse ambiente.

Não está disponível no idioma português, mas existe um projeto em andamento na “Universidade Federal do Rio Grande do Sul” (UFRGS) para tal implementação.

2.3.2 Emacspeak

O *software* “Emacspeak”, distribuído livremente sob os termos da GPL, foi desenvolvido pelo pesquisador “TV Raman” em 1996, está em constante evolução e atualmente encontra-se na versão 22.0 (EMACSPEAK, 2005). Foi o primeiro recurso que possibilitou aos DVs a utilização de sistemas operacionais “livres” (RAMAN, 1996). Devido ao alto nível de estabilidade atingido por essa tecnologia assistiva, é seguramente a tecnologia assistiva “livre” de melhor aceitação e também o de maior utilização a nível mundial.

Seu funcionamento é semelhante ao de um leitor de telas, pois trabalha em conjunto com outras aplicações, mas ao invés de “ler” o que está sendo exibido na tela, essa ferramenta faz com que as aplicações falem com o usuário (RAMAN, 1996). Isso faz com que essa interação seja mais eficiente, mais rápida e reduz a incidência de erros na interpretação das telas.

O principal requisito para seu funcionamento é o ambiente “Emacs” (ambiente de trabalho integrado capaz de executar rapidamente uma infinidade de aplicativos, com os quais é possível realizar praticamente qualquer tipo de atividade no computador), pois para que uma aplicação seja habilitada para dialogar com o usuário, é necessário que esteja sendo executada dentro desse ambiente. Por esse motivo é que o nome dessa ferramenta é “Emacspeak”.

“Emacs” + “Speak” = “Emacspeak”

Não está disponível no idioma português e não existem projetos em andamento para tal implementação. Mais detalhes sobre o funcionamento dessa tecnologia serão apresentados na próxima seção deste capítulo, onde serão abordados também aspectos técnicos de funcionamento para verificar a viabilidade de adaptação para o idioma português do Brasil.

2.4 Análise Comparativa: Gnopernicus X Emacspeak

Como visto anteriormente na seção 2.3, em matéria de *software* são apenas duas as tecnologias assistivas na área de deficiência visual que são realmente "livres" de custo para sua utilização, o "Gnopernicus" e o "Emacspeak". Nenhuma delas encontra-se disponível no idioma português, mas ambas oferecem viabilidade para adaptação, pois possuem o código fonte aberto de acordo com os termos da GPL. Resta agora definir qual das duas oferece melhor viabilidade para tal adaptação e este é exatamente o objetivo da análise comparativa apresentada nesta seção.

Entre os quesitos a serem comparados estão a estabilidade de funcionamento e utilização, a portabilidade entre os diferentes sistemas operacionais, a facilidade de manuseio para os usuários sejam eles iniciantes ou não, a facilidade de desenvolvimento para os mantenedores do projeto e o equipamento mínimo necessário para o funcionamento da tecnologia.

2.4.1 Estabilidade

O nível de estabilidade de uma ferramenta pode ser obtido de acordo com sua capacidade de tolerância à erros cometidos pelos usuários ou pelo próprio sistema operacional. Podemos dizer que a virtude "estabilidade" está diretamente relacionada à "maturidade", ou seja, quanto mais "maduro" for um determinado sistema, maior será a "estabilidade" oferecida pelo mesmo. Note também que "estabilidade" nada tem haver com "sofisticação", pois para este quesito, de nada adianta uma tecnologia oferecer recursos inéditos e inovadores que ainda não tenham sidos testados e aprovados nas mais diversas circunstâncias e com as mais diversas categorias de usuários.

Não é difícil concluir que a tecnologia escolhida nesse quesito foi o "Emacspeak", pois além de ser uma ferramenta muito mais "madura" (versão 22.0), utiliza como base o *software* "Emacs" que é um ambiente de estabilidade incontestável que vem sendo desenvolvido desde 1976, portanto também incontestavelmente "maduro". Enquanto o "Gnopernicus", apesar de oferecer recursos de maior sofisticação, encontra-se em fase de testes e ainda não atingiu uma versão estável.

Gnopernicus $\boxed{0}$ X $\boxed{1}$ Emacspeak

2.4.2 Portabilidade

A portabilidade de uma ferramenta entre os diferentes sistemas operacionais se faz importante principalmente quando a intenção é o uso corporativo. Em geral as empresas adotam

um determinado sistema operacional e a implantação do mesmo é feita de forma generalizada em todos os computadores pertencentes à corporação. Mesmo tendo sua eficácia comprovada, quando uma determinada ferramenta não está disponível para o sistema operacional adotado pela corporação, ela é na grande maioria das vezes simplesmente "descartada". Portanto, quanto maior a portabilidade, ou seja, quanto maior o número de sistemas operacionais para os quais está disponível uma determinada tecnologia, melhor será sua aceitação e maior será a possibilidade de implantação da mesma.

Vimos na subseção 2.3.1 que o "Gnopernicus" funciona em qualquer sistema operacional derivado do "Unix", ex. "Linux", "FreeBSD" e "Solaris", desde que como gerenciador de janelas seja habilitado o "Gnome". Mas o "Emacspeak", além de funcionar em todos esses sistemas independente de qual seja o gerenciador de janelas utilizado, pode funcionar também fora de qualquer gerenciador de janelas (fora do ambiente gráfico, em modo texto) e com algumas adaptações pode também funcionar em sistemas operacionais proprietários, ex. "Microsoft Windows®" e "Apple MacOS®", o que amplia consideravelmente a portabilidade desse software. Portanto a tecnologia escolhida nesse quesito foi também o "Emacspeak".

Gnopernicus $\boxed{0}$ X $\boxed{2}$ Emacspeak

2.4.3 Facilidade de Utilização

Este é provavelmente o mais importante dos quesitos para a seleção da tecnologia a ser adaptada, pois esse será o de maior relevância para que seja efetivada a aceitação do usuário. A importância deste item ganha um peso ainda maior quando observa-se que não serão apenas usuários experientes que farão uso de tal tecnologia, entre os usuários propícios estão desde crianças ainda não alfabetizadas (com deficiência visual congênita) até idosos com perda recente da visão (ocasionada por problemas como glaucoma, catarata ou acidentes). Portanto, quanto menor forem os conhecimentos prévios necessários para utilização da tecnologia em questão, melhor será a sua íntegra aceitação.

O procedimento necessário para realizar uma determinada tarefa com o "Gnopernicus" é obviamente diferente do procedimento para realizar a mesma tarefa com o "Emacspeak". O "Gnopernicus" segue o padrão dos já conhecidos leitores de telas, enquanto o "Emacspeak" utiliza comandos específicos do ambiente "Emacs". Por exemplo, para editar um arquivo de texto com o "Gnopernicus", é necessário que primeiramente seja iniciado o aplicativo editor de textos (*acionar o menu de programas pressionando simultaneamente as teclas "Control" e "Escape", localizar o ícone do editor utilizando as setas direcionais e iniciá-lo pressionando a tecla "Enter"*) e em seguida que o arquivo seja carregado (*acionar a barra de menu pres-*

sionando a tecla "Alt", localizar a opção "Abrir" utilizando as setas direcionais, pressionar "Enter" para acionar a opção, digitar o nome do arquivo e finalizar pressionando a tecla "Enter"). Já com o "Emacspeak", a inicialização do editor de textos e o carregamento do arquivo podem ser executados de forma integrada, para isso basta pressionar simultaneamente as teclas "Control" e " x ", "Control" e " f ", digitar o nome do arquivo e finalizar pressionando "Enter".

Esse exemplo é apenas uma das diversas circunstâncias que foram comparadas para definir qual das ferramentas apresenta maior facilidade de utilização, mas praticamente em todas as comparações o resultado foi o mesmo. O procedimento do "Emacspeak" sempre se mostrou mais fácil, mais rápido e mais eficiente, enquanto que no "Gnopernicus" a incidência de erros foi muito maior devido à complexidade dos procedimentos que exigem profundos conhecimentos de ambientes gráficos (menus, janelas, etc.). Apesar de agregar uma elevada carga mnemônica de combinação de teclas, o "Emacspeak" foi a tecnologia escolhida nesse quesito devido à possibilidade de implementação de menus interativos (semelhante aos apresentados pelo "Dosvox" na subseção 2.2.1), e essa implementação de recursos adicionais está detalhada e documentada no capítulo 3.

Gnopernicus $\boxed{0}$ X $\boxed{3}$ Emacspeak

2.4.4 Facilidade de Desenvolvimento

A princípio este não seria um quesito de muita importância para a seleção da tecnologia, pois o desenvolvimento é uma tarefa para os mantenedores do projeto, e não para os usuários do sistema. Mas essa situação se inverte completamente quando se cogita a possibilidade de que os próprios usuários DVs possam também colaborar no desenvolvimento do projeto, pois isso tornaria ilimitada as possibilidades de tal tecnologia e seria uma garantia de constante evolução.

Para analisar a complexibilidade de desenvolvimento de uma determinada ferramenta, podemos seguir o mesmo raciocínio apresentado na subseção 2.4.3: quanto menor forem os conhecimentos prévios necessários para atuar no desenvolvimento da tecnologia em questão, maior será a possibilidade de que os usuários DVs possam também participar desse projeto.

- Para atuar no desenvolvimento do "Gnopernicus" é preciso conhecer:
  - A linguagem " C ";
  - A biblioteca de desenvolvimento Xlib (X Library);
  - A biblioteca de desenvolvimento Glib (Gimp Library);

  - A ferramenta de desenvolvimento GTK (Gimp Toolkit);
  - A ferramenta auxiliar de desenvolvimento GDK (Gimp Draw Kit);
  - A ferramenta auxiliar de desenvolvimento ATK (Access Toolkit);
  - E avançados conhecimento de programação em ambientes gráficos.

- E para atuar no desenvolvimento do "Emacspeak" é preciso conhecer apenas:

  - A linguagem " C ";
  - E a linguagem "Lisp".

O "Emacspeak" foi também a tecnologia escolhida nesse quesito, uma vez que é visivelmente maior a quantidade de conhecimentos necessários para atuar no desenvolvimento do "Gnopernicus".

Gnopernicus $\boxed{0}$ X $\boxed{4}$ Emacspeak

2.4.5 Requisitos de *Hardware*

A relevância deste quesito está no intuito de aproveitar da melhor maneira possível o maquinário disponível, pois de nada adianta estimar a redução de custos com a implantação do software "livre" se para o bom funcionamento do mesmo faz se necessária a aquisição de novos computadores.

Gnopernicus: vimos na subseção 2.3.1 que esta é uma tecnologia que trabalha em conjunto com o gerenciador de janelas "Gnome". Em geral, para que seja utilizado um ambiente gráfico do tipo "Gnome", o tamanho mínimo de memória RAM recomendado é de 64 Mega Bytes, a velocidade mínima de processamento recomendada é de 300 Mega Hertz e o espaço que esse ambiente e as ferramentas básicas vão utilizar no disco rígido é de aproximadamente 500 Mega Bytes. Mas além do ambiente gráfico, o computador tem que executar simultaneamente o "Gnopernicus" e o aplicativo do qual será feita a leitura da tela. Portanto, as especificações mínimas do equipamento necessário para utilização dessa tecnologia são:

- Computador Pentium ou AMD K6;
- Processador de 500MHz;
- 128MB de memória RAM;
- 800MB de disco rígido;

Emacspeak: vimos na subseção 2.3.2 que esta é uma tecnologia que trabalha em conjunto com o ambiente "Emacs". Vimos também na subseção 2.4.2 que o "Emacs" é um ambiente muito versátil que pode também ser executado em modo texto, o que reduz consideravelmente os requisitos de *hardware* para seu perfeito funcionamento. Em geral, para que seja utilizado o modo texto de um sistema operacional qualquer, o tamanho mínimo de memória RAM recomendado é de 8 Mega Bytes, a velocidade mínima de processamento recomendada é de 66 Mega Hertz e o espaço que esse sistema e as ferramentas básicas vão utilizar no disco rígido é de aproximadamente 50 Mega Bytes. Além do modo texto, o computador tem que executar simultaneamente o ambiente "Emacs" e o "Emacspeak", mas como a grande maioria de *softwares* para modo texto, esses também exigem baixas especificações de *hardware* para sua utilização. Portanto, as especificações mínimas do equipamento necessário para utilização dessa tecnologia são:

- Computador 486;
- Processador de 100MHz;
- 16MB de memória RAM;
- 100MB de disco rígido;

Gnopernicus $\boxed{0}$ X $\boxed{5}$ Emacspeak

2.4.6 Resultado e Seleção

Para encerrar a fase de seleção da tecnologia assistiva a ser adaptada para o idioma português do Brasil, a tabela 2.1 apresenta o resultado da análise comparativa iniciada na seção 2.4.

Quesito	**Gnopernicus**	**Emacspeak**
Estabilidade	0	1
Portabilidade	0	1
Facilidade de Utilização	0	1
Facilidade de Desenvolvimento	0	1
Requisitos de *Hardware*	0	1
Total	$\boxed{0}$	$\boxed{5}$

Tab. 2.1: Resultado e Seleção

O **"Emacspeak"** foi então a tecnologia escolhida para ser traduzida e adaptada. Os detalhes referente à essa adaptação e à implementação de novos recursos facilitadores estão apresentados e documentados no capítulo seguinte.

Capítulo 3

Implementação

Este capítulo apresenta a implementação necessária para que a tecnologia assistiva selecionada no capítulo anterior seja disponibilizada no idioma português do Brasil. Esse processo está basicamente dividido em duas etapas:

- Implementação do sintetizador de voz;
- Tradução da tecnologia assistiva.

Por se tratar de uma base operacional "livre", o processo de implementação e tradução baseia-se em adaptações inseridas no código fonte de *softwares* já existentes. Porém, foi necessária a implementação integral do conversor "*Text-to-Speech*" (TTS) para o idioma português, que é o dispositivo que transforma uma determinada entrada de conteúdo textual em fonemas vocais (conversor de texto para fala). Também foi necessária a implementação de recursos inéditos relativos à interação com o usuário para satisfazer o nível de "facilidade de utilização" estipulado na subseção 2.4.3.

3.1 Implementação do Sintetizador de Voz

Devido à complexidade deste assunto, que envolve diversos conceitos nas áreas de "*Natural Language Processing*" (NLP) e "*Digital Signal Processing*" (DSP), não é a intenção desta seção explorar detalhadamente todos os aspectos envolvidos na geração digital de síntese de voz. Entretanto, apesar de introdutório, com os conceitos aqui apresentados será possível não apenas entender o processo de adaptação, como também habilitar desenvolvedores para que possam futuramente implementar adaptações para outros idiomas ou até mesmo para outros dialetos.

Portanto, esta será uma breve, mas compreensiva explanação dos procedimentos que foram necessários para a obtenção de um sintetizador de voz para o idioma português do Brasil.

3.1.1 Introdução à Síntese de Voz

Um gerador digital de síntese de voz, ou simplesmente, um sintetizador de voz, é um sistema computacional capaz de ler um texto qualquer em "voz alta", ou seja, um aplicativo capaz de transformar uma entrada de texto em saída de áudio (DUTOIT, 1996).

Existem vários tipos de sintetizadores de voz, mas de uma forma geral, para que o áudio seja gerado a partir de um texto qualquer, é necessário que esse texto seja submetido a dois mecanismos fundamentais:

- **O NLP:** que é o processamento de linguagem natural, dispositivo que transforma o conteúdo textual em informações fonéticas, também referenciado por diversos autores da área como o TTS do sistema;
- **E o DSP:** que é o processamento digital de sinais, responsável pela geração do áudio a partir das informações fonéticas recebidas do NLP (figura 3.1).

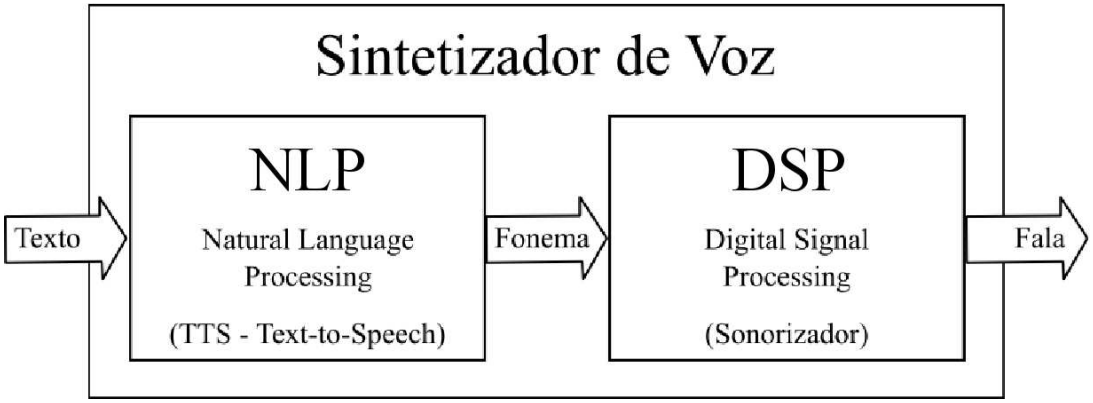


Fig. 3.1: Sintetizador de Voz - Diagrama Geral

Fazendo uma analogia ao sistema humano de pronúncia, podemos dizer que as funções do NLP (ou TTS) são desempenhadas pelo "cérebro", que é o responsável pela captação das informações a serem pronunciadas e pela manipulação e transposição desse conteúdo para o "sistema emissor de áudio", que nessa analogia desempenha as funções do DSP (ou sonorizador) e é composto pela boca em conjunto com pulmões e cordas vocais.

Texto $\Longrightarrow$ **TTS** (cérebro) $\Longrightarrow$ **Sonorizador** (boca, pulmões e cordas vocais) $\Longrightarrow$ *Fala*

3.1.2 Sonorizador - DSP

Apesar de ser apresentado pela figura 3.1 como sendo o segundo mecanismo pertencente ao gerador digital de síntese de voz, o sonorizador é o primeiro a ser explorado devido á existência de pré-requisitos necessário para um melhor entendimento do TTS, que está posteriormente detalhado na subseção 3.1.3.

Com os conceitos apresentados na subseção 3.1.1, podemos concluir intuitivamente que o sonorizador é um módulo de processamento digital de sinais que desempenha a tarefa de simular articulações musculares (boca e pulmões) e freqüências vibratórias (cordas vocais), de modo que o resultado na saída de áudio se assemelhe à voz humana.

Segundo DUTOIT (1996), essa simulação é geradas pelo sonorizador a partir de conceitos fonéticos que podem ser armazenado basicamente de duas maneiras:

- **Explicitamente:** na forma de uma série de regras que formalmente descrevem a influência de determinados fonemas vocálicos sobre outros;
- **Implicitamente:** pela gravação de pequenas amostras auditivas de transições fonéticas e co-articulações em uma espécie de "base de dados fonéticos", para que sejam utilizados sempre que necessário de forma concatenativa.

Esses dois métodos de armazenamento de conceitos vocálico originam também dois tipos distintos de sonorizadores:

- **O Sonorizador Baseado em Regras** - armazenamento de modo explícito;
- **E o Sonorizador Concatenativo** - armazenamento de modo implícito.

A quantidade de informações a serem processadas pelos sonorizadores é consideravelmente menor quando o sistema de armazenamento fonético adotado é o de modo implícito. Isso se deve ao fato de que enquanto o de modo explícito tem que analisar uma infinidade de regras para gerar o áudio referente a um determinado fonema, o modo implícito simplesmente vai buscar esses segmentos auditivos pré-gravados em uma base de dados fonéticos pré-estabelecida.

Em contrapartida, o modo implícito requer um espaço muito maior em disco rígido para que seja armazenada a base de dados fonéticos. Porém, graças aos sofisticados algoritmos de compressão de dados atualmente disponíveis para o processamento digital desse tipo de informação, esse quesito torna-se praticamente irrelevante (DUTOIT et al., 1996). Como exemplos de algoritmos de compressão para base de dados fonéticos podemos citar:

- **LPC** - *Classical Auto-Regressive*;

- **H/S Model** - *Hybrid Harmonic/Stochastic Model*;
- **TD-PSOLA** - *Time-Domain Pitch-Synchronous OverLap-Add*;
- **MBR-PSOLA** - *Mult-Band Re-Synthesis Pitch-Synchronous OverLap-Add*.

Como já dito anteriormente, não é a intenção desta seção explorar detalhadamente todos os aspectos envolvidos na geração digital de síntese de voz, portanto não serão apresentados detalhes referentes à esses diferentes algoritmos de compactação. Mas para um entendimento geral, consideremos que a atuação desses algoritmos é semelhante à do famoso e largamente utilizado sistema de compactação de arquivos no formato "ZIP" (que utiliza o algoritmo de Lempel-Ziv), com a adição de características e rotinas específicas para manipulação de segmentações de áudio.

Sendo assim, para minimizar os "requisitos de hardware" e manter estável as necessidade de acordo com o estipulado na subseção 2.4.5, o tipo de sonorizador selecionado para construção do sintetizador de voz apresentado nesse projeto é o "concatenativo", ou seja, aquele que utiliza o modo implícito para armazenamento dos conceitos fonéticos.

O diagrama então referente ao sonorizador a partir deste ponto, passa a contar com dois módulos operacionais, o "Processador Digital de Sinais" e a "Base de Dados Fonéticos" (figura 3.2):

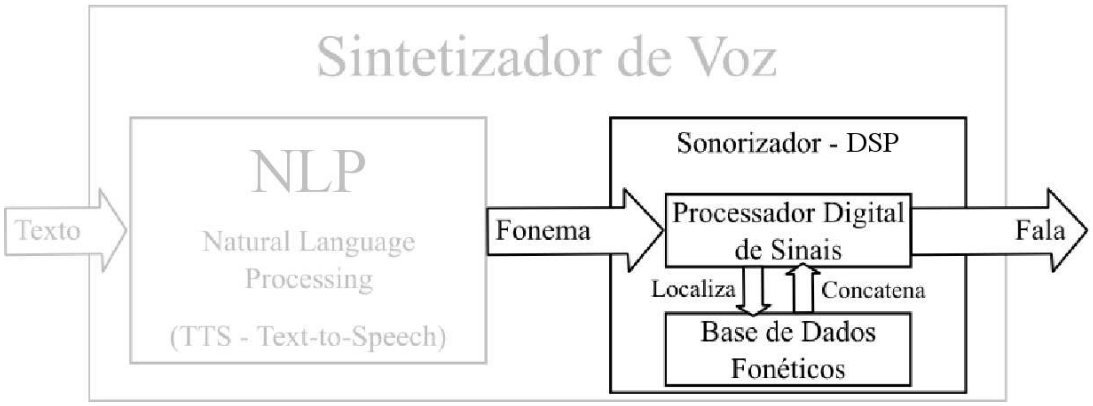


Fig. 3.2: Sonorizador - DSP

Processador Digital de Sinais: de acordo com a diagramação atual, este é o dispositivo que recebe a informação fonética do TTS e localiza na base de dados fonéticos o segmento correspondente. Depois de localizado, esse segmento é então extraído (descompactado) da base de dados e concatenado aos demais fonemas requisitados pelo TTS. Somente após finalizada a solicitação do TTS é que então esse conjunto de segmentos concatenados é finalmente enviado para a saída de áudio do computador;

Base de Dados Fonéticos: é basicamente um arquivo composto por dezenas de segmentos de áudio, devidamente organizados e compactados de forma que possam ser facilmente localizados e extraídos pelo processador digital de sinais.

Esses segmentos são basicamente frações de segundos de áudio de uma voz humana captados por intermédio de um microfone comum. Mas ao contrário do que possa parecer, esses segmentos de áudio não se referem ao armazenamento de sílabas, e sim de frações de sílabas. Por exemplo:

- Para que seja gerada a síntese do texto “ba” é necessário que o TTS gere um fonema requisitando o segmento “ b ” concatenado ao segmento “ a ” (figura 3.3);

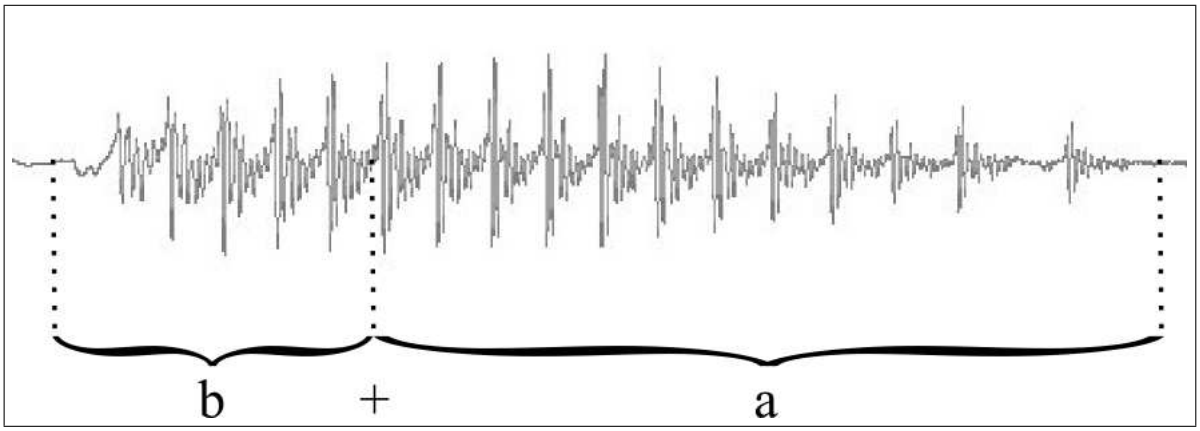


Fig. 3.3: Onda Sonora - “ba”

- E para que seja gerada a síntese do texto “Brasil” é necessário que o fonema gerado pelo TTS requisite a concatenação dos segmentos: “ br ”, “ a ”, “ z ”, “ i ” e “ u ” (figura 3.4).

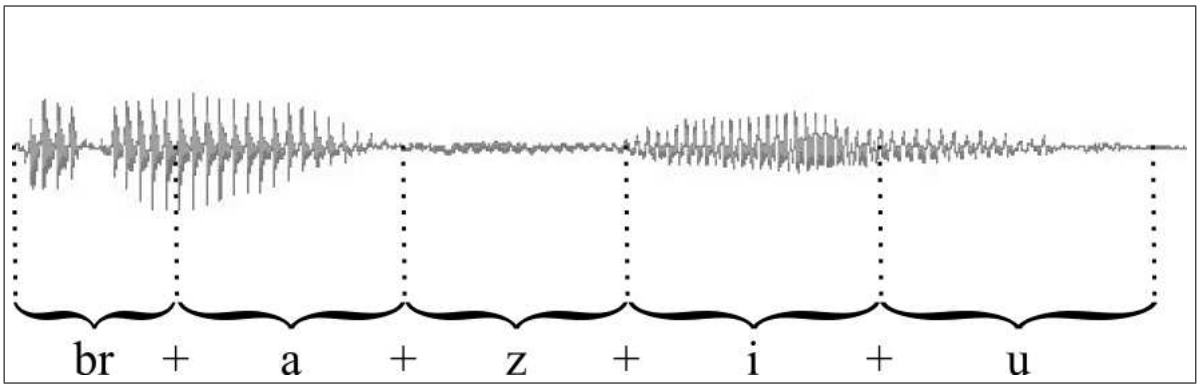


Fig. 3.4: Onda Sonora - “Brasil”

Para garantir a eficácia do sonorizador, seja ele concatenativo ou não, é essencial que o projetista conheça detalhadamente as especificações e características desse dispositivo antes que seja iniciado o desenvolvimento do TTS. No caso especial do concatenativo, as informações que devem essencialmente estar presentes nos fonemas gerados pelo TTS são três:

- **O código dos segmentos fonéticos:** que é a referência pela qual será solicitada a sonorização de uma determinada fração de sílaba;
- **A faixa de duração:** que são os períodos (mais curto ou mais longo) aceitáveis para que seja sustentado cada um dos segmentos fonéticos concatenados;
- **E a faixa de freqüência:** que são as tonalidades (mais grave ou mais agudo) aceitáveis para que seja pronunciado cada um dos segmentos fonéticos concatenados.

Mais detalhes referentes à concatenação de segmentos e geração se fonemas estão apresentados na subseção seguinte, que destina-se à exploração do primeiro mecanismo pertencente ao gerador digital de síntese de voz, o TTS.

3.1.3 *Text-to-Speech* - TTS

Levando em consideração a analogia e os conceitos apresentados na subseção anterior, podemos afirmar que entre os dois dispositivos básicos que compõem o gerador digital de síntese de voz (TTS e sonorizador), o TTS é seguramente o de maior importância. Podemos afirmar também, de uma forma bastante resumida, que o sonorizador é um reservatório de sons inteiramente controlado pelo TTS. Porém, isso não minimiza a importância do sonorizador, pois uma boa qualidade de áudio depende do bom funcionamento de ambos.

Esta subseção apresenta os principais conceitos necessários para que seja implementado um fiel conversor TTS. Como todo conversor, este é também um dispositivo que fornece um determinado tipo de informação a partir de uma dada entrada, ou seja, é um dispositivo que possui uma entrada e uma saída.

$$\textit{Texto} \Longrightarrow \boxed{\textbf{TTS}} \Longrightarrow \textit{Fonema}$$

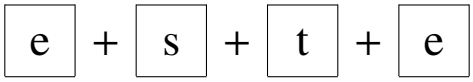
Mesmo sendo este conversor projetado para um idioma específico, a entrada será sempre imprevisível. Apesar de um conteúdo sempre textual, jamais será possível prever as informações que estarão sendo submetidas à entrada do TTS. Por esse motivo o TTS deve estar sempre preparado para receber qualquer tipo de informação, sinais, números ou palavras, sejam elas pertencentes ou não ao dicionário do idioma para o qual foi projetado.

Vimos na subseção anterior que o tipo de sonorizador selecionado para a construção do sintetizador de voz foi o concatenativo, ou seja, aquele que trabalha em conjunto com uma base de dados fonéticos. Vimos também que as informações que devem essencialmente estar presentes nos fonemas a serem produzidos são: código fonético, duração e freqüência. Portanto, para que seja gerado um fonema a partir de uma palavra qualquer, cada fração de sílaba será codificada sempre preenchendo esses três diferentes campos.

O valor dos campos a serem codificados pode variar, mesmo quando codificando caracteres iguais pertencentes a uma mesma palavra, de acordo com o posicionamento da fração de sílaba dentre as demais. Por exemplo:

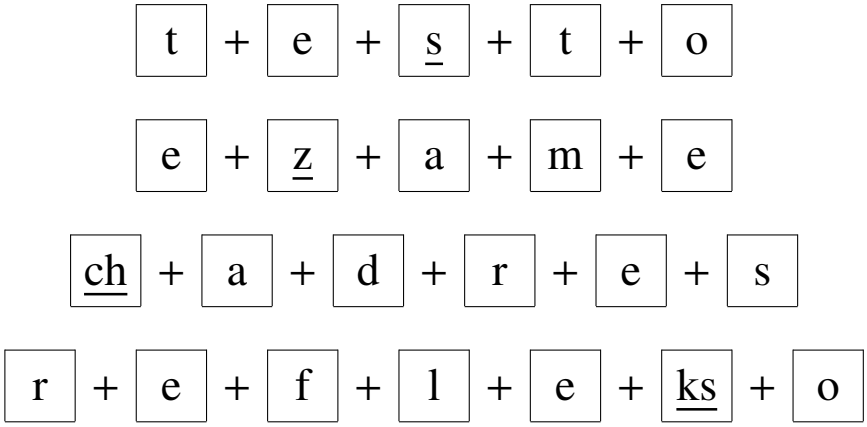
- **Na palavra "teste",** a grafia do caractere " e " localizado na segunda posição é exatamente igual à grafia do " e " localizado na quinta posição. Entretanto a pronúncia de ambos é completamente distinta, o primeiro deve ser pronunciado com a boca aberta sonorizando como se fosse um " é " (acentuado), enquanto o segundo deve ser pronunciado com a boca fechada sonorizando mesmo como sendo um " e " (não acentuado);

- **Já na palavra "este",** essa correspondência é praticamente linear, pois tanto a grafia quanto a pronúncia do caractere " e " localizado na primeira posição são idênticas à grafia e à pronúncia do " e " localizado na quarta posição, sonorizando mesmo como sendo um " e " (não acentuado);



- **Na palavra "bolo",** a grafia do caractere " o " localizado na segunda posição é exatamente igual à grafia do " o " localizado na quarta posição e a pronúncia de ambos é também exatamente igual, sonorizando como sendo mesmo um " o " (não acentuado). Mas essa correspondência lógica se distorce novamente quando o termo a ser convertido para fonemas é **a palavra "colo",** pois nesse caso o " o " localizado na segunda posição deve ser pronunciado com a boca aberta sonorizando como se fosse um " ó " (acentuado) enquanto esse mesmo caractere quando localizado na quarta posição deve ser pronunciado com a boca fechada sonorizando mesmo como sendo um " o " (não acentuado);

- **Nas palavras “texto”, “exame” e “xadrez”,** a grafia do caractere “ x ” é exatamente igual diferindo-se apenas no posicionamento, mas a pronúncia é completamente diferente, sendo na primeira palavra sonorizado como sendo um “ s ”, na segunda como sendo um “ z ” e na terceira como sendo um “ ch ”. E existe ainda uma quarta diferente situação onde o caractere “ x ” deve ser sonorizado como sendo uma espécie de “ ks ”, que é o caso por exemplo **da palavra “reflexo”.**



Estas são apenas algumas das infinitas particularidades que estão presentes no idioma português. Identificar e sanar as necessidades do maior número possível dessas particularidades é a principal função do TTS. Mais detalhes referentes à essas e outras particularidades e as soluções propostas nesse projeto para o saneamento das mesmas estão apresentados na subseção 3.1.5, que refere-se à construção do TTS.

3.1.4 Construção do Sintetizador de Voz

Como prática comum no desenvolvimento de *softwares* de utilização “livre”, antes que fosse iniciada a construção do sintetizador de voz, uma vasta pesquisa foi realizada com a intenção de encontrar, estudar e até mesmo reutilizar trabalhos que tenham sido desenvolvidos com esse mesmo objetivo para outros idiomas, afim de reduzir os esforços e agilizar o processo de construção.

Como conclusão dessa pesquisa, que foi realizada por diversos mecanismos de busca na internet e em bases de dados de publicações nacionais e internacionais, dois sintetizadores de utilização “livre” para idiomas estrangeiros foram indicados como viáveis para adaptação na construção deste projeto específico para o idioma português do Brasil: o “Festival” e o “Mbrola”.

A pesquisa indicou também o “Aiuruetê” que é um sintetizador desenvolvido aqui mesmo na FEEC da UNICAMP sob orientação do Professor Fábio Violaro (VIOLARO et al., 1999),

mas infelizmente esse não pode ser utilizado por estar operando somente em conjunto com o sistema operacional proprietário "Microsoft Windows®".

Tanto o "Festival" quanto o "Mbrola" haviam sido originalmente desenvolvidos para o idioma inglês e o tipo de sonorizador adotado pelos mesmos foi também o concatenativo, sendo que o "Festival" adotara como método de compressão da base de dados fonéticos o algoritmo "LPC", enquanto o "Mbrola" adotara para a mesma função o algoritmo "MBR-PSOLA". O resultado da pesquisa indicou também diversas experiências realizadas com sucesso de adaptação desses projetos para outros idiomas, o que propiciou ainda mais a adesão por uma dessas tecnologias.

O "Festival", desenvolvido pela Universidade de Edinburgo - Inglaterra (FESTIVAL, 2005) conta inclusive com um projeto denominado "Festvox" desenvolvido exclusivamente para ajudar desenvolvedores interessados na adaptação do mesmo para outros idiomas (FESTVOX, 2005). E o "Mbrola", desenvolvido pela Faculdade Politécnica de Mons - Bélgica (MBROLA, 2005), além de um projeto semelhante para ajudar nessa adaptação, conta com uma excelente e já concluída base de dados fonéticos para o idioma português do Brasil.

Vimos na subseção 3.1.1 que um sintetizador de voz é composto por dois mecanismos fundamentais: o TTS e o sonorizador. Portanto, é claro que o fato de contar com uma base de dados fonéticos para o idioma português do Brasil não significa que "Mbrola" já esteja pronto para falar em português, mas sim que o mecanismo do sonorizador já está pronto para receber informações fonéticas de um TTS brasileiro. Existe também no repositório de projetos relacionados ao "Mbrola" algumas tentativas de construção do tal TTS para o idioma português do Brasil, mas infelizmente nenhum deles obteve sucesso provavelmente pelo fato de terem sido desenvolvidos por estrangeiros que certamente não possuíam domínio suficiente sobre as particularidades que compõem esse idioma. Outra razão que descartou a utilização desses esboços de TTS foi o fato de terem sido desenvolvidos em linguagem de programação "PERL", pois prosseguir com o desenvolvimento nessa direção significaria a inclusão de mais um "conhecimento prévio" necessário para atuar no desenvolvimento da tecnologia em questão e isso estaria contra os princípios estipulados na subseção 2.4.4.

Então, o "Mbrola" foi selecionado para compor a construção do sintetizador de voz para o idioma português do Brasil. Desse modo fica eliminada a necessidade de construção de um novo sonorizador e a partir deste ponto trataremos apenas da construção do TTS.

3.1.5 Construção do TTS

Antes de iniciar a construção do TTS para o idioma português do Brasil é preciso conhecer as especificações e características do sonorizador do "Mbrola" para que os campos código

fonético, duração e freqüência, sejam eficientemente preenchidos.

Código Fonético: as tabelas 3.1 e 3.2 relacionam cada um dos caracteres de distinta sonorização necessários para geração de síntese de voz no idioma português do Brasil, o código correspondente para que sejam acessados na base de dados fonéticos do sonorizador "Mbrola" e um exemplo de palavra contendo o caractere em questão destacado para um melhor entendimento do segmento auditivo correspondente:

Caractere	Código	Exemplo
b	b	(b) a r c o
c	k	(c) a m a
d	d	(d) o c e
f	f	(f) á c i l
g	g	(g) r a n d e
h	-	-
j	j	(j) a t o
l	l	(l) a n c h e
lh	lh	a (lh) o
m	m	(m) e s a
n	n	(n) u v e m
nh	nh	g a l i (nh) a
p	p	(p) a i
q	k	(q) u e i j o
r	r	p u (r) o
r	r2	a (r) p a
rr	rr	t o (rr) e
s	s	(s) a l a
s	s2	c a (s) c a
s	z	a (s) a
t	t	(t) a c o
v	v	(v) i n h o
x	x	(x) a d r e z
x	s2	t e (x) t o
x	z	e (x) a m e
x	k + i + s	r e f l e (x) o
z	z	(z) i n c o

Tab. 3.1: Sonorizador - Código Fonético (consoantes)

Caractere	Código	Exemplo
a	a	v (a) l e
	@	r (a) m o
	am	c (am) p o
e	e	p (e) n a
	ee	c (é) u
	em	t (em) p o
i	i	p (i) c o
	y	m a (i) s
	im	l (im) p o
o	o	b (o) l o
	oo	b (ó) i a
	om	(on) t e m
u	u	d (u) r o
	w	m a (u)
	um	a l g (um)

Tab. 3.2: Sonorizador - Código Fonético (vogais)

Duração: o funcionamento desse campo é bastante simples, deve ser preenchido com um valor numérico que representa o tempo em mili segundos no qual permanecerá soando o segmento fonético solicitado (tabela 3.3). A dificuldade nesse ponto está em dividir esses espaços de tempos de forma natural e não linear de maneira que o resultado final se assemelhe da melhor maneira possível à voz humana.

Duração	Segundos
1	0,001
10	0,01
100	0,1
1000	1
10000	10

Tab. 3.3: Sonorizador - Duracao

Freqüência: o funcionamento desse campo é também bastante simples, deve ser preenchido com um ou mais valores numéricos que representam a freqüência em Hertz na qual será submetido o segmento fonético solicitado (tabela 3.4). E mais uma vez a dificuldade desse ponto está em escolher freqüências mais graves e mais agudas de forma natural e não linear de maneira que o resultado final se assemelhe da melhor maneira possível à voz humana.

Freqüência	Descrição
100	O segmento será enviado para a saída de áudio a 100 Hz.
100 150	Inicia a 100 Hz e sobe linearmente até 150 Hz.
100 50 200	Inicia a 100 Hz, desce para 50 Hz e sobe para 200 Hz.
50 50 50 100	Mantém a 50 Hz por 3/4 da duração e finalmente sobe para 100 Hz.

Tab. 3.4: Sonorizador - Freqüência

Esses são então os possíveis valores que devem ser atribuídos a cada um dos três campos necessários para que um texto qualquer seja convertido para fonemas. Os caracteres são codificados um a um, de acordo com a posição ou a disposição em que se encontram dentre os demais.

Portanto na prática, o fonema resultante da conversão realizada pelo TTS a partir por exemplo da sílaba "ba" apresentada na figura 3.3 se apresentaria da seguinte forma:

$$\textit{"ba"} \Longrightarrow \boxed{\textbf{TTS}} \Longrightarrow \textit{"b 60 120" + "a 90 125"}$$

Onde:

- **" b "** = concatenar o segmento auditivo de código " b ";
- **"60"** = soar este segmento auditivo por 60 mili segundos;
- **"120"** = e manter a freqüência auditiva deste segmento em 120 Hz.
- **" + "** = inicia nova concatenação;
- **" a "** = concatenar o segmento auditivo de código " a ";
- **"90"** = soar este segmento auditivo por 90 mili segundos;
- **"125"** = e manter a freqüência auditiva deste segmento em 125 Hz.

A itemização a seguir relaciona cada um dos possíveis caracteres de entrada e explica os procedimentos implementados para conversão dos mesmos no TTS desenvolvido para esse projeto (o programa completo escrito em linguagem de programação " C " está apresentado no apêndice A):

- " *a* " $\Longrightarrow$ **TTS** . . .
  - Quando seguido por "m" ou "n" ...
    - e quando "m" ou "n" seguido por vogal:
      TTS $\Longrightarrow$ *"@ 80 120"* (ex.: ano, ramo, etc.);
    - e quando "m" ou "n" não seguido por vogal:
      TTS $\Longrightarrow$ *"am 120 120"* (ex.: antes, campo, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *"a 90 120"* (ex.: ave, barco, etc.).

- " $\hat{a}$ " $\Longrightarrow$ **TTS** . . .
  - Quando seguido por "m" ou "n" ...
    - e quando "m" ou "n" seguido por vogal:
      TTS $\Longrightarrow$ *"@ 80 130"* (ex.: ânimo, lâmina, etc.);
    - e quando "m" ou "n" não seguido por vogal:
      TTS $\Longrightarrow$ *"am 120 130"* (ex.: âncora, lâmpada, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *"@ 100 130"* (ocorrência rara, mas deve estar prevista).

- " $\tilde{a}$ " $\Longrightarrow$ **TTS** . . .
  - Quando seguido por "m" ou "n" ...
    - e quando "m" ou "n" seguido por vogal:
      TTS $\Longrightarrow$ *"@ 80 130"* (ocorrência rara, mas deve estar prevista);
    - e quando "m" ou "n" não seguido por vogal:
      TTS $\Longrightarrow$ *"am 120 130"* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *"@ 80 130"* (ex.: distorção, sermão, etc.).

- *“ à ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” ...
    - e quando “m” ou “n” seguido por vogal:
      TTS $\Longrightarrow$ *“@ 80 120”* (ocorrência rara, mas deve estar prevista);
    - e quando “m” ou “n” não seguido por vogal:
      TTS $\Longrightarrow$ *“am 120 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“a 90 120”* + *“a 90 125”* (ex.: às, àquele, etc.).

- *“ á ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” ...
    - e quando “m” ou “n” seguido por vogal:
      TTS $\Longrightarrow$ *“@ 80 130”* (ocorrência rara, mas deve estar prevista);
    - e quando “m” ou “n” não seguido por vogal:
      TTS $\Longrightarrow$ *“am 120 130”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“a 100 130”* (ex.: árvore, mármore, etc.).

- *“ b ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“b 60 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“b 60 120”* (ex.: banco, brasileiro, tablado, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“b 60 120”* + *“i 20 120”* (ex.: abdômen, subsídio, etc.).

- *“ c ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“s 110 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por “e” ou “i”:
    TTS $\Longrightarrow$ *“s 110 120”* (ex.: cérebro, ciúme, etc.).
  - Quando seguido por “a”, “o”, “u”, “l” ou “r”:
    TTS $\Longrightarrow$ *“k 100 120”* (ex.: carro, couro, cume, clima, crista, etc.).
  - Quando seguido por “h”:
    TTS $\Longrightarrow$ *“x 80 120”* + *“h 0 0”* (ex.: chave, lanche, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“k 100 120”* + *“i 20 120”* (ex.: infecção, tecnologia, etc.).

- *“ ç ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“s 110 120”* + *“e 90 120”* + *“s 110 120”* + *“i 80 120”* + *“d 60 120”* + *“i 80 130”* + *“l 80 120”* + *“i 40 120”* + *“a 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“s 110 120”* (ex.: ação, maço, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“s 110 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ d ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“d 60 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“d 60 120”* (ex.: dama, dupla, quadra, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“d 60 120”* + *“i 20 120”* (ex.: adjunta, coadjuvar, etc.).

- “ *e* ” $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“em 120 120”* (ex.: trem, vento, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“e 90 120”* (ex.: avestruz, delicada, etc.).
  - Existe ainda um sistema de detecção para identificar se o caractere “ e ” deve ser pronunciado com a boca fechada sonorizando mesmo como um “ e ” (não acentuado) ou com a boca aberta sonorizando como se fosse um “ é ” (acentuado). Esse sistema se aplica também na identificação da pronúncia do caractere “ o ” e está apresentado nessa mesma subseção, logo após essa itemização.

- “ *ê* ” $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“em 120 130”* (ex.: êmbolo, têmpora, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“e 100 130”* (ex.: freguês, você, etc.).

- “ *é* ” $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“em 120 130”* (ex.: além, parabéns, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“ee 100 130”* (ex.: balé, régua, etc.).

- “ *f* ” $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“ee 100 130”* + *“f 100 120”* + *“e 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“f 100 120”* (ex.: farol, fluído, fração, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“f 60 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ g ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“j 60 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por “e” ou “i”:
    TTS $\Longrightarrow$ *“j 60 120”* (ex.: gelo, vigia, etc.).
  - Quando seguido por “ue” ou “ui”:
    TTS $\Longrightarrow$ *“g 50 120”* + *“u 0 0”* (ex.: guerra, guincho, etc.);
  - Quando seguido por “ua” ou “uo”:
    TTS $\Longrightarrow$ *“g 50 120”* + *“u 50 120”* (ex.: guarda, ambíguo, etc.).
  - Quando seguido por “a”, “o”, “u”, “l” ou “r”:
    TTS $\Longrightarrow$ *“g 50 120”* (ex.: gato, glicerina, grave, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“g 50 120”* + *“i 20 120”* (ex.: designado, diagnóstico, etc.).

- *“ h ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“a 90 130”* + *“g 50 120”* + *“a 90 120”* (ex.: para leitura soletrada).
  - Ou então:
    TTS $\Longrightarrow$ *“h 0 0”* (ex.: hotel, hortênsia, etc.).

- *“ i ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“im 120 120”* (ex.: amendoim, limpo, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“i 80 120”* (ex.: figo, mito, etc.).

- *“ í ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“im 120 130”* (ex.: ímpar, íntegro, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“i 100 130”* (ex.: legível, vírgula, etc.).

- *“ j ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“j 60 120”* + *“oo 80 130”* + *“t 75 120”* + *“a 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“j 60 120”* (ex.: jardim, viajem, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“j 60 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).
- *“ k ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“k 100 120”* + *“a 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“k 100 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“k 100 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).
- *“ l ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“ee 100 130”* + *“l 80 120”* + *“e 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“l 80 120”* (ex.: lápis, lente, etc.).
  - Quando seguido por “h”:
    TTS $\Longrightarrow$ *“l 80 120”* + *“i 40 120”* + *“h 0 0”* (ex.: alho, joelho, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“w 100 120”* (ex.: mel, viral, etc.).
- *“ m ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“e 90 140”* + *“m 140 120”* + *“e 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“m 100 120”* (ex.: mar, música, etc.).

- *“ n ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“e 90 130” + “n 80 120” + “e 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“n 80 120”* (ex.: nariz, maneira, etc.).
  - Quando seguido por “h”:
    TTS $\Longrightarrow$ *“nh 80 120”* (ex.: manhã, pinho, etc.).

- *“ o ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“om 120 120”* (ex.: compra, ontem, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“o 90 120”* (ex.: jornal, lixo, etc.).
  - Existe ainda um sistema de detecção para identificar se o caractere “ o ” deve ser pronunciado com a boca fechada sonorizando mesmo como um “ o ” (não acentuado) ou com a boca aberta sonorizando como se fosse um “ ó ” (acentuado). Esse sistema se aplica também na identificação da pronúncia do caractere “ e ” e está apresentado nessa mesma subseção, logo após essa itemização.

- *“ ó ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“om 120 130”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“oo 80 130”* (ex.: dedicatória, hóspede, etc.).

- *“ ô ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“om 120 130”* (ex.: colômbia, cômputo, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“o 90 130”* (ex.: ônibus, termômetro, etc.).

- “ *õ* ” $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“om 120 130”* (ocorrência rara, mas deve estar prevista);
  - Ou então:
    TTS $\Longrightarrow$ *“o 90 130”* (ex.: bilhões, nações, etc.).

- “ *p* ” $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“p 100 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“p 100 120”* (ex.: piano, simples, prato, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“p 100 120”* + *“i 20 120”* (ex.: helicóptero, recepção, etc.).

- “ *q* ” $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“k 100 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por “ue” ou “ui”:
    TTS $\Longrightarrow$ *“k 100 120”* + *“u 0 0”* (ex.: queijo, quinta, etc.);
  - Quando seguido por “ua” ou “uo”:
    TTS $\Longrightarrow$ *“k 100 120”* + *“u 50 120”* (ex.: quando, oblíquo, etc.).
  - Quando seguido por “a”, “e”, “i” ou “o”:
    TTS $\Longrightarrow$ *“k 100 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“k 100 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ r ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“ee 100 130”* + *“rr 80 120”* + *“e 90 120”* (ex.: para leitura soletrada).
  - Quando antecedido por uma letra qualquer ...
    - e quando seguido por “r”:
      TTS $\Longrightarrow$ *“rr 80 120”* (ex.: arroz, corrimão, etc.);
    - e quando seguido por vogal:
      TTS $\Longrightarrow$ *“r 50 120”* (ex.: areia, pura, etc.);
    - e quando não seguido por vogal:
      TTS $\Longrightarrow$ *“r2 50 120”* (ex.: certo, mar, etc.).
  - Quando não antecedido por uma letra qualquer e seguido por vogal:
    TTS $\Longrightarrow$ *“rr 80 120”* (ex.: rato, rima, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“rr 80 120”* + *“i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ s ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“ee 100 130”* + *“s 110 120”* + *“e 90 120”* (ex.: para leitura soletrada).
  - Quando antecedido por vogal ...
    - e quando seguido por “h”:
      TTS $\Longrightarrow$ *“x 80 120”* + *“h 0 0”* (ocorrência rara, mas deve estar prevista);
    - e quando seguido por “s”:
      TTS $\Longrightarrow$ *“s 110 120”* + *“s 0 0”* (ex.: assado, vassoura, etc.);
    - e quando seguido por vogal:
      TTS $\Longrightarrow$ *“z 60 120”* (ex.: asa, análise, etc.);
    - e quando não seguido por vogal:
      TTS $\Longrightarrow$ *“s2 70 120”* (ex.: casca, vespa, etc.).
  - Quando não antecedido por uma letra qualquer e seguido por vogal:
    TTS $\Longrightarrow$ *“s 110 120”* (ex.: salto, soma, etc.).
  - Quando não antecedido por uma letra qualquer e seguido por “h”:
    TTS $\Longrightarrow$ *“x 80 120”* + *“h 0 0”* (ex.: short, show, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“i 20 120”* + *“s2 110 120”* (ocorrência rara, mas deve estar prevista).

- *“ t ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“t 75 120”* + *“e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“t 75 120”* (ex.: talco, atlas, trave, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“t 75 120”* + *“i 20 120”* (ex.: algoritmo, etnologia, etc.).

- *“ u ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“um 120 120”* (ex.: bumbo, fecundo, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“u 70 120”* (ex.: luto, uva, etc.).

- *“ ú ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“um 120 130”* (ex.: cúmplice, latifúndio, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“u 100 130”* (ex.: acústico, úlcera, etc.).

- *“ ü ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“um 120 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“u 50 120”* (ex.: seqüestro, tranqüilo etc.).

- *“ v ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“v 60 120” + “e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“v 60 120”* (ex.: aval, breve, livre, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“v 60 120” + “i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ x ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“x 80 130” + “i 80 120” + “s2 70 120”* (ex.: para leitura soletrada).
  - Quando antecedido por “a” e seguido por “a”:
    TTS $\Longrightarrow$ *“x 80 120”* (ex.: relaxar, taxa, etc.).
  - Quando seguido por vogal ...
    - e quando antecedido por “me”:
      TTS $\Longrightarrow$ *“x 80 120”* (ex.: mexer, mexerica, etc.);
    - e quando antecedido por “e” e esse “e” é a primeira letra da palavra:
      TTS $\Longrightarrow$ *“z 60 120”* (ex.: exame, exercício, etc.);
    - e quando antecedido por “i” e esse “i” não é antecedido por “m”:
      TTS $\Longrightarrow$ *“x 80 120”* (ex.: abaixo, lixo, etc. $\neq$ mixagem, mixer, etc.);
    - e quando seguido por “i”, antecedido por “a” ou “o” e o caractere que antecede esse “a” ou “o” é uma letra qualquer diferente de “t”:
      TTS $\Longrightarrow$ *“s 110 120”* (ex.: aproximar, máximo, etc. $\neq$ taxi, tóxico, etc.);
    - e quando antecedido por vogal:
      TTS $\Longrightarrow$ *“k 100 120” + “i 20 120” + “s 110 120”* (ex.: axila, flexão, etc.);
    - Ou então:
      TTS $\Longrightarrow$ *“x 80 120”* (ex.: xadrez, xarope, xerife, xícara, etc.).
  - Quando antecedido por vogal ...
    - e é a última letra da palavra:
      TTS $\Longrightarrow$ *“k 100 120” + “i 20 120” + “s2 70 120”* (ex.: flex, max, etc.);
    - Ou então:
      TTS $\Longrightarrow$ *“s2 70 120”* (ex.: exclusivo, extra, extraordinário, textos, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“x 80 120” + “i 20 120”* (ocorrência rara, mas deve estar prevista).

- *“ w ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“d 60 120” + “a 90 130” + “b 60 120” + “l 80 120” + “i 80 120” + “o 90 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal, “l” ou “r”:
    TTS $\Longrightarrow$ *“v 60 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“w 100 120”* (ocorrência rara, mas deve estar prevista).

- *“ y ”* $\Longrightarrow$ **TTS** . . .
  - Quando seguido por “m” ou “n” e na seqüência não seguido por vogal:
    TTS $\Longrightarrow$ *“im 120 120”* (ocorrência rara, mas deve estar prevista).
  - Ou então:
    TTS $\Longrightarrow$ *“i 90 120”* (ocorrência rara, mas deve estar prevista).

- *“ z ”* $\Longrightarrow$ **TTS** . . .
  - Quando sozinho:
    TTS $\Longrightarrow$ *“z 60 120” + “e 135 120”* (ex.: para leitura soletrada).
  - Quando seguido por vogal:
    TTS $\Longrightarrow$ *“z 60 120”* (ex.: vazar, trazer, etc.).
  - Quando não seguido por vogal e antecedido por vogal:
    TTS $\Longrightarrow$ *“s2 70 120”* (ex.: vez, xadrez, etc.).
  - Ou então:
    TTS $\Longrightarrow$ *“z 60 120” + “i 20 120”* (ocorrência rara, mas deve estar prevista).

- “ . ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“p 100 120” + “om 120 130” + “t 75 120” + “o 90 120”*

- “ , ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“v 60 120” + “i 100 130” + “r2 50 120” + “g 50 120” + “u 70 120” + “l 80 120” + “a 90 120”*

- “ : ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“d 60 120” + “o 90 130” + “i 80 120” + “s2 70 120” + “p 100 120” + “om 120 130” + “t 75 120” + “o 90 120” + “s2 70 120”*

- “ ; ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“p 100 120” + “om 120 130” + “t 75 120” + “o 90 120” + “e 90 120” + “v 60 120” + “i 100 130” + “r2 50 120” + “g 50 120” + “u 70 120” + “l 80 120” + “a 90 120”*

- “ (” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“a 90 120” + “b 60 120” + “r 50 120” + “e 90 120” + “p 100 120” + “a 90 120” + “r 50 120” + “em 120 130” + “t 75 120” + “e 90 120” + “z 60 120” + “e 90 120”*

- “) ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“f 100 120” + “ee 90 120” + “x 80 120” + “a 90 120” + “p 100 120” + “a 90 120” + “r 50 120” + “em 120 130” + “t 75 120” + “e 90 120” + “z 60 120” + “e 90 120”*

- “ [” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“a 90 120” + “b 60 120” + “r 50 120” + “e 90 120” + “k 100 120” + “o 90 120” + “w 100 120” + “x 80 120” + “e 90 130” + “t 75 120” + “e 90 120”*

- “] ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“f 100 120” + “ee 90 120” + “x 80 120” + “a 90 120” + “k 100 120” + “o 90 120” + “w 100 120” + “x 80 120” + “e 90 130” + “t 75 120” + “e 90 120”*

- " { " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"a 90 120" + "b 60 120" + "r 50 120" + "e 90 120" + "x 80 120" + "a 90 130" + "v 60 120" + "e 90 120"*

- " } " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"f 100 120" + "ee 90 120" + "x 80 120" + "a 90 130" + "x 80 120" + "a 90 130" + "v 60 120" + "e 90 120"*

- " ! " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"e 90 120" + "s2 70 120" + "k 100 120" + "l 80 120" + "@ 80 120" + "m 70 120" + "a 90 120" + "s 110 120" + "@ 80 130" + "o 90 120"*

- " ? " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"im 120 120" + "t 75 120" + "e 90 120" + "rr 80 120" + "o 90 120" + "g 50 120" + "a 90 120" + "s 110 120" + "@ 80 130" + "o 90 120"*

- " @ " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"a 90 120" + "rr 80 120" + "o 90 130" + "b 60 120" + "a 90 120"*

- " # " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"n 80 120" + "am 120 130" + "b 60 120" + "e 90 120" + "r2 50 120"*

- " % " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"p 100 120" + "o 90 120" + "r2 50 120" + "s 110 120" + "em 120 130" + "t 75 120" + "o 90 120"*

- " * " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"a 90 120" + "s2 70 120" + "t 75 120" + "e 90 120" + "r 50 120" + "i 80 130" + "s2 70 120" + "k 100 120" + "o 90 120"*

- " + " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"s 110 120" + "o 90 130" + "m 70 120" + "a 90 120"*

- " - " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"i 100 130" + "f 100 120" + "em 120 120"*

- " _ " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"s 110 120" + "u 70 120" + "b 60 120" + "l 80 120" + "im 120 120" + "nh 80 120" + "a 90 130" + "d 60 120" + "o 90 120"*

- " / " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"b 60 120" + "a 90 130" + "rr 80 120" + "a 90 120"*

- " \ " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"b 60 120" + "a 90 130" + "rr 80 120" + "a 90 120" + "im 120 120" + "v 60 120" + "e 90 120" + "r2 50 120" + "t 75 120" + "i 80 130" + "d 60 120" + "a 90 120"*

- " | " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"b 60 120" + "a 90 130" + "rr 80 120" + "a 90 120" + "v 60 120" + "e 90 120" + "r2 50 120" + "t 75 120" + "i 80 120" + "k 100 120" + "a 90 130" + "w 100 120"*

- " & " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"e 100 130" + "k 100 120" + "o 90 120" + "m 70 120" + "e 90 120" + "r2 50 120" + "s 110 120" + "i 80 120" + "a 100 130" + "w 100 120"*

- " < " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"m 70 120" + "e 90 120" + "n 80 120" + "oo 80 130" + "r2 50 120"*

- " > " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"m 70 120" + "a 90 120" + "i 80 120" + "oo 80 130" + "r2 50 120"*

- " = " $\Longrightarrow$ **TTS** $\Longrightarrow$ *"i 80 120" + "g 50 120" + "u 50 120" + "a 90 130" + "w 100 120"*

- “ " ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“a 100 130” + “s2 70 120” + “p 100 120” + “a 90 120” + “s2 70 120”*

- “ ´ ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“a 90 120” + “p 100 120” + “oo 80 130” + “s2 70 120” + “t 75 120” + “r 50 120” + “o 90 120” + “f 100 120” + “o 90 120”*

- “ ` ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“k 100 120” + “r 50 120” + “a 90 130” + “z 60 120” + “e 90 120”*

- “ ^ ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“s 110 120” + “i 80 120” + “r2 50 120” + “k 100 120” + “um 120 120” + “f 100 120” + “l 80 120” + “ee 100 130” + “k 100 120” + “i 0 120” + “s 110 120” + “o 90 120”*

- “ ~ ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“t 75 120” + “i 80 130” + “w 100 120”*

- “ $ ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“s 110 120” + “i 80 120” + “f 100 120” + “r 50 120” + “@ 80 130” + “o 90 120”*

- “ a ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“o 90 120” + “r2 50 120” + “d 60 120” + “i 80 120” + “n 80 120” + “a 90 130” + “w 100 120” + “f 100 120” + “e 90 120” + “m 100 120” + “i 80 120” + “n 80 120” + “i 100 130” + “n 80 120” + “o 90 120”*

- “ o ” $\Longrightarrow$ **TTS** $\Longrightarrow$ *“o 90 120” + “r2 50 120” + “d 60 120” + “i 80 120” + “n 80 120” + “a 90 130” + “w 100 120” + “m 100 120” + “a 90 120” + “s2 70 120” + “k 100 120” + “u 70 120” + “l 80 120” + “i 100 130” + “n 80 120” + “o 90 120”*

Vimos na subseção 3.1.3 que o TTS tem também a responsabilidade de identificar se o caractere “ e ” deve ser pronunciado com a boca fechada sonorizando mesmo como um “ e ” (não acentuado) ou com a boca aberta sonorizando como se fosse um “ é ” (acentuado). Essa característica diferenciação de pronúncia ocorre também quando o caractere em questão é a letra “ o ” e a correta identificação da sonorização dessa outra vogal é também mais uma das responsabilidades do TTS.

O sistema de identificação de pronúncia dessas duas vogais é semelhante ao de qualquer outro caractere, porém a quantidade necessária de testes de comparação é muito maior. Para se ter uma idéia, antes que seja pronunciado, o caractere “e” sofre aproximadamente oitenta testes de comparação. Por esse motivo, antes que seja iniciada a explicação desses procedimentos, faz se necessária a definição de algumas convenções (tabela 3.5) para minimizar o desenvolvimento e facilitar a visualização.

Convenção	Significado
se (+1 = m)	se o próximo caractere for igual a “m”
se (+2 ≠ m)	se o segundo próximo caractere for diferente de “m”
se (-1 = n)	se o caractere anterior for igual a “n”
& , | , !	(& ≡ e) , (| ≡ ou) , (! ≡ inversor de resultado binário)
se (+1 = m | n)	se o próximo caractere for igual a “m” ou “n”
se (-1 & -2 = consoante)	se é antecedido por duas consoantes
se ((+1 = m) & (-1 = b))	se o próximo caractere for igual a “m” e o anterior igual a “b”
se (+1 = ∄)	se é o último caractere da palavra (se o próximo não existe)

Tab. 3.5: Pronúncia - Convenções para Testes de Comparação

- *“ e ”* $\Longrightarrow$ **TTS** . . .

  ↓
  ↓→ se (+1 = m | n) ...
  ↓ ↓
  ↓ ↓→ se (+2 ≠ vogal):
  ↓ ↓ **TTS** $\Longrightarrow$ *“em 120 120”* (ex.: pente, trem, vento);
  ↓ ↓
  ↓ \ → senão:
  ↓ **TTS** $\Longrightarrow$ *“e 90 120”* (ex.: academia, gema, morena).
  ↓

↓→ senão, se ((palavra não é acentuada) & (+1 ≠ vogal)) ...
↓ ↓
↓ ↓→ se ((+2 = ∄) | ((+2 = s) & (+3 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se (+1 = l):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: anel, corcel, coronel, mel, pastel, pincel);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = z) & (-1 = d) & (-2 = ∄)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: dez ≠ acidez, fez, vez);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = r) & (((-1 = h) & (-2 = l)) | ((-1 = u) & (-2 = q)))):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: mulher, qualquer ≠ erguer, manter);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: absorver, valer, ver).
↓ ↓
↓ ↓→ senão, se ((+3 = ∄) | ((+3 = s) & (+4 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se (+1 = g) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 ≠ vogal) & (-2 = consoante)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: aconchego, chega, chego, morcego);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: cego, cega).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = j) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (((-1 = r) & (-2 ≠ vogal)) | ((-1 = v) & (-2 = n))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: brejo, inveja);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: azulejo, bandeja, cerveja, gargarejo, molejo, percevejo).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = l) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = d) & (+2 ≠ a)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: dele, modelo ≠ dela);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = g) & (-2 = ∄) & (+2 = o)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: gelo ≠ singelo);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = p) & (((-2 = ∄) | (+2 ≠ e)) | ((-2 = a) & (+2 = o)))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: apelo, pelo, pela ≠ apela, capela, pele);

↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = b | m) & (-2 = a) & (+2 = o)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: cabelo, camelo ≠ amarelo, bela, belo, chinelo, cogumelo);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = u) & (+2 = e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: aquele, daquele, naquele ≠ aquela, branquelo, guela);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: apele, duelo, marmelo, martelo).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = r) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = d) & (+2 = e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: poderes);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: espera, espere).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = s) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (+2 ≠ e):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: aceso, ileso, mesa, preso);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: catequese, maionese, tese).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = t) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (-1 = b | i | j | s):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: alfabeto, beta, projeto, quieto, sete);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = l) & (-2 ≠ vogal)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: completo, repleto);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = l) & (((-2 = o) & (-3 = c)) | ((-2 = a) & (+2 =o)) | ((-2 = e) & (-3 = s | d)))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: coleta, dialeto, seleto ≠ borboleta, esqueleto, roleta);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = m | n) & (-2 = ∄)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: meta, neta ≠ cometa, corneto);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = p) & (+2 = e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: compete, repete ≠ chupeta);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = r) & (-2 = r)) ...

↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓→ se (-3 ≠ a):
↓ ↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: correto, derrete);
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ \ → senão:
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: canaleta, carreto).
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = r) & (-2 ≠ a & b & o & p & u)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: concreto, direto, ereto, reta, vinagrete ≠ careta, coreto, lambreta, mureta);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = t) & (((+2 = o) & ((-2 = ∄) | (-2 = vowel))) | ((+2 = a) & (-2 = vowel)))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: arquiteto, teto, pateta ≠ teta, quinteto, quarteto, sexteto);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = u) & (-2 = g)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: baguete, espaguete);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = u) & (-2 = q) & (-3 = vogal)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: maquete, raquete ≠ banqueta, banquete);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: caneta, corneta, dueto, gazeta, maleta, muleta, palheta, sorvete).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = v) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = t) | (-1 = r)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: teve, conteve, esteve, frevo, trevo);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: leve, neve).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = z) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-2 = ∃) | (-1 = v)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: beleza, leveza, natureza, proeza, pureza, treze, vezes);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: fezes, reza).
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 ≠ d & ç) | (-1 = vogal)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: acelera, adega, ameba, bolero, caneca, chefe, colega, coleta,
↓ ↓ ↓ entrega, ereto, espera != aparelho, coelho, entregar, esfregar, joelho, vermelho);
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: adereço, dedo, endereço, mereço, medo, parede, rede, preço).
↓ ↓

↓ ↓→ senão, se ((+4 = ∄) | ((+4 = s) & (+5 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+3 = m | n) & (+2 = vogal) & (-1 ≠ t)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: bebem, fedem, querem ≠ conterem, terem);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = c) & (+2 = h) & (-1 ≠ h) & (+3 ≠ o)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: brecha, flecha, mecha ≠ bochecha, trecho);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = g | q) & (+2 = u)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: cheque, entregue, esfregue, jegue, moleque);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = r) & (+2 ≠ vogal) & (+3 ≠ r)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: alerta, alicerce, berro, coberta, conserva, conserve, esperta
↓ ↓ ↓ esperto, eterno, ferro, lerda, merda, terno ≠ acelerar, alicerce, esperar, seria);
↓ ↓ ↓→ senão, se ((+2 = r) & (-1 ≠ n) & !((-1 = l) & (+1 = t))):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: alegre, casebre, febre, poliedro, regra ≠ letra, negro, negra);
↓ ↓ ↓
↓ ↓ ↓→ senão, se (((+2 = l) | ((+1 = l) & (+2 ≠ vogal))) & (-2 = ∄)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: belga, velho, velha, tecla ≠ abelha, orelha, ovelha);
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = s) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = ∄) | ((-1 = n | d) & (-2 = ∄))) ...
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓→ se (+3 = e):
↓ ↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: esse este, desde, desse, nesse, neste);
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ \ → senão:
↓ ↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: essa, esta, nessa, nesta).
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((-1 = m | n | r) & ((-2 = ∄) | (-2 = a | e))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: esclaressa, interesse, mesmo, mesma, permanessa);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: acesso, inverso, leste, oeste, promessa, universo, verso);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: alicerçar, conservar).
↓ ↓
↓ ↓→ senão, se ((+5 = ∄) | ((+5 = s) & (+6 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+4 = m | n) & (+2 ≠ vogal)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: penetrem);

↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = g | q) & (+2 = u) & (+4 = m | n)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: carreguem, peguem, pequem);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = r) & (+2 = g | q) & (+3 = u)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: albergue, albuquerque);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = s) & (+2 = t) & (+3 = r)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: campestre, palestra, silvestre);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: melado, menino).
↓ ↓
↓ ↓→ senão, se ((+6 = #) | ((+6 = s) & (+7 = #))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+1 = r) & (+2 = g | q) & (+3 = u) & (+5 = m | n)):
↓ ↓ ↓ **TTS** ⟹ *"ee 100 130"* (ex.: enverguem, erguem);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"e 90 120"* (ex.: zelador).
↓ ↓
↓ \ → senão:
↓ **TTS** ⟹ *"e 90 120"* (ex.: depravado).
↓
\ → senão: **TTS** ⟹ *"e 90 120"* (ex.: ceia, deitar, teima).

- *" o "* ⟹ **TTS** . . .

↓
↓→ se (+1 = m | n) ...
↓ ↓
↓ ↓→ se (+2 ≠ vogal):
↓ ↓ **TTS** ⟹ *"om 120 120"* (ex.: conter, dom, pomba, tombo);
↓ ↓
↓ \ → senão:
↓ **TTS** ⟹ *"o 90 120"* (ex.: comer, goma, tomar).
↓
↓→ senão, se ((palavra não é acentuada) & (+1 ≠ vogal)) ...
↓ ↓
↓ ↓→ se ((+2 = #) | ((+2 = s) & (+3 = #))) ...
↓ ↓ ↓
↓ ↓ ↓→ se (+1 = l):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: anzol, girassol, sol);

↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = z):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: noz, voz);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: cor, flor, som, tom, trator).
↓ ↓
↓ ↓→ senão, se ((+3 = ∄) | ((+3 = s) & (+4 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se (+1 = c) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((+2 = a) & (-1 ≠ b)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: broca, carroça, coca, coloca, maloca, mandioca, minhoca,
↓ ↓ ↓ ↓ maloca, oca, paçoca, pipoca, toca);
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: boca, caroço, coco, doce, oco, soco, toco).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = l) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (+2 = o) ...
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓→ se (-1 = c | m | p | s):
↓ ↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: amolo, colo, solo);
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓→ senão, se ((-1 = r) & (-2 = t)):
↓ ↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: controlo);
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ \ → senão:
↓ ↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: bolo, tolo).
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((+2 = a | e) & (-3 ≠ c & n)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: bola, cola, escola, estola, gole, pistola, rebola, rebole, sacola);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: atolo, cebola, controle, rolo).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = r) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (-1 = l | s) ...
↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓→ se (+3 ≠ s):
↓ ↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: cloro, flora, folclore);
↓ ↓ ↓ ↓ ↓

↓ ↓ ↓ ↓ \ → senão:
↓ ↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: bolores, professores, valores).
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((+2 = a | e) & (-1 ≠ d | t)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: agora, amora, chore, embora, hora, namore, ora, outrora ≠
↓ ↓ ↓ ↓ amadores, calculadora, programadores);
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: autora, cantora, choro, doutora).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = s) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((+2 ≠ o) & (-1 ≠ p)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: babosa, caprichosa, carinhosa, dengosa, dose, geniosa,
↓ ↓ ↓ ↓ gostosa, leprosa, maldosa, maliciosa, manhosa, mimosa,
↓ ↓ ↓ ↓ ociosa, ondulosa, onerosa, overdose, tosa);
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: caprichoso, carinhoso, dengoso, esposo, esposa, genioso,
↓ ↓ ↓ gostoso, leproso, maldoso, malicioso, manhoso, mariposa,
↓ ↓ ↓ mimoso, ocioso, onduloso, oneroso, raposa).
↓ ↓ ↓→ senão, se (+1 = t) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = r) & ((-2 = a) | ((-2 = b) & (+2 = o)) | (-2 = c | r))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: arroto, broto, escroto, garota, garoto, maroto, marota);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: bota, boto, brota, foto, moto, nota, remoto, rota, voto ≠
↓ ↓ ↓ botar, brotar, votar).
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+2 = a | e) & (-1 ≠ d & t)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: chove, cova, nova, nove, nozes, ova, roda, soda, vozes != doce, doze, toda);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+2 = o) & (+3 = s) & (-1 ≠ r & t)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: jogos, novos, ovos, votos ≠ marotos, rodos, todos);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((-1 = c) | ((-1 = m) & (+1 = d))):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: copo, modo);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: bobo, estojo, lodo, mofo, nono, novo, outono, ovo, rodo, todo).
↓ ↓
↓ ↓→ senão, se ((+4 = ∄) | ((+4 = s) & (+5 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+3 = m | n) & (+2 = vogal) & ((-1 ≠ f) | (+1 ≠ r))):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: fogem, morem, rolem, sobem ≠ foram, forem);

↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = b) & (+2 = r)) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((-1 = s) & (+3 = e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: sobre);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: nobre, pobre, sobra).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = c) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (+2 = h):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 120"* (ex.: broche, deboche, fantoche);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: cabocla, cocar, colocar, tocar).
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = g | q) & (+2 = u)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: choque, coloque, invoque);
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = l) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se (-1 = v):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 120"* (ex.: volta, volte, volto);
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ senão, se ((+2 ≠ f) & (+3 = a | e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 120"* (ex.: escolta, folga, molha, solda);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: bolar, colar, molho, polar, golfe, golfo, olho, piolho, repolho, solto, toldo).
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = r) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((+2 ≠ vogal) & (+2 ≠ ç & z) & (+3 = a | e)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 120"* (ex.: morde, porta, socorre, torta);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: coral, corvo, dorso, força, forno, gordo, melhoria, morto, orar, socorro, torça).
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 ≠ d) & (+2 = r) & (-1 ≠ x)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: cobra, cofre, manobra, manobro, obra, pobre ≠ enxofre, podre);
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = s) ...
↓ ↓ ↓ ↓

↓ ↓ ↓ ↓→ se ((+2 = c) | ((+2 = t) & (+3 = o) & (-1 = g | p | r)) | ((+3 = o) & (+4 = 0) & (-1 = ∄)) |
↓ ↓ ↓ ↓ ((+3 = o) & (+4 = 0) & (-1 = r))):
↓ ↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: deposto, grosso, mosca, osso, posto, rosca, rosto, suposto ≠ aposta, aposte);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"oo 80 120"* (ex.: grossa, grossos, nosso, ossos, posso, vosso).
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: chover, cobrar, elogiar, logia, mover, obrar, ocasionar, odor, opor).
↓ ↓
↓ ↓→ senão, se ((+5 = ∄) | ((+5 = s) & (+6 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+4 = m | n) & (+2 ≠ vogal)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: cortem, soltem);
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((+1 = g | q) & (+2 = u) & (+4 = m | n)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: joguem);
↓ ↓ ↓
↓ ↓ ↓→ senão, se (+1 = l | r) ...
↓ ↓ ↓ ↓
↓ ↓ ↓ ↓→ se ((+2 = g | q) & (+3 = u) & (-1 ≠ p)):
↓ ↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: enforque, folgue, folque, porque, torque);
↓ ↓ ↓ ↓
↓ ↓ ↓ \ → senão:
↓ ↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: absorver, morder, olhar, orçar, soltar, voltar, torcer).
↓ ↓ ↓
↓ ↓ ↓→ senão, se ((-1 = ∃) & (+1 = s) & (+2 ≠ vogal) & (+4 ≠ r)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: bosque, mostra, mostre, mostro ≠ apostar, gostar, ostra);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: morada, polida).
↓ ↓
↓ ↓→ senão, se ((+6 = ∄) | ((+6 = s) & (+7 = ∄))) ...
↓ ↓ ↓
↓ ↓ ↓→ se ((+1 = r | l) & (+2 = g | q) & (+3 = u) & (+5 = m | n)):
↓ ↓ ↓ **TTS** ⟹ *"oo 80 130"* (ex.: enforquem, folguem);
↓ ↓ ↓
↓ ↓ \ → senão:
↓ ↓ **TTS** ⟹ *"o 90 120"* (ex.: cobiçar).
↓ ↓
↓ \ → senão:
↓ **TTS** ⟹ *"o 90 120"* (ex.: hortaliça).
↓
\ → senão: **TTS** ⟹ *"o 90 120"* (ex.: doer, ouro, ousam, roer, touro, voar).

Um outra conversão que também é de responsabilidade do TTS é a transformação de conteúdos textuais numéricos para "pronúncia numeral" (por extenso), ou seja, ao receber como entrada por exemplo o texto "1234", ao invés de gerar a pronúncia para "um, dois, três, quatro", o TTS deve gerar para "um mil, duzentos e trinta e quatro". Essas e outras particularidades do TTS podem também ser observadas na íntegra no apêndice A, que apresenta o programa completo escrito em linguagem de programação " C ".

3.2 Tradução da Tecnologia Assistiva

Como segunda e última etapa do processo necessário para que a tecnologia assistiva selecionada no capítulo 2 seja disponibilizada no idioma português do Brasil, esta seção apresenta o processo de tradução dos *softwares* que compõem essa tecnologia, o ambiente "Emacs" e o "Emacspeak". E para encerrar esse processo de adaptação, essa seção apresenta também a implementação de recursos inéditos que foram adicionados ao ambiente para facilitar a interação com o usuário e satisfazer as expectativas estipuladas durante a seleção dessa tecnologia na subseção 2.4.3.

Para garantir que o resultado final da tradução e adaptação da interface do aplicativo resultante desse projeto seguisse um padrão de linguagem constituído apenas por palavras e expressões comuns em programas computacionais, todas as alterações foram realizadas respeitando os termos do "Vocabulário Padrão" disponível no projeto de documentação para *softwares* "*The Linux Documentation Project*" que é basicamente um glossário de termos técnicos e expressões para padronização de traduções (TLDP, 2005).

Todo esse processo de adaptação foi desenvolvido utilizando-se da linguagem de programação nativa do ambiente "Emacs", a linguagem "Lisp". Para uma visão integral do código fonte dos *softwares* já com as alterações e adaptações inseridas, consulte o apêndice B que contém o programa completo escrito nessa histórica linguagem de programação (que completa no ano de 2006, 48 anos de existência).

3.2.1 Tradução do Ambiente

Vimos na subseção 2.3.2 que o "Emacspeak" é uma espécie de leitor de telas que adiciona o recurso de "fala" aos aplicativos que estão sendo executados dentro do ambiente "Emacs", ou seja, faz com que as informações que estariam sendo passadas para o usuário apenas através da tela do monitor sejam também enviadas para o sistema sintetizador de voz.

Tendo em mãos o sintetizador de voz para o idioma português do Brasil apresentado na

seção 3.1, resta agora fazer com que o "Emacspeak" envie para esse gerador de fala digital as informações também traduzidas para esse idioma. A itemização a seguir relaciona as principais alterações que foram efetuadas no sistema de comunicação do "Emacspeak" para a nacionalização desse projeto, citando sempre a mensagem original e a adaptação sugerida:

- Mensagem de saudação:
  - Original: *Press C-h C-e to get an overview of emacspeak - I am completely operational, and all my circuits are functioning perfectly*.
  - Adaptação: Olá. O ambiente Emacs está ativado. Pressione <F1>, para obter ajuda.

- Solicitação para abertura de arquivo:
  - Original: *Find file: <nome do arquivo>*.
  - Adaptação: Abrir arquivo: <nome do arquivo>.

- Aviso de erro na leitura do arquivo:
  - Original: *File exists, but cannot be read*.
  - Adaptação: Arquivo existe, mas não pode ser lido.

- Solicitação para salvamento do arquivo:
  - Original: *Save file: <nome do arquivo>*.
  - Adaptação: Salvar arquivo: <nome do arquivo>.

- Solicitação para salvamento do arquivo com nome alternativo:
  - Original: *Write file: <nome do arquivo>*.
  - Adaptação: Salvar arquivo como: <nome do arquivo>.

- Solicitação de confirmação de sobreposição no salvamento do arquivo:
  - Original: *File <nome do arquivo> exists; overwrite? (y or n)*.
  - Adaptação: Já existe um arquivo chamado <nome do arquivo>; deseja substituí-lo? (s ou n).

- Aviso de erro no salvamento do arquivo quando protegido contra gravação:
  - Original: *Note: file is write protected*.
  - Adaptação: Nota: arquivo está protegido contra gravação.

- Aviso de confirmação de salvamento do arquivo:
  - Original: *Wrote: <nome do arquivo>*.
  - Adaptação: Arquivo salvo: <nome do arquivo>.

- Aviso de não confirmação de salvamento do arquivo:
  - Original: *Save not confirmed*.
  - Adaptação: Salvar não confirmado.

- Aviso de salvamento desnecessário:
  - Original: *No changes need to be saved*.
  - Adaptação: Nenhuma alteração precisa ser salva.

- Aviso de salvamento recomendado:
  - Original: *Save file <nome do arquivo>? (y or n)*.
  - Adaptação: Salvar arquivo <nome do arquivo>? (s or n).

- Solicitação para impressão do arquivo:
  - Original: *(Print: <nome do arquivo> with: <nome da impressora>)*.
  - Adaptação: Imprimir: <nome do arquivo> no dispositivo: <nome da impressora>.

- Solicitação de confirmação de impressão do arquivo:
  - Original: *(recurso inédito, portanto não possui mensagem original)*.
  - Adaptação: Pressione Enter para confirmar a impressão, para cancelar pressione Escape.

- Aviso de confirmação de impressão do arquivo:
  - Original: *(recurso inédito, portanto não possui mensagem original)*.
  - Adaptação: Imprimindo.

- Aviso de cancelamento de uma operação qualquer:
  - Original: *Canceled*.
  - Adaptação: Cancelado.

- Solicitação para localização no conteúdo do arquivo:
  - Original: *Find: <conteúdo a ser localizado>*.
  - Adaptação: Localizar: <conteúdo a ser localizado>.

- Aviso de falha quando localizando no conteúdo do arquivo:
  - Original: *failing*.
  - Adaptação: (não sucedido).

- Solicitação para localização e substituição no conteúdo do arquivo:
  - Original: *Query replace: <conteúdo a ser localizado> with: <novo conteúdo>*.
  - Adaptação: Localizar e substituir: <conteúdo a ser localizado> por: <novo conteúdo>.

- Confirmação de cópia de conteúdo em região selecionada:
  - Original: *region copied to kill ring*.
  - Adaptação: Região copiada.

- Confirmação de recorte de conteúdo em região selecionada:
  - Original: *Killed region*.
  - Adaptação: Região recortada.

- Aviso de cancelamento da última operação realizada:
  - Original: *Undo*.
  - Adaptação: Desfeito.

- Informação de localização vertical do cursor:
  - Original: *line: <número da linha>*.
  - Adaptação: Linha: <número da linha>.

- Informação de localização horizontal do cursor:
  - Original: *Point at column: <número da coluna>*.
  - Adaptação: Coluna: <número da coluna>.

- Aviso quando posicionado o cursor nos limites da janela do aplicativo:
  - Original: *Beginning of buffer* - ou então - *End of buffer*.
  - Adaptação: Começo da janela - ou então - Fim da janela.

- Solicitação de confirmação de encerramento de um aplicativo ou de uma janela qualquer:
  - Original: *Kill buffer: <nome do aplicativo>*.
  - Adaptação: Pressione Enter, para confirmar o encerramento da janela: <nome do aplicativo>.

- Aviso de confirmação de encerramento de um aplicativo qualquer:
  - Original: *(recurso inédito, portanto não possui mensagem original)*.
  - Adaptação: Janela encerrada. Pressione F1, para obter ajuda.

- Aviso de sucesso na execução de um aplicativo qualquer:
  - Original: *Shell command succeeded with no output*.
  - Adaptação: Aplicativo executado com sucesso.

- Aviso de execução interrompida de um aplicativo qualquer:
  - Original: *Shell command succeeded with some error output*.
  - Adaptação: A execução do aplicativo foi interrompida.

- Solicitação de confirmação de encerramento do ambiente “Emacs”:
  - Original: *(recurso inédito, portanto não possui mensagem original)*.
  - Adaptação: Pressione Enter para confirmar o encerramento do ambiente “Emacs”, para cancelar pressione Escape.

- Solicitação de confirmação de encerramento do ambiente “Emacs” quando ainda existem janelas de aplicativos com alteração no conteúdo original:
  - Original: *Modified buffers exist; exit anyway? (y or n)*.
  - Adaptação: Existem janelas modificadas; Mesmo assim deseja sair? (s ou n).

- Solicitação de confirmação de encerramento do ambiente “Emacs” quando ainda existem processos ativos nos aplicativos em execução:
  - Original: *Active processes exist; kill them and exit anyway? (y or n)*.
  - Adaptação: Existem processos ativos; Mesmo assim deseja encerrá-los e sair? (s ou n).

- Solicitação para deleção de arquivo:
  - Original: *Delete: <nome do arquivo>*.
  - Adaptação: Deletar: <nome do arquivo>? (s ou n).

- Aviso de confirmação de deleção de arquivo:
  - Original: *Deletion done*.
  - Adaptação: Deleção concluída.

- Aviso de cancelamento de deleção de arquivo:
  - Original: *No deletion performed*.
  - Adaptação: Deleção não concluída.

- Solicitação para cópia de arquivo:
  - Original: *Copy: <nome do arquivo> to <novo arquivo>*.
  - Adaptação: Copiar: <nome do arquivo> para <novo arquivo>.

- Aviso de confirmação de cópia de arquivo:
  - Original: *Copied*.
  - Adaptação: Copiado.

- Solicitação para alteração de nome de arquivo:
  - Original: *Rename: <nome do arquivo> to <novo nome>*.
  - Adaptação: Renomear: <nome do arquivo> para <novo nome>.

- Aviso de confirmação de alteração de nome de arquivo:
  - Original: *Moved*.
  - Adaptação: Renomeado.

- Solicitação para criação de diretório:
  - Original: *Create directory: <nome do diretório>*.
  - Adaptação: Criar diretório: <nome do diretório>.

- Informação de tamanho de arquivo:
  - Original: *File size: <tamanho do arquivo>*.
  - Adaptação: Tamanho do arquivo: <tamanho do arquivo>.

- Aviso de atualização da lista de arquivos do diretório:
  - Original: *Reading directory: <nome do diretório>*.
  - Adaptação: Lendo diretório: <nome do diretório>.

- Solicitação do navegador para acesso a endereço eletrônico:
  - Original: *URL (default HOME)*.
  - Adaptação: Digite o endereço e pressione Enter.

- Solicitação do navegador para download de endereço eletrônico:
  - Original: *Download <endereço eletrônico> to: <nome do arquivo>*.
  - Adaptação: : Download <endereço eletrônico> para: <nome do arquivo>.

- Solicitação do navegador de confirmação para download de endereço eletrônico:
  - Original: *Input <endereço eletrônico>'s content type (default Download)*.
  - Adaptação: : Pressione Enter para confirmar o download de <endereço eletrônico>, para cancelar pressione Escape.

- Aviso do navegador de inexistência de endereço eletrônico sob o cursor:
  - Original: *No URL at point*.
  - Adaptação: Nenhum endereço sob o cursor.

- Aviso do navegador quando não encontrado o endereço eletrônico solicitado:
  - Original: *Cannot retrieve URL: <endereço eletrônico>*.
  - Adaptação: Não foi possível acessar o endereço: <endereço eletrônico>.

- Aviso do navegador quando saindo de um endereço eletrônico seguro:
  - Original: *You are leaving secure page. Continue? (y or n)*.
  - Adaptação: Você está saindo de uma página segura. Deseja continuar? (s ou n).

- Aviso do navegador quando acessando campo de formulário para entrada de texto:
  - Original: *Text input: <nome do campo>*.
  - Adaptação: Entrada de texto: <nome do campo>. Pressione Enter para editar, e Enter para concluir.

- Aviso do navegador quando acessando campo de formulário para entrada de senha:
  - Original: *Password input: <nome do campo>*.
  - Adaptação: Entrada de senha: <nome do campo>. Pressione Enter para editar, e Enter para concluir.

- Aviso do navegador quando acessando botão de formulário:
  - Original: *button: <nome do botão>*.
  - Adaptação: Botão: <nome do botão>.

- Aviso do navegador quando acessando botão tipo reset de formulário:
  - Original: *Reset button*.
  - Adaptação: Botão reset.

- Aviso do navegador quando acessando botão do tipo seleção de formulário:
  - Original: *Select: <nome do botão>*.
  - Adaptação: Botão de seleção <nome do botão>. Pressione Enter e use as setas para selecionar.

- Aviso do navegador quando acessando botão do tipo radio de formulário:
  - Original: *Unset radio: <nome do botão>*.
  - Adaptação: Botão rádio <nome do botão> não marcado. Pressione Enter para marcar.

- Aviso do navegador quando acessando área de texto de formulário:
  - Original: *Text area: <nome do campo>*.
  - Adaptação: Área de texto <nome do campo>. Pressione Enter para iniciar a edição, e Control-c, Control-c, para finalizar.

- Aviso do navegador quando enviando dados de formulário:
  - Original: *Request sent, waiting for response*.
  - Adaptação: Requisição enviada, aguardando resposta.

- Aviso do navegador quando acessando a primeira página do histórico:
  - Original: *Beginning of history; no preceding item*.
  - Adaptação: Começo do histórico.

- Aviso do navegador quando acessando a última página do histórico:
  - Original: *End of history; no default available*.
  - Adaptação: Fim do histórico.

- **Pronúncia de caracteres não alfabéticos:** a tabela 3.6 apresenta uma relação de caracteres não alfabéticos que devem essencialmente ser representados por um nome específico, e não simplesmente por um som característico.

Original	Adaptação
space	espaço
exclamation	exclamação
question[*]mark	interrogação
underscore	sublinhado
quotes	aspas
pound	sustenido
dollar	cifrão
percent	porcento
ampersand	E comercial
apostrophe	apostrofo
left[*]paren	abre parêntese
right[*]paren	fecha parêntese
star	asterisco
plus	soma
comma	vírgula
dash	hífen
dot	ponto

Original	Adaptação
new line	nova linha
slash	barra
backslash	barra invertida
pipe	barra vertical
colon	dois pontos
semi	ponto e vírgula
less[*]than	menor
greater[*]than	maior
equals	igual
at	arroba
left[*]bracket	abre colchete
right[*]bracket	fecha colchete
left[*]brace	abre chave
right[*]brace	fecha chave
caret	circunflexo
backquote	crase
tilde	til

Tab. 3.6: Pronúncia de Caracteres Não Alfabéticos

- **Pronúncia de caracteres alfabéticos especiais:** a tabela 3.7 apresenta uma relação de caracteres alfabéticos dotados de sinais modificadores de pronúncia. Estes devem essencialmente ser representados por um nome específico somente quando apresentados

individualmente, mas ao invés da tradução, tiveram que ser implementados na íntegra devido à inexistência dos mesmos no idioma inglês.

Caractere	Pronúncia
À	“ A ” com acento crase
Á	“ A ” com acento agudo
Â	“ A ” com acento circunflexo
Ã	“ A ” com acento til
Ç	“ C ” cedilha
É	“ E ” com acento agudo
Ê	“ E ” com acento circunflexo
Í	“ I ” com acento agudo
Ó	“ O ” com acento agudo
Ô	“ O ” com acento circunflexo
Õ	“ O ” com acento til
Ú	“ U ” com acento agudo
Ü	“ U ” com acento trema

Tab. 3.7: Pronúncia de Caracteres Alfabéticos Especiais

O ambiente “Emacs” e o “Emacspeak” são *softwares* de grande porte, dotados de uma gigantesca lista de aplicativos com infinitas possibilidades de utilização. Por esse motivo, existem ainda, além das traduções apresentadas nesse projeto, dezenas de milhares de linhas de código para serem nacionalizadas. Mas apesar de não integralizada, com a tradução oferecida nesse projeto, todas as funções apresentadas como objetivo dessa dissertação na seção 1.2 estão disponíveis com seu conteúdo 100% em português do Brasil.

3.2.2 Implementação de Recursos Facilitadores

O “Emacspeak” foi construído seguindo os mesmo moldes do ambiente “Emacs”, que por sua vez foi desenvolvido de forma que praticamente todas as suas funções podem ser acessadas por intermédio de teclas de atalho a qualquer momento, em frações de segundos. É claro que essa é uma característica espetacular, pois essa possibilidade mantém o conceito desse

ambiente como uma das mais práticas e mais eficientes ferramentas de trabalho de todos os tempos.

Em contrapartida, o gigantesco número de funções disponíveis chega a comprometer a atribuição de teclas de acesso rápido, fazendo com que em alguns casos o atalho para determinados recursos seja composto por duas ou até três diferentes combinações de teclas. Mas essa necessidade de elevadas cargas mnemônicas para um satisfatório domínio sobre o ambiente pode complicar demais a utilização para usuários menos experientes. Por esse motivo é que mostra-se necessária a implementação de novos "recursos facilitadores".

A itemização a seguir relaciona os principais recursos que foram implementados e adicionados ao funcionamento do "Emacspeak" para facilitar a utilização e tornar mais intuitiva a execução de algumas tarefas mais cotidianas.

- **Menu de ajuda interativa:** A qualquer momento dentro do ambiente a tecla "F1" pode ser utilizada para acionar o "menu de ajuda interativa", que nada mais é do que uma lista com as principais funções disponíveis no aplicativo em execução no momento da solicitação. Por exemplo:

  - Menu exibido quando pressionada a tecla "F1" (padrão):

```
Pressione:
<F5> para iniciar o treino de teclado.
<F6> para iniciar o editor de textos.
<F7> para iniciar o gerenciador de arquivos.
<F8> para iniciar o navegador de internet.
<F9> para iniciar o gerenciador de emails.
<F4> para encerrar qualquer janela.
<Escape> para silenciar e cancelar qualquer opera  o.
<F2> para conhecer outros comandos ou <F1> para repetir este menu.
```

  - Menu exibido quando pressionada a tecla "F2" (padrão):

```
Pressione:
<Control-h> <F1> para conhecer comandos de gerenciamento de janelas.
<Control-h> <F2> para conhecer comandos de leitura b sica de janelas.
<Control-h> <F3> para conhecer comandos de leitura avan ada de janelas.
<Control-h> <F4> para conhecer comandos de configura  o do leitor.
<Control-h> <F5> para acessar o tutorial do Ambiente Emacs.
<Alt-x> para executar aplicativos do Ambiente Emacs.
<Control-x> <Control-c> para encerrar o Ambiente Emacs.
<Control-x> <Shift-c> para desligar o computador.
```

  - Menu exibido quando pressionada a tecla "F1" no "editor de textos":

```
Voc  est  no editor de textos, editando o arquivo:  <nome do arquivo>.  Pressione:
<Control-x> <Control-f> para abrir um arquivo.
<Control-x> <Control-s> para salvar o arquivo.
<Control-x> <Control-w> para salvar o arquivo com nome diferente.
<Control-x> <Control-p> para imprimir o arquivo.
<Control-f> para localizar de forma incremental no conte do do arquivo.
<Control-s> para localizar e substituir no conte do do arquivo.
<F4> para encerrar o editor de textos.
<F2> para conhecer outros comandos ou <F1> para repetir este menu.
```

- Menu exibido quando pressionada a tecla “F2” no “editor de textos”:

```
Pressione:  <Control-Espa o>, para definir a posi  o do cursor como o in cio de uma sele  o.
  - Ap s iniciada a sele  o, o final ser  limitado pela posi  o atual do cursor.
<Control-e> <r> para ler a regi o selecionada.
<Control-x> <c> para copiar a regi o selecionada.
<Control-x> <x>, para recortar a regi o selecionada.
<Control-x> <v> para colar a regi o copiada.
<Control-x> <u> para desfazer a  ltima edi  o.
<Control-e> <Control-l> para ler o n mero da linha atual.
<Control-e> < = > para ler o n mero da coluna atual.
```

- Menu exibido quando pressionada a tecla “F1” no “gerenciador de arquivos”:

```
Voc  est  no gerenciador de arquivos, explorando o diret rio:  <nome do diret rio>.  Pressione:
seta <Abaixo> ou <Acima> para avan ar ou retroceder o cursor pelos arquivos e diret rios.
<PageDown> ou <PageUp> para avan ar ou retroceder o cursor em 10 posi  es.
  - Um bip ser  sempre emitido antes de anunciar os diret rios para diferenci -los dos arquivos.
<Enter> para acessar o diret rio ou editar o arquivo sob o cursor.
<Backspace> para sair do diret rio e explorar um n vel anterior.
<g> para informar o caminho completo e atualizar o diret rio em explora  o.
<F4> para encerrar o gerenciador de arquivos.
<F2> para conhecer outros comandos ou <F1> para repetir este menu.
```

- Menu exibido quando pressionada a tecla “F2” no “gerenciador de arquivos”:

```
Pressione:  <Shift-c> para copiar o arquivo sob o cursor.
<Shift-d> para deletar o arquivo sob o cursor.
<Shift-p> para imprimir o arquivo sob o cursor.
<Shift-r> para renomear o arquivo ou diret rio sob o cursor.
< + > para criar um diret rio.
<Shift-x> para executar um aplicativo com o arquivo sob o cursor.
  - Durante a execu  o do aplicativo, pressione <Control-g> para encerrar.
<z> para informar o tamanho do arquivo ou diret rio sob o cursor.
<Shift-z> para compactar ou descompactar o arquivo no formato "gz".
```

- Menu exibido quando pressionada a tecla “F1” no “navegador de internet”:

```
Voc  est  no navegador de internet, acessando o site:  <nome do site>.  Pressione:
<g> para acessar um novo endere o.
<w> para salvar o conte do do endere o atual.
<Tab> ou <Shift-Tab> para posicionar o cursor no link seguinte ou no anterior.
<d> para fazer download do endere o do link sob o cursor.
<Enter> para acessar o endere o do link sob o cursor.
<Shift-b> para voltar ao endere o anterior.
<F4> para encerrar o navegador de internet.
<F2> para conhecer outros comandos ou <F1> para repetir este menu.
```

- Menu exibido quando pressionada a tecla “F2” no “navegador de internet”:

```
Pressione <c> para informar e copiar na  rea de transfer ncia o endere o atual.
<u> para informar e copiar na  rea de transfer ncia o endere o do link sob o cursor.
<Shifit-r> para recarregar o conte do do endere o atual.
< ] > ou < [ > para posicionar o cursor no campo de formul rio seguinte ou no anterior.
<Control-c> <Control-c> para enviar o formul rio.
<a> para adicionar o endere o atual   lista de favoritos.
<Alt-a> para adicionar o endere o do link sob o cursor   lista de favoritos.
<v> para acessar a lista de favoritos.
  - Na lista de favoritos pressione <Control-k> para excluir o registro sob o cursor.
```

- Menu exibido quando pressionada a tecla “F1” no “gerenciador de emails”:

```
Voc  est  no gerenciador de emails.  Pressione:
<m> para enviar um novo email.
<g> para receber novos emails.
<n> para ler o pr ximo email.
<p> para ler o email anterior.
<r> para responder o email selecionado.
<f> para encaminhar o email selecionado.
<d> para deletar o email selecionado.
<F4> para encerrar o gerenciador de emails.
```

- Menu exibido quando pressionada as teclas “Control-h” e “F1” (global):

```
Comandos de gerenciamento de janelas.  Pressione:
<Control-e> <f> para ler o nome da janela.
<Control-e> <m> para ler a barra de estado da janela.
<Control-f> para localizar de forma incremental no conte do da janela.
<Control-x> <Control-p> para imprimir o conte do da janela.
<Control-x> <Control-w> para salvar o conte do da janela.
<Control-x> <Control-b> para exibir uma lista com as janelas iniciadas,
  - Use a seta <Abaixo> ou <Acima> para selecionar e pressione <Enter> para alternar.
<F4>, para encerrar qualquer janela.
```

- Menu exibido quando pressionada as teclas “Control-h” e “F2” (global):

```
Comandos de leitura b sica de janelas (com o acompanhamento do cursor).  Pressione:
seta <Abaixo> ou <Acima> para ler a linha seguinte ou a anterior.
<Control-Abaixo> ou <Control-Acima>, para ler o par grafo seguinte ou o anterior.
seta <Direita> ou <Esquerda> para ler o caractere seguinte ou o anterior.
<Control-Direita> ou <Control-Esquerda> para ler a palavra seguinte ou a anterior.
<Home> ou <End> para posicionar o cursor no in cio ou no final da linha.
<Control-Home> ou <Control-End> para posicionar o cursor no in cio ou no final da janela.
<Escape> para silenciar e cancelar qualquer opera  o.
<Control-z> para silenciar n o interferindo em qualquer opera  o.
```

- Menu exibido quando pressionada as teclas “Control-h” e “F3” (global):

```
Comandos de leitura avan ada de janelas (sem o acompanhamento do cursor).  Pressione:
<Control-e> <l> para ler a linha sob o cursor.
<Control-e> <w> para ler a palavra sob o cursor.
<Control-e> <Shift-w> para soletrar a palavra sob o cursor.
<Control-e> <c> para ler o caractere sob o cursor.
<Control-e> <b> para ler a janela toda.
<Control-e> <n> para ler a janela toda a partir da posi  o do cursor.
<Control-e> <p> para pausar a leitura.
<Control-e> <Espa o> para continuar a leitura ap s um comando de pausa.
```

- Menu exibido quando pressionada as teclas “Control-h” e “F4” (global):

```
Comandos de configura  o do leitor.  Pressione:
<Alt-Acima> ou <Alt-Abaixo> para aumentar ou reduzir o volume em 5 porcento.
<Alt-Direita> ou <Alt-Esquerda> para aumentar ou reduzir a velocidade em 5 porcento.
<F12> para alternar entre os leitores portugu s e ingl s.
<Control-e> <d> <k> para ativar ou desativar o retorno auditivo de teclas.
<Control-e> <d> <w> para ativar ou desativar o retorno auditivo de palavras.
<Control-e> <d> <l> para ativar ou desativar o retorno auditivo de linhas.
<Control-e> <d> <s> para ativar ou desativar a pron ncia soletrada de letras capitalizadas.
<Control-e> <d> ( <,> ou <.> ou <;> ) para setar a leitura de pontua  o.
```

- **Mensagens de localização freqüente:** toda vez que um novo aplicativo é iniciado o ambiente informa a nova localização ao usuário mantendo-o sempre atualizado com relação ao foco do sistema. E quando encerrada a execução do aplicativo ou uma janela qualquer o ambiente também confirma essa operação com uma mensagem padrão. Por exemplo:

  - Mensagem de localização quando iniciado o "editor de textos":
  `Editor de textos.  Pressione <F1> para obter ajuda.`

  - Mensagem de localização quando iniciado o "gerenciador de arquivos":
  `Gerenciador de arquivos.  Pressione <F1> para obter ajuda.`

  - Mensagem de localização quando iniciado o "navegador de internet":
  `Navegador de internet.  Pressione <F1> para obter ajuda.`

  - Mensagem de localização quando iniciado o "gerenciador de emails":
  `Gerenciador de emails.  Pressione <F1> para obter ajuda.`

  - Mensagem de localização quando encerrado um aplicativo um uma janela qualquer:
  `Janela encerrada.  Pressione <F1> para obter ajuda.`

- **Aplicativo de treino de teclado:** o primeiro desafio que enfrenta o DV para iniciar-se na utilização de computadores é o reconhecimento do teclado. Um teclado comum possui em média 104 teclas e esse utilitário para treino foi desenvolvido especialmente para o reconhecimento integral desse conjunto de botões. O atalho para o acionamento desse aplicativo é a tecla <F5>, pode ser ativado a qualquer momento e para encerrar o treino a tecla <Escape> deve ser pressionada.

- **Confirmação de operações:** para evitar a execução instantânea de operações solicitadas por engano, uma confirmação será sempre solicitada antes de executar algumas das operações irreversíveis. Por exemplo:

  - Mensagem para confirmar a impressão de arquivos:
  `Pressione Enter para confirmar a impress o.  Para cancelar pressione Escape.`

  - Mensagem para confirmar o encerramento do ambiente "Emacs":
  `Pressione Enter para confirmar o encerramento do ambiente Emacs.  Para cancelar pressione Escape.`

  - Mensagem para confirmar o desligamento do computador:
  `Pressione Enter para confirmar o desligamento do computador.  Para cancelar pressione Escape.`

- **Utilitários para configuração da síntese de voz:** para minimizar o estresse ocasionado por prolongados períodos de utilização do ambiente, é de extrema importância que as propriedades da síntese de voz possam ser facilmente alteradas pelo usuário. Algumas das propriedades do sintetizador de voz que podem ser instantaneamente alteradas são:

  - Volume:
  `Alt + Acima` = aumenta em 5% e `Alt + Abaixo` = reduz em 5%

  - Velocidade:
  `Alt + Direita` = aumenta em 5% e `Alt + Esquerda` = reduz em 5%

  - Idioma:
  `F12` = alterna sistema de pronúncia entre Inglês e Português.

  - Retorno auditivo de caracteres: depois de adquirir prática, o retorno auditivo de teclas individuais torna-se não mais necessário, podendo então ser desativado ou ativado pelo usuário instantaneamente (Control-e, d, k), assim como também o retorno auditivo de palavras (Control-e, d, w) ou até mesmo o retorno auditivo de linhas (Control-e, d, l).

  - Para uma relação completa das propriedades que podem ser alteradas pelo usuário, consulte o apêndice C que contém o tutorial completo para utilização da tecnologia desenvolvida nesse projeto.

Para uma visão integral do código fonte dos recursos adicionados, consulte o apêndice B que contém o programa completo escrito em linguagem de programação “Lisp”.

Capítulo 4

Computador Pessoal Compacto

Para possibilitar que a tecnologia assistiva apresentada no capítulo anterior possa ser facilmente transportada da casa para a escola ou da casa para o trabalho, este capítulo apresenta o desenvolvimento de um modelo compacto de computador pessoal. Este modelo de utilidade baseia-se na reestruturação de um computador pessoal comum, visando redução de tamanho, peso e principalmente redução de custo. O objetivo maior é transformá-lo num dispositivo portátil de fácil locomoção, dispensando assim altos investimentos na aquisição de *notebooks*.

4.1 Computador Pessoal Comum

Para entender melhor o processo de compactação desse conjunto de periféricos é preciso antes conhecer alguns detalhes referente ao funcionamento e à construção de um computador pessoal comum (figura 4.1). Esta seção relaciona e explica cada um dos componentes externos e internos á "Unidade Central de Processamento" (CPU), atribuindo a cada um desses elementos sua devida importância no sistema como um todo para determinar o emprego ou não do mesmo na versão compacta sugerida.

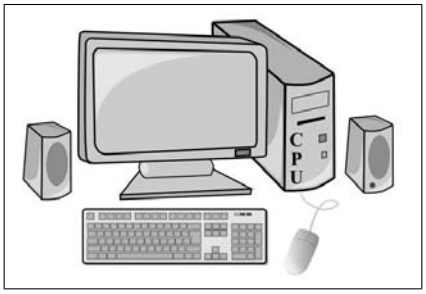


Fig. 4.1: Computador Pessoal Comum

4.1.1 Componentes Externos

Monitor de Vídeo (figura 4.2): esse é o dispositivo que possibilita ao usuário um acompanhamento visual de cada uma das operações executadas no sistema. É um componente de extrema importância para usuários com visão normal e subnormal, mas para cegos não oferece vantagem nenhuma, visto que quando nessa condição o acompanhamento das operações é feito exclusivamente por meios auditivos. Portanto, fica como opcional o emprego desse dispositivo na versão compacta do computador pessoal.

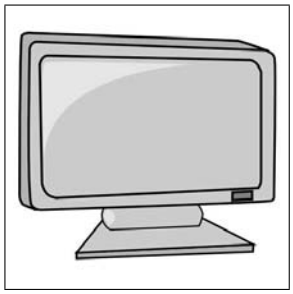


Fig. 4.2: Computador Pessoal Comum - Monitor

Teclado (figura 4.3): esse é um dos dispositivo que possibilita ao usuário controlar e manipular o sistema. O controle oferecido por esse elemento é baseado em comandos e inserção de informações. É um componente de grande importância e de fácil aprendizado para utilização por usuários DVs, portanto, deve ser empregado na versão compacta do computador pessoal.

Fig. 4.3: Computador Pessoal Comum - Teclado

***Mouse* (figura 4.4):** esse é um outro dispositivo que também possibilita ao usuário controlar e manipular o sistema, mas o controle oferecido por esse elemento exige um acompanhamento visual do posicionamento do ponteiro na tela do Monitor de Vídeo. Portanto, é um outro componente de grande importância para usuários com visão normal e subnormal, mas para cegos não oferece vantagens. Portanto, fica também como opcional o emprego desse dispositivo na versão compacta do computador pessoal.

Fig. 4.4: Computador Pessoal Comum - *Mouse*

CPU (figura 4.5): esse é um dispositivo indispensável, responsável pelo gerenciamento de todos os demais componentes do sistema. A grosso modo, esse é o computador propriamente dito e os demais elementos externos são apenas periféricos responsáveis pela entrada e saída de informações. É composto internamente por uma série de componentes que estão relacionados e explicados na subseção seguinte.

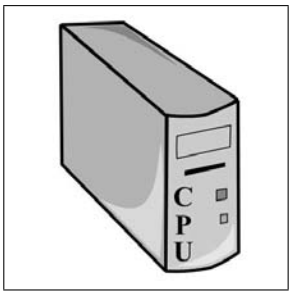


Fig. 4.5: Computador Pessoal Comum - CPU

Caixa(s) de Som (figura 4.6) ou fone de ouvido: esse é o dispositivo que possibilita ao sistema a emissão de áudio. É um componente de baixa importância para usuários com visão normal, mas essencial para usuários DVs, visto que quando nessa condição o acompanhamento das operações é feito exclusivamente por meios auditivos. Portanto, esse dispositivo deve ser empregado na versão compacta do computador pessoal.

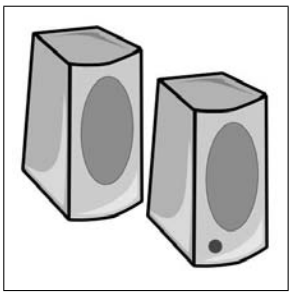


Fig. 4.6: Computador Pessoal Comum - Caixa(s) de Som

4.1.2 Componentes Internos à CPU

Placa Mãe: esse é o dispositivo central do computador, onde são conectados (por intermédio de cabos ou *sockets*) todos os demais componentes internos à CPU e também grande parte dos externos. É indispensável para o funcionamento do sistema, e portanto, deve ser empregado na versão compacta do computador pessoal.

Processador: como o próprio nome já diz, esse é o dispositivo responsável pelo processamento das informações do sistema. É também indispensável para o funcionamento do conjunto porque gerencia praticamente todos os componentes conectados à Placa Mãe, e portanto, deve também ser empregado na versão compacta do computador pessoal.

Ventilador e Dissipador de Calor: os Processadores atuais são geralmente submetidos a altíssimas freqüências de trabalho, que podem variar desde a faixa de 100 Mega Hertz até valores superiores a 1 Giga Hertz. Mesmo sendo alimentado por baixas tensões, essa alta freqüência gera um elevado aquecimento e na grande maioria dos casos não é possível manter estável a temperatura de trabalho apenas com a utilização de dissipadores metálicos de calor, mostrando-se então necessário também o emprego de ventiladores auxiliares para acelerar a redução do aquecimento. Portanto, fica evidente a necessidade de adotar também esses dispositivos na versão compacta do computador pessoal.

Hard Disk: esse é o dispositivo que armazena toda a parte virtual do sistema (sistema operacional, *softwares* e arquivos). É um objeto pesado e bastante frágil (muito sensível a impactos). Poderia até ser substituído por outros métodos de armazenamento, como por exemplo cartões de memória, mas como essas alternativas ainda representam um investimento financeiro muito elevado e dentre os objetivos desse projeto está também a redução de custos, o Hard Disk torna-se também um elemento indispensável. Portanto, esse dispositivo deve também ser empregado na versão compacta do computador pessoal.

Memória RAM: esse é o dispositivo intermediário entre o Hard Disk e o Processador, ou seja, para que as informações virtuais armazenadas no Hard Disk sejam enviadas para o Processador, devem antes ser carregadas na memória temporária de acesso aleatório (Random Access Memory). É indispensável para o funcionamento do sistema, e portanto, deve também ser empregado na versão compacta do computador pessoal.

Placa de Vídeo: esse é o dispositivo intermediário entre o Processador e o Monitor de Vídeo, ou seja é o dispositivo que manipula as informações para que sejam devidamente

exibidas na tela do elemento externo de acompanhamento visual. Poderia até ser descartado devido ao fato da não necessidade de utilização do Monitor de Vídeo por parte dos DVs, mas os atuais modelos de Placa Mãe já oferecem esse componente integrado na mesma placa (Placa de Vídeo *on-board*). Portanto, fica como opcional o emprego desse dispositivo na versão compacta do computador pessoal.

Placa de Som: esse é o dispositivo intermediário entre o Processador e a(s) Caixa(s) de Som, ou seja é o dispositivo que manipula as informações para que sejam devidamente sonorizadas nos auto-falantes do elemento externo de acompanhamento auditivo. Além da Placa de Vídeo *on-board*, os atuais modelos de Placa Mãe oferecem também esse outro elemento integrado na mesma placa. Como a utilização de computadores por DVs é guiada pelo retorno auditivo, que depende também desse dispositivo, fica evidente a necessidade do emprego desse elemento na versão compacta do computador pessoal.

Drive de Disquete: esse é um dos dispositivos responsáveis pela transferência de informações entre computadores. Não é essencial para o funcionamento do sistema, mas é de extrema utilidade quando a intenção é compartilhar informações entre usuários de diferentes computadores, como por exemplo, entre aluno e professor. Portanto, fica como opcional o emprego desse dispositivo na versão compacta do computador pessoal, com a ressalva de que é importantíssima a possibilidade de compartilhamento de informações.

Drive de CD: esse é outro dispositivos também responsável pela transferência de informações entre computadores, mas nesse caso a maior necessidade mostra-se no momento de preparação do computador (instalação do sistema operacional e dos *softwares* básicos). É um elemento pesado (1 Kg), não é essencial para o funcionamento do sistema, mas pode oferecer maior independência aos usuários DVs possibilitando que os mesmos instalem novas aplicações quando necessário. Portanto, fica também como opcional o emprego desse dispositivo na versão compacta do computador pessoal.

Fonte de Alimentação: esse é o dispositivo que fornece energia para todos os elementos internos à CPU e também grande parte dos externos. Por esse motivo, é indispensável para o funcionamento do conjunto, e portanto, deve também ser empregado na versão compacta do computador pessoal.

Gabinete: esse é o recipiente onde são fixados todos os elementos internos à CPU. É um elemento pesado (8 Kg), mas primordial para a organização dos componentes. A adaptação deste dispositivo é o ponto chave da compactação, pois a seção seguinte destina-se principalmente ao remodelamento das características físicas e de composição desse compar-

timento envoltório, de modo que agregue internamente todos os elementos selecionados para integrar a versão compacta do computador pessoal.

4.2 Desenvolvimento - Compactação

Para que o computador pessoal torne-se um acessório portátil é preciso que seja transformado em um único objeto e que esse objeto seja planejado de modo que seu peso mantenha-se dentro dos padrões suportados por um ser humano, seja ele com idade adulta ou infantil. Essa não é uma tarefa fácil, mas de acordo com a atribuição de importâncias observada na descrição dos elementos na seção 4.1, alguns dos dispositivos que compõem um computador pessoal comum podem opcionalmente ser descartados no caso de adaptação para construção de um computador pessoal compacto para DVs.

A tabela 4.1 expõe de forma resumida todos os elementos que foram citados na seção 4.1 e o nível de prioridade a que foram atribuído para inclusão do mesmo na versão compacta do computador pessoal.

	Elemento	**Prioridade**
Externos	Monitor de Vídeo	Baixa
	Teclado	Alta
	Mouse	Baixa
	CPU	Alta
	Caixa(s) de Som	Alta
Internos	Placa Mãe	Alta
	Processador	Alta
	Ventilador e Dissipador	Alta
	Hard Disk	Alta
	Memória RAM	Alta
	Placa de Vídeo	Baixa
	Placa de Som	Alta
	Drive de Disquete	Média
	Drive de CD	Baixa
	Fonte de Alimentação	Alta
	Gabinete	Alta (com adaptações)

Tab. 4.1: Computador Pessoal Comum - Elementos e Prioridades

A relação de componentes que podem opcionalmente ser descartados no caso de adaptação para construção de uma versão compacta de computador pessoal para DVs é relativamente pequena em termos de quantidade, pois apenas 4 dos 16 dispositivos podem efetivamente ser eliminados com convicção (Monitor, *Mouse*, Placa de Vídeo e Drive de CD). Mas quando comparados sobre uma balança de pesagem, essa pequena quantidade de elementos pode até ultrapassar a marca dos 50% em termos de redução no peso total do computador (alguns Monitores de Vídeo chegam a pesar até 14 Kg, enquanto que a soma do peso de todos os demais componentes internos e externos, raramente chega a atingir essa mesma faixa).

Em termos de alteração no custo, a porcentagem de redução também se mantém elevada, pois em média o valor de mercado de um Monitor de Vídeo comum representa na faixa de 25% a 35% do preço total de um computador pessoal.

Resta finalmente então, propiciar também uma satisfatória redução de tamanho. E esse objetivo pode ser conquistado através de uma reformulação geral do gabinete, de forma que este novo modelo se torne menor, mais leve e unificado aos demais componentes externos que de acordo com a análise de prioridades precisam também ser integrados à essa versão compacta de computador.

4.2.1 Reformulação Geral do Gabinete

Um gabinete comum pode ser descrito fisicamente como uma caixa retangular de metal, medindo aproximadamente 40 centímetros de altura, 20 centímetros de largura e 50 centímetros de profundidade (figura 4.7). O peso varia de acordo com o modelo, mas em média encontra-se na faixa de 8 Kg.

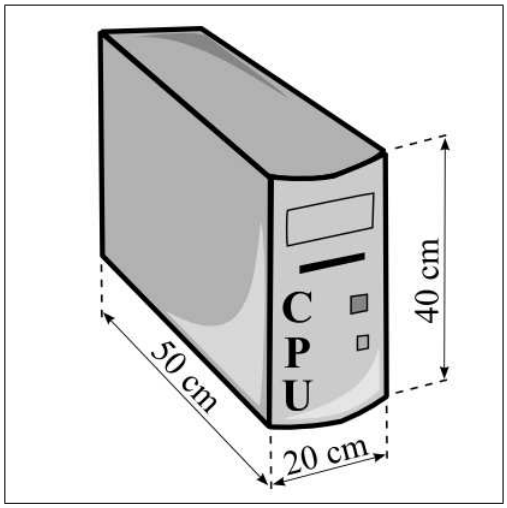


Fig. 4.7: Computador Pessoal Comum - Gabinete

Para reduzir o tamanho deste compartimento, uma versão compacta foi desenvolvida visando a eliminação de praticamente todo o espaço interno vazio, deixando apenas pequenas lacunas para a fixação dos componentes internos e para interligação dos cabos conectores.

Ao invés de ferro, chapas de alumínio foram utilizadas na construção da versão compacta para garantir a redução do peso e para reforçar a estrutura do protótipo, essas chapas foram fixadas sobre uma armação pré-moldada construída com cantoneiras também de alumínio.

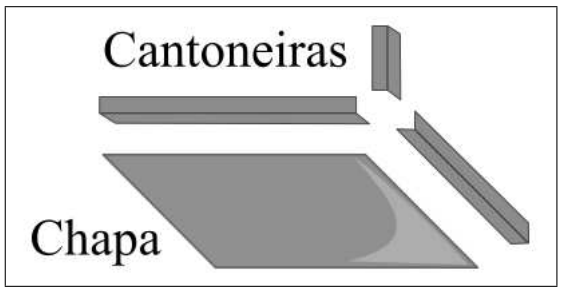


Fig. 4.8: Computador Pessoal Compacto - Estrutura

Para proteger contra possíveis impactos durante o transporte, uma fina camada de espuma sintética (E.V.A.) e um acabamento de tecido de nylon impermeável foram adicionados à superfície externa do protótipo.

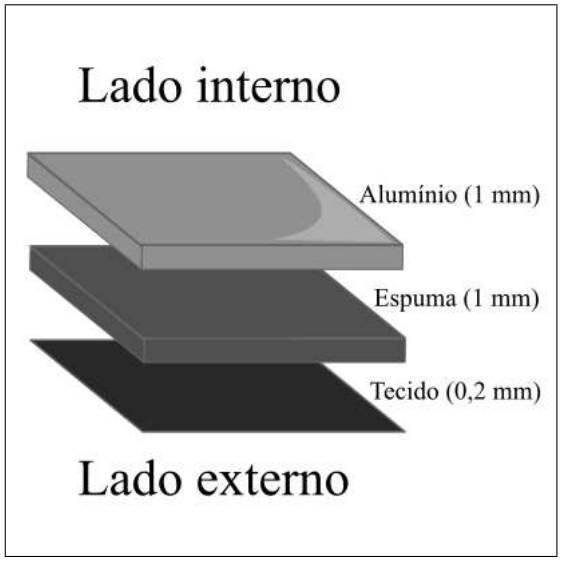


Fig. 4.9: Computador Pessoal Compacto - Revestimento

Para auxiliar na ventilação interna, além do ventilador do Processador, um segundo ventilador exaustor com 10 centímetro de diâmetro foi instalado no gabinete compacto, garantindo assim estabilidade na temperatura dos componentes, mesmo quando submetidos a longos períodos de utilização.

Para facilitar o transporte, uma alça foi adicionada á estrutura e os elementos externos à CPU selecionados para composição dessa versão compacta do computador pessoal (Teclado e Caixa(s) de Som), foram também embutidos nesse protótipo de gabinete.

Depois da construção experimental de 3 diferentes modelos de gabinetes, o protótipo final foi concluído no formato retangular (figura 4.10), medindo aproximadamente 45 centímetros de largura, 30 centímetros de profundidade e 8 centímetros de altura. O peso total após a fixação de todos os componentes ficou na faixa de 4 Kg sem Drive de CD ou 5 Kg com o Drive.

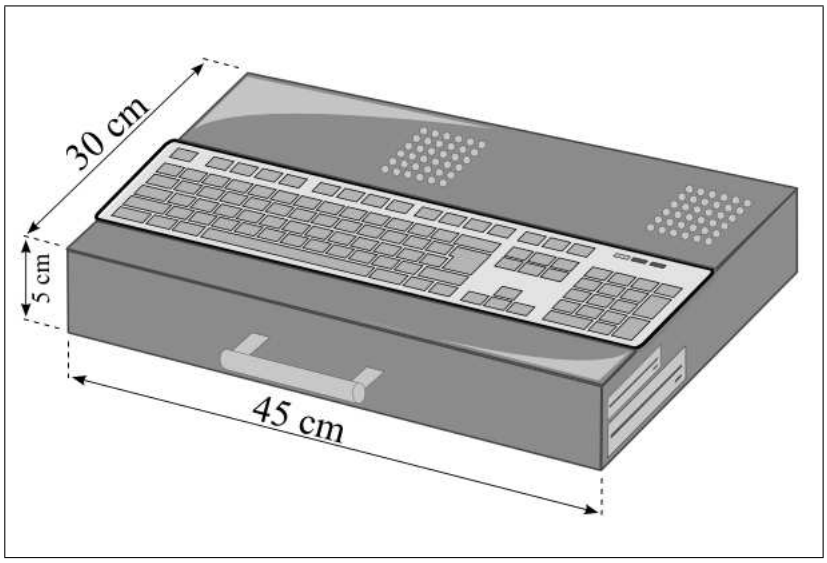


Fig. 4.10: Computador Pessoal Compacto - Protótipo

Comparando este modelo compacto com um computador pessoal comum, os únicos elementos ausentes são o monitor e o *mouse*, mas isso não impede que sejam também utilizados, pois existe na lateral esquerda do protótipo um painel com diversos tipos de conectores para ligação dos mais variados tipos de dispositivos externos:

- Conexão para monitor de vídeo;
- Conexão PS/2 (para *mouse*);
- Conexões USB (para impressoras, câmeras digitais, *scanners*, etc.);
- Conexão para Rede;
- Porta Serial;
- Porta Paralela;
- Entrada de áudio;
- Saída de áudio;
- Entrada para microfone;

- Saída para *headphone*;

Para chegar a esse protótipo, vários modelos experimentais foram desenvolvidos, inclusive uma versão plástica foi construída com poliestireno objetivando uma redução de peso ainda maior. Mas essa última foi a escolhida porque apresentou maior resistência física, melhor estabilidade de temperatura, melhor disposição interna de componentes, maior facilidade para manutenção e menor cansaço físico durante utilizações prolongadas devido à existência de uma pequena base para apoio das mãos.

As figuras apresentadas a seguir são imagens reais do protótipo final, sendo a figura 4.11 o gabinete vazio, as figuras 4.12 e 4.13 a parte interna do gabinete já com os componentes fixados, a figura 4.14 o gabinete final, a figura 4.15 o gabinete fechado para transporte, a figura 4.16 a lateral esquerda com o painel de conectores e a figura 4.17 a lateral direita com os drives de disquete e CD. Mais informações sobre experiências e resultados de utilização serão apresentadas no capítulo seguinte que refere-se às conclusões gerais desse trabalho.

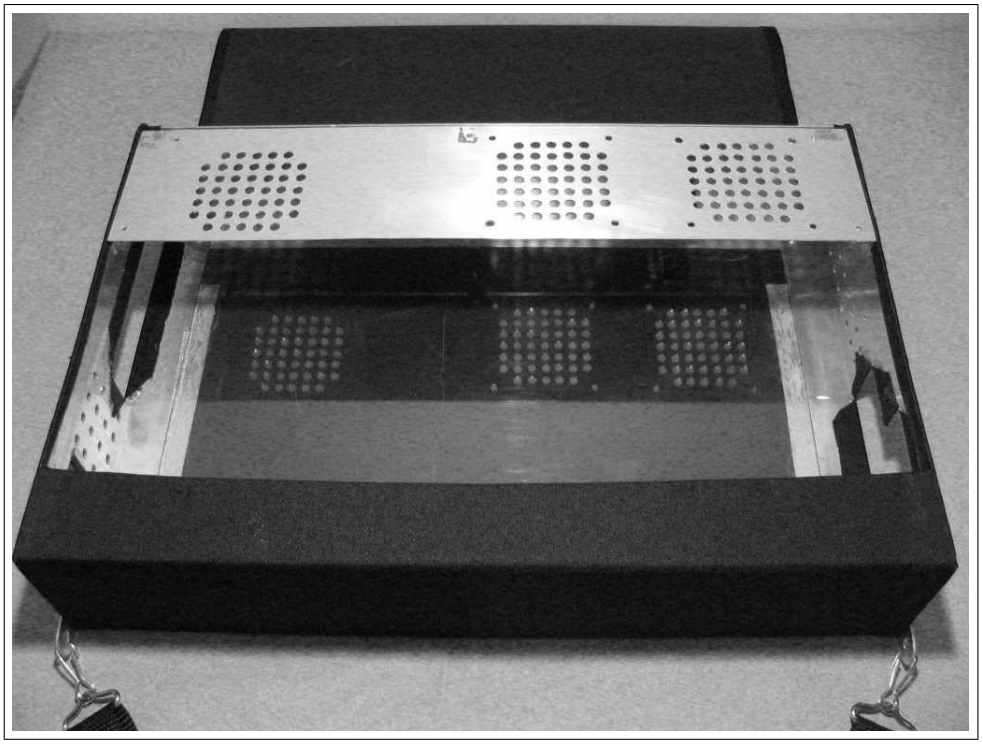


Fig. 4.11: Computador Pessoal Compacto - Protótipo Vazio

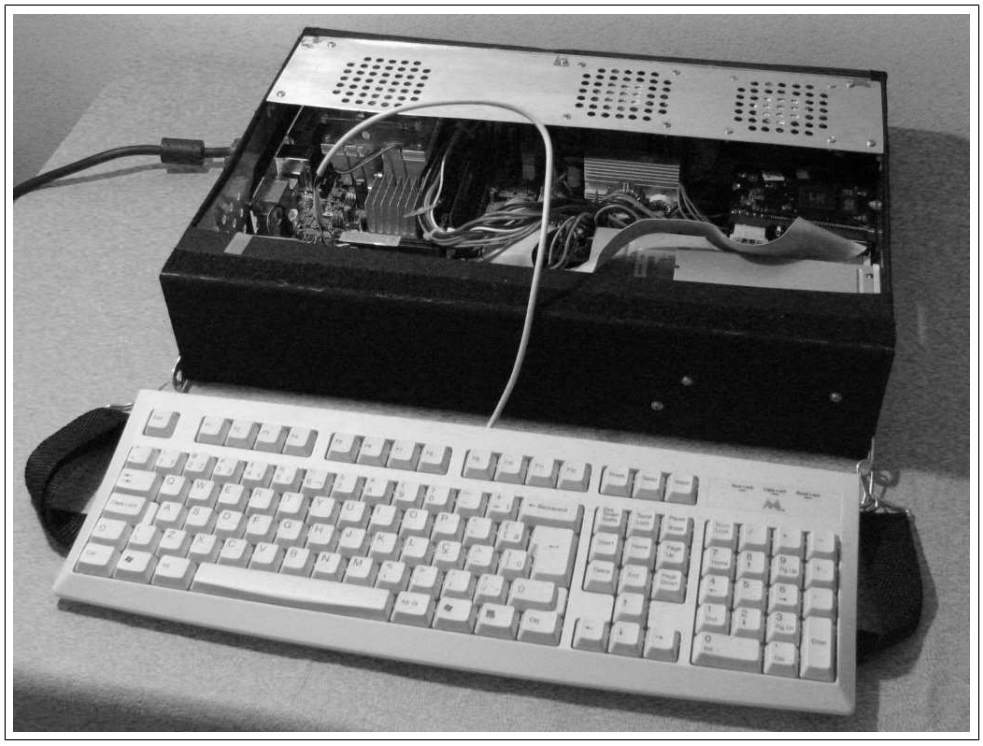


Fig. 4.12: Computador Pessoal Compacto - Protótipo Componentes (a)

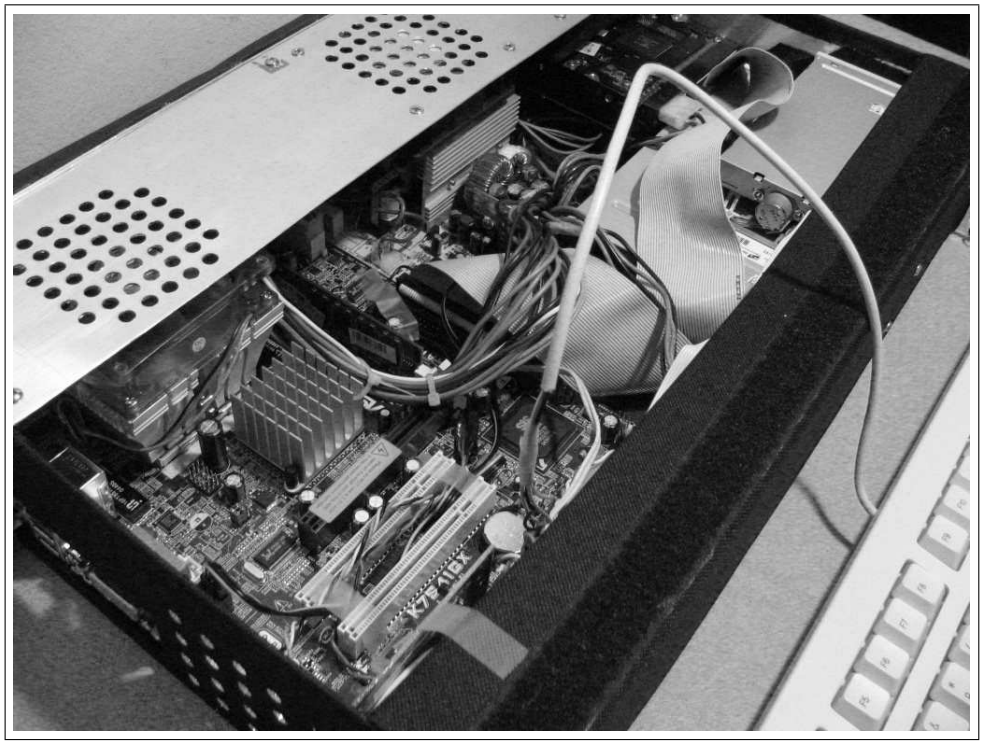


Fig. 4.13: Computador Pessoal Compacto - Protótipo Componentes (b)

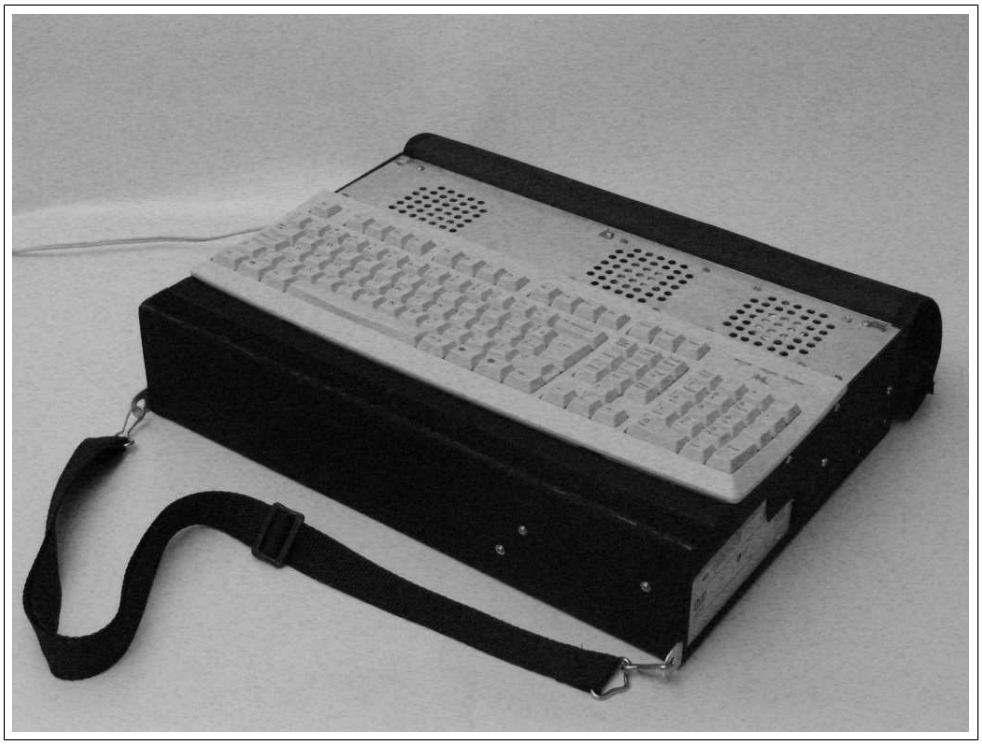


Fig. 4.14: Computador Pessoal Compacto - Protótipo Final

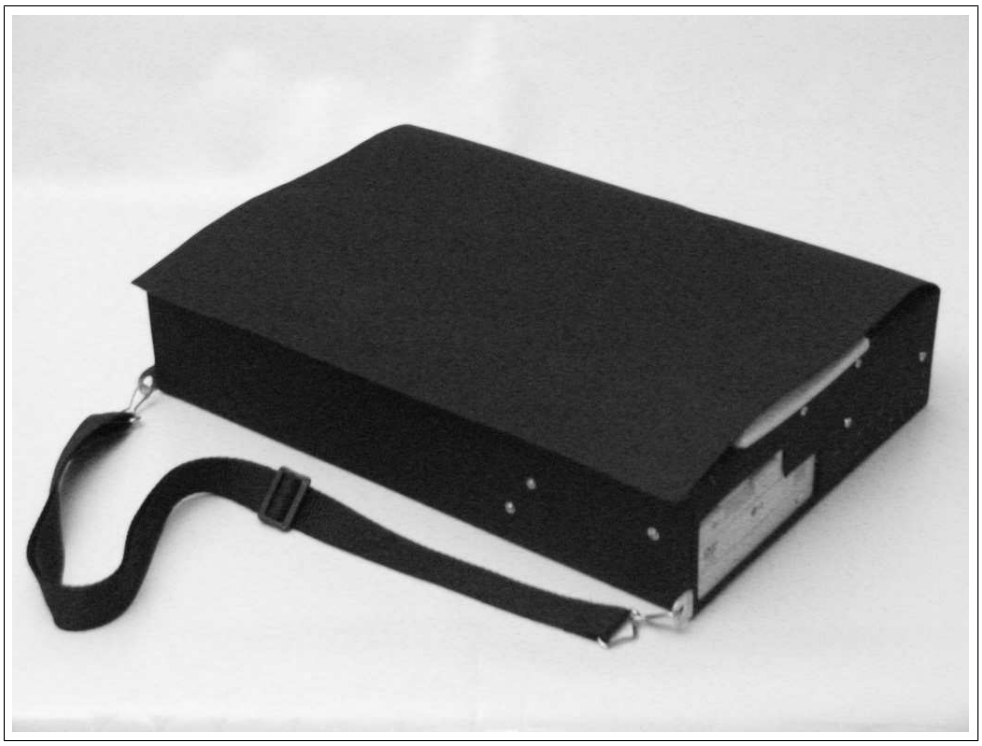


Fig. 4.15: Computador Pessoal Compacto - Protótipo Fechado

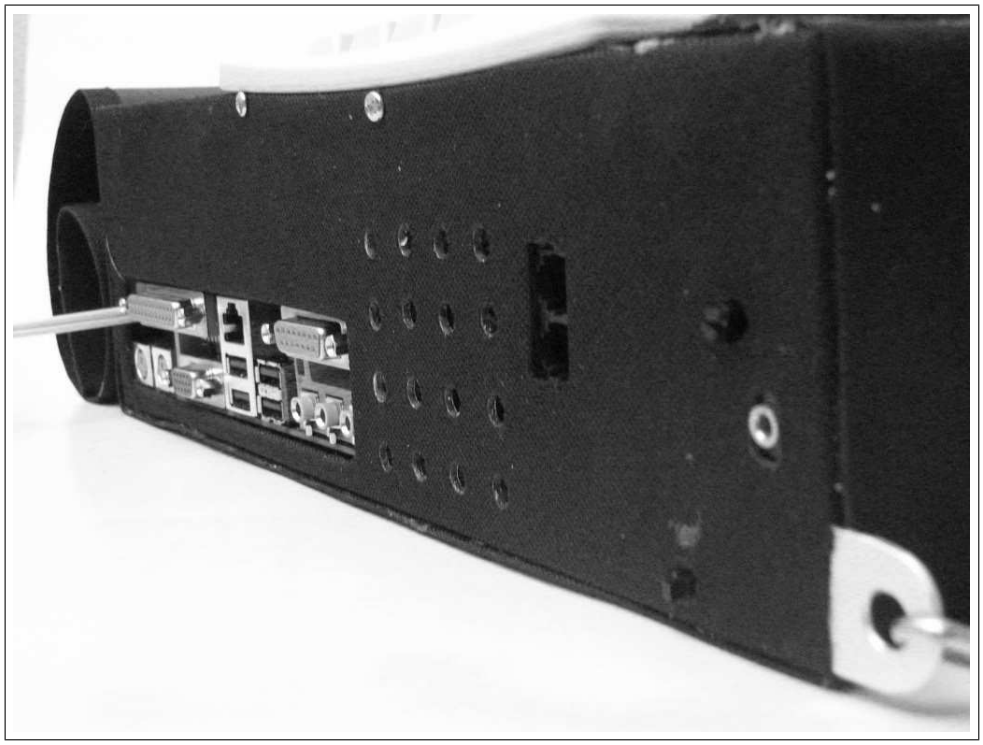


Fig. 4.16: Computador Pessoal Compacto - Protótipo Painel

Fig. 4.17: Computador Pessoal Compacto - Protótipo Drives

Capítulo 5

Resultados, Conclusões e Trabalhos Futuros

Para encerrar a documentação deste projeto, além das sugestões para trabalhos futuros e das conclusões, este último capítulo apresenta também o resultado das experiências com a utilização do computador pessoal compacto desenvolvido no capítulo 4 e o resultado dos testes realizados com o software desenvolvido no capítulo 3.

O objetivo principal dos testes realizados com o *software* e com o computador pessoal compacto foi comprovar não só o funcionamento como também a eficiência oferecida por essas alternativas. Em nenhum momento a intenção foi avaliar o desempenho das demais alternativas concorrentes, mas algumas comparações se tornaram inevitáveis para comprovar sucesso dos resultados desse projeto.

5.1 Computador Pessoal Compacto

5.1.1 Resultado dos Testes

O resultado das experiências com a utilização do computador pessoal compacto foi bastante positivo. Nenhuma comparação matemática com relação à utilização de computadores pessoais comuns foi realizada para se chegar à essa conclusão, mas o resultado foi classificado como positivo devido às bem sucedidas experiências cotidianas realizadas por usuários que adquiriram o equipamento.

Um exemplo de experiência bem sucedida é o de uma DV, aluna de primeiro grau (13 anos, 7ª série) na cidade de Santa Bárbara D'Oeste, que ao invés da pesada "máquina *Perkins*" e espessos maços de papel cartão (8 Kg), agora leva todos os dias para a sala de aula um

protótipo dessa versão compacta de computador pessoal (4,5 Kg). E graças a essa inovação, não precisa mais que suas tarefas escolares sejam manualmente transcritas do sistema *Braille* para o escrito e vice-versa, pois agora essa troca de informação entre aluno e professor é feita diretamente por intermédio de arquivos no formato texto, gravados em disquetes no decorrer das aulas, que por sua vez são perfeitamente acessíveis para ambos.

Outro exemplo de experiência bem sucedida é o de uma DV, aluna de mestrado (28 anos) no Instituto de Artes da UNICAMP, que também dispensou o transporte diário da pesada "máquina *Perkins*" a troco de um protótipo com o peso de aproximadamente 3,5 Kg (mais leve devido à ausência do drive de CD). Nesse caso, além da troca de informações entre aluno e professor sem a necessidade de transcrição manual, o uso do dispositivo possibilitou também que a aluna incrementasse suas apresentações em seminários com a exibição de conteúdos em formato digital (*slides*) acoplando o protótipo em projetores de imagem disponíveis na instituição.

Ao todo, cinco protótipos foram produzidos e destinados a usuários com diferentes características, níveis de estudo e conhecimentos, conforme a relação apresentada na tabela 5.1.

	Idade	**Escolaridade**	**Atividade**	**Deficiência**
Usuário 1	28	Mestrado	Estudante	Visual total
Usuário 2	13	1º Grau	Estudante	Visual total
Usuário 3	45	Doutorado	Professor	Visual total
Usuário 4	65	2º Grau	Aposentado	Visual total
Usuário 5	56	Superior	Médico	Visual total

Tab. 5.1: Utilização do Computador Pessoal Compacto - Usuários

Inicialmente o foco da dissertação estava somente no desenvolvimento do *software*, portanto, devido à escassez de tempo hábil e à indisponibilidade de matéria prima para uma imediata produção do conjunto de protótipos, na medida do possível os dispositivos foram construídos individualmente, separados por intervalos com períodos de 3 a 6 meses. Por esse motivo as experiências não foram realizadas simultaneamente e nem de forma seqüencial, mas felizmente todas apresentaram resultados positivos satisfazendo de forma integral as necessidades dos usuários.

5.1.2 Conclusão

O custo final dos dispositivos se manteve sempre bem abaixo da média praticada em modelos equivalentes comerciais de computadores pessoais. A relação "custo X benefício" foi tão positiva, que além dos usuários DVs, familiares desses usuários passaram também a utilizar o dispositivo graças à possibilidade de conexão da um monitor de vídeo avulso no painel de conectores na lateral esquerda do protótipo.

Fica então comprovada a eficiência da versão compacta do computador pessoal para DVs, que desempenha um papel semelhante ao de um *notebook*, porém com um custo em média oito vezes menor.

5.1.3 Sugestão para Trabalhos Futuros

Como sugestão para trabalhos futuros com relação ao computador pessoal compacto, fica então a tentativa de redução ainda maior no peso total do dispositivo, substituindo o elemento principal de armazenamento (Hard Disk) por cartões de memória, de modo que o preço final se mantenha abaixo da média praticada em modelos equivalentes comerciais de computador pessoal.

5.2 Tecnologia Assistiva

5.2.1 Resultado dos Testes

Para testar a versão traduzida da tecnologia assistiva com os inéditos recursos facilitadores, foram selecionados cinco usuários voluntários também com diferentes características, níveis de estudo e conhecimentos, conforme a relação apresentada na tabela 5.2.

	Idade	**Escolaridade**	**Atividade**	**Deficiência**
Usuário 1	28	2º Grau	Aux. de escritório	Visual total
Usuário 2	13	1º Grau	Estudante	Visual total
Usuário 3	42	Superior	Pedagoga	Visual parcial
Usuário 4	56	Superior	Médico	Visual total
Usuário 5	31	Doutorado	Professora	Nenhuma

Tab. 5.2: Testes Realizados com o Software - Usuários

Para avaliar o desempenho do *software* produto final desse projeto, os voluntários foram submetidos a uma série de testes compreendidos pela análise da execução de tarefas cotidianas

utilizando-se da tecnologia assistiva aqui oferecida e do “Jaws” que é a mais utilizada alternativa comercial. Como critérios para classificação da ferramenta foram utilizados o tempo de execução e a incidência de erros, conforme o apresentado nas tabelas 5.3, 5.4, 5.5, 5.6 e 5.7.

Antes de iniciar a bateria de testes, os participantes receberam um treinamento básico, com duração de aproximadamente 30 minutos, sobre a utilização de cada uma das ferramentas adotadas para a análise. Tanto o treinamento quanto os testes foram realizados individualmente e no caso do participante sem deficiência visual as tarefas foram todas executadas com o monitor de vídeo desligado.

Tarefa - Usuário 1	**Emacspeak**		**Jaws**	
	Tempo	**Erros**	**Tempo**	**Erros**
Iniciar o aplicativo “Treino de Teclado” e localizar as teclas “Home”, “End”, “Page Up” e “Page Down”.	2,8 min	3	3,1 min	3
Iniciar o “Editor de Textos”, digitar uma frase, salvar o arquivo com o nome “treino.txt” e encerrar o editor.	3,8 min	3	4,2 min	4
Iniciar o “Gerenciador de Arquivos”, localizar, imprimir e apagar o arquivo “teste.txt”.	3,9 min	2	4,1 min	2
Iniciar o “Navegador de Internet”, acessar o endereço “google.com.br” e pesquisar pelo termo “Emacs”.	4,3 min	1	4,6 min	2
Desligar o computador	0,8 min	0	0,9 min	0

Tab. 5.3: Testes Realizados com o Software - Usuário 1

Tarefa - Usuário 2	**Emacspeak**		**Jaws**	
	Tempo	**Erros**	**Tempo**	**Erros**
Iniciar o aplicativo “Treino de Teclado” e localizar as teclas “Home”, “End”, “Page Up” e “Page Down”.	2,9 min	2	3,2 min	2
Iniciar o “Editor de Textos”, digitar uma frase, salvar o arquivo com o nome “treino.txt” e encerrar o editor.	3,7 min	2	4,1 min	4
Iniciar o “Gerenciador de Arquivos”, localizar, imprimir e apagar o arquivo “teste.txt”.	4,2 min	3	4,5 min	3
Iniciar o “Navegador de Internet”, acessar o endereço “google.com.br” e pesquisar pelo termo “Emacs”.	4,1 min	1	4,8 min	2
Desligar o computador	1,8 min	2	2,2 min	3

Tab. 5.4: Testes Realizados com o Software - Usuário 2

Tarefa - Usuário 3	Emacspeak		Jaws	
	Tempo	Erros	Tempo	Erros
Iniciar o aplicativo “Treino de Teclado” e localizar as teclas “Home”, “End”, “Page Up” e “Page Down”.	1,8 min	1	2,5 min	1
Iniciar o “Editor de Textos”, digitar uma frase, salvar o arquivo com o nome “treino.txt” e encerrar o editor.	3,4 min	2	3,8 min	3
Iniciar o “Gerenciador de Arquivos”, localizar, imprimir e apagar o arquivo “teste.txt”.	4,1 min	1	4,2 min	1
Iniciar o “Navegador de Internet”, acessar o endereço “google.com.br” e pesquisar pelo termo “Emacs”.	2,8 min	0	3,1 min	2
Desligar o computador	0,9 min	0	1,1 min	0

Tab. 5.5: Testes Realizados com o Software - Usuário 3

Tarefa - Usuário 4	Emacspeak		Jaws	
	Tempo	Erros	Tempo	Erros
Iniciar o aplicativo “Treino de Teclado” e localizar as teclas “Home”, “End”, “Page Up” e “Page Down”.	1,4 min	0	2,5 min	1
Iniciar o “Editor de Textos”, digitar uma frase, salvar o arquivo com o nome “treino.txt” e encerrar o editor.	2,4 min	2	2,8 min	2
Iniciar o “Gerenciador de Arquivos”, localizar, imprimir e apagar o arquivo “teste.txt”.	2,1 min	1	3,2 min	1
Iniciar o “Navegador de Internet”, acessar o endereço “google.com.br” e pesquisar pelo termo “Emacs”.	3,5 min	1	4,1 min	2
Desligar o computador	1,2 min	0	1,4 min	1

Tab. 5.6: Testes Realizados com o Software - Usuário 4

Tarefa - Usuário 5	Emacspeak		Jaws	
	Tempo	Erros	Tempo	Erros
Iniciar o aplicativo “Treino de Teclado” e localizar as teclas “Home”, “End”, “Page Up” e “Page Down”.	1,2 min	0	2,3 min	1
Iniciar o “Editor de Textos”, digitar uma frase, salvar o arquivo com o nome “treino.txt” e encerrar o editor.	1,6 min	1	1,8 min	1
Iniciar o “Gerenciador de Arquivos”, localizar, imprimir e apagar o arquivo “teste.txt”.	1,4 min	0	2,1 min	0
Iniciar o “Navegador de Internet”, acessar o endereço “google.com.br” e pesquisar pelo termo “Emacs”.	2,4 min	0	4,2 min	2
Desligar o computador	0,2 min	0	0,4 min	0

Tab. 5.7: Testes Realizados com o Software - Usuário 5

5.2.2 Conclusão

De acordo com o resultado matemático apresentado nas tabelas, podemos perceber claramente a eficiência da tecnologia oferecida nesse projeto, pois apesar da não familiaridade com o *software* em questão, todos os usuários obtiveram sempre um desempenho semelhante e às vezes até melhor do que quando utilizando-se do “Jaws” que é uma ferramentas já bastante tradicional.

O trabalho resultou também na criação de um tutorial (apêndice C), que é na realidade um manual produzido com a intenção de viabilizar a utilização do *software* apresentado. Portanto, com a disponibilização pública dessa ferramenta, fica definitivamente rompida a barreira para que usuários DVs possam também migrar dos sistemas operacionais “proprietários” para os “livres”.

5.2.3 Sugestão para Trabalhos Futuros

Como sugestão para trabalhos futuros com relação ao software, fica então a finalização da tradução das demais funções do ambiente “Emacs” e do utilitário “Emacspeak”, pois como visto anteriormente na seção 3.2, as principais funções já estão traduzidas, mas restam ainda dezenas de milhares de linhas de código para serem nacionalizadas.

Outra implementação também interessante seria a adição de expressividade no sistema de

pronúncia do sintetizador de voz, ou seja,fazer com que sejam identificadas as funções de cada uma das frases como afirmativa, negativa, exclamativa ou interrogativa, e alterar a freqüência de pronúncia de cada uma das sílabas de modo que a representação digital se assemelhe da melhor maneira possível à expressão humana.

Referências Bibliográficas

AI SQUARED. ZoomText Magnifier/Reader. Disponível em: <http://www.aisquared.com/Products/zoomtextmrd/index.cfm>. Acesso em: 14 set. 2005.

BAUM ENGINEERING SRL. Gnopernicus - Software Tools for Blind and Visually Impaired in GNOME2. Disponível em: <http://www.baum.ro/gnopernicus.html>. Acesso em: 16 set. 2005.

BEZERRA, Cláudia Maria Caixeta; BARROS, Rodrigo de Passos; KAWAI, Adriana Keiko; BUTTON, Vera Lúcia da Silveira Nantes. Construção de um software transcritor do sistema Braille para o sistema óptico da língua portuguesa através de técnicas de processamento digital de imagens. In: CONGRESSO LATINO-AMERICANO DE ENGENHARIA BIOMÉDICA, 3, 2004, João Pessoa, Brasil. **Anais...** João Pessoa: Proceedings of the International Federation for Medical and Biological Engineering, 2004. p.449-452.

BORGES, José Antonio. Dosvox - Um Novo Acesso dos Cegos à Cultura e ao Trabalho. **Revista Benjamin Constant - IBCENTRO/MEC**, Rio de Janeiro, no 03, p.24-29, mai. 1996.

BR BRAILLE. Programa transcritor de textos em caracteres Braille para caracteres alfanuméricos em português. Disponível em: <http://www.fee.unicamp.br/deb/brbraille>. Acesso em: 16 set. 2005.

BRAILLE FÁCIL. Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro. Disponível em: <http://intervox.nce.ufrj.br/brfacil/>. Acesso em: 15 set. 2005.

BRASIL. Decreto nº 10007, de 29 de outubro de 2003. Institui Comitês Técnicos do Comitê Executivo do Governo Eletrônico e dá outras providências. **Diário Oficial da União**, Brasília, DF, 30 out. 2003.

DANCING DOTS. Goodfeel package with Lime and SharpEye. Disponível em: <http://www.dancingdots.com/goodfeel.htm>. Acesso em: 14 set. 2005.

DODIESIS. BME - Braille Music Editor. Disponível em: <http://www.dodiesis.com/asp/bmk.asp?language=2>. Acesso em: 14 set. 2005.

DOSVOX. Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro. Disponível em: <intervox.nce.ufrj.br/dosvox>. Acesso em: 15 set. 2005.

DUTOIT, Thierry. High-Quality Text-to-Speech Synthesis: an Overview. **Journal of Electrical and Electronics Engineering**, Australia, v.17, p.25-37, 1997.

DUTOIT, Thierry; BATAILLE, François; PAGEL, Vincent; PIERRET, Nicolas; VAN DER VREKEN, Olivier. The MBROLA Project: Towards a Set of High-Quality Speech Synthesizers Free of Use for Non-Commercial Purposes. In: INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING, 1996, Philadelphia, USA. **Anais...** Philadelphia: Proc. ICSLP, 1996.

DUXBURY SYSTEMS. TGD Pro - Tactile Graphic Designer. Disponível em: <http://www.duxburysystems.com>. Acesso em: 14 set. 2005.

EMACSPEAK. Emacs Auditory Interface - The Complete Audio Desktop. Disponível em: <http://emacspeak.sourceforge.net/>. Acesso em: 05 out. 2005.

FESTIVAL. The Festival Speech Synthesis System. The University of Edinburgh - The Center for Speech Technology Research. Disponível em: <http://www.cstr.ed.ac.uk/projects/festival/>. Acesso em: 6 dez. 2005.

FESTVOX. CMU Speech Software - Carnegie Mellon University's speech group. Disponível em: <http://festvox.org/>. Acesso em: 6 dez. 2005.

FREEDOM SCIENTIFIC. JAWS - Job Access With Speech. Disponível em: <http://www.freedomscientific.com/fs_products/JAWS_HQ.asp>. Acesso em: 14 set. 2005.

MBROLA. The MBROLA Project - Towards a Freely Available Multilingual Speech Synthesizer. TCTS Lab of the Faculté Polytechnique de Mons (Belgium). Disponível em: <http://tcts.fpms.ac.be/synthesis/mbrola.html>. Acesso em: 6 dez. 2005.

RAMAN, TV. Emacspeak - Direct Speech Access. In: ACM SIGACCESS CONFERENCE ON ASSISTIVE TECHNOLOGIES, 1996, Vancouver, Canada. **Anais...** Vancouver: Proceedings of the Second Annual ACM Conference on Assistive Technologies, 1996, p. 32-36.

STALLMAN, Richard. GPL - The GNU General Public License. Copyright© Free Software Foundation, version 1 released in January 1989. Disponível em: <http://www.gnu.org/licenses/gpl.html>. Acesso em: 21 set. 2005.

INDEX BRAILLE. WinBraille Transcription System. Disponível em: <http://www.braille.se/downloads/index.htm>. Acesso em: 14 set. 2005.

LARAMARA. Associação Brasileira de Assistência ao Deficiente Visual. Disponível em: <http://www.laramara.org.br/jaws.htm>. Acesso em: 15 set. 2005.

MICROPOWER. Virtual Vision - Tecnologia em Educação e Negócios. Disponível em: <http://www.micropower.com.br/dv/vvision5/index.asp>. Acesso em: 14 set. 2005.

PC Conectado (Projeto Cidadão Conectado) - Computador para Todos. Programa Brasileiro de Inclusão Digital do Governo Federal. Disponível em: <http://www.computadorparatodos.gov.br>. Acesso em: 19 dez. 2005.

PC-WORLD. Ouça e seja ouvido: Tecnologia do reconhecimento de voz - um breve histórico. Disponível em: <http://pcworld.uol.com.br/AdPortalV3/adCmsDocumentoShow.aspx?Documento=105049>. Acesso em: 15 set. 2005.

SOFTEX; UNICAMP. Com o apoio do MCT. **O Impacto do Software Livre e de Código Aberto na Indústria de Software do Brasil**. Campinas: Softex, 2005. 76 p.

SONZA, Andréa Polleto; SANTAROSA, Lucila Maria Costi. In: CICLO DE PALESTRAS SOBRE NOVAS TECNOLOGIAS NA EDUCAÇÃO, 2003, Porto Alegre, Brasil. **Ambientes Digitais Virtuais: Acessibilidade aos Deficientes Visuais**. Porto Alegre: CINTED-UFRGS Novas Tecnologias em Educação, 2003.

TLDP - The Linux Documentation Project. Vocabulário Padrão LDP-BR - Projeto de Documentação Linux. Disponível em: <http://br.tldp.org/>. Acesso em: 20 out. 2005.

VIOLARO, Fábio; BARBOSA, Plínio; ALBANO, Eleonora; SIMÕES, Flávio; AQUINO, Patrícia; MADUREIRA, Sandra; FRANÇOSO, Edson. Aiuruetê: A High-Quality Concatenative Text-to-Speech System for Brazilian Portuguese with Demisyllabic Analysis-Based Units and a Hierarchical Model of Rhythim Production. In: 6th European Conference on Speech Communication and Technology, 1999, Budapest, Hungria. **Anais...** Budapest: Eurospeech, 1999, Vol. 5, p. 2059-2062.

WIKIPEDIA. The Free Encyclopedia. Provided by Wikimedia Foundation. Disponível em: <http://en.wikipedia.org/wiki/Accessibility>. Acesso em: 14 set. 2005.

Apêndice A - Sintetizador de Voz

Este apêndice apresenta o código fonte completo do sintetizador de voz apresentado no capítulo 3. Para facilitar o desenvolvimento e a manutenção do *software*, o código foi dividido em três diferentes arquivos:

- **Interface Emacspeak:** que é o mecanismo intermediário entre o ambiente "Emacs" e o sintetizador de voz, responsável por recepcionar, identificar e alocar as mensagens provenientes do "Emacspeak" em variáveis específicas para o sistema de pronúncia;
- **Speak:** que é o sistema de pronúncia (TTS), responsável por analisar o texto e colocar em prática cada uma das regras implementadas para o idioma português do Brasil.
- **E Tone:** que é o gerador de tons; responsável pela emissão de *beeps* sinalizadores durante a execução de programas no ambiente "Emacs".

Interface Emacspeak:

```
/*****************************************************************************\
*  EmBrasil - Copyright (C) 2005 - Version 1.0
*  Author: Samer Eberlin <samereberlin@gmail.com>
*
*  Released under the terms of the GNU Library General Public License.
*
*  EmBrasil is a brazilian portuguese speech server for "Emacspeak"
*  <http://emacspeak.sf.net> and works with "Mbrola Speech Synthesizer"
*  <http://tcts.fpms.ac.be/synthesis/mbrola.html>
\*****************************************************************************/

#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include "speak.h"
#include "tone.h"
```

```
#define WORDSIZE 64    /* 2^6 (~1 line) */
#define ARGSSIZE 4096    /* 2^12 (~50 lines) */
#define BUFFSIZE 1048576    /* 2^20 (~15000 lines) */

/*****************************************************************************/
/* Global variables: *********************************************************/
/*****************************************************************************/
int stop = 0, pause = 0, punct = 2, split_caps = 1, buffrun = 0, charrun = 0;
float rate = 1.0;

/*****************************************************************************/
/* Main function: ************************************************************/
/*****************************************************************************/
int main (int argc, char *argv[])
{
    /**************/
    /* variables: */
    unsigned char ch, lc, temp[4], word[WORDSIZE], tone[WORDSIZE], *args,
        *buff = NULL;
    int i;
    pthread_t thread_b = 1, thread_c = 1, thread_t = 1;

    /************/
    /* malloc: */
    args = (char *) malloc (ARGSSIZE);
    buff = (unsigned char *) malloc (BUFFSIZE);

    /*********/
    /* argv: */
    if (argv[1]) if ((!strcmp (argv[1], "tts")) || (!strcmp (argv[1], "-tts")))
    {
        free (buff);
        if (!argv[2])
        {
            printf ("========== Seja Bem Vindo ao EmBrasil ==========\n");
            printf ("    Entrando em modo TTS (Text To Speech)...\n");
            printf ("- Digite o texto e pressione <ENTER> para ouv -lo.\n");
            printf ("- Para encerrar o programa pressione Control+Z.\n");
            while (fgets(args, ARGSSIZE, stdin))
            {
                spk ((void*) args);
            }
        }
        else
        {
            strcpy (args, argv[2]);
            spk ((void*) args);
            free (args);
            return 0;
        }
    }
```

```
/*********/
/* loop: */
while (1)
{
    /*********/
    /* word: */
    while ((ch = getchar ()) == ' ');
    i = 0;
    while ((ch != '\n') && (ch != ' ') && (i < (WORDSIZE - 1)))
    {
        word[i] = ch;
        i++;
        ch = getchar ();
    }
    word[i] = '\0';

    /*********/
    /* args: */
    if (ch == ' ')
    {
        while ((ch = getchar ()) == ' ');
        i = 0;
        if (ch == '{')
        {
            while (((ch = getchar ()) != '}') && (i < (ARGSSIZE - 1)))
            {
                if (ch == '[')
                {
                    if ((ch = getchar ()) == ':')
                        while ((ch = getchar ()) != ']');
                    else if (ch == '}') { args[i] = '['; i++; break; }
                    else if ((ch == ' ') && ((ch = getchar ()) == ':'))
                        while ((ch = getchar ()) != ']');
                    else { args[i] = '['; i++; args[i] = ch; i++; }
                    ch = getchar ();
                }
                if (ch == '}') break;
                args[i] = ch;
                i++;
            }
            args[i] = '\0';
            if (ch != '\n') while ((ch = getchar ()) != '\n');
        }
        else
        {
            while ((ch != '\n') && (i < (ARGSSIZE - 1)))
            {
                args[i] = ch;
                i++;
                ch = getchar ();
            }
```

```
        args[i] = '\0';
        if (ch != '\n') while ((ch = getchar ()) != '\n');
    }
}
else args[0] = '\0';

/***********/
/* strcmp: */
if (!strcmp (word, "q"))
{
    if (buffrun)
    {
        stop = 1;
        while (buffrun) usleep (10000);
    }
    strcat (buff, args);
    strcat (buff, "\n");
    continue;
}
if (!strcmp (word, "d"))
{
    while (charrun)
    {
        while (charrun) usleep (10000);
        usleep (50000);
    }
    if (!buffrun)
    {
        stop = 0; pause = 0;
        pthread_create (&thread_b, NULL, spk, (void*) buff);
        usleep (50000);
    }
    continue;
}
if (!strcmp (word, "l"))
{
    if (charrun)
        while (charrun) usleep (10000);
    if (buffrun)
        while (buffrun) usleep (10000);
    lc = args[0];
    pthread_create (&thread_c, NULL, spk_char, (void*) &lc);
    usleep (50000);
    continue;
}
if (!strcmp (word, "s"))
{
    stop = 1;
    usleep (50000);
    if (buffrun)
        while (buffrun) usleep (10000);
    continue;
}
```

```
if (!strcmp (word, "t"))
{
    strcpy (tone, args);
    if (charrun)
        while (charrun) usleep (10000);
    if (buffrun)
        while (buffrun) usleep (10000);
    pthread_create (&thread_t, NULL, spk_tone, (void*) tone);
    usleep (50000);
    continue;
}
if (!strcmp (word, "tts_say"))
{
    if (buffrun)
    {
        stop = 1;
        while (buffrun) usleep (10000);
    }
    strcpy (buff, args);
    stop = 0; pause = 0;
    pthread_create (&thread_b, NULL, spk, (void*) buff);
    usleep (50000);
    continue;
}
if (!strcmp (word, "tts_pause"))
{
    pause = 1;
    continue;
}
if (!strcmp (word, "tts_resume"))
{
    pause = 0;
    continue;
}
if (!strcmp (word, "tts_set_punctuations"))
{
    if (!strncmp (args, "none", 4)) punct = 0;
    else if (!strncmp (args, "some", 4)) punct = 1;
    else if (!strncmp (args, "all", 3)) punct = 2;
    continue;
}
if (!strcmp (word, "tts_set_speech_rate"))
{
    rate = 1/(atoi (args) / 180.0);
    continue;
}
if (!strcmp (word, "tts_split_caps"))
{
    split_caps = atoi (args);
    continue;
}
```

```
        if (!strcmp (word, "tts_sync_state"))
        {
            if (!strncmp (args, "none", 4)) punct = 0;
            else if (!strncmp (args, "some", 4)) punct = 1;
            else if (!strncmp (args, "all", 3)) punct = 2;
            i = 0;
            while (args[i] != ' ') i++;
            temp[0] = args[i+5];
            temp[1] = '\0';
                split_caps = atoi (temp);
            temp[0] = args[i+7];
            temp[1] = args[i+8];
            temp[2] = args[i+9];
            temp[3] = '\0';
                rate = 1/(atoi (temp) / 225.0);
            continue;
        }
    }

    /*********/
    /* free: */
    free (args);
    free (buff);
    return 0;
}
```

Speak:

```
/*************************************************************************\
*  EmBrasil - Copyright (C) 2005 - Version 1.0
*  Author: Samer Eberlin <samereberlin@gmail.com>
*
*  Released under the terms of the GNU Library General Public License.
*
*  EmBrasil is a brazilian portuguese speech server for "Emacspeak"
*  <http://emacspeak.sf.net> and works with "Mbrola Speech Synthesizer"
*  <http://tcts.fpms.ac.be/synthesis/mbrola.html>
\*************************************************************************/

#define PARTSIZE 64    /* 2^6 (~1 line) */
#define COMMSIZE 128    /* 2^7 (~1 line) */
#define SYNCSIZE 512    /* 2^9 (~1 line) */
#define NUMSIZE 1024    /* 2^10 (~1 line) */
#define SYNBSIZE 1024    /* 2^10 (~1 line) */
#define UPPERFRQ 1.4
```

```
/*****************************************************************************/
/* Global variables: *********************************************************/
/*****************************************************************************/
extern int stop, pause, punct, split_caps, buffrun, charrun;
extern float rate;
int accent, vowel, upper;

/*****************************************************************************/
/* isvowel function: *********************************************************/
/*****************************************************************************/
int isvowel (unsigned char c)
{
    switch (c)
    {
        case 'a': return 1;
        case 'A': return 1;
        case 192: return 1;    /*   */
        case 193: return 1;    /*   */
        case 194: return 1;    /*   */
        case 195: return 1;    /*   */
        case 224: return 1;    /*   */
        case 225: return 1;    /*   */
        case 226: return 1;    /*   */
        case 227: return 1;    /*   */
        case 'e': return 1;
        case 'E': return 1;
        case 201: return 1;    /*   */
        case 202: return 1;    /*   */
        case 233: return 1;    /*   */
        case 234: return 1;    /*   */
        case 'i': return 1;
        case 'I': return 1;
        case 205: return 1;    /*   */
        case 237: return 1;    /*   */
        case 'o': return 1;
        case 'O': return 1;
        case 211: return 1;    /*   */
        case 212: return 1;    /*   */
        case 213: return 1;    /*   */
        case 243: return 1;    /*   */
        case 244: return 1;    /*   */
        case 245: return 1;    /*   */
        case 'u': return 1;
        case 'U': return 1;
        case 218: return 1;    /*   */
        case 220: return 1;    /*   */
        case 250: return 1;    /*   */
        case 252: return 1;    /*   */
        case 'y': return 1;
        case 'Y': return 1;
        default: return 0;
    }
}
```

```
/*****************************************************************************/
/* isaccvow function: ********************************************************/
/*****************************************************************************/
int isaccvow (unsigned char c)
{
    switch (c)
    {
        case 'a': vowel = 1; return 1;
        case 'A': vowel = 1; return 1;
        case 192: accent = 1; vowel = 1; return 1;    /*   */
        case 193: accent = 1; vowel = 1; return 1;    /*   */
        case 194: accent = 1; vowel = 1; return 1;    /*   */
        case 195: accent = 1; vowel = 1; return 1;    /*   */
        case 224: accent = 1; vowel = 1; return 1;    /*   */
        case 225: accent = 1; vowel = 1; return 1;    /*   */
        case 226: accent = 1; vowel = 1; return 1;    /*   */
        case 227: accent = 1; vowel = 1; return 1;    /*   */
        case 'e': vowel = 1; return 1;
        case 'E': vowel = 1; return 1;
        case 201: accent = 1; vowel = 1; return 1;    /*   */
        case 202: accent = 1; vowel = 1; return 1;    /*   */
        case 233: accent = 1; vowel = 1; return 1;    /*   */
        case 234: accent = 1; vowel = 1; return 1;    /*   */
        case 'i': vowel = 1; return 1;
        case 'I': vowel = 1; return 1;
        case 205: accent = 1; vowel = 1; return 1;    /*   */
        case 237: accent = 1; vowel = 1; return 1;    /*   */
        case 'o': vowel = 1; return 1;
        case 'O': vowel = 1; return 1;
        case 211: accent = 1; vowel = 1; return 1;    /*   */
        case 212: accent = 1; vowel = 1; return 1;    /*   */
        case 213: accent = 1; vowel = 1; return 1;    /*   */
        case 243: accent = 1; vowel = 1; return 1;    /*   */
        case 244: accent = 1; vowel = 1; return 1;    /*   */
        case 245: accent = 1; vowel = 1; return 1;    /*   */
        case 'u': vowel = 1; return 1;
        case 'U': vowel = 1; return 1;
        case 218: accent = 1; vowel = 1; return 1;    /*   */
        case 220: accent = 1; vowel = 1; return 1;    /*   */
        case 250: accent = 1; vowel = 1; return 1;    /*   */
        case 252: accent = 1; vowel = 1; return 1;    /*   */
        case 'y': vowel = 1; return 1;
        case 'Y': vowel = 1; return 1;
        default: return 0;
    }
}
```

```
/*****************************************************************************/
/* isaccupp function: ********************************************************/
/*****************************************************************************/
int isaccupp (unsigned char c)
{
    switch (c)
    {
        case 192: return 1;    /*   */
        case 193: return 1;    /*   */
        case 194: return 1;    /*   */
        case 195: return 1;    /*   */
        case 199: return 1;    /*   */
        case 201: return 1;    /*   */
        case 202: return 1;    /*   */
        case 205: return 1;    /*   */
        case 211: return 1;    /*   */
        case 212: return 1;    /*   */
        case 213: return 1;    /*   */
        case 218: return 1;    /*   */
        case 220: return 1;    /*   */
        default: return 0;
    }
}

/*****************************************************************************/
/* isei function: ************************************************************/
/*****************************************************************************/
int isei (unsigned char c)
{
    switch (c)
    {
        case 'e': return 1;
        case 'E': return 1;
        case 201: return 1;    /*   */
        case 202: return 1;    /*   */
        case 233: return 1;    /*   */
        case 234: return 1;    /*   */
        case 'i': return 1;
        case 'I': return 1;
        case 205: return 1;    /*   */
        case 237: return 1;    /*   */
        default: return 0;
    }
}

/*****************************************************************************/
/* iscedila function: ********************************************************/
/*****************************************************************************/
int iscedila (unsigned char c)
{
    if ((c == 199) || (c == 231)) return 1;
    return 0;
}
```

```
/*****************************************************************************/
/* punctsome function: *******************************************************/
/*****************************************************************************/
int punctsome (unsigned char c)
{
    if ((c == 44) || (c == 46) || (c == 59)) return 0;    /* . , ;*/
    return 1;
}

/*****************************************************************************/
/* tts function: *************************************************************/
/*****************************************************************************/
void tts (unsigned char *part, int j)
{
    /**************/
    /* variables: */
    unsigned char synt[SYNBSIZE] = "echo -e \"", comm[COMMSIZE];
    float freq = 1.0;
    int i = 0;
    if (upper) freq = UPPERFRQ;

    /*********/
    /* char: */
    if (j == 1)
    {
        switch (part[0])
        {
            case ' ': strcat(synt, "_ 100 50 120\n"); break;
            case '!': strcat(synt, "e 90 50 120\ns2 70 50 120\nk 100 50 120\n
                l 80 50 120\nam 120 50 120\nm 70 50 120\na 90 50 120\n
                s 110 50 120\n@ 80 50 130\no 90 50 120\n_ 100 50 120\n"); break;
            case '\"': strcat(synt, "a 100 50 130\ns2 70 50 120\np 100 50 120\n
                a 90 50 120\ns2 70 50 120\n"); break;
            case '#': strcat(synt, "n 80 50 120\nam 120 50 130\nb 60 50 120\n
                e 90 50 120\nr2 50 50 120\n"); break;
            case '$': strcat(synt, "s 110 50 120\ni 80 50 120\nf 100 50 120\n
                r 50 50 120\n@ 80 50 130\no 90 50 120\n"); break;
            case '%': strcat(synt, "p 100 50 120\no 90 50 120\nr2 50 50 120\n
                s 110 50 120\nem 120 50 130\nt 75 50 120\no 90 50 120\n"); break;
            case '&': strcat(synt, "e 100 50 130\nk 100 50 120\nom 120 50 120\n
                m 70 50 120\ne 90 50 120\nr2 50 50 120\ns 110 50 120\n
                i 80 50 120\na 90 50 120\nu 50 50 120\n"); break;
            case '\'': strcat(synt, "a 90 50 120\np 100 50 120\noo 80 50 130\n
                s2 70 50 120\nt 75 50 120\nr 50 50 120\no 90 50 120\n
                f 100 50 120\no 90 50 120\n"); break;
            case '(': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\n
                e 90 50 120\np 100 50 120\na 90 50 120\nr 50 50 120\n
                em 120 50 130\nt 75 50 120\ne 90 50 120\nz 60 50 120\n
                e 90 50 120\n"); break;
            case ')': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
                a 90 50 120\np 100 50 120\na 90 50 120\nr 50 50 120\n
                em 120 50 130\nt 75 50 120\ne 90 50 120\nz 60 50 120\n
                e 90 50 120\n"); break;
```

```
case '*': strcat(synt, "a 90 50 120\ns2 70 50 120\nt 75 50 120\n
    e 90 50 120\nr 50 50 120\ni 80 50 130\ns2 70 50 120\n
    k 100 50 120\no 90 50 120\n"); break;
case '+': strcat(synt, "s 110 50 120\nom 120 50 130\nm 70 50 120\n
    a 90 50 120\n"); break;
case ',': strcat(synt, "v 60 50 120\ni 100 50 130\nr2 50 50 120\n
    g 50 50 120\nu 70 50 120\nl 80 50 120\na 90 50 120\n"); break;
case '-': strcat(synt, "i 100 50 130\nf 100 50 120\nem 120 50 120\n");
    break;
case '.': strcat(synt, "p 100 50 120\nom 120 50 130\nt 75 50 120\n
    o 90 50 120\n_ 100 50 120\n"); break;
case '/': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
    a 90 50 120\n"); break;
case '0': strcat (synt, "z 60 50 120\nee 100 50 130\nr 50 50 120\n
    o 90 50 120\n"); break;
case '1': strcat (synt, "um 120 50 120\n"); break;
case '2': strcat (synt, "d 60 50 120\no 90 50 130\ni 80 50 120\n
    s2 70 50 120\n")  ; break;
case '3': strcat (synt, "t 75 50 120\nr 50 50 120\ne 100 50 130\n
    s2 70 50 120\n"); break;
case '4': strcat (synt, "k 100 50 120\nu 50 50 120\na 90 50 130\n
    t 75 50 120\nr 50 50 120\no 90 50 120\n"); break;
case '5': strcat (synt, "s 110 50 120\nim 120 50 130\nk 100 50 120\n
    o 90 50 120\n"); break;
case '6': strcat (synt, "s 110 50 120\ne 90 50 130\ni 80 50 120\n
    s2 70 50 120\n"); break;
case '7': strcat (synt, "s 110 50 120\nee 100 50 120\nt 75 50 120\n
    e 90 50 120\n"); break;
case '8': strcat (synt, "o 90 50 130\ni 80 50 120\nt 75 50 120\n
    o 90 50 120\n"); break;
case '9': strcat (synt, "n 80 50 120\noo 80 50 130\nv 60 50 120\n
    e 90 50 120\n"); break;
case ':': strcat(synt, "d 60 50 120\no 90 50 130\ni 80 50 120\n
    s2 70 50 120\np 100 50 120\nom 120 50 130\nt 75 50 120\n
    o 90 50 120\ns2 70 50 120\n_ 100 50 120\n"); break;
case ';': strcat(synt, "p 100 50 120\nom 120 50 130\nt 75 50 120\n
    o 90 50 120\ne 90 50 120\nv 60 50 120\ni 100 50 130\n
    r2 50 50 120\ng 50 50 120\nu 70 50 120\nl 80 50 120\n
    a 90 50 120\n_ 100 50 120\n"); break;
case '<': strcat(synt, "m 70 50 120\ne 90 50 120\nn 80 50 120\n
    oo 80 50 130\nr2 50 50 120\n"); break;
case '=': strcat(synt, "i 80 50 120\ng 50 50 120\nu 50 50 120\n
    a 90 50 130\nu 50 50 120\n"); break;
case '>': strcat(synt, "m 70 50 120\na 90 50 120\ni 80 50 120\n
    oo 80 50 130\nr2 50 50 120\n"); break;
case '?': strcat(synt, "im 120 50 120\nt 75 50 120\ne 90 50 120\n
    rr 80 50 120\no 90 50 120\ng 50 50 120\na 90 50 120\n
    s 110 50 120\n@ 80 50 130\no 90 50 120\n_ 100 50 120\n"); break;
case '@': strcat(synt, "a 90 50 120\nrr 80 50 120\no 90 50 130\n
    b 60 50 120\na 90 50 120\n"); break;
case '[': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\n
    e 90 50 120\nk 100 50 120\no 90 50 120\nu 50 50 120\n
    x 80 50 120\ne 90 50 130\nt 75 50 120\ne 90 50 120\n"); break;
```

```
case '\\': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
    a 90 50 120\nim 120 50 120\nv 60 50 120\ne 90 50 120\n
    r2 50 50 120\nt 75 50 120\ni 80 50 130\nd 60 50 120\n
    a 90 50 120\n "); break;
case ']': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
    a 90 50 120\nk 100 50 120\no 90 50 120\nu 50 50 120\n
    x 80 50 120\ne 90 50 130\nt 75 50 120\ne 90 50 120\n"); break;
case '^': strcat(synt, "s 110 50 120\ni 80 50 120\nr2 50 50 120\n
    k 100 50 120\num 120 50 120\nf 100 50 120\nl 80 50 120\n
    ee 100 50 130\nk 100 50 120\ni 0 50 120\ns 110 50 120\n
    o 90 50 120\n"); break;
case '_': strcat(synt, "s 110 50 120\nu 70 50 120\nb 60 50 120\n
    l 80 50 120\nim 120 50 120\nnh 80 50 120\na 90 50 130\n
    d 60 50 120\no 90 50 120\n"); break;
case '`': strcat(synt, "k 100 50 120\nr 50 50 120\na 90 50 130\n
    z 60 50 120\ne 90 50 120\n"); break;
case 'a': strcat(synt, "a 135 50 120\n"); break;
    case 192:/*   */
    case 224:/*   */ strcat(synt, "a 90 50 120\na 90 50 125\n"); break;
    case 193:/*   */
    case 225:/*   */ strcat(synt, "a 150 50 130\n"); break;
    case 194:/*   */
    case 226:/*   */ strcat(synt, "@ 150 50 130\n"); break;
    case 195:/*   */
    case 227:/*   */ strcat(synt, "@ 150 50 130\n"); break;
case 'b': strcat(synt, "b 60 50 120\ne 135 50 120\n"); break;
case 'c': strcat(synt, "s 110 50 120\ne 135 50 120\n"); break;
    case 199:/*   */
    case 231:/*   */ strcat(synt, "s 110 50 120\ne 90 50 120\n
        s 110 50 120\ni 80 50 120\nd 60 50 120\ni 80 50 130\n
        l 80 50 120\i 40 50 120\na 90 50 120\n"); break;
case 'd': strcat(synt, "d 60 50 120\ne 135 50 120\n"); break;
case 'e': strcat(synt, "e 135 50 120\n"); break;
    case 201:/*   */
    case 233:/*   */ strcat(synt, "ee 150 50 130\n"); break;
    case 202:/*   */
    case 234:/*   */ strcat(synt, "e 150 50 130\n"); break;
case 'f': strcat(synt, "ee 100 50 130\nf 100 50 120\ne 90 50 120\n");
    break;
case 'g': strcat(synt, "j 60 50 120\ne 135 50 120\n"); break;
case 'h': strcat(synt, "a 90 50 120\ng 50 50 120\na 90 50 120\n");
    break;
case 'i': strcat(synt, "i 120 50 120\n"); break;
    case 205:/*   */
    case 237:/*   */ strcat(synt, "i 150 50 130\n"); break;
case 'j': strcat(synt, "j 60 50 120\noo 80 50 130\nt 75 50 120\n
    a 90 50 120\n"); break;
case 'k': strcat(synt, "k 100 50 120\na 135 50 120\n"); break;
case 'l': strcat(synt, "ee 100 50 130\nl 80 50 120\ne 90 50 120\n");
    break;
case 'm': strcat(synt, "e 90 50 140\nm 140 50 120\ne 90 50 120\n");
    break;
```

```
    case 'n': strcat(synt, "e 90 50 130\nn 80 50 120\ne 90 50 120\n");
        break;
    case 'o': strcat(synt, "o 135 50 120\n"); break;
        case 211:/*   */
        case 243:/*   */ strcat(synt, "oo 150 50 130\n"); break;
        case 212:/*   */
        case 244:/*   */ strcat(synt, "o 150 50 130\n"); break;
        case 213:/*   */
        case 245:/*   */ strcat(synt, "o 150 50 130\n"); break;
    case 'p': strcat(synt, "p 100 50 120\ne 135 50 120\n"); break;
    case 'q': strcat(synt, "k 100 50 120\ne 135 50 120\n"); break;
    case 'r': strcat(synt, "ee 100 50 130\nrr 80 50 120\ne 90 50 120\n");
        break;
    case 's': strcat(synt, "ee 100 50 130\ns 110 50 120\ne 90 50 120\n");
        break;
    case 't': strcat(synt, "t 75 50 120\ne 135 50 120\n"); break;
    case 'u': strcat(synt, "u 105 50 120\n"); break;
        case 218:/*   */
        case 250:/*   */ strcat(synt, "u 150 50 130\n"); break;
        case 220:/*   */
        case 252:/*   */ strcat(synt, "u 100 50 120\n"); break;
    case 'v': strcat(synt, "v 60 50 120\ne 135 50 120\n"); break;
    case 'w': strcat(synt, "d 60 50 120\na 90 50 130\nb 60 50 120\n
        l 80 50 120\ni 80 50 120\no 90 50 120\n"); break;
    case 'x': strcat(synt, "x 80 50 120\ni 80 50 120\ns2 70 50 120\n");
        break;
    case 'y': strcat(synt, "i 80 50 130\np 50 50 120\ni 0 50 120\n
        s 110 50 120\nu 70 50 120\nl 80 50 120\nom 120 50 120\n"); break;
    case 'z': strcat(synt, "z 60 50 120\ne 135 50 120\n"); break;
    case '{': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\n
        e 90 50 120\nx 80 50 120\na 90 50 130\nv 60 50 120\n
        e 90 50 120\n"); break;
    case '|': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
        a 90 50 120\nv 60 50 120\ne 90 50 120\nr2 50 50 120\n
        t 75 50 120\ni 80 50 120\nk 100 50 120\na 90 50 130\n
        u 50 50 120\n"); break;
    case '}': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
        a 90 50 130\nx 80 50 120\na 90 50 130\nv 60 50 120\n
        e 90 50 120\n"); break;
    case '~': strcat(synt, "t 75 50 120\ni 80 50 130\nu 70 50 120\n");
        break;
    case 170: strcat(synt, "o 90 50 120\nr2 50 50 120\nd 60 50 120\n
        i 80 50 120\nn 80 50 120\na 90 50 130\nw 100 50 120\n
        f 100 50 120\ne 90 50 120\nm 100 50 120\ni 80 50 120\n
        n 80 50 120\ni 100 50 130\nn 80 50 120\no 90 50 120\n"); break;
    case 186: strcat(synt, "o 90 50 120\nr2 50 50 120\nd 60 50 120\n
        i 80 50 120\nn 80 50 120\na 90 50 130\nw 100 50 120\n
        m 100 50 120\na 90 50 120\ns2 70 50 120\nk 100 50 120\n
        u 70 50 120\nl 80 50 120\ni 100 50 130\nn 80 50 120\n
        o 90 50 120\n"); break;
    default: strcat(synt, "_ 100 50 120\n");
}
```

```
    strcat(synt, "_ 20 50 120\n");
}

/*********/
/* word: */
else
{
    while (part[i])
    {
        switch (part[i])
        {
            case ' ': strcat(synt, "_ 100 50 120\n"); break;
            /*-----------*/
            /*- Vowels -*/
            case 'a':
                if ((part[i+1] == 'm') || (part[i+1] == 'n'))
                {
                    if (!isvowel (part[i+2])) strcat (synt, "am 120 50 120\n");
                    else if (isalpha (part[i-1]) || isvowel (part[i-1]) ||
                        (!isalpha (part[i+3]) && !isvowel (part[i-1])))
                        strcat (synt, "@ 80 50 120\n");
                    else strcat (synt, "a 90 50 120\n");
                }
                else if (!accent && ((((j - i) == 2) && (part[i+1] != 's')) ||
                    (((j - i) == 3) && (isvowel (part[i-2]) ||
                    isalpha (part[i-2]))) || (((j - i) == 4) &&
                    (isvowel (part[i-2]) || isalpha (part[i-2])) &&
                    (part[i+3] == 's')))) strcat (synt, "a 100 50 130\n");
                else strcat (synt, "a 90 50 120\n");
                break;
            case 192:/*   */
            case 224:/*   */
                if ((part[i+1] == 'm') || (part[i+1] == 'n'))
                {
                    if (!isvowel (part[i+2])) strcat (synt, "am 120 50 120\n");
                    else if (isalpha (part[i-1]) || isvowel (part[i-1]))
                        strcat (synt, "@ 80 50 120\n");
                    else strcat (synt, "a 90 50 120\n");
                }
                else strcat (synt, "a 90 50 120\na 90 50 125\n");
                break;
            case 193:/*   */
            case 225:/*   */
                if ((part[i+1] == 'm') || (part[i+1] == 'n'))
                {
                    if (!isvowel (part[i+2])) strcat (synt, "am 120 50 130\n");
                    else if (isalpha (part[i-1]) || isvowel (part[i-1]))
                        strcat (synt, "@ 80 50 130\n");
                    else strcat (synt, "a 90 50 130\n");
                }
                else strcat (synt, "a 100 50 130\n");
                break;
```

```
case 194:/*   */
case 226:/*   */
    if ((part[i+1] == 'm') || (part[i+1] == 'n'))
    {
        if (!isvowel (part[i+2])) strcat (synt, "am 120 50 130\n");
        else if (isalpha (part[i-1]) || isvowel (part[i-1]))
            strcat (synt, "@ 80 50 130\n");
        else strcat (synt, "a 90 50 130\n");
    }
    else strcat (synt, "@ 100 50 130\n");
    break;
case 195:/*   */
case 227:/*   */
    if ((part[i+1] == 'm') || (part[i+1] == 'n'))
    {
        if (!isvowel (part[i+2])) strcat (synt, "am 120 50 130\n");
        else if (isalpha (part[i-1]) || isvowel (part[i-1]))
            strcat (synt, "@ 80 50 130\n");
        else strcat (synt, "a 90 50 130\n");
    }
    else strcat (synt, "@ 80 50 130\n");
    break;
case 'e':
    if ((part[i+1] == 'm') || (part[i+1] == 'n'))
    {
        if (!isvowel (part[i+2])) strcat (synt, "em 120 50 120\n");
        else strcat (synt, "e 90 50 120\n");
    }
    else if (!accent && !isvowel (part[i+1]))
    {
        if (((j - i) > 1) && ((part[i+2] == '\0') ||
            ((part[i+3] == '\0') && (part[i+2] == 's'))))
        {
            if (part[i+1] == 'l') { strcat (synt,
                "ee 100 50 130\n"); accent = 1;}
            else if ((part[i+1] == 'z') && (part[i-1] == 'd') &&
                (!isalpha (part[i-2]))) { strcat (synt,
                "ee 100 50 130\n"); accent = 1;}
            else if ((part[i+1] == 'r') && (((part[i-1] == 'h') &&
                (part[i-2] == 'l')) || ((part[i-1] == 'u') &&
                (part[i-2] == 'q')))) { strcat (synt,
                "ee 100 50 130\n"); accent = 1;}
            else strcat (synt, "e 90 50 120\n");
        }
        else if (((j - i) > 2) && ((part[i+3] == '\0') ||
            ((part[i+4] == '\0') && (part[i+3] == 's'))))
        {
            if (part[i+1] == 'g')
            {
                if (!isvowel (part[i-1]) && isalpha (part[i-2]) &&
                    !isvowel (part[i-2])) strcat (synt,
                    "e 90 50 120\n");
```

```
    else { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
}
else if (part[i+1] == 'j')
{
    if (((part[i-1] == 'r') && !isvowel (part[i-2])) ||
        ((part[i-1] == 'v') && (part[i-2] == 'n')))
        { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
    else strcat (synt, "e 90 50 120\n");
}
else if (part[i+1] == 'l')
{
    if ((part[i-1] == 'd') && (part[i+2] != 'a'))
        strcat (synt, "e 90 50 120\n");
    else if ((part[i-1] == 'g') && !isalpha (part[i-2])
        && (part[i+2] == 'o')) strcat (synt,
        "e 90 50 120\n");
    else if ((part[i-1] == 'p') &&
        ((!isalpha (part[i-2]) && (part[i+2] != 'e'))
        || ((part[i-2] == 'a') && (part[i+2] == 'o'))))
        strcat (synt, "e 90 50 120\n");
    else if (((part[i-1] == 'b') || (part[i-1] == 'm'))
        && (part[i-2] == 'a') && (part[i+2] == 'o'))
        strcat (synt, "e 90 50 120\n");
    else if ((part[i-1] == 'u') && (part[i+2] == 'e'))
        strcat (synt, "e 90 50 120\n");
    else { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
}
else if (part[i+1] == 'r')
{
    if ((part[i-1] == 'd') && (part[i+2] == 'e'))
        strcat (synt, "e 90 50 120\n");
    else { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
}
else if (part[i+1] == 's')
{
    if (part[i+2] != 'e') strcat (synt,
        "e 90 50 120\n");
    else { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
}
else if (part[i+1] == 't')
{
    if ((part[i-1] == 'b') || (part[i-1] == 'i') ||
        (part[i-1] == 'j') || (part[i-1] == 's'))
        { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
    else if ((part[i-1] == 'l') &&
        !isvowel (part[i-2])) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
```

```
    else if ((part[i-1] == 'l') && (((part[i-2] == 'o')
        && (part[i-3] == 'c')) || ((part[i-2] == 'a')
        && (part[i+2] == 'o')) || ((part[i-2] == 'e')
        && ((part[i-3] == 's') ||
        (part[i-3] == 'd'))))) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if (((part[i-1] == 'm') || (part[i-1] == 'n'))
        && !isalpha (part[i-2])) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if ((part[i-1] == 'p') && (part[i+2] == 'e'))
        { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
    else if ((part[i-1] == 'r') && (part[i-2] == 'r'))
    {
        if (part[i-3] != 'a') { strcat (synt,
            "ee 100 50 130\n"); accent = 1;}
        else strcat (synt, "e 90 50 120\n");
    }
    else if ((part[i-1] == 'r') && ((part[i-2] != 'a')
        && (part[i-2] != 'b') && (part[i-2] != 'o') &&
        (part[i-2] != 'p') && (part[i-2] != 'u')))
        { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
    else if ((part[i-1] == 't') && (((part[i+2] == 'o')
        && (!isalpha (part[i-2]) ||
        isvowel (part[i-2]))) || ((part[i+2] == 'a') &&
        isvowel (part[i-2])))) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if ((part[i-1] == 'u') && (part[i-2] == 'g'))
        { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
    else if ((part[i-1] == 'u')  && (part[i-2] == 'q')
        && isvowel (part[i-3])) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else strcat (synt, "e 90 50 120\n");
}
else if (part[i+1] == 'v')
{
    if ((part[i-1] == 't')  || (part[i-1] == 'r'))
        strcat (synt, "e 90 50 120\n");
    else { strcat (synt, "ee 100 50 130\n"); accent = 1;}
}
else if (part[i+1] == 'z')
{
    if (isalpha (part[i-2]) || (part[i-1] == 'v'))
        strcat (synt, "e 90 50 120\n");
    else { strcat (synt, "ee 100 50 130\n");
        accent = 1;}
}
else if (((part[i+1] != 'd') && !iscedila (part[i+1]))
    || isvowel (part[i-1])) { strcat (synt,
    "ee 100 50 130\n"); accent = 1;}
```

```
    else strcat (synt, "e 90 50 120\n");
}
else if (((j - i) > 3) && ((part[i+4] == '\0') ||
    ((part[i+5] == '\0') && (part[i+4] == 's'))))
{
    if (((part[i+3] == 'm') || (part[i+3] == 'n')) &&
        isvowel (part[i+2]) && (part[i-1] != 't')) {
        strcat (synt, "ee 100 50 130\n"); accent = 1;}
    else if ((part[i+1] == 'c') && (part[i+2] == 'h') &&
        (part[i-1] != 'h') && (part[i+3] != 'o')) {
        strcat (synt, "ee 100 50 130\n"); accent = 1;}
    else if (((part[i+1] == 'g') || (part[i+1] == 'q')) &&
        (part[i+2] == 'u')) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if ((part[i+1] == 'r') && !isvowel (part[i+2]) &&
        (part[i+3] != 'r')) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if ((part[i+2] == 'r') && (part[i-1] != 'n') &&
        !((part[i-1] == 'l') && (part[i+1] == 't'))) {
        strcat (synt, "ee 100 50 130\n"); accent = 1;}
    else if (((part[i+2] == 'l') || ((part[i+1] == 'l') &&
        !isvowel (part[i+2]))) && !isalpha (part[i-2])) {
        strcat (synt, "ee 100 50 130\n"); accent = 1;}
    else if (part[i+1] == 's')
    {
        if (!isalpha (part[i-1]) || (((part[i-1] == 'n') ||
            (part[i-1] == 'd')) && !isalpha (part[i-2])))
        {
            if (part[i+3] == 'e') strcat (synt,
                "e 90 50 120\n");
            else { strcat (synt, "ee 100 50 130\n");
                accent = 1;}
        }
        else if (((part[i-1] == 'm') ||
            (part[i-1] == 'n') || (part[i-1] == 'r')) &&
            (!isalpha (part[i-2]) || ((part[i-2] == 'a') ||
            (part[i-2] == 'e')))) strcat (synt,
            "e 90 50 120\n");
        else { strcat (synt, "ee 100 50 130\n");
            accent = 1;}
    }
    else strcat (synt, "e 90 50 120\n");
}
else if (((j - i) > 4) && ((part[i+5] == '\0') ||
    ((part[i+6] == '\0') && (part[i+5] == 's'))))
{
    if (((part[i+4] == 'm') || (part[i+4] == 'n')) &&
        !isvowel (part[i+2])) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
    else if (((part[i+1] == 'g') || (part[i+1] == 'q')) &&
        (part[i+2] == 'u') && ((part[i+4] == 'm') ||
        (part[i+4] == 'n'))) { strcat (synt,
        "ee 100 50 130\n"); accent = 1;}
```

```
                else if ((part[i+1] == 'r') && ((part[i+2] == 'g') ||
                    (part[i+2] == 'q')) && (part[i+3] == 'u')) {
                    strcat (synt, "ee 100 50 130\n"); accent = 1;}
                else if ((part[i+1] == 's') && (part[i+2] == 't') &&
                    (part[i+3] == 'r')) { strcat (synt,
                    "ee 100 50 130\n"); accent = 1;}
                else strcat (synt, "e 90 50 120\n");
            }
            else if (((j - i) > 5) && ((part[i+6] == '\0') ||
                ((part[i+7] == '\0') && (part[i+6] == 's'))))
            {
                if ((part[i+1] == 'r') && ((part[i+2] == 'g') ||
                    (part[i+2] == 'q')) && (part[i+3] == 'u') &&
                    ((part[i+5] == 'm') || (part[i+5] == 'n'))) {
                    strcat (synt, "ee 100 50 130\n"); accent = 1;}
                else strcat (synt, "e 90 50 120\n");
            }
            else strcat (synt, "e 90 50 120\n");
        }
        else strcat (synt, "e 90 50 120\n");
        break;
    case 201:/*   */
    case 233:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "em 120 50 130\n");
        else strcat (synt, "ee 100 50 130\n");
        break;
    case 202:/*   */
    case 234:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "em 120 50 130\n");
        else strcat (synt, "e 100 50 130\n");
        break;
    case 'i':
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "im 120 50 120\n");
        else if (!accent && ((((j - i) == 2) && (part[i+1] != 's')) ||
            (((j - i) == 3) && !isvowel (part[i+1]) &&
            !isvowel (part[i-1])) || (((j - i) == 4) &&
            !isvowel (part[i+1]) && !isvowel (part[i-1]) &&
            (part[i+3] == 's')))) strcat (synt, "i 100 50 130\n");
        else strcat (synt, "i 80 50 120\n");
        break;
    case 205:/*   */
    case 237:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "im 120 50 130\n");
        else strcat (synt, "i 100 50 130\n");
        break;
    case 'o':
        if ((part[i+1] == 'm') || (part[i+1] == 'n'))
        {
            if (!isvowel (part[i+2])) strcat (synt, "om 120 50 120\n");
```

```
    else strcat (synt, "o 90 50 120\n");
}
else if (!accent && !isvowel (part[i+1]))
{
    if (((j - i) > 1) && ((part[i+2] == '\0') ||
        ((part[i+3] == '\0') && (part[i+2] == 's'))))
    {
        if (part[i+1] == 'l') { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else if (part[i+1] == 'z') { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else strcat (synt, "o 90 50 120\n");
    }
    else if (((j - i) > 2) && ((part[i+3] == '\0') ||
        ((part[i+4] == '\0') && (part[i+3] == 's'))))
    {
        if ((part[i+1] == 'c') || iscedila (part[i+1]))
        {
            if ((part[i+2] == 'a') && (part[i-1] != 'b')) {
                strcat (synt, "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if (part[i+1] == 'l')
        {
            if (part[i+2] == 'o')
            {
                if ((part[i-1] == 'c') || (part[i-1] == 'm') ||
                    (part[i-1] == 'p') || (part[i-1] == 's')) {
                    strcat (synt, "oo 80 50 130\n");
                    accent = 1; }
                else if ((part[i-1] == 'r') &&
                    (part[i-2] == 't')) { strcat (synt,
                    "oo 80 50 130\n"); accent = 1; }
                else strcat (synt, "o 90 50 120\n");
            }
            else if (((part[i+2] == 'a') || (part[i+2] == 'e'))
                && ((part[i-3] != 'c') && (part[i-3] != 'n')))
                { strcat (synt, "oo 80 50 130\n");
                accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if (part[i+1] == 'r')
        {
            if ((part[i-1] == 'l') || (part[i-1] == 's'))
            {
                if (part[i+3] != 's') { strcat (synt,
                    "oo 80 50 130\n"); accent = 1; }
                else strcat (synt, "o 90 50 120\n");
            }
            else if (((part[i+2] == 'a') || (part[i+2] == 'e'))
                && ((part[i-1] != 'd') && (part[i-1] != 't')))
                { strcat (synt, "oo 80 50 130\n");
                accent = 1; }
```

```
            else strcat (synt, "o 90 50 120\n");
        }
        else if (part[i+1] == 's')
        {
            if ((part[i+2] != 'o') && (part[i-1] != 'p')) {
                strcat (synt, "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if (part[i+1] == 't')
        {
            if ((part[i-1] == 'r') && ((part[i-2] == 'a') ||
                ((part[i-2] == 'b') && (part[i+2] == 'o')) ||
                (part[i-2] == 'c') || (part[i-2] == 'r')))
                strcat (synt, "o 90 50 120\n");
            else { strcat (synt, "oo 80 50 130\n");
                accent = 1; }
        }
        else if (((part[i+2] == 'a') || (part[i+2] == 'e')) &&
            (part[i-1] != 'd') && (part[i-1] != 't')) {
            strcat (synt, "oo 80 50 130\n"); accent = 1; }
        else if ((part[i+2] == 'o') && (part[i+3] == 's') &&
            (part[i-1] != 'r') && (part[i-1] != 't')) {
            strcat (synt, "oo 80 50 130\n"); accent = 1; }
        else if ((part[i-1] == 'c') || ((part[i-1] == 'm') &&
            (part[i+1] == 'd'))) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else strcat (synt, "o 90 50 120\n");
    }
    else if (((j - i) > 3) && ((part[i+4] == '\0') ||
        ((part[i+5] == '\0') && (part[i+4] == 's'))))
    {
        if (((part[i+3] == 'm') || (part[i+3] == 'n')) &&
            isvowel (part[i+2]) && ((part[i-1] != 'f') ||
            (part[i+1] != 'r'))) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else if ((part[i+1] == 'b') && (part[i+2] == 'r'))
        {
            if ((part[i-1] == 's') && (part[i+3] == 'e'))
                strcat (synt, "o 90 50 120\n");
            else { strcat (synt, "oo 80 50 130\n");
                accent = 1; }
        }
        else if (part[i+1] == 'c')
        {
            if (part[i+2] == 'h') { strcat (synt,
                "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if (((part[i+1] == 'g') || (part[i+1] == 'q')) &&
            (part[i+2] == 'u')) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
```

```
        else if (part[i+1] == 'l')
        {
            if (part[i-1] == 'v') { strcat (synt,
                "oo 80 50 130\n"); accent = 1; }
            else if ((part[i+2] != 'f') && ((part[i+3] == 'a')
                || (part[i+3] == 'e'))) { strcat (synt,
                "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if (part[i+1] == 'r')
        {
            if (!isvowel (part[i+2]) && !iscedila (part[i+2])
                && (part[i+2] != 'z') && ((part[i+3] == 'a') ||
                (part[i+3] == 'e'))) { strcat (synt,
                "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
        else if ((part[i+1] != 'd') && (part[i+2] == 'r') &&
            (part[i-1] != 'x')) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else if (part[i+1] == 's')
        {
            if ((part[i+2] == 'c') || ((part[i+2] == 't') &&
                (part[i+3] == 'o') && ((part[i-1] == 'g') ||
                (part[i-1] == 'p') || (part[i-1] == 'r'))) ||
                ((part[i+3] == 'o') && (part[i+4] == '\0') &&
                !isalpha (part[i-1])) || ((part[i+3] == 'o') &&
                (part[i+4] == '\0') && (part[i-1] == 'r')))
                strcat (synt, "o 90 50 120\n");
            else { strcat (synt, "oo 80 50 130\n");
                accent = 1; }
        }
        else strcat (synt, "o 90 50 120\n");
    }
    else if (((j - i) > 4) && ((part[i+5] == '\0') ||
        ((part[i+6] == '\0') && (part[i+5] == 's'))))
    {
        if (((part[i+4] == 'm') || (part[i+4] == 'n')) &&
            !isvowel (part[i+2])) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else if (((part[i+1] == 'g') || (part[i+1] == 'q')) &&
            (part[i+2] == 'u') && ((part[i+4] == 'm') ||
            (part[i+4] == 'n'))) { strcat (synt,
            "oo 80 50 130\n"); accent = 1; }
        else if ((part[i+1] == 'l') || (part[i+1] == 'r'))
        {
            if (((part[i+2] == 'g') || (part[i+2] == 'q')) &&
                (part[i+3] == 'u') && (part[i-1] != 'p')) {
                strcat (synt, "oo 80 50 130\n"); accent = 1; }
            else strcat (synt, "o 90 50 120\n");
        }
```

```
                else if (isalpha (part[i-1]) && (part[i+1] == 's') &&
                    !isvowel (part[i+2]) && (part[i+4] != 'r')) {
                    strcat (synt, "oo 80 50 130\n"); accent = 1; }
                else strcat (synt, "o 90 50 120\n");
            }
            else if (((j - i) > 5) && ((part[i+6] == '\0') ||
                ((part[i+7] == '\0') && (part[i+6] == 's'))))
            {
                if (((part[i+1] == 'l') || (part[i+1] == 'r')) &&
                    ((part[i+2] == 'g') || (part[i+2] == 'q')) &&
                    (part[i+3] == 'u') && ((part[i+5] == 'm') ||
                    (part[i+5] == 'n'))) { strcat (synt,
                    "oo 80 50 130\n"); accent = 1; }
                else strcat (synt, "o 90 50 120\n");
            }
            else strcat (synt, "o 90 50 120\n");
        }
        else strcat (synt, "o 90 50 120\n");
        break;
    case 211:/*   */
    case 243:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "om 120 50 130\n");
        else strcat (synt, "oo 80 50 130\n");
        break;
    case 212:/*   */
    case 244:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "om 120 50 130\n");
        else strcat (synt, "o 90 50 130\n");
        break;
    case 213:/*   */
    case 245:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "om 120 50 130\n");
        else strcat (synt, "o 90 50 130\n");
        break;
    case 'u':
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "um 120 50 120\n");
        else if (!accent && ((((j - i) == 2) && (part[i+1] != 's')) ||
            (((j - i) == 3) && !isvowel (part[i+1]) &&
            !isvowel (part[i-1])) || (((j - i) == 4) &&
            !isvowel (part[i+1]) && !isvowel (part[i-1]) &&
            (part[i+3] == 's')))) strcat (synt, "u 100 50 130\n");
        else strcat (synt, "u 70 50 120\n");
        break;
    case 218:/*   */
    case 250:/*   */
        if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
            (!isvowel (part[i+2]))) strcat (synt, "um 120 50 130\n");
        else strcat (synt, "u 100 50 130\n");
        break;
```

```
case 220:/*   */
case 252:/*   */
    if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
        (!isvowel (part[i+2]))) strcat (synt, "um 120 50 120\n");
    else strcat (synt, "u 50 50 120\n");
    break;
case 'y':
    if (((part[i+1] == 'm') || (part[i+1] == 'n')) &&
        (!isvowel (part[i+2]))) strcat (synt, "im 120 50 120\n");
    else strcat (synt, "i 90 50 120\n");
    break;
/*--------------*/
/*- Consonants -*/
case 'b':
    if (!vowel) strcat(synt, "b 60 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "b 60 50 120\n");
    else strcat (synt, "b 60 50 120\ni 20 50 120\n");
    break;
case 'c':
    if (!vowel) strcat(synt, "s 110 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r'))
    {
        if (isei (part[i+1])) strcat (synt, "s 110 50 120\n");
        else strcat (synt, "k 100 50 120\n");
    }
    else if (part[i+1] == 'h') { strcat (synt, "x 80 50 120\n");
        i++; }
    else if (part[i+1] != 'k') strcat (synt,
        "k 100 50 120\ni 20 50 120\n");
    break;
case 199:/*   */
case 231:/*   */
    if (!vowel) strcat(synt, "s 110 50 120\ne 90 50 120\n
        s 110 50 120\ni 80 50 120\nd 60 50 120\ni 80 50 130\n
        l 80 50 120\ni 40 50 120\na 90 50 120\n");
    else if (isvowel (part[i+1])) strcat (synt, "s 110 50 120\n");
    else strcat (synt, "s 110 50 120\ni 20 50 120\n");
    break;
case 'd':
    if (!vowel) strcat(synt, "d 60 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "d 60 50 120\n");
    else strcat (synt, "d 60 50 120\ni 20 50 120\n");
    break;
case 'f':
    if (!vowel) strcat(synt, "ee 100 50 130\nf 100 50 120\n
        e 90 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "f 100 50 120\n");
    else strcat (synt, "f 100 50 120\ni 20 50 120\n");
    break;
```

```
case 'g':
    if (!vowel) strcat(synt, "j 60 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r'))
    {
        if (isei (part[i+1])) strcat (synt, "j 60 50 120\n");
        else if ((part[i+1] == 'u') && isvowel (part[i+2]))
        {
            if (isei (part[i+2])) strcat (synt, "g 50 50 120\n");
            else strcat (synt, "g 50 50 120\nu 50 50 120\n");
            i++;
        }
        else strcat (synt, "g 50 50 120\n");
    }
    else strcat (synt, "g 50 50 120\ni 20 50 120\n");
    break;
case 'h':
    if (!vowel) strcat(synt, "a 90 50 120\ng 50 50 120\n
        a 90 50 120\n");
    break;
case 'j':
    if (!vowel) strcat(synt, "j 60 50 120\noo 80 50 130\n
        t 75 50 120\na 90 50 120\n");
    else if (isvowel (part[i+1])) strcat (synt, "j 60 50 120\n");
    else strcat (synt, "j 60 50 120\ni 20 50 120\n");
    break;
case 'k':
    if (!vowel) strcat(synt, "k 100 50 120\na 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "k 100 50 120\n");
    else strcat (synt, "k 100 50 120\ni 20 50 120\n");
    break;
case 'l':
    if (!vowel) strcat(synt, "ee 100 50 130\nl 80 50 120\n
        e 90 50 120\n");
    else if (isvowel (part[i+1])) strcat (synt, "l 80 50 120\n");
    else if (part[i+1] == 'h') { strcat (synt, "l 80 50 120\n
        i 40 50 120\n"); i++; }
    else strcat (synt, "w 100 50 120\n");
    break;
case 'm':
    if (!vowel) strcat(synt, "e 90 50 140\nm 140 50 120\n
        e 90 50 120\n");
    else if (isvowel (part[i+1])) strcat (synt, "m 100 50 120\n");
    break;
case 'n':
    if (!vowel) strcat(synt, "e 90 50 130\nn 80 50 120\n
        e 90 50 120\n");
    else if (isvowel (part[i+1])) strcat (synt, "n 80 50 120\n");
    else if (part[i+1] == 'h') { strcat (synt, "nh 80 50 120\n");
        i++; }
    break;
```

```
case 'p':
    if (!vowel) strcat(synt, "p 100 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "p 100 50 120\n");
    else strcat (synt, "p 100 50 120\ni 20 50 120\n");
    break;
case 'q':
    if (!vowel) strcat(synt, "k 100 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r'))
    {
        if ((part[i+1] == 'u') && isvowel (part[i+2]))
        {
            if (isei (part[i+2])) strcat (synt, "k 100 50 120\n");
            else strcat (synt, "k 100 50 120\nu 50 50 120\n");
            i++;
        }
        else strcat (synt, "k 100 50 120\n");
    }
    else strcat (synt, "k 100 50 120\ni 20 50 120\n");
    break;
case 'r':
    if (!vowel) strcat(synt, "ee 100 50 130\nrr 80 50 120\n
        e 90 50 120\n");
    else if (isalpha (part[i-1]) || isvowel (part[i-1]))
    {
        if (isvowel (part[i+1])) strcat (synt, "r 50 50 120\n");
        else if (part[i+1] == 'r') { strcat (synt,
            "rr 80 50 120\n"); i++; }
        else strcat (synt, "r2 50 50 120\n");
    }
    else if (isvowel (part[i+1])) strcat (synt, "rr 80 50 120\n");
    else strcat (synt, "rr 80 50 120\ni 20 50 120\n");
    break;
case 's':
    if (!vowel) strcat(synt, "ee 100 50 130\ns 110 50 120\n
        e 90 50 120\n");
    else if (isvowel (part[i-1]))
    {
        if (isvowel (part[i+1])) strcat (synt, "z 60 50 120\n");
        else if (part[i+1] == 'h') { strcat (synt,
            "x 80 50 120\n"); i++; }
        else if (part[i+1] == 's') { strcat (synt,
            "s 110 50 120\n"); i++; }
        else strcat (synt, "s2 70 50 120\n");
    }
    else if (isvowel (part[i+1])) strcat (synt, "s 110 50 120\n");
    else if (part[i+1] == 'h') { strcat (synt, "x 80 50 120\n");
        i++; }
    else strcat (synt, "i 0 50 120\ns2 70 50 120\n");
    break;
```

```
case 't':
    if (!vowel) strcat(synt, "t 75 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "t 75 50 120\n");
    else strcat (synt, "t 75 50 120\ni 20 50 120\n");
    break;
case 'v':
    if (!vowel) strcat(synt, "v 60 50 120\ne 135 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "v 60 50 120\n");
    else strcat (synt, "v 60 50 120\ni 20 50 120\n");
    break;
case 'w':
    if (!vowel) strcat(synt, "d 60 50 120\na 90 50 130\n
        b 60 50 120\nl 80 50 120\ni 80 50 120\no 90 50 120\n");
    else if (isvowel (part[i+1]) || (part[i+1] == 'l') ||
        (part[i+1] == 'r')) strcat (synt, "v 60 50 120\n");
    else strcat (synt, "w 100 50 120\n");
    break;
case 'x':
    if (!vowel) strcat(synt, "x 80 50 120\ni 80 50 120\n
        s2 70 50 120\n");
    else if (isvowel (part[i+1]))
    {
        if ((part[i-1] == 'a') && (part[i+1] == 'a')) strcat (synt,
            "x 80 50 120\n");
        else if ((part[i-1] == 'e') && (part[i-2] == 'm'))
            strcat (synt, "x 80 50 120\n");
        else if ((part[i-1] == 'e') && !isalpha (part[i-2]))
            strcat (synt, "z 60 50 120\n");
        else if ((part[i-1] == 'i') && (part[i-2] != 'm'))
            strcat (synt, "x 80 50 120\n");
        else if (((part[i-1] == 'a') || (part[i-1] == 'o') ||
            (part[i-1] == 193) || (part[i-1] == 225) ||
            (part[i-1] == 211) || (part[i-1] == 243)) &&
            isalpha (part[i-2]) && (part[i-2] != 't') &&
            (part[i+1] == 'i'))  strcat (synt, "s 110 50 120\n");
        else if (isvowel (part[i-1]))  strcat (synt,
            "k 100 50 120\ni 0 50 120\ns 110 50 120\n");
        else strcat (synt, "x 80 50 120\n");
    }
    else if (isvowel (part[i-1]))
    {
        if (!isalpha (part[i+1]))  strcat (synt, "k 100 50 120\n
            i 0 50 120\ns2 70 50 120\n");
        else strcat (synt, "s2 70 50 120\n");
    }
    else strcat (synt, "x 80 50 120\ni 20 50 120\n");
    break;
```

```
                case 'z':
                    if (!vowel) strcat(synt, "z 60 50 120\ne 135 50 120\n");
                    else if (isvowel (part[i+1])) strcat (synt, "z 60 50 120\n");
                    else if (isvowel (part[i-1])) strcat (synt, "s2 70 50 120\n");
                    else strcat (synt, "z 60 50 120\ni 20 50 120\n");
                    break;
                /*-----------*/
                /*- Numbers -*/
                case '0': strcat (synt, "z 60 50 120\nee 100 50 130\nr 50 50 120\n
                    o 90 50 120\n"); break;
                case '1': strcat (synt, "um 120 50 120\n"); break;
                case '2': strcat (synt, "d 60 50 120\no 90 50 130\ni 80 50 120\n
                    s2 70 50 120\n")  ; break;
                case '3': strcat (synt, "t 75 50 120\nr 50 50 120\ne 100 50 130\n
                    s2 70 50 120\n"); break;
                case '4': strcat (synt, "k 100 50 120\nu 50 50 120\na 90 50 130\n
                    t 75 50 120\nr 50 50 120\no 90 50 120\n"); break;
                case '5': strcat (synt, "s 110 50 120\nim 120 50 130\n
                    k 100 50 120\no 90 50 120\n"); break;
                case '6': strcat (synt, "s 110 50 120\ne 90 50 130\ni 80 50 120\n
                    s2 70 50 120\n"); break;
                case '7': strcat (synt, "s 110 50 120\nee 100 50 120\nt 75 50 120\n
                    e 90 50 120\n"); break;
                case '8': strcat (synt, "o 90 50 130\ni 80 50 120\nt 75 50 120\n
                    o 90 50 120\n"); break;
                case '9': strcat (synt, "n 80 50 120\noo 80 50 130\nv 60 50 120\n
                    e 90 50 120\n"); break;
                default: strcat(synt, "_ 100 50 120\n");
            }
            i++;
        }
        strcat(synt, "_ 20 50 120\n");
    }

    /************/
    /* command: */
    sprintf (comm, "\"| mbrola -e -t %3.2f -f %2.1f /usr/lib/mbrola/br1/br1 - - |
        sox -t raw -sw -r 16000 - -t ossdsp /dev/dsp", rate, freq);
    strcat (synt, comm);
    system (synt);
}

/*****************************************************************************/
/* spk_num: ******************************************************************/
/*****************************************************************************/
void spk_num (char* num, char* part, int j, int nzero)
{
    /**************/
    /* variables: */
    char a, b, temp[NUMSIZE];
    int m = 0;
```

```
/*********/
/* num: */
while (j > 0)
{
    /***********/
    /* milhar: */
    if (m > 0)
    {
        strcpy (temp, num);
        if (isdigit (part[j-2])) a = part[j-2]; else a = '0';
        if (isdigit (part[j-3])) b = part[j-3]; else b = '0';
        if (((part[j] != '0') && (part[j+1] == '0') && (part[j+2] == '0')) ||
            ((part[j] == '0') && (part[j+1] == '0') && (part[j+2] != '0')) ||
            ((part[j] == '0') && (part[j+1] != '0') && (part[j+2] == '0')) ||
            ((part[j] == '0') && (part[j+1] != '0') && (part[j+2] != '0')))
            {
                sprintf (num, "e %s", temp);
                strcpy (temp, num);
            }
        if ((part[j-1] == '1') && (a == '0') && (b == '0'))
            switch (m)
            {
                case 1: sprintf (num, "mil %s", temp); break;
                case 2: sprintf (num, "milh o %s", temp); break;
                case 3: sprintf (num, "bilh o %s", temp); break;
                case 4: sprintf (num, "trilh o %s", temp); break;
                case 5: sprintf (num, "quatrilh o %s", temp); break;
                case 6: sprintf (num, "quintilh o %s", temp); break;
                case 7: sprintf (num, "sextilh o %s", temp); break;
                case 8: sprintf (num, "setilh o %s", temp); break;
                case 9: sprintf (num, "octilh o %s", temp); break;
                case 10: sprintf (num, "nonilh o %s", temp); break;
                case 11: sprintf (num, "decilh o %s", temp); break;
                case 12: sprintf (num, "undecilh o %s", temp); break;
                case 13: sprintf (num, "dodecilh o %s", temp); break;
                case 14: sprintf (num, "tredecilh o %s", temp); break;
                case 15: sprintf (num, "quatordecilh o %s", temp); break;
                case 16: sprintf (num, "quindecilh o %s", temp); break;
                case 17: sprintf (num, "sedecilh o %s", temp); break;
                case 18: sprintf (num, "septendecilh o %s", temp); break;
                case 19: sprintf (num, "zilh o %s", temp); break;
            }
         else if ((part[j-1] != '0') || (a != '0') || (b != '0'))
            switch (m)
            {
                case 1: sprintf (num, "mil %s", temp); break;
                case 2: sprintf (num, "milh es %s", temp); break;
                case 3: sprintf (num, "bilh es %s", temp); break;
                case 4: sprintf (num, "trilh es %s", temp); break;
                case 5: sprintf (num, "quatrilh es %s", temp); break;
                case 6: sprintf (num, "quintilh es %s", temp); break;
                case 7: sprintf (num, "sextilh es %s", temp); break;
                case 8: sprintf (num, "setilh es %s", temp); break;
```

```
            case 9: sprintf (num, "octilh es %s", temp); break;
            case 10: sprintf (num, "nonilh es %s", temp); break;
            case 11: sprintf (num, "decilh es %s", temp); break;
            case 12: sprintf (num, "undecilh es %s", temp); break;
            case 13: sprintf (num, "dodecilh es %s", temp); break;
            case 14: sprintf (num, "tredecilh es %s", temp); break;
            case 15: sprintf (num, "quatordecilh es %s", temp); break;
            case 16: sprintf (num, "quindecilh es %s", temp); break;
            case 17: sprintf (num, "sedecilh es %s", temp); break;
            case 18: sprintf (num, "septendecilh es %s", temp); break;
            case 19: sprintf (num, "zilh es %s", temp); break;
        }
}
m++;

/************/
/* unidade: */
j--;
if (j < 0) break;
if (part[j-1] != '1')
{
    strcpy (temp, num);
    switch (part[j])
    {
        case '1': sprintf (num, "um %s", temp); break;
        case '2': sprintf (num, "dois %s", temp); break;
        case '3': sprintf (num, "tr s %s", temp); break;
        case '4': sprintf (num, "quatro %s", temp); break;
        case '5': sprintf (num, "cinco %s", temp); break;
        case '6': sprintf (num, "seis %s", temp); break;
        case '7': sprintf (num, "sete %s", temp); break;
        case '8': sprintf (num, "oito %s", temp); break;
        case '9': sprintf (num, "nove %s", temp); break;
    }
}

/***********/
/* dezena: */
j--;
if (j < 0) break;
strcpy (temp, num);
switch (part[j])
{
    case '1':
        switch (part[j+1])
        {
            case '0': sprintf (num, "dez %s", temp); break;
            case '1': sprintf (num, "onze %s", temp); break;
            case '2': sprintf (num, "doze %s", temp); break;
            case '3': sprintf (num, "treze %s", temp); break;
            case '4': sprintf (num, "quatorze %s", temp); break;
            case '5': sprintf (num, "quinze %s", temp); break;
            case '6': sprintf (num, "dezessseis %s", temp); break;
```

```
            case '7': sprintf (num, "dezessete %s", temp); break;
            case '8': sprintf (num, "dezoito %s", temp); break;
            case '9': sprintf (num, "dezenove %s", temp); break;
        }
        break;
    case '2': if (part[j+1] == '0') sprintf (num, "vinte %s", temp);
        else sprintf (num, "vinte e %s", temp); break;
    case '3': if (part[j+1] == '0') sprintf (num, "trinta %s", temp);
        else sprintf (num, "trinta e %s", temp); break;
    case '4': if (part[j+1] == '0') sprintf (num, "quarenta %s", temp);
        else sprintf (num, "quarenta e %s", temp); break;
    case '5': if (part[j+1] == '0') sprintf (num, "cinq enta %s", temp);
        else sprintf (num, "cinq enta e %s", temp); break;
    case '6': if (part[j+1] == '0') sprintf (num, "sessenta %s", temp);
        else sprintf (num, "sessenta e %s", temp); break;
    case '7': if (part[j+1] == '0') sprintf (num, "setenta %s", temp);
        else sprintf (num, "setenta e %s", temp); break;
    case '8': if (part[j+1] == '0') sprintf (num, "oitenta %s", temp);
        else sprintf (num, "oitenta e %s", temp); break;
    case '9': if (part[j+1] == '0') sprintf (num, "noventa %s", temp);
        else sprintf (num, "noventa e %s", temp); break;
}

/*************/
/* centena: */
j--;
if (j < 0) break;
strcpy (temp, num);
switch (part[j])
{
    case '1': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "cem %s", temp); else sprintf (num, "cento e %s", temp); break;
    case '2': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "duzentos %s", temp); else sprintf (num, "duzentos e %s", temp);
        break;
    case '3': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "trezentos %s", temp); else sprintf (num, "trezentos e %s", temp);
        break;
    case '4': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "quatrocentos %s", temp); else sprintf (num, "quatrocentos e %s",
        temp); break;
    case '5': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "quinhentos %s", temp); else sprintf (num, "quinhentos e %s",
        temp); break;
    case '6': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "seiscentos %s", temp); else sprintf (num, "seiscentos e %s",
        temp); break;
    case '7': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "setecentos %s", temp); else sprintf (num, "setecentos e %s",
        temp); break;
    case '8': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
        "oitocentos %s", temp); else sprintf (num, "oitocentos e %s",
        temp); break;
```

```
            case '9': if ((part[j+1] == '0') && (part[j+2] == '0')) sprintf (num,
                "novecentos %s", temp); else sprintf (num, "novecentos e %s",
                temp); break;
        }
    }

    /**********/
    /* nzero: */
    if (nzero)
    {
        strcpy (temp, num);
        num[0] = '\0';
        while (nzero > 0) { strcat (num, "zero "); nzero--; }
        strcat (num, temp);
    }
}

/*****************************************************************************/
/* spk_char: *****************************************************************/
/*****************************************************************************/
void* spk_char (void* arg)
{
    /**************/
    /* variables: */
    unsigned char synt[SYNCSIZE] = "echo -e \"", comm[COMMSIZE];
    unsigned char *lc = (unsigned char *) arg;
    float freq = 1.0;
    charrun = 1;

    /*********/
    /* char: */
    if (isupper (*lc)) freq = UPPERFRQ;
    switch (tolower (*lc))
    {
        case ' ': strcat(synt, "_ 100 50 120\n"); break;
        case '!': strcat(synt, "e 90 50 120\ns2 70 50 120\nk 100 50 120\n
            l 80 50 120\nam 120 50 120\nm 70 50 120\na 90 50 120\ns 110 50 120\n
            @ 80 50 130\no 90 50 120\n"); break;
        case '\"': strcat(synt, "a 100 50 130\ns2 70 50 120\np 100 50 120\n
            a 90 50 120\ns2 70 50 120\n"); break;
        case '#': strcat(synt, "n 80 50 120\nam 120 50 130\nb 60 50 120\n
            e 90 50 120\nr2 50 50 120\n"); break;
        case '$': strcat(synt, "s 110 50 120\ni 80 50 120\nf 100 50 120\n
            r 50 50 120\n@ 80 50 130\no 90 50 120\n"); break;
        case '%': strcat(synt, "p 100 50 120\no 90 50 120\nr2 50 50 120\n
            s 110 50 120\nem 120 50 130\nt 75 50 120\no 90 50 120\n"); break;
        case '&': strcat(synt, "e 100 50 130\nk 100 50 120\nom 120 50 120\n
            m 70 50 120\ne 90 50 120\nr2 50 50 120\ns 110 50 120\ni 80 50 120\n
            a 90 50 120\nu 50 50 120\n"); break;
        case '\'': strcat(synt, "a 90 50 120\np 100 50 120\noo 80 50 130\n
            s2 70 50 120\nt 75 50 120\nr 50 50 120\no 90 50 120\nf 100 50 120\n
            o 90 50 120\n"); break;
```

```
case '(': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\ne 90 50 120\n
    p 100 50 120\na 90 50 120\nr 50 50 120\nem 120 50 130\nt 75 50 120\n
    e 90 50 120\nz 60 50 120\ne 90 50 120\n"); break;
case ')': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
    a 90 50 120\np 100 50 120\na 90 50 120\nr 50 50 120\nem 120 50 130\n
    t 75 50 120\ne 90 50 120\nz 60 50 120\ne 90 50 120\n"); break;
case '*': strcat(synt, "a 90 50 120\ns2 70 50 120\nt 75 50 120\n
    e 90 50 120\nr 50 50 120\ni 80 50 130\ns2 70 50 120\nk 100 50 120\n
    o 90 50 120\n"); break;
case '+': strcat(synt, "s 110 50 120\nom 120 50 130\nm 70 50 120\n
    a 90 50 120\n"); break;
case ',': strcat(synt, "v 60 50 120\ni 100 50 130\nr2 50 50 120\n
    g 50 50 120\nu 70 50 120\nl 80 50 120\na 90 50 120\n"); break;
case '-': strcat(synt, "i 100 50 130\nf 100 50 120\nem 120 50 120\n");
    break;
case '.': strcat(synt, "p 100 50 120\nom 120 50 130\nt 75 50 120\n
    o 90 50 120\n"); break;
case '/': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
    a 90 50 120\n"); break;
case '0': strcat (synt, "z 60 50 120\nee 100 50 130\nr 50 50 120\n
    o 90 50 120\n"); break;
case '1': strcat (synt, "um 120 50 120\n"); break;
case '2': strcat (synt, "d 60 50 120\no 90 50 130\ni 80 50 120\n
    s2 70 50 120\n")  ; break;
case '3': strcat (synt, "t 75 50 120\nr 50 50 120\ne 100 50 130\n
    s2 70 50 120\n"); break;
case '4': strcat (synt, "k 100 50 120\nu 50 50 120\na 90 50 130\n
    t 75 50 120\nr 50 50 120\no 90 50 120\n"); break;
case '5': strcat (synt, "s 110 50 120\nim 120 50 130\nk 100 50 120\n
    o 90 50 120\n"); break;
case '6': strcat (synt, "s 110 50 120\ne 90 50 130\ni 80 50 120\n
    s2 70 50 120\n"); break;
case '7': strcat (synt, "s 110 50 120\nee 100 50 120\nt 75 50 120\n
    e 90 50 120\n"); break;
case '8': strcat (synt, "o 90 50 130\ni 80 50 120\nt 75 50 120\n
    o 90 50 120\n"); break;
case '9': strcat (synt, "n 80 50 120\noo 80 50 130\nv 60 50 120\n
    e 90 50 120\n"); break;
case ':': strcat(synt, "d 60 50 120\no 90 50 130\ni 80 50 120\n
    s2 70 50 120\np 100 50 120\nom 120 50 130\nt 75 50 120\no 90 50 120\n
    s2 70 50 120\n"); break;
case ';': strcat(synt, "p 100 50 120\nom 120 50 130\nt 75 50 120\n
    o 90 50 120\ne 90 50 120\nv 60 50 120\ni 100 50 130\nr2 50 50 120\n
    g 50 50 120\nu 70 50 120\nl 80 50 120\na 90 50 120\n"); break;
case '<': strcat(synt, "m 70 50 120\ne 90 50 120\nn 80 50 120\n
    oo 80 50 130\nr2 50 50 120\n"); break;
case '=': strcat(synt, "i 80 50 120\ng 50 50 120\nu 50 50 120\n
    a 90 50 130\nu 50 50 120\n"); break;
case '>': strcat(synt, "m 70 50 120\na 90 50 120\ni 80 50 120\n
    oo 80 50 130\nr2 50 50 120\n"); break;
case '?': strcat(synt, "im 120 50 120\nt 75 50 120\ne 90 50 120\n
    rr 80 50 120\no 90 50 120\ng 50 50 120\na 90 50 120\ns 110 50 120\n
    @ 80 50 130\no 90 50 120\n"); break;
```

```
case '@': strcat(synt, "a 90 50 120\nrr 80 50 120\no 90 50 130\n
    b 60 50 120\na 90 50 120\n"); break;
case '[': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\n
    e 90 50 120\nk 100 50 120\no 90 50 120\nu 50 50 120\nx 80 50 120\n
    e 90 50 130\nt 75 50 120\ne 90 50 120\n"); break;
case '\\': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
    a 90 50 120\nim 120 50 120\nv 60 50 120\ne 90 50 120\nr2 50 50 120\n
    t 75 50 120\ni 80 50 130\nd 60 50 120\na 90 50 120\n "); break;
case ']': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
    a 90 50 120\nk 100 50 120\no 90 50 120\nu 50 50 120\nx 80 50 120\n
    e 90 50 130\nt 75 50 120\ne 90 50 120\n"); break;
case '^': strcat(synt, "s 110 50 120\ni 80 50 120\nr2 50 50 120\n
    k 100 50 120\num 120 50 120\nf 100 50 120\nl 80 50 120\nee 100 50 130\n
    k 100 50 120\ni 0 50 120\ns 110 50 120\no 90 50 120\n"); break;
case '_': strcat(synt, "s 110 50 120\nu 70 50 120\nb 60 50 120\n
    l 80 50 120\nim 120 50 120\nnh 80 50 120\na 90 50 130\nd 60 50 120\n
    o 90 50 120\n"); break;
case '`': strcat(synt, "k 100 50 120\nr 50 50 120\na 90 50 130\n
    z 60 50 120\ne 90 50 120\n"); break;
case 'a': strcat(synt, "a 90 50 120\n"); break;
    case 192:/*   */ freq = UPPERFRQ; strcat(synt, "a 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\nk 100 50 120\n
        r 50 50 120\na 90 50 130\nz 60 50 120\ne 90 50 120\n"); break;
    case 193:/*   */ freq = UPPERFRQ; strcat(synt, "a 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\na 90 50 120\n
        g 50 50 120\nu 70 50 130\nd 60 50 120\no 90 50 120\n"); break;
    case 194:/*   */ freq = UPPERFRQ; strcat(synt, "a 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\ns 110 50 120\n
        i 80 50 120\nr2 50 50 120\nk 100 50 120\num 120 50 120\n
        f 100 50 120\nl 80 50 120\nee 100 50 130\nk 100 50 120\n
        i 0 50 120\ns 110 50 120\no 90 50 120\n"); break;
    case 195:/*   */ freq = UPPERFRQ; strcat(synt, "a 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\nt 75 50 120\n
        i 80 50 130\nu 70 50 120\n"); break;
    case 224:/*   */ strcat(synt, "a 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\nk 100 50 120\nr 50 50 120\na 90 50 130\n
        z 60 50 120\ne 90 50 120\n"); break;
    case 225:/*   */ strcat(synt, "a 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\na 90 50 120\ng 50 50 120\nu 70 50 130\n
        d 60 50 120\no 90 50 120\n"); break;
    case 226:/*   */ strcat(synt, "a 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\ns 110 50 120\ni 80 50 120\nr2 50 50 120\n
        k 100 50 120\num 120 50 120\nf 100 50 120\nl 80 50 120\n
        ee 100 50 130\nk 100 50 120\ni 0 50 120\ns 110 50 120\n
        o 90 50 120\n"); break;
```

```
    case 227:/*   */ strcat(synt, "a 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\nt 75 50 120\ni 80 50 130\n
        u 70 50 120\n"); break;
case 'b': strcat(synt, "b 60 50 120\ne 90 50 120\n"); break;
case 'c': strcat(synt, "s 110 50 120\ne 90 50 120\n"); break;
    case 199:/*   */ freq = UPPERFRQ; strcat(synt, "s 110 50 120\n
        e 90 50 120\ns 110 50 120\ni 80 50 120\nd 60 50 120\ni 80 50 130\n
        l 80 50 120\ni 40 50 120\na 90 50 120\n"); break;
    case 231:/*   */ strcat(synt, "s 110 50 120\ne 90 50 120\n
        s 110 50 120\ni 80 50 120\nd 60 50 120\ni 80 50 130\nl 80 50 120\n
        i 40 50 120\na 90 50 120\n"); break;
case 'd': strcat(synt, "d 60 50 120\ne 90 50 120\n"); break;
case 'e': strcat(synt, "e 90 50 120\n"); break;
    case 201:/*   */ freq = UPPERFRQ; strcat(synt, "e 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\na 90 50 120\ng 50 50 120\n
        u 70 50 130\nd 60 50 120\no 90 50 120\n"); break;
    case 202:/*   */ freq = UPPERFRQ; strcat(synt, "e 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\ns 110 50 120\n
        i 80 50 120\nr2 50 50 120\nk 100 50 120\num 120 50 120\n
        f 100 50 120\nl 80 50 120\nee 100 50 130\nk 100 50 120\n
        i 0 50 120\ns 110 50 120\no 90 50 120\n"); break;
    case 233:/*   */ strcat(synt, "e 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\na 90 50 120\ng 50 50 120\nu 70 50 130\n
        d 60 50 120\no 90 50 120\n"); break;
    case 234:/*   */ strcat(synt, "e 90 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\ns 110 50 120\ni 80 50 120\nr2 50 50 120\n
        k 100 50 120\num 120 50 120\nf 100 50 120\nl 80 50 120\n
        ee 100 50 130\nk 100 50 120\ni 0 50 120\ns 110 50 120\n
        o 90 50 120\n"); break;
case 'f': strcat(synt, "ee 100 50 130\nf 100 50 120\ne 90 50 120\n");
    break;
case 'g': strcat(synt, "j 60 50 120\ne 90 50 120\n"); break;
case 'h': strcat(synt, "a 90 50 120\ng 50 50 120\na 90 50 120\n"); break;
case 'i': strcat(synt, "i 80 50 120\n"); break;
    case 205:/*   */ freq = UPPERFRQ; strcat(synt, "i 80 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\na 90 50 120\ng 50 50 120\n
        u 70 50 130\nd 60 50 120\no 90 50 120\n"); break;
    case 237:/*   */ strcat(synt, "i 80 50 130\nk 100 50 120\n
        om 120 50 120\na 90 50 120\ns 110 50 120\nem 120 50 120\n
        t 75 50 120\no 90 50 120\na 90 50 120\ng 50 50 120\nu 70 50 130\n
        d 60 50 120\no 90 50 120\n"); break;
case 'j': strcat(synt, "j 60 50 120\noo 80 50 130\nt 75 50 120\n
    a 90 50 120\n"); break;
case 'k': strcat(synt, "k 100 50 120\na 90 50 120\n"); break;
case 'l': strcat(synt, "ee 100 50 130\nl 80 50 120\ne 90 50 120\n"); break;
case 'm': strcat(synt, "e 90 50 140\nm 140 50 120\ne 90 50 120\n"); break;
case 'n': strcat(synt, "e 90 50 130\nn 80 50 120\ne 90 50 120\n"); break;
```

```
case 'o': strcat(synt, "o 90 50 120\n"); break;
    case 211:/*   */ freq = UPPERFRQ; strcat(synt, "o 90 50 130\n
        k 100 50 120\nom 120 50 120\na 90 50 120\ns 110 50 120\n
        em 120 50 120\nt 75 50 120\no 90 50 120\a 90 50 120\g 50 50 120\n
        u 70 50 130\d 60 50 120\o 90 50 120\n"); break;
    case 212:/*   */ freq = UPPERFRQ; strcat(synt, "o 90 50 130\n
        k 100 50 120\nom 120 50 120\a 90 50 120\s 110 50 120\n
        em 120 50 120\t 75 50 120\o 90 50 120\s 110 50 120\n
        i 80 50 120\r2 50 50 120\k 100 50 120\um 120 50 120\n
        f 100 50 120\l 80 50 120\ee 100 50 130\k 100 50 120\n
        i 0 50 120\s 110 50 120\o 90 50 120\n"); break;
    case 213:/*   */ freq = UPPERFRQ; strcat(synt, "o 90 50 130\n
        k 100 50 120\nom 120 50 120\a 90 50 120\s 110 50 120\n
        em 120 50 120\t 75 50 120\o 90 50 120\t 75 50 120\n
        i 80 50 130\u 70 50 120\n"); break;
    case 243:/*   */ strcat(synt, "o 90 50 130\k 100 50 120\n
        om 120 50 120\a 90 50 120\s 110 50 120\em 120 50 120\n
        t 75 50 120\o 90 50 120\a 90 50 120\g 50 50 120\u 70 50 130\n
        d 60 50 120\o 90 50 120\n"); break;
    case 244:/*   */ strcat(synt, "o 90 50 130\k 100 50 120\n
        om 120 50 120\a 90 50 120\s 110 50 120\em 120 50 120\n
        t 75 50 120\o 90 50 120\s 110 50 120\i 80 50 120\r2 50 50 120\n
        k 100 50 120\um 120 50 120\f 100 50 120\l 80 50 120\n
        ee 100 50 130\k 100 50 120\i 0 50 120\s 110 50 120\n
        o 90 50 120\n"); break;
    case 245:/*   */ strcat(synt, "o 90 50 130\k 100 50 120\n
        om 120 50 120\a 90 50 120\s 110 50 120\em 120 50 120\n
        t 75 50 120\o 90 50 120\t 75 50 120\i 80 50 130\n
        u 70 50 120\n"); break;
case 'p': strcat(synt, "p 100 50 120\e 90 50 120\n"); break;
case 'q': strcat(synt, "k 100 50 120\e 90 50 120\n"); break;
case 'r': strcat(synt, "ee 100 50 130\rr 80 50 120\e 90 50 120\n");
    break;
case 's': strcat(synt, "ee 100 50 130\s 110 50 120\e 90 50 120\n");
    break;
case 't': strcat(synt, "t 75 50 120\e 90 50 120\n"); break;
case 'u': strcat(synt, "u 70 50 120\n"); break;
    case 218:/*   */ freq = UPPERFRQ; strcat(synt, "u 70 50 130\n
        k 100 50 120\nom 120 50 120\a 90 50 120\s 110 50 120\n
        em 120 50 120\t 75 50 120\o 90 50 120\a 90 50 120\g 50 50 120\n
        u 70 50 130\d 60 50 120\o 90 50 120\n"); break;
    case 220:/*   */ freq = UPPERFRQ; strcat(synt, "u 70 50 130\n
        k 100 50 120\nom 120 50 120\a 90 50 120\s 110 50 120\n
        em 120 50 120\t 75 50 120\o 90 50 120\t 75 50 120\r 50 50 120\n
        e 90 50 130\m 70 50 120\a 90 50 120\n"); break;
    case 250:/*   */ strcat(synt, "u 70 50 130\k 100 50 120\n
        om 120 50 120\a 90 50 120\s 110 50 120\em 120 50 120\n
        t 75 50 120\o 90 50 120\a 90 50 120\g 50 50 120\u 70 50 130\n
        d 60 50 120\o 90 50 120\n"); break;
    case 252:/*   */ strcat(synt, "u 70 50 130\k 100 50 120\n
        om 120 50 120\a 90 50 120\s 110 50 120\em 120 50 120\n
        t 75 50 120\o 90 50 120\t 75 50 120\r 50 50 120\e 90 50 130\n
        m 70 50 120\a 90 50 120\n"); break;
```

```
        case 'v': strcat(synt, "v 60 50 120\ne 90 50 120\n"); break;
        case 'w': strcat(synt, "d 60 50 120\na 90 50 130\nb 60 50 120\n
            l 80 50 120\ni 80 50 120\no 90 50 120\n"); break;
        case 'x': strcat(synt, "x 80 50 120\ni 80 50 120\ns2 70 50 120\n"); break;
        case 'y': strcat(synt, "i 80 50 130\np 50 50 120\ni 0 50 120\n
            s 110 50 120\nu 70 50 120\nl 80 50 120\nom 120 50 120\n"); break;
        case 'z': strcat(synt, "z 60 50 120\ne 90 50 120\n"); break;
        case '{': strcat(synt, "a 90 50 120\nb 60 50 120\nr 50 50 120\n
            e 90 50 120\nx 80 50 120\na 90 50 130\nv 60 50 120\ne 90 50 120\n");
            break;
        case '|': strcat(synt, "b 60 50 120\na 90 50 130\nrr 80 50 120\n
            a 90 50 120\nv 60 50 120\ne 90 50 120\nr2 50 50 120\nt 75 50 120\n
            i 80 50 120\nk 100 50 120\na 90 50 130\nu 50 50 120\n"); break;
        case '}': strcat(synt, "f 100 50 120\nee 90 50 120\nx 80 50 120\n
            a 90 50 130\nx 80 50 120\na 90 50 130\nv 60 50 120\ne 90 50 120\n");
            break;
        case '~': strcat(synt, "t 75 50 120\ni 80 50 130\nu 70 50 120\n"); break;
        case 170: strcat(synt, "o 90 50 120\nr2 50 50 120\nd 60 50 120\n
            i 80 50 120\nn 80 50 120\na 90 50 130\nw 100 50 120\nf 100 50 120\n
            e 90 50 120\nm 100 50 120\ni 80 50 120\nn 80 50 120\ni 100 50 130\n
            n 80 50 120\no 90 50 120\n"); break;
        case 186: strcat(synt, "o 90 50 120\nr2 50 50 120\nd 60 50 120\n
            i 80 50 120\nn 80 50 120\na 90 50 130\nw 100 50 120\nm 100 50 120\n
            a 90 50 120\ns2 70 50 120\nk 100 50 120\nu 70 50 120\nl 80 50 120\n
            i 100 50 130\nn 80 50 120\no 90 50 120\n"); break;
        default: strcat(synt, "_ 100 50 120\n");
    }
    strcat(synt, "_ 20 50 120\n");

    /************/
    /* command: */
    sprintf (comm, "\"| mbrola -e -t %3.2f -f %2.1f /usr/lib/mbrola/br1/br1 - - |
        sox -t raw -sw -r 16000 - -t ossdsp /dev/dsp", rate, freq);
    strcat (synt, comm);
    system (synt);
    charrun = 0;
}

/*****************************************************************************/
/* spk_buff: *****************************************************************/
/*****************************************************************************/
void* spk (void* arg)
{
    /**************/
    /* variables: */
    unsigned char part[PARTSIZE], num[NUMSIZE];
    unsigned char *buff = (unsigned char*) arg;
    int i = 0, j, nzero;
    buffrun = 1;
```

```
/*********/
/* part: */
while (buff[i] != '\0')
{
    while ((buff[i] == ' ') || (buff[i] == '\n')) i++;
    if (buff[i] == '\0') break;
    j = 0;
    accent = 0;
    vowel = 0;
    upper = 0;
    if (ispunct (buff[i]))
    {
        if (((punct > 0) && punctsome (buff[i])) || (punct > 1))
        {
            if ((buff[i] == '-') && (punct == 1) && (isalpha (buff[i-1]) ||
                isvowel (buff[i-1])))
            {
                i++;
                continue;
            }
            else part[j] = buff[i];
        }
        else if ((buff[i] == '.') && (((buff[i+1] != ' ') &&
            (buff[i+1] != '\n') && (buff[i+1] != '\0')) || (buff[i-1] == '.')))
        {
            part[j] = buff[i];
        }
        else if ((buff[i] == '.') || (buff[i] == ';') || (buff[i] == ':'))
        {
            part[j] = ' '; part[++j] = ' ';
        }
        else
        {
            part[j] = ' ';
        }
        i++; j++;
    }
    else if (isdigit (buff[i]))
    {
        num[0] = '\0';
        nzero = 0;
        while (buff[i] == '0') { nzero++; i++; }
        while (isdigit (buff[i]) && (j < (PARTSIZE - 1)))
        {
            part[j] = buff[i];
            i++; j++;
        }
        part[j] = '\0';
        spk_num (num, part, j, nzero);
        spk ((void*) num); buffrun = 1;
        if (((buff[i] == '.') || (buff[i] == ',')) && (isdigit (buff[i+1])))
        {
            j = 0;
```

```
        part[j] = buff[i];
        i++; j++;
    }
    else continue;
}
else if (split_caps && (isupper (buff[i]) || isaccupp (buff[i])) &&
    (isupper (buff[i+1]) || isaccupp (buff[i+1])))
{
    part[j] = tolower (buff[i]);
    i++; j++;
}
else if (isalpha (buff[i]) || iscedila (buff[i]) || isaccvow (buff[i]))
{
    while ((isaccvow (buff[i]) || iscedila (buff[i]) || isalpha (buff[i]))
        && (j < (PARTSIZE - 1)))
    {
        part[j] = tolower (buff[i]);
        i++; j++;
    }
}
else
{
    part[j] = buff[i];
    i++; j++;
}
part[j] = '\0';
if ((j == 1) && (isupper (buff[i-1]) || isaccupp (buff[i-1]))) upper = 1;
if (!accent)
{
    if (!strcmp (part, "cds")) { strcpy (part, "ced s"); j = 5; accent = 1;
        vowel = 1; }
    /*en*/ else if (!strcmp (part, "alt")) { strcpy (part, " lti"); j = 4;
        accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "bytes")) { strcpy (part, "b itis");
        j = 6; accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "backspace")) { strcpy (part,
        "b csp ici"); j = 9; accent = 1; vowel = 1; }
    else if (!strcmp (part, "cdrom")) { strcpy (part, "cederr m"); j = 8;
        accent = 1; vowel = 1; }
    else if (!strcmp (part, "cdroms")) { strcpy (part, "cederr ms"); j = 9;
        accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "control")) { strcpy (part, "c ntrol");
        j = 7; accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "down")) { strcpy (part, "d um"); j = 4;
        accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "download")) { strcpy (part,
        "daunl uad"); j = 9; accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "downloads")) { strcpy (part,
        "daunl uads"); j = 10; accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "emacspeak")) { strcpy (part,
        "emacspik"); j = 8; accent = 1; vowel = 1; }
    /*en*/ else if (!strcmp (part, "emacspeakbrasil")) { strcpy (part,
        "emacspikbrasil"); j = 14; accent = 1; vowel = 1; }
```

```
            /*en*/ else if (!strcmp (part, "email")) { strcpy (part, "im il");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "emails")) { strcpy (part, "im ils");
                j = 6; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "end")) { strcpy (part, " ndi"); j = 4;
                accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "enter")) { strcpy (part, " nter");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "esc")) { strcpy (part, " ski"); j = 4;
                accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "google")) { strcpy (part, "g gol");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "help")) { strcpy (part, "r lp"); j = 4;
                accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "home")) { strcpy (part, "r mi"); j = 4;
                accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "internet")) { strcpy (part,
                "intern t"); j = 8; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "load")) { strcpy (part, "l uad");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "mail")) { strcpy (part, "m il"); j = 4;
                accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "mails")) { strcpy (part, "m ils");
                j = 4; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "page")) { strcpy (part, "p igi");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "pagedown")) { strcpy (part,
                "p igid um"); j = 9; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "pageup")) { strcpy (part, "p igi p");
                j = 7; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "shift")) { strcpy (part, "x fit");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "site")) { strcpy (part, "s iti");
                j = 5; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "software")) { strcpy (part, "s ftuer");
                j = 7; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "softwares")) { strcpy (part,
                "s ftuers"); j = 8; accent = 1; vowel = 1; }
            /*en*/ else if (!strcmp (part, "up")) { strcpy (part, " p"); j = 2;
                accent = 1; vowel = 1; }
        }
        while (pause && !stop) usleep (100000);
        if (stop) break;
        tts (part, j);
    }
    buff[0] = '\0';
    buffrun = 0;
}
```

Tone:

```
/*****************************************************************************\
*  EmBrasil - Copyright (C) 2005 - Version 1.0
*  Author: Samer Eberlin <samereberlin@gmail.com>
*
*  Released under the terms of the GNU Library General Public License.
*
*  EmBrasil is a brazilian portuguese speech server for "Emacspeak"
*  <http://emacspeak.sf.net> and works with "Mbrola Speech Synthesizer"
*  <http://tcts.fpms.ac.be/synthesis/mbrola.html>
\*****************************************************************************/

#include <math.h>    /* sin */
#include <fcntl.h>    /* O_WRONLY */
#include <linux/soundcard.h>    /* SNDCTL_DSP, AFMT_S16_LE */

#define PI 3.1416    /* 3.14159265358979323846264338327 */
#define TONESIZE 8    /* 2^3 (~1 number) */

/*****************************************************************************/
/* spk_tone function: ********************************************************/
/*****************************************************************************/
void* spk_tone (void* arg)
{
    /**************/
    /* variables: */
    unsigned char *tone = (unsigned char*) arg;
    char ch_freq[TONESIZE], ch_length[TONESIZE];
    int fd, freq, length, i = 0, j = 0;
    int fmt = AFMT_S16_LE;
    int stereo = 0;
    int speed = 44100;
    float max, step;
    float n;
    short val;

    /**********/
    /* freq: */
    while (tone[i] == ' ') i++;
    while ((tone[i] != ' ') && (j < (TONESIZE - 1)))
    {
        ch_freq[j] = tone[i];
        i++; j++;
    }
    ch_freq[j] = '\0';
    freq = atoi (ch_freq);
```

```
    /***********/
    /* length: */
    j = 0;
    while (tone[i] == ' ') i++;
    while ((tone[i] != ' ') && (tone[i] != '\0') && (j < (TONESIZE - 1)))
    {
        ch_length[j] = tone[i];
        i++; j++;
    }
    ch_length[j] = '\0';
    length = atoi (ch_length);

    /*********/
    /* tone: */
    max = 2 * PI * freq * length / 1000;
    step = 2 * PI * freq / speed;
    fd = open("/dev/dsp", O_WRONLY);
    if (fd == -1) return;
    ioctl(fd, SNDCTL_DSP_SETFMT, &fmt);
    ioctl(fd, SNDCTL_DSP_STEREO, &stereo);
    ioctl(fd, SNDCTL_DSP_SPEED, &speed);
    for (n = 0; n < max; n += step)
    {
        val = sin(n) * 8192;
        write(fd, &val, 2);
    }
    close(fd);
}
```

Apêndice B - Emacs e Emacspeak

Como todo grande *software*, este também tem seu código fonte subdividido em diversos arquivos. O arquivo referente à implementação dos recursos inéditos (emacspeak-keymap-brasil.el) será integralmente apresentado e os demais arquivos, referente às adaptações e traduções, serão apresentados em blocos, indicando sempre o nome do arquivo que sofreu alterações, o número referente à localização das linhas alteradas no arquivo original e finalmente a seção com as alterações já implementadas.

emacspeak-keymap-brasil.el:

```
/*****************************************************************************\
*  EmacspeakBrasil - Copyright (C) 2005 - Version 1.0
*  Author: Samer Eberlin <samereberlin@gmail.com>
*
*  Released under the terms of the GNU Library General Public License.
*
*  EmacspeakBrasil is a brazilian portuguese version of "Emacspeak"
*  <http://emacspeak.sf.net> and works with "Mbrola Speech Synthesizer"
*  <http://tcts.fpms.ac.be/synthesis/mbrola.html>
\*****************************************************************************/

(eval-when-compile (require 'cl))
(declaim  (optimize  (safety 0) (speed 3)))
(provide 'emacspeak-keymap-brasil)

;;*****************************************************************************
;;* The global settings: ******************************************************
;;*****************************************************************************
;;{{ auto-save:
(setq auto-save-default nil)
;;}}

;;{{ loadkeys:
(if (equal window-system nil) (shell-command (format "sudo loadkeys %s"
    (expand-file-name "console.kmap" emacspeak-etc-directory))))
;;}}
```

```
;;*****************************************************************************
;;* The global bindings: ******************************************************
;;*****************************************************************************
(define-key global-map [f1] 'emacspeakbrasil-help)
(define-key global-map [f2] 'emacspeakbrasil-help-b)
(define-key global-map [f3] 'emacspeakbrasil-forward-word)
(define-key global-map [C-f3] 'emacspeakbrasil-backward-word)
(define-key global-map [f4] 'emacspeak-kill-buffer)
(define-key global-map [f5] 'emacspeakbrasil-keyboard-test)
(define-key global-map [f6] 'emacspeakbrasil-text)
(define-key global-map [f7] 'emacspeakbrasil-dired)
(define-key global-map [f8] 'emacspeakbrasil-w3m)
(define-key global-map [f9] 'emacspeakbrasil-email)
(define-key global-map [f12] 'emacspeakbrasil-language)

(define-key global-map [escape] "\C-g")
(define-key global-map "\C-a" 'dtk-stop)
(define-key global-map "\C-z" 'tts-restart)
(define-key global-map "\M-X" 'shell-command)

(define-key global-map [C-right] 'emacspeakbrasil-forward-word)
(define-key global-map [C-left] 'emacspeakbrasil-backward-word)
(define-key global-map "\M-f" 'emacspeakbrasil-forward-word)
(define-key global-map "\M-b" 'emacspeakbrasil-backward-word)
(define-key global-map "\C-f" 'isearch-forward)
(define-key global-map "\C-s" 'emacspeakbrasil-query-replace)

(define-key global-map [M-up] 'emacspeakbrasil-volume-increase)
(define-key global-map [M-down] 'emacspeakbrasil-volume-decrease)
(define-key esc-map "s" 'emacspeakbrasil-volume-increase)
(define-key esc-map "n" 'emacspeakbrasil-volume-decrease)
(define-key global-map [M-right] 'emacspeakbrasil-rate-increase)
(define-key global-map [M-left] 'emacspeakbrasil-rate-decrease)
(define-key esc-map "p" 'emacspeakbrasil-rate-increase)
(define-key esc-map "o" 'emacspeakbrasil-rate-decrease)

(define-key help-map [f1] 'emacspeakbrasil-help-1)
(define-key help-map [f2] 'emacspeakbrasil-help-2)
(define-key help-map [f3] 'emacspeakbrasil-help-3)
(define-key help-map [f4] 'emacspeakbrasil-help-4)
(define-key help-map [f5] 'emacspeakbrasil-tutorial)

(define-key ctl-x-map "\C-p" 'emacspeakbrasil-print-buffer)
(define-key ctl-x-map "\S-c" 'emacspeakbrasil-shutdown)
(define-key ctl-x-map "x" 'kill-region)
(define-key ctl-x-map "c" 'kill-ring-save)
(define-key ctl-x-map "v" 'yank)

(define-key emacspeak-dtk-submap "," 'emacspeakbrasil-set-punct-none)
(define-key emacspeak-dtk-submap "." 'emacspeakbrasil-set-punct-some)
(define-key emacspeak-dtk-submap ";" 'emacspeakbrasil-set-punct-all)
```

```
(define-key emacspeak-keymap "c" 'emacspeakbrasil-speak-char)

(defun emacspeakbrasil-help ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "pressi ne«")
    (dtk-tone 500 50) (dtk-speak "« fec nco« parainici rotr inodetecl do««")
    (dtk-tone 500 50) (dtk-speak "« fec is« parainici roedit rdet xtos««")
    (dtk-tone 500 50) (dtk-speak "« fec te« parainici rogerenciad rdearqu vos««")
    (dtk-tone 500 50) (dtk-speak "« fe ito« parainici ronavegad rdeintern t««")
    (dtk-tone 500 50) (dtk-speak "« fen ve« parainici rogerenciad rdeim ils««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro« paraencerr rqualqu rjan la««")
    (dtk-tone 500 50) (dtk-speak "«escape« paracilenci r«
        ecancel rqualqu ropera  o««")
    (dtk-tone 500 50) (dtk-speak "« fed is« paraconhec routroscom ndos« ou  feu m«
        pararrepetir estemen ")
    (message "\
Pressione:
<F5> para iniciar o treino de teclado.
<F6> para iniciar o editor de textos.
<F7> para iniciar o gerenciador de arquivos.
<F8> para iniciar o navegador de internet.
<F9> para iniciar o gerenciador de emails.
<F4> para encerrar qualquer janela.
<Escape> para silenciar e cancelar qualquer opera  o.
<F2> para conhecer outros comandos ou <F1> para repetir este menu.")))

(defun emacspeakbrasil-help-b ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«c ntrol h«") (dtk-tone 450 50) (dtk-speak
        "« feu m« paraconhec rcom ndosdegerenciam ntodejan las««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol h«") (dtk-tone 450 50) (dtk-speak
        "« fed is« paraconhec rcom ndosdeleit rab sicadejan las««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol h«") (dtk-tone 450 50) (dtk-speak
        "« fetr s« paraconhec rcom ndosdeleit raavan  dadejan las««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol h«") (dtk-tone 450 50) (dtk-speak
        "« fequ tro« paraconhec rcom ndosdeconfigura  odoleit r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol h«") (dtk-tone 450 50) (dtk-speak
        "« fec nco« paraacess rotutori ldoambi nteem cs««")
    (dtk-tone 500 50) (dtk-speak "« lti x«
        paraezecut raplicat vosdoambi nteem cs««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol c « paraencerr roambi nteem cs««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«shift c « paradeslig rocomputad r")
    (message "\
Pressione:
<Control-h> <F1> para conhecer comandos de gerenciamento de janelas.
<Control-h> <F2> para conhecer comandos de leitura b sica de janelas.
<Control-h> <F3> para conhecer comandos de leitura avan ada de janelas.
<Control-h> <F4> para conhecer comandos de configura  o do leitor.
```

```
<Control-h> <F5> para acessar o tutorial do Ambiente Emacs.
<Alt-x> para executar aplicativos do Ambiente Emacs.
<Control-x> <Control-c> para encerrar o Ambiente Emacs.
<Control-x> <Shift-c> para desligar o computador.")))

(defun emacspeakbrasil-keyboard-test ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "Tr inodetecl do«« pressi ne escape« ou c ntrol g «
        paraencerrar")
    (message "Treino de teclado. Pressione <Escape> ou <Control-g> para
        encerrar."))
    (while (not (or (equal last-input-event 7) (equal last-input-event 'escape)))
    (setq char (read-event))
    (dtk-stop)
    (if (equal char 1) (dtk-speak "control a")
    (if (equal char 2) (dtk-speak "control b")
    (if (equal char 3) (dtk-speak "control c")
    (if (equal char 4) (dtk-speak "control d")
    (if (equal char 5) (dtk-speak "control e")
    (if (equal char 6) (dtk-speak "control f")
    (if (equal char 7) (dtk-speak "control g")
    (if (equal char 8) (dtk-speak "control h")
    (if (equal char 9) (if (equal window-system nil) (dtk-speak "tab, ou control
        i") (dtk-speak "control i"))
    (if (equal char 10) (dtk-speak "control j")
    (if (equal char 11) (dtk-speak "control k")
    (if (equal char 12) (dtk-speak "control l")
    (if (equal char 13) (if (equal window-system nil) (dtk-speak "enter, ou control
        m") (dtk-speak "control m"))
    (if (equal char 14) (dtk-speak "control n")
    (if (equal char 15) (dtk-speak "control o")
    (if (equal char 16) (dtk-speak "control p")
    (if (equal char 17) (dtk-speak "control q")
    (if (equal char 18) (dtk-speak "control r")
    (if (equal char 19) (dtk-speak "control s")
    (if (equal char 20) (dtk-speak "control t")
    (if (equal char 21) (dtk-speak "control u")
    (if (equal char 22) (dtk-speak "control v")
    (if (equal char 23) (dtk-speak "control w")
    (if (equal char 24) (dtk-speak "control x")
    (if (equal char 25) (dtk-speak "control y")
    (if (equal char 26) (dtk-speak "control z")
    (if (equal char 27)
        (progn (setq char (read-event))
        (if (equal char 91)
            (progn (setq char (read-event))
            (if (equal char 49)
                (progn (setq char (read-event))
                (if (equal char 126) (dtk-speak "home")
                (if (equal char 55) (progn (read-event) (dtk-speak "f6"))
                (if (equal char 56) (progn (read-event) (dtk-speak "f7"))
                (if (equal char 57) (progn (read-event) (dtk-speak "f8"))
```

```
            )))))
        (if (equal char 50)
            (progn (setq char (read-event))
            (if (equal char 126) (dtk-speak "insert")
            (if (equal char 48) (progn (read-event) (dtk-speak "f9"))
            (if (equal char 49) (progn (read-event) (dtk-speak "f10"))
            (if (equal char 51) (progn (read-event) (dtk-speak "f11"))
            (if (equal char 52) (progn (read-event) (dtk-speak "f12"))
            ))))))
        (if (equal char 51) (progn (setq char (read-event)) (dtk-speak
            "delete"))
        (if (equal char 52) (progn (setq char (read-event)) (dtk-speak "end"))
        (if (equal char 53) (progn (setq char (read-event)) (dtk-speak
            "pageup"))
        (if (equal char 54) (progn (setq char (read-event)) (dtk-speak
            "pagedown"))
        (if (equal char 65) (dtk-speak "ac ma")
        (if (equal char 66) (dtk-speak "ab ixo")
        (if (equal char 67) (dtk-speak "dir ita")
        (if (equal char 68) (dtk-speak "esqu rda")
        (if (equal char 80) (dtk-speak "pause")
        (if (equal char 91)
            (progn (setq char (read-event))
            (if (equal char 65) (dtk-speak "f1")
            (if (equal char 66) (dtk-speak "f2")
            (if (equal char 67) (dtk-speak "f3")
            (if (equal char 68) (dtk-speak "f4")
            (if (equal char 69) (dtk-speak "f5")
            ))))))
        (dtk-speak (format "alt abrecolch te %c" char)))))))))))))))
    (dtk-speak (format "alt %c" char))))
(if (equal char 28) (dtk-speak "print")
(if (equal char 32) (dtk-speak "Espa o")
(if (equal char 91) (dtk-speak "abrecolch te")
(if (equal char 93) (dtk-speak "f chacolch te")
(if (equal char 123) (dtk-speak "abrech ve")
(if (equal char 125) (dtk-speak "f chach ve")
(if (equal char 127) (dtk-speak "backspace")
(if (equal char 170) (dtk-letter "ª")
(if (equal char 186) (dtk-letter "º")
(if (equal char 2218) (dtk-letter "ª")
(if (equal char 2234) (dtk-letter "º")
(if (equal char 199) (dtk-letter " ")
(if (equal char 231) (dtk-letter " ")
(if (equal char 2247) (dtk-letter " ")
(if (equal char 2279) (dtk-letter " ")
(if (equal char -134217631) (dtk-speak "alt a")
(if (equal char -134217630) (dtk-speak "alt b")
(if (equal char -134217629) (dtk-speak "alt c")
(if (equal char -134217628) (dtk-speak "alt d")
(if (equal char -134217627) (dtk-speak "alt e")
(if (equal char -134217626) (dtk-speak "alt f")
(if (equal char -134217625) (dtk-speak "alt g")
```

```
    (if (equal char -134217624) (dtk-speak "alt h")
    (if (equal char -134217623) (dtk-speak "alt i")
    (if (equal char -134217622) (dtk-speak "alt j")
    (if (equal char -134217621) (dtk-speak "alt k")
    (if (equal char -134217620) (dtk-speak "alt l")
    (if (equal char -134217619) (dtk-speak "alt m")
    (if (equal char -134217618) (dtk-speak "alt n")
    (if (equal char -134217617) (dtk-speak "alt o")
    (if (equal char -134217616) (dtk-speak "alt p")
    (if (equal char -134217615) (dtk-speak "alt q")
    (if (equal char -134217614) (dtk-speak "alt r")
    (if (equal char -134217613) (dtk-speak "alt s")
    (if (equal char -134217612) (dtk-speak "alt t")
    (if (equal char -134217611) (dtk-speak "alt u")
    (if (equal char -134217610) (dtk-speak "alt v")
    (if (equal char -134217609) (dtk-speak "alt w")
    (if (equal char -134217608) (dtk-speak "alt x")
    (if (equal char -134217607) (dtk-speak "alt y")
    (if (equal char -134217606) (dtk-speak "alt z")
    (if (numberp char) (if (and (> char '32) (< char '127)) (dtk-letter
        (char-to-string char)) (message (format "%i" char)))
    (if (eq char 'return) (dtk-speak "enter")
    (if (eq char 'up) (dtk-speak "ac ma")
    (if (eq char 'down) (dtk-speak "ab ixo")
    (if (eq char 'left) (dtk-speak "esqu rda")
    (if (eq char 'right) (dtk-speak "dir ita")
    (if (eq char 'prior) (dtk-speak "pageup")
    (if (eq char 'next) (dtk-speak "pagedown")
    (dtk-speak (format "%s" char))
    ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "tr inoemcerr do«« pressi ne  feu m« para obt raj da")
    (message "Treino encerrado. Pressione <F1> para obter ajuda.")))

(defun emacspeakbrasil-text ()
    (interactive)
    (switch-to-buffer "Arquivo novo")
    (text-mode)
    (setq buffer-offer-save t)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "edit rdet xtos«« pressi ne  feu m« para obt raj da")
    (message "Editor de textos. Pressione <F1> para obter ajuda.")))

(defun emacspeakbrasil-dired ()
    (interactive)
    (dired-at-point "~/")
    (sleep-for 1)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "gerenciad rdearqu vos«« pressi ne  feu m« para
        obt raj da")
    (message "Gerenciador de arquivos. Pressione <F1> para obter ajuda.")))

(defun emacspeakbrasil-w3m ()
```

```
    (interactive)
    (w3m)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "navegad rdeintern t«« pressi ne  feu m« para
        obt raj da")
    (message "Navegador de internet. Pressione <F1> para obter ajuda.")))

(defun emacspeakbrasil-email ()
    (interactive)
    (rmail)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "gerenciad rdeim ils«« pressi ne  feu m« para
        obt raj da")
    (message "Gerenciador de emails. Pressione <F1> para obter ajuda.")))

(defun emacspeakbrasil-help-1 ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "Com ndosdegerenciam ntodejan las«« pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«f«
        paral ron medajan la««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«m«
        paral rab rradeest dodajan la««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol f«
        paralocaliz rdef rmaincrement lnoconte dodajan la««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol p « paraimprim roconte dodajan la««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol w« parassalv roconte dodajan la««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol b « paraezib rumal stacom asjan lasinici das««  seass ta ab ixo«
        ou ac ma« parasselecion r« epressi ne  nter« paraaltern r««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro« paraencerr rqualqu rjan la")
    (message "\
Comandos de gerenciamento de janelas. Pressione:
<Control-e> <f> para ler o nome da janela.
<Control-e> <m> para ler a barra de estado da janela.
<Control-f> para localizar de forma incremental no conte do da janela.
<Control-x> <Control-p> para imprimir o conte do da janela.
<Control-x> <Control-w> para salvar o conte do da janela.
<Control-x> <Control-b> para exibir uma lista com as janelas iniciadas,
  - Use a seta <Abaixo> ou <Acima> para selecionar e pressione <Enter> para
      alternar.
<F4>, para encerrar qualquer janela.")))

(defun emacspeakbrasil-help-2 ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "Com ndosdeleit rab sicadejan las« c m
        oacompanham ntodocurs r«« pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«seta ab ixo« ou ac ma«
        paral ralinhacegu nteouaanteri r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol ab ixo« ou c ntrol ac ma«
```

```
        paral ropar grafocegu nteouoanteri r««")
    (dtk-tone 500 50) (dtk-speak "«seta dir ita« ou esqu rda«
        paral rocaract rsegu nteouoanteri r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol dir ita« ou c ntrol esqu rda«
        paral rapalavracegu nteouaanteri r««")
    (dtk-tone 500 50) (dtk-speak "«r mi« ou  ndi«
        paraposicion rocurs rnoin cioounofin ldal nha««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol r mi« ou c ntrol  ndi«
        paraposicion rocurs rnoin cioounofin ldajan la««")
    (dtk-tone 500 50) (dtk-speak "«escape« paracilenci r«
        ecancel rqualqu ropera  o««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol z « paracilenci r«
        n ointerfer ndoemqualqu ropera  o")
    (message "\
Comandos de leitura b sica de janelas (com o acompanhamento do cursor). Pressione:
seta <Abaixo> ou <Acima> para ler a linha seguinte ou a anterior.
<Control-Abaixo> ou <Control-Acima>, para ler o par grafo seguinte ou o anterior.
seta <Direita> ou <Esquerda> para ler o caractere seguinte ou o anterior.
<Control-Direita> ou <Control-Esquerda> para ler a palavra seguinte ou a anterior.
<Home> ou <End> para posicionar o cursor no in cio ou no final da linha.
<Control-Home> ou <Control-End> para posicionar o cursor no in cio ou no final da
    janela.
<Escape> para silenciar e cancelar qualquer opera  o.
<Control-z> para silenciar n o interferindo em qualquer opera  o.")))

(defun emacspeakbrasil-help-3 ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "com ndosdeleit raavan  dadejan las« s m
        oacompanham ntodocurs r«« pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«l«
        paral ral nha ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«w«
        paral rapal vra ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«sh ft w« para oletr rapal vra ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«c «
        paral rocaract rsob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«b «
        paral rajan lat da««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«n«
        paral rajan lat daapart rdaposi  odocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«p «
        parapaus raleit ra««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«esp  o« paracontinu raleit raap sumcom ndodep usa")
    (message "\
Comandos de leitura avan ada de janelas (sem o acompanhamento do cursor).
    Pressione:
<Control-e> <l> para ler a linha sob o cursor.
<Control-e> <w> para ler a palavra sob o cursor.
<Control-e> <Shift-w> para soletrar a palavra sob o cursor.
<Control-e> <c> para ler o caractere sob o cursor.
```

```
<Control-e> <b> para ler a janela toda.
<Control-e> <n> para ler a janela toda a partir da posi  o do cursor.
<Control-e> <p> para pausar a leitura.
<Control-e> <Espa o> para continuar a leitura ap s um comando de pausa.")))

(defun emacspeakbrasil-help-4 ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "com ndosdeconfigura  odoleit r«« pressi ne«")
    (dtk-tone 500 50) (dtk-speak "« lti ac ma« ou  lti ab ixo«
        paraaument rourreduzirovol meemcincoporc nto««")
    (dtk-tone 500 50) (dtk-speak "« lti dir ita« ou  lti esqu rda«
        paraaument rourreduz ravelocid deemcincoporc nto««")
    (dtk-tone 500 50) (dtk-speak "« fed ze« paraaltern rentreosleit res portugu s
        e ingl s««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«d «") (dtk-tone 400 50) (dtk-speak "«k«
        paraativ roudesativ rorretornoaudit vodet clas««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«d «") (dtk-tone 400 50) (dtk-speak "«w«
        paraativ roudesativ rorretornoaudit vodepal vras««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«d «") (dtk-tone 400 50) (dtk-speak "«l«
        paraativ roudesativ rorretornoaudit vodel nhas««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«d «") (dtk-tone 400 50) (dtk-speak "«s«
        paraativ roudesativ rapron ncia oletr dadeletrascaptaliz das««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«d «") (dtk-tone 400 50) (dtk-speak "«v rgula« ou p nto« ou p ntoev rgula«
        paracet raleit radepontua  o")
    (message "\
Comandos de configura  o do leitor. Pressione:
<Alt-Acima> ou <Alt-Abaixo> para aumentar ou reduzir o volume em 5 porcento.
<Alt-Direita> ou <Alt-Esquerda> para aumentar ou reduzir a velocidade em 5
    porcento.
<F12> para alternar entre os leitores portugu s e ingl s.
<Control-e> <d> <k> para ativar ou desativar o retorno auditivo de teclas.
<Control-e> <d> <w> para ativar ou desativar o retorno auditivo de palavras.
<Control-e> <d> <l> para ativar ou desativar o retorno auditivo de linhas.
<Control-e> <d> <s> para ativar ou desativar a pron ncia soletrada de letras
    capitalizadas.
<Control-e> <d> ( <,> ou <.> ou <;> ) para setar a leitura de pontua  o.")))

(defun emacspeakbrasil-tutorial ()
    (interactive)
    (find-file (expand-file-name "EmacspeakBrasil.txt" emacspeak-etc-directory))
    (emacspeak-speak-buffer))

(defun emacspeakbrasil-print-buffer (buffer)
    (interactive (list (read-buffer "Pressione Enter, para confirmar a impress o.
        Para cancelar pressione Escape." nil)))
    (print-buffer)
    (tts-restart)
```

```
    (dtk-speak "Imprim ndo")
    (message "Imprimindo"))

(defun emacspeakbrasil-volume-increase ()
    (interactive)
    (shell-command "aumix -v +5")
    (sleep-for 1)
    (message "Volume aumentado em 5 porcento."))

(defun emacspeakbrasil-volume-decrease ()
    (interactive)
    (shell-command "aumix -v -5")
    (sleep-for 1)
    (message "Volume reduzido em 5 porcento."))

(defun emacspeakbrasil-rate-increase ()
    (interactive)
    (setq dtk-default-speech-rate (+ dtk-default-speech-rate 25))
    (dtk-set-rate dtk-default-speech-rate t)
    (message "Velocidade aumentada em 5 porcento."))

(defun emacspeakbrasil-rate-decrease ()
    (interactive)
    (setq dtk-default-speech-rate (- dtk-default-speech-rate 25))
    (dtk-set-rate dtk-default-speech-rate t)
    (message "Velocidade reduzida em 5 porcento."))

(defun emacspeakbrasil-set-punct-none ()
    (interactive)
    (dtk-set-punctuations "none" t)
    (message "Leitura de pontua  o setada para: nada."))

(defun emacspeakbrasil-set-punct-some ()
    (interactive)
    (dtk-set-punctuations "some" t)
    (message "Leitura de pontua  o setada para: algo."))

(defun emacspeakbrasil-set-punct-all ()
    (interactive)
    (dtk-set-punctuations "all" t)
    (message "Leitura de pontua  o setada para: tudo."))

(defun emacspeakbrasil-speak-char ()
    (interactive)
    (emacspeak-speak-char t))

(defun emacspeakbrasil-language ()
    (interactive)
    (if (equal dtk-tcl "/usr/bin/embrasil")
    (progn (setq dtk-tcl "/usr/bin/eflite") (tts-restart) (message "English reader
        activated"))
    (setq dtk-tcl "/usr/bin/embrasil") (tts-restart) (message "Leitor portugu s
        ativado")))
```

```
;;*****************************************************************************
;;* The text bindings: ********************************************************
;;*****************************************************************************
(require 'info nil t)
(declaim (special text-mode-map))
(define-key text-mode-map [f1] 'emacspeakbrasil-help-text)
(define-key text-mode-map [f2] 'emacspeakbrasil-help-text-b)

(defun emacspeakbrasil-help-text ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak (format "voc est  noedit rdet xtos« edit ndooarqu vo
        %s«« pressi ne«" (buffer-name)))
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol f« paraabr r um arqu vo««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol s« para alv roarqu vo««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol w« para alv roarqu vocomnomedifer nte««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak
        "«c ntrol p « paraimprim roarqu vo««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol f«
        paralocaliz rdef rmaincrement lnoconte dodoarqu vo««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol s«
        paralocaliz re ubstitu rnoconte dodoarqu vo««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro« paraencerr roeditordet xtos««")
    (dtk-tone 500 50) (dtk-speak "« fed is« paraconhec routroscom ndos« ou  feu m«
        pararrepetir estemen ")
    (message (format "\
Voc  est  no editor de textos, editando o arquivo: %s. Pressione:
<Control-x> <Control-f> para abrir um arquivo.
<Control-x> <Control-s> para salvar o arquivo.
<Control-x> <Control-w> para salvar o arquivo com nome diferente.
<Control-x> <Control-p> para imprimir o arquivo.
<Control-f> para localizar de forma incremental no conte do do arquivo.
<Control-s> para localizar e substituir no conte do do arquivo.
<F4> para encerrar o editor de textos.
<F2> para conhecer outros comandos ou <F1> para repetir este menu."
    (buffer-name)))))

(defun emacspeakbrasil-help-text-b ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«c ntrol esp  o« paradefin raposi  odocurs r como
        o in ciodeumacele  o«« Ap s inici daacele  o« ofin lser limit do
        pelaposi  oatu ldocurs r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak "«r«
        paral rarregi ocelecion da««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak "«c «
        paracopi rarregi ocelecion da««")
```

```
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak "«x«
        pararrecort rarregi ocelecion da««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak "«v «
        paracol rarregi ocopi da««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol x«") (dtk-tone 450 50) (dtk-speak "« «
        paradesfaz ra ltimaedi  o««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«c ntrol l« paral ron merodal nhaatu l««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol  «") (dtk-tone 450 50) (dtk-speak
        "«igu l« paral ron merodacol naatu l")
    (message (format "\
Pressione: <Control-Espa o>, para definir a posi  o do cursor como o in cio de uma
    sele  o.
  - Ap s iniciada a sele  o, o final ser  limitado pela posi  o atual do cursor.
<Control-e> <r> para ler a regi o selecionada.
<Control-x> <c> para copiar a regi o selecionada.
<Control-x> <x>, para recortar a regi o selecionada.
<Control-x> <v> para colar a regi o copiada.
<Control-x> <u> para desfazer a  ltima edi  o.
<Control-e> <Control-l> para ler o n mero da linha atual.
<Control-e> < = > para ler o n mero da coluna atual.")))


;;*****************************************************************************
;;* The dired bindings: *******************************************************
;;*****************************************************************************
(require 'dired nil t)
(require 'dired-aux nil t)
(declaim (special dired-mode-map))
(define-key dired-mode-map [f1] 'emacspeakbrasil-help-dired)
(define-key dired-mode-map [f2] 'emacspeakbrasil-help-dired-b)
(define-key dired-mode-map [next] 'emacspeakbrasil-dired-next)
(define-key dired-mode-map [prior] 'emacspeakbrasil-dired-previous)
(define-key dired-mode-map "\r" 'dired-explore-file)
(define-key dired-mode-map "\d" 'dired-explore-back)
(require 'ls-lisp)
(setq ls-lisp-dirs-first t)
(setq ls-lisp-ignore-case t)

(defun emacspeakbrasil-help-dired ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak (format "voc est  nogerenciad rdearqu vos«
        explor ndoodiret rio %s«« pressi ne«" (buffer-name)))
    (dtk-tone 500 50) (dtk-speak "«seta ab ixo« ou ac ma«
        paraavan  rouretroced rocurs rpelosarqu vosediret rios««")
    (dtk-tone 500 50) (dtk-speak "«pagedown« ou pageup«
        paraavan  rouretroced rocurs r emd zposi  es««
        umb pser c mpreemit doantesdeanunci rosdiret rios
        paradiferenci losdosarqu vos««")
    (dtk-tone 500 50) (dtk-speak "« nter« paraacess rodiret rio
        ouedit roarqu vo ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«backspace« para a rdodiret rio
```

```
        eexplor rumn veuanteri r««")
    (dtk-tone 500 50) (dtk-speak "«g « parainform rocaminhocompl to
        eatualiz rodiret rio em explora  o««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro«
        paraencerr rogerenciad rdearqu vos««")
    (dtk-tone 500 50) (dtk-speak "« fed is« paraconhec routroscom ndos« ou  feu m«
        pararrepetir estemen ")
    (message (format "\
Voc  est  no gerenciador de arquivos, explorando o diret rio: %s. Pressione:
seta <Abaixo> ou <Acima> para avan ar ou retroceder o cursor pelos arquivos e
    diret rios.
<PageDown> ou <PageUp> para avan ar ou retroceder o cursor em 10 posi  es.
  - Um bip ser  sempre emitido antes de anunciar os diret rios para diferenci -los
      dos arquivos.
<Enter> para acessar o diret rio ou editar o arquivo sob o cursor.
<Backspace> para sair do diret rio e explorar um n vel anterior.
<g> para informar o caminho completo e atualizar o diret rio em explora  o.
<F4> para encerrar o gerenciador de arquivos.
<F2> para conhecer outros comandos ou <F1> para repetir este menu."
    (buffer-name)))))

(defun emacspeakbrasil-help-dired-b ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«x fit c « paracopi roarqu vo ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit d « paradelet roarqu vo ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit p « paraimprim roarqu vo ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit r« pararrenome roarqu vooudiret rio ob
        ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«+« paracri rumdiret rio««")
    (dtk-tone 500 50) (dtk-speak "«x fit x« paraezecut rumaplicat vocomoarqu vo ob
        ocurs r«« dur nteaezecu  odoaplicat vo« pressi ne escape« paraencerrar««")
    (dtk-tone 500 50) (dtk-speak "«z « parainform rotam nhodoarqu vooudiret rio ob
        ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit z «
        paracompact roudescompact roarqu vonoform to gez ")
    (message (format "\
Pressione: <Shift-c> para copiar o arquivo sob o cursor.
<Shift-d> para deletar o arquivo sob o cursor.
<Shift-p> para imprimir o arquivo sob o cursor.
<Shift-r> para renomear o arquivo ou diret rio sob o cursor.
< + > para criar um diret rio.
<Shift-x> para executar um aplicativo com o arquivo sob o cursor.
  - Durante a execu  o do aplicativo, pressione <Escape> para encerrar.
<z> para informar o tamanho do arquivo ou diret rio sob o cursor.
<Shift-z> para compactar ou descompactar o arquivo no formato \"gz\".")))

(defun emacspeakbrasil-dired-next ()
    (interactive) (call-interactively (dired-next-line 10)))

(defun emacspeakbrasil-dired-previous ()
    (interactive) (call-interactively (dired-previous-line 10)))
```

```
(defun dired-explore-file ()
  "In dired, visit *in full window* the file or directory named on this line.
If a directory, open it in the *current* buffer.
Alternative to ‘dired-advertised-find-file’, bound to
    \\[dired-advertised-find-file]."
  (interactive)
  (dired-explore (dired-get-filename)))

(defun dired-explore-back ()
  (interactive)
  (dired-explore ".."))

(defun dired-explore (file)
  "In dired, visit FILE *in full window*.
If a directory, open it in the *current* buffer."
  (setq file (file-name-sans-versions file t))
  (if (file-exists-p file)
      (if (file-directory-p file)
          (find-alternate-file file)
        (find-file file))
    (if (file-symlink-p file)
        (error "File is a symlink to a nonexistent target")
      (error "File no longer exists; type ‘g’ to update Dired buffer"))))


;;;****************************************************************************
;;;* The w3m bindings: ********************************************************
;;;****************************************************************************
(require 'w3m nil t)
(declaim (special w3m-mode-map))
(define-key w3m-mode-map [f1] 'emacspeakbrasil-help-w3m)
(define-key w3m-mode-map [f2] 'emacspeakbrasil-help-w3m-b)
(define-key w3m-mode-map "w" 'emacspeakbrasil-download)
(setq w3m-command "w3m_notables")
(setq w3m-use-cookies t)
(setq w3m-default-save-directory "~/")

(defun emacspeakbrasil-help-w3m ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak (format "voc est  nonavegad rdeintern t«
        acess ndoo  iti %s«« pressi ne«" (or w3m-current-title "sem t tulo")))
    (dtk-tone 500 50) (dtk-speak "«g « paraacess rumn voender  o««")
    (dtk-tone 500 50) (dtk-speak "«w« para alv roconte dodoender  oatu l««")
    (dtk-tone 500 50) (dtk-speak "«tab« ou x fit tab«
        paraposicion rocurs rnolinksegu nteounoanteri r««")
    (dtk-tone 500 50) (dtk-speak "«d « parafaz rdaunl uaddoender  odolink ob
        ocurs r««")
    (dtk-tone 500 50) (dtk-speak "« nter« paraacess roender  odolink ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit b « paravolt raoender  oanteri r««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro« paraencerr ronavegad rdeintern t««")
    (dtk-tone 500 50) (dtk-speak "« fed is« paraconhec routroscom ndos« ou  feu m«
```

```
        pararrepetir estemen ")
    (message (format "\
Voc  est  no navegador de internet, acessando o site: %s. Pressione:
<g> para acessar um novo endere o.
<w> para salvar o conte do do endere o atual.
<Tab> ou <Shift-Tab> para posicionar o cursor no link seguinte ou no anterior.
<d> para fazer download do endere o do link sob o cursor.
<Enter> para acessar o endere o do link sob o cursor.
<Shift-b> para voltar ao endere o anterior.
<F4> para encerrar o navegador de internet.
<F2> para conhecer outros comandos ou <F1> para repetir este menu."
    (or w3m-current-title "sem t tulo")))))

(defun emacspeakbrasil-help-w3m-b ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«c « parainform r«
        ecopi rna readetransfer nciaoender  oatu l««")
    (dtk-tone 500 50) (dtk-speak "« « parainform r«
        ecopi rna readetransfer nciaoender  odolink ob ocurs r««")
    (dtk-tone 500 50) (dtk-speak "«x fit r«
        pararrecarreg roconte dodoender  oatu l««")
    (dtk-tone 500 50) (dtk-speak "«f chacolch te« ou abrecolch te«
        paraposicion rocurs rnoc mpodeformul riocegu nteounoanteri r««")
    (dtk-tone 500 50) (dtk-speak "«c ntrol c «") (dtk-tone 450 50) (dtk-speak
        "«c ntrol c « paraenvi roformul rio««")
    (dtk-tone 500 50) (dtk-speak "« « paraadicion roender  oatu l
        al stadefavor tos««")
    (dtk-tone 500 50) (dtk-speak "« lti  « paraadicion roender  odolink ob ocurs r
        al stadefavor tos««")
    (dtk-tone 500 50) (dtk-speak "«v « paraacess ral stadefavor tos««
        nal stadefavor tospressi ne« c ntrol k « paraexclu roreg stro ob ocurs r")
    (message (format "\
Pressione <c> para informar e copiar na  rea de transfer ncia o endere o atual.
<u> para informar e copiar na  rea de transfer ncia o endere o do link sob o
    cursor.
<Shifit-r> para recarregar o conte do do endere o atual.
< ] > ou < [ > para posicionar o cursor no campo de formul rio seguinte ou no
    anterior.
<Control-c> <Control-c> para enviar o formul rio.
<a> para adicionar o endere o atual   lista de favoritos.
<Alt-a> para adicionar o endere o do link sob o cursor   lista de favoritos.
<v> para acessar a lista de favoritos.
  - Na lista de favoritos pressione <Control-k> para excluir o registro sob o
      cursor.")))

(defun emacspeakbrasil-download ()
    (interactive)
    (w3m-download w3m-current-url))
```

```
;;*****************************************************************************
;;* The mail bindings: ********************************************************
;;*****************************************************************************
(require 'rmail)
(require 'rmailsort)
(declaim (special rmail-mode-map))
(define-key rmail-mode-map [f1] 'emacspeakbrasil-help-mail)

(defun emacspeakbrasil-help-mail ()
    (interactive)
    (let ((emacspeak-speak-messages nil))
    (tts-restart) (dtk-speak "voc est  nogerenciad rdeim ils«« pressi ne«")
    (dtk-tone 500 50) (dtk-speak "«m« paraenvi rumnovoim il««")
    (dtk-tone 500 50) (dtk-speak "«g « pararreceb rn vosim ils««")
    (dtk-tone 500 50) (dtk-speak "«n« paral ropr ximoim il««")
    (dtk-tone 500 50) (dtk-speak "«p « paral roim ilanteri r««")
    (dtk-tone 500 50) (dtk-speak "« rre« pararrespond roim ilselecion do««")
    (dtk-tone 500 50) (dtk-speak "« fe« paraencaminh roim ilselecion do««")
    (dtk-tone 500 50) (dtk-speak "«d « paradelet roim ilselecion do««")
    (dtk-tone 500 50) (dtk-speak "« fequ tro« paraencerr rogerenciadordeim ils«")
    (message "\
Voc  est  no gerenciador de emails. Pressione:
<m> para enviar um novo email.
<g> para receber novos emails.
<n> para ler o pr ximo email.
<p> para ler o email anterior.
<r> para responder o email selecionado.
<f> para encaminhar o email selecionado.
<d> para deletar o email selecionado.
<F4> para encerrar o gerenciador de emails.")))

;(setq user-full-name "User Name"
;      user-login-name "username"
;      mail-from-style 'angles
;      user-mail-address "username@provider.com")

;; pop:
;(setenv "MAILHOST" "pop.provider.com")
;  (setq rmail-primary-inbox-list '("po:username") rmail-pop-password-required
;    t)

;; smtp:
;(setq send-mail-function 'smtpmail-send-it)
;(setq message-send-mail-function 'smtpmail-send-it)
;(setq smtpmail-smtp-server "smtp.provider.com")
;(setq smtpmail-smtp-service "smtp")


;;*****************************************************************************
;;* The EmacspeakBrasil redefine: *********************************************
;;*****************************************************************************
```

```
;;* (y or n): -> (s ou n): ****************************************************
(defun y-or-n-p (prompt)
    (setq char (read-char (format "%s (s ou n) " prompt)))
    (if (or (char-equal char 83) (char-equal char 115)) (setq action t) (setq
        action nil))
    (message "")
    action)

;;* (yes or no): -> (s ou n): ***************************************************
(defun yes-or-no-p (prompt)
    (setq char (read-char (format "%s (s ou n) " prompt)))
    (if (or (char-equal char 83) (char-equal char 115)) (setq action t) (setq
        action nil))
    (message "")
    action)

;;* forward-word -> emacspeakbrasil-forward-word ******************************
(defun emacspeakbrasil-forward-word (point)
  (interactive "p")
    (if (or (equal (following-char) 9) (equal (following-char) 32))
      (skip-chars-forward "\t ")
      (skip-chars-forward "^\n\t ") (skip-chars-forward "\t "))
    (when (equal (following-char) 10) (dtk-tone 250 75) (forward-char))
    (if (equal (point) (point-max))
      (progn (dtk-tone 500 75) (message "Fim da janela"))
      (emacspeak-speak-word)))

;;* backward-word -> emacspeakbrasil-backward-word *****************************
(defun emacspeakbrasil-backward-word (point)
  (interactive "p")
    (when (or (equal (preceding-char) 9) (equal (preceding-char) 32))
        (skip-chars-backward "\t "))
    (when (equal (preceding-char) 10) (dtk-tone 250 75) (backward-char))
    (if (and (equal (skip-chars-backward "^\n\t ") 0) (equal (point) (point-min)))
      (progn (dtk-tone 500 75) (message "Come o da janela"))
      (emacspeak-speak-word)))

;;* (files.el @ 3741 @) save-buffers-kill-emacs -> emacspeakbrasil-shutdown ****
(defun emacspeakbrasil-shutdown (&optional buffer &optional arg)
  "Offer to save each buffer, then turn off the computer. With prefix arg, silently
      save all file-visiting buffers, then turn off."
  (interactive "P") (if (not (active-minibuffer-window)) (list (read-buffer
      "Pressione Enter, para confirmar o desligamento do computador. Para cancelar
      pressione Escape." nil)) (abort-recursive-edit))
  (save-some-buffers arg t)
  (and (or (not (memq t (mapcar (function
                                  (lambda (buf) (and (buffer-file-name buf)
                                                     (buffer-modified-p buf))))
                                (buffer-list))))
           (y-or-n-p "Existem janelas modificadas; Mesmo assim deseja desligar? "))
       (or (not (fboundp 'process-list))
           ;; process-list is not defined on VMS.
           (let ((processes (process-list))
```

```
                active)
              (while processes
                (and (memq (process-status (car processes)) '(run stop open))
                     (let ((val (process-kill-without-query (car processes))))
                       (process-kill-without-query (car processes) val)
                       val)
                     (setq active t))
                (setq processes (cdr processes)))
              (or (not active)
                  (list-processes)
                  (s-ou-no-p "Existem processos ativos; Mesmo assim deseja
                      encerr -los e desligar? "))))
        ;; Query the user for other things, perhaps.
        (run-hook-with-args-until-failure 'kill-emacs-query-functions)
        (or (null confirm-kill-emacs)
            (funcall confirm-kill-emacs "Deseja realmente desligar o computador? "))
;;{{ Add:
        (progn
        (tts-restart)
        (dtk-speak "Deslig ndo.")
        (sleep-for 2)
        (shell-command "sudo shutdown -h now")
;;}}
        (kill-emacs))))

;;* (replace.el @ 59 @) query-replace-read-args ->
;;    emacspeakbrasil-query-replace-read-args **********************************
(defun emacspeakbrasil-query-replace-read-args (string regexp-flag)
  (let (from to)
    (if query-replace-interactive
        (setq from (car (if regexp-flag regexp-search-ring search-ring)))
      (setq from (read-from-minibuffer (format "%s: " string)
                                       nil nil nil
                                       query-replace-from-history-variable
                                       nil t)))
    (setq to (read-from-minibuffer (format "%s %s por: " string from)
                                   nil nil nil nil nil t))
    (if (and transient-mark-mode mark-active)
        (list from to current-prefix-arg (region-beginning) (region-end))
      (list from to current-prefix-arg nil nil))))

;;* (replace.el @ 74 @) query-replace -> emacspeakbrasil-query-replace *********
(defun emacspeakbrasil-query-replace (from-string to-string &optional delimited
    start end)
  "Replace some occurrences of FROM-STRING with TO-STRING.
As each match is found, the user must type a character saying
what to do with it.  For directions, type \\[help-command] at that time.

In Transient Mark mode, if the mark is active, operate on the contents
of the region.  Otherwise, operate from point to the end of the buffer.

If `query-replace-interactive' is non-nil, the last incremental search
string is used as FROM-STRING--you don't have to specify it with the
```

```
minibuffer.

Replacement transfers the case of the old text to the new text,
if ‘case-replace’ and ‘case-fold-search’
are non-nil and FROM-STRING has no uppercase letters.
\(Preserving case means that if the string matched is all caps, or capitalized,
then its replacement is upcased or capitalized.)

Third arg DELIMITED (prefix arg if interactive), if non-nil, means replace
only matches surrounded by word boundaries.
Fourth and fifth arg START and END specify the region to operate on.

To customize possible responses, change the \"bindings\" in ‘query-replace-map’."
  (interactive (emacspeakbrasil-query-replace-read-args "Localizar e substituir"
      nil))
;;{{ Add (from emacspeak-advice):
  (declare (special voice-lock-mode ))
  (let ((saved-voice-lock voice-lock-mode) (emacspeak-speak-messages nil))
  (dtk-stop)
;;}}
  (perform-replace from-string to-string t nil delimited nil nil start end)
;;{{ Add (from emacspeak-advice):
    (unwind-protect
        (progn
          (setq voice-lock-mode 1)
          (setq emacspeak-replace-start nil
                emacspeak-replace-end nil
                emacspeak-replace-highlight-on nil ))
      (emacspeak-auditory-icon 'task-done)
      (setq voice-lock-mode saved-voice-lock
            emacspeak-speak-messages t))))
;;}}


;;*****************************************************************************
;;* The EmacspeakBrasil advice: ***********************************************
;;*****************************************************************************

;;* (w3m.el @ 5483 @) (ler a mensagem apenas quando chamada interativamente) ****
(defadvice w3m-print-this-url (around emacspeak pre act)
  (if (interactive-p)
    (let ((emacspeak-speak-messages t)) ad-do-it)
    (let ((emacspeak-speak-messages nil)) ad-do-it))
    ad-return-value)


;;*****************************************************************************
;;* The Emacs redefine: *******************************************************
;;*****************************************************************************

;;* (dired-aux.el @ 171 @) Print %s with: -> Imprimir %s no dispositivo: *******
(defun dired-do-print (&optional arg)
  "Print the marked (or next ARG) files.
```

```
Uses the shell command coming from variables ‘lpr-command’ and
‘lpr-switches’ as default."
  (interactive "P")
  (let* ((file-list (dired-get-marked-files t arg))
         (command (dired-mark-read-string
                   "Imprimir %s no dispositivo: "
                    (mapconcat 'identity
                               (cons lpr-command
                                     (if (stringp lpr-switches)
                                         (list lpr-switches)
                                       lpr-switches))
                               " ")
                   'print arg file-list)))
    (dired-run-shell-command (dired-shell-stuff-it command file-list nil))))

;;* (dired-aux.el @ 313 @) ! on -> Executar com ********************************
(defun dired-do-shell-command (command &optional arg file-list)
  "Run a shell command COMMAND on the marked files.
If no files are marked or a specific numeric prefix arg is given,
the next ARG files are used.  Just \\[universal-argument] means the current file.
The prompt mentions the file(s) or the marker, as appropriate.

If there is output, it goes to a separate buffer.

Normally the command is run on each file individually.
However, if there is a ‘*’ in the command then it is run
just once with the entire file list substituted there.

If there is no ‘*’, but a ‘?’ in the command then it is still run
on each file individually but with the filename substituted there
instead of at the end of the command.

No automatic redisplay of dired buffers is attempted, as there’s no
telling what files the command may have changed.  Type
\\[dired-do-redisplay] to redisplay the marked files.

The shell command has the top level directory as working directory, so
output files usually are created there instead of in a subdir.

In a noninteractive call (from Lisp code), you must specify
the list of file names explicitly with the FILE-LIST argument."
;;Functions dired-run-shell-command and dired-shell-stuff-it do the
;;actual work and can be redefined for customization.
  (interactive
   (let ((files (dired-get-marked-files t current-prefix-arg)))
     (list
      ;; Want to give feedback whether this file or marked files are used:
      (dired-read-shell-command (concat "Executar com "
                                        "%s: ")
                                current-prefix-arg
                                files)
      current-prefix-arg
      files)))
```

```
  (let* ((on-each (not (string-match "\\*" command))))
    (if on-each
        (dired-bunch-files
         (- 10000 (length command))
         (function (lambda (&rest files)
                     (dired-run-shell-command
                      (dired-shell-stuff-it command files t arg))))
         nil
         file-list)
      ;; execute the shell command
      (dired-run-shell-command
       (dired-shell-stuff-it command file-list nil arg)))))

;;* (dired-aux.el @ 598 @) Compress or uncompress -> Compactar ou descompactar **
(defun dired-mark-confirm (op-symbol arg)
  ;; Request confirmation from the user that the operation described
  ;; by OP-SYMBOL is to be performed on the marked files.
  ;; Confirmation consists in a y-or-n question with a file list
  ;; pop-up unless OP-SYMBOL is a member of `dired-no-confirm'.
  ;; The files used are determined by ARG (as in dired-get-marked-files).
  (or (eq dired-no-confirm t)
      (memq op-symbol dired-no-confirm)
      (let ((files (dired-get-marked-files t arg))
            (string (if (eq op-symbol 'compress) "Compactar ou descompactar"
                      (capitalize (symbol-name op-symbol)))))
        (dired-mark-pop-up nil op-symbol files (function y-or-n-p)
                           (concat string " "
                                   (dired-mark-prompt arg files) "? "))))))

;;* (dired-aux.el @ 1190 @) to: -> para: ************************************
(defun dired-do-create-files (op-symbol file-creator operation arg
                                        &optional marker-char op1
                                        how-to)
  "Create a new file for each marked file.
Prompts user for target, which is a directory in which to create
  the new files.  Target may be a plain file if only one marked
  file exists.  The way the default for the target directory is
  computed depends on the value of `dired-dwim-target-directory'.
OP-SYMBOL is the symbol for the operation.  Function `dired-mark-pop-up'
  will determine whether pop-ups are appropriate for this OP-SYMBOL.
FILE-CREATOR and OPERATION as in `dired-create-files'.
ARG as in `dired-get-marked-files'.
Optional arg MARKER-CHAR as in `dired-create-files'.
Optional arg OP1 is an alternate form for OPERATION if there is
  only one file.
Optional arg HOW-TO is used to set the value of the into-dir variable
  which determines how to treat target.
  If into-dir is set to nil then target is not regarded as a directory,
    there must be exactly one marked file, else error.
  Else if into-dir is set to a list, then target is a generalized
    directory (e.g. some sort of archive).  The first element of into-dir
    must be a function with at least four arguments:
      operation as OPERATION above.
```

```
    rfn-list a list of the relative names for the marked files.
    fn-list a list of the absolute names for the marked files.
    target.
    The rest of into-dir are optional arguments.
Else into-dir is not a list.  Target is a directory.
  The marked file(s) are created inside the target directory.

If HOW-TO is not given (or nil), then into-dir is set to true if
  target is a directory and otherwise to nil.
Else if HOW-TO is t, then into-dir is set to nil.
Else HOW-TO is assumed to be a function of one argument, target,
  that looks at target and returns a value for the into-dir
  variable.  The function 'dired-into-dir-with-symlinks' is provided
  for the case (common when creating symlinks) that symbolic
  links to directories are not to be considered as directories
  (as 'file-directory-p' would if HOW-TO had been nil)."
(or op1 (setq op1 operation))
(let* ((fn-list (dired-get-marked-files nil arg))
       (rfn-list (mapcar (function dired-make-relative) fn-list))
       (dired-one-file        ; fluid variable inside dired-create-files
        (and (consp fn-list) (null (cdr fn-list)) (car fn-list)))
       (target-dir (dired-dwim-target-directory))
       (default (and dired-one-file
                     (expand-file-name (file-name-nondirectory (car fn-list))
                                       target-dir)))
       (target (expand-file-name ; fluid variable inside dired-create-files
                 (dired-mark-read-file-name
                  (concat (if dired-one-file op1 operation) " %s para: ")
                  target-dir op-symbol arg rfn-list)))
       (into-dir (cond ((null how-to)
                        ;; Allow DOS/Windows users to change the letter
                        ;; case of a directory.  If we don't test these
                        ;; conditions up front, file-directory-p below
                        ;; will return t because the filesystem is
                        ;; case-insensitive, and Emacs will try to move
                        ;; foo -> foo/foo, which fails.
                        (if (and (memq system-type '(ms-dos windows-nt))
                                 (eq op-symbol 'move)
                                 dired-one-file
                                 (string= (downcase
                                           (expand-file-name (car fn-list)))
                                          (downcase
                                           (expand-file-name target)))
                                 (not (string=
                                       (file-name-nondirectory (car fn-list))
                                       (file-name-nondirectory target))))
                            nil
                          (file-directory-p target)))
                       ((eq how-to t) nil)
                       (t (funcall how-to target)))))
  (if (and (consp into-dir) (functionp (car into-dir)))
      (apply (car into-dir) operation rfn-list fn-list target (cdr into-dir))
    (if (not (or dired-one-file into-dir))
```

```
          (error "Marked %s: target must be a directory: %s" operation target))
      ;; rename-file bombs when moving directories unless we do this:
      (or into-dir (setq target (directory-file-name target)))
      (dired-create-files
       file-creator operation fn-list
       (if into-dir                       ; target is a directory
           ;; This function uses fluid variable target when called
           ;; inside dired-create-files:
           (function
            (lambda (from)
              (expand-file-name (file-name-nondirectory from) target)))
         (function (lambda (from) target)))
       marker-char))))

;;* (dired-aux.el @ 1313 @) Create directory: -> Criar diret rio: **************
(defun dired-create-directory (directory)
  "Create a directory called DIRECTORY."
  (interactive
   (list (read-file-name "Criar diret rio: " (dired-current-directory))))
  (let ((expanded (directory-file-name (expand-file-name directory))))
    (make-directory expanded)
    (dired-add-file expanded)
    (dired-move-to-filename)))

;;* (dired-aux.el @ 1343 @) Copy [-p] -> Copiar ********************************
(defun dired-do-copy (&optional arg)
  "Copy all marked (or next ARG) files, or copy the current file.
This normally preserves the last-modified date when copying.
When operating on just the current file, you specify the new name.
When operating on multiple or marked files, you specify a directory,
and new copies of these files are made in that directory
with the same names that the files currently have.  The default
suggested for the target directory depends on the value of
‘dired-dwim-target’, which see."
  (interactive "P")
  (let ((dired-recursive-copies dired-recursive-copies))
    (dired-do-create-files 'copy (function dired-copy-file)
                           "Copiado"
                           arg dired-keep-marker-copy
                           "Copiar")))

;;* (dired-aux.el @ 1386 @) Rename -> Renomear **********************************
(defun dired-do-rename (&optional arg)
  "Rename current file or all marked (or next ARG) files.
When renaming just the current file, you specify the new name.
When renaming multiple or marked files, you specify a directory.
The default suggested for the target directory depends on the value
of ‘dired-dwim-target’, which see."
  (interactive "P")
  (dired-do-create-files 'move (function dired-rename-file)
                         "Renomeado" arg dired-keep-marker-rename "Renomear"))

;;* (dired.el @ 597 @) Reading directory -> Lendo diret rio ********************
```

```
(defun dired-readin (dir-or-list buffer)
  ;; default-directory and dired-actual-switches must be buffer-local
  ;; and initialized by now.
  ;; Thus we can test (equal default-directory dirname) instead of
  ;; (file-directory-p dirname) and save a filesystem transaction.
  ;; Also, we can run this hook which may want to modify the switches
  ;; based on default-directory, e.g. with ange-ftp to a SysV host
  ;; where ls won't understand -Al switches.
  (let (dirname
        (indent-tabs-mode nil))
    (if (consp dir-or-list)
        (setq dirname (car dir-or-list))
      (setq dirname dir-or-list))
    (setq dirname (expand-file-name dirname))
    (if (consp dir-or-list)
        (setq dir-or-list (cons dirname (cdr dir-or-list))))
    (run-hooks 'dired-before-readin-hook)
    (save-excursion
      (message "Lendo diret rio %s" dirname)
      (set-buffer buffer)
      (let (buffer-read-only (failed t))
        (widen)
        (erase-buffer)
        (dired-readin-insert dir-or-list)
        (indent-rigidly (point-min) (point-max) 2)
        ;; We need this to make the root dir have a header line as all
        ;; other subdirs have:
        (goto-char (point-min))
        (if (not (looking-at "^  /.*:$"))
            (dired-insert-headerline default-directory))
        ;; can't run dired-after-readin-hook here, it may depend on the subdir
        ;; alist to be OK.
        )
      (message "Lendo diret rio %s" dirname)
      ;; Must first make alist buffer local and set it to nil because
      ;; dired-build-subdir-alist will call dired-clear-alist first
      (set (make-local-variable 'dired-subdir-alist) nil)
      (dired-build-subdir-alist)
      (let ((attributes (file-attributes dirname)))
        (if (eq (car attributes) t)
            (set-visited-file-modtime (nth 5 attributes))))
      (set-buffer-modified-p nil))))

;;* (dired.el @ 2038 @) Delete -> Deletar ************************************
(defun dired-internal-do-deletions (l arg)
  ;; L is an alist of files to delete, with their buffer positions.
  ;; ARG is the prefix arg.
  ;; Filenames are absolute (VMS needs this for logical search paths).
  ;; (car L) *must* be the *last* (bottommost) file in the dired buffer.
  ;; That way as changes are made in the buffer they do not shift the
  ;; lines still to be changed, so the (point) values in L stay valid.
  ;; Also, for subdirs in natural order, a subdir's files are deleted
  ;; before the subdir itself - the other way around would not work.
```

```
  (let ((files (mapcar (function car) l))
        (count (length l))
        (succ 0))
    ;; canonicalize file list for pop up
    (setq files (nreverse (mapcar (function dired-make-relative) files)))
    (if (dired-mark-pop-up
         " *Deletions*" 'delete files dired-deletion-confirmer
         (format "Deletar %s " (dired-mark-prompt arg files)))
        (save-excursion
          (let (failures);; files better be in reverse order for this loop!
            (while l
              (goto-char (cdr (car l)))
              (let (buffer-read-only)
                (condition-case err
                    (let ((fn (car (car l))))
                      (dired-delete-file fn dired-recursive-deletes)
                      ;; if we get here, removing worked
                      (setq succ (1+ succ))
                      (message "%s de %s dele  es" succ count)
                      (delete-region (progn (beginning-of-line) (point))
                                     (progn (forward-line 1) (point)))
                      (dired-clean-up-after-deletion fn))
                  (error;; catch errors from failed deletions
                   (dired-log "%s\n" err)
                   (setq failures (cons (car (car l)) failures)))))
              (setq l (cdr l)))
            (if (not failures)
                (message "Dele  o conclu da")
              (dired-log-summary
               (format "%d de %d dele  es n o conclu das"
                       (length failures) count)
               failures))))
      (message "(Dele  o n o concu da)")))
  (dired-move-to-filename))

;;* (files.el @ 727 @) Find file: -> Abrir arquivo: ****************************
(defun find-file (filename &optional wildcards)
  "Edit file FILENAME.
Switch to a buffer visiting file FILENAME,
creating one if none already exists.
Interactively, or if WILDCARDS is non-nil in a call from Lisp,
expand wildcards (if any) and visit multiple files.  Wildcard expansion
can be suppressed by setting `find-file-wildcards'."
  (interactive "FAbrir arquivo: \np")
  (let ((value (find-file-noselect filename nil nil wildcards)))
    (if (listp value)
        (mapcar 'switch-to-buffer (nreverse value))
      (switch-to-buffer value))))

;;* (files.el @ 817 @) Buffer %s modified; kill anyway? -> A janela %s foi
;;    modificada; Mesmo assim deseja encerrar? ********************************
(defun find-alternate-file (filename)
  "Find file FILENAME, select its buffer, kill previous buffer.
```

```
If the current buffer now contains an empty file that you just visited
\(presumably by mistake), use this command to visit the file you really want."
  (interactive
   (let ((file buffer-file-name)
         (file-name nil)
         (file-dir nil))
     (and file
          (setq file-name (file-name-nondirectory file)
                file-dir (file-name-directory file)))
     (list (read-file-name
            "Alternar arquivo: " file-dir nil nil file-name))))
  (and (buffer-modified-p) (buffer-file-name)
       ;; (not buffer-read-only)
       (not (yes-or-no-p (format "A janela %s foi modificada; Mesmo assim deseja
           encerrar? "
                                 (buffer-name))))
       (error "Cancelado"))
  (let ((obuf (current-buffer))
        (ofile buffer-file-name)
        (onum buffer-file-number)
        (otrue buffer-file-truename)
        (oname (buffer-name)))
    (if (get-buffer " **lose**")
        (kill-buffer " **lose**"))
    (rename-buffer " **lose**")
    (unwind-protect
        (progn
          (unlock-buffer)
          (setq buffer-file-name nil)
          (setq buffer-file-number nil)
          (setq buffer-file-truename nil)
          (find-file filename))
      (cond ((eq obuf (current-buffer))
             (setq buffer-file-name ofile)
             (setq buffer-file-number onum)
             (setq buffer-file-truename otrue)
             (lock-buffer)
             (rename-buffer oname))))
    (or (eq (current-buffer) obuf)
        (kill-buffer obuf))))

;;* (files.el @ 1249 @) Note: file is write protected -> Nota: arquivo est
;;    protegido contra grava  o ************************************************
(defun after-find-file (&optional error warn noauto
                                  after-find-file-from-revert-buffer
                                  nomodes)
  "Called after finding a file and by the default revert function.
Sets buffer mode, parses local variables.
Optional args ERROR, WARN, and NOAUTO: ERROR non-nil means there was an
error in reading the file.  WARN non-nil means warn if there
exists an auto-save file more recent than the visited file.
NOAUTO means don't mess with auto-save mode.
Fourth arg AFTER-FIND-FILE-FROM-REVERT-BUFFER non-nil
```

```
 means this call was from ‘revert-buffer’.
Fifth arg NOMODES non-nil means don’t alter the file’s modes.
Finishes by calling the functions in ‘find-file-hooks’
unless NOMODES is non-nil."
  (setq buffer-read-only (not (file-writable-p buffer-file-name)))
  (if noninteractive
      nil
    (let* (not-serious
           (msg
            (cond
             ((not warn) nil)
             ((and error (file-attributes buffer-file-name))
              (setq buffer-read-only t)
              "Arquivo existe, mas n o pode ser lido")
             ((not buffer-read-only)
              (if (and warn
                       ;; No need to warn if buffer is auto-saved
                       ;; under the name of the visited file.
                       (not (and buffer-file-name
                                 auto-save-visited-file-name))
                       (file-newer-than-file-p (or buffer-auto-save-file-name
                                                   (make-auto-save-file-name))
                                               buffer-file-name))
                  (format "%s possui dados automaticamente salvos; Para recuperar
                      M-x recover-file"
                          (file-name-nondirectory buffer-file-name))
                (setq not-serious t)
                (if error "(Arquivo novo)" nil)))
             ((not error)
              (setq not-serious t)
              "Nota: arquivo est  protegido contra grava  o")
             ((file-attributes (directory-file-name default-directory))
              "Arquivo n o encontrado e diret rio est  protegido contra grava  o")
             ((file-exists-p (file-name-directory buffer-file-name))
              (setq buffer-read-only nil))
             (t
              (setq buffer-read-only nil)
              (if (file-exists-p (file-name-directory (directory-file-name
                  (file-name-directory buffer-file-name))))
                  "Use M-x make-directory Enter Enter para criar o diret rio"
                "Use C-u M-x make-directory Enter Enter para criar diret rio e
                    subdiret rios")))))
      (when msg
        (message msg)
        (or not-serious (sit-for 1 nil t))))
    (when (and auto-save-default (not noauto))
      (auto-save-mode t)))
  ;; Make people do a little extra work (C-x C-q)
  ;; before altering a backup file.
  (when (backup-file-name-p buffer-file-name)
    (setq buffer-read-only t))
  (unless nomodes
    (when (and view-read-only view-mode)
```

```
      (view-mode-disable))
    (normal-mode t)
    (when (and buffer-read-only
               view-read-only
               (not (eq (get major-mode 'mode-class) 'special)))
      (view-mode-enter))
    (run-hooks 'find-file-hooks)))

;;* (files.el @ 2102 @) Write file: -> Salvar arquivo como: ********************
(defun write-file (filename &optional confirm)
  "Write current buffer into file FILENAME.
This makes the buffer visit that file, and marks it as not modified.

If you specify just a directory name as FILENAME, that means to use
the default file name but in that directory.  You can also yank
the default file name into the minibuffer to edit it, using M-n.

If the buffer is not already visiting a file, the default file name
for the output file is the buffer name.

If optional second arg CONFIRM is non-nil, this function
asks for confirmation before overwriting an existing file.
Interactively, confirmation is required unless you supply a prefix argument."
;;  (interactive "FWrite file: ")
  (interactive
   (list (if buffer-file-name
             (read-file-name "Salvar arquivo como: "
                             nil nil nil nil)
           (read-file-name "Salvar arquivo como: "
                           nil nil))
         (not current-prefix-arg)))
  (or (null filename) (string-equal filename "")
      (progn
        ;; If arg is just a directory,
        ;; use the default file name, but in that directory.
        (if (file-directory-p filename)
            (setq filename (concat (file-name-as-directory filename)
                                   (file-name-nondirectory
                                    (or buffer-file-name (buffer-name))))))
        (and confirm
             (file-exists-p filename)
             (or (y-or-n-p (format "J  existe um arquivo chamado %s; deseja
                 substitu -lo? " filename))
                 (error "Cancelado")))
        (set-visited-file-name filename (not confirm))))
  (set-buffer-modified-p t)
  ;; Make buffer writable if file is writable.
  (and buffer-file-name
       (file-writable-p buffer-file-name)
       (setq buffer-read-only nil))
  (save-buffer) (message "Arquivo salvo %s" (buffer-file-name)))

;;* (files.el @ 2609 @) File to save: -> Salvar arquivo: ***********************
```

```
(defun basic-save-buffer ()
  "Save the current buffer in its visited file, if it has been modified.
The hooks ‘write-contents-hooks’, ‘local-write-file-hooks’ and
‘write-file-hooks’ get a chance to do the job of saving; if they do not,
then the buffer is saved in the visited file file in the usual way.
After saving the buffer, this function runs ‘after-save-hook’."
  (interactive)
  (save-current-buffer
    ;; In an indirect buffer, save its base buffer instead.
    (if (buffer-base-buffer)
        (set-buffer (buffer-base-buffer)))
    (if (buffer-modified-p)
        (let ((recent-save (recent-auto-save-p))
              setmodes tempsetmodes)
          ;; On VMS, rename file and buffer to get rid of version number.
          (if (and (eq system-type 'vax-vms)
                   (not (string= buffer-file-name
                                 (file-name-sans-versions buffer-file-name))))
              (let (buffer-new-name)
                ;; Strip VMS version number before save.
                (setq buffer-file-name
                      (file-name-sans-versions buffer-file-name))
                ;; Construct a (unique) buffer name to correspond.
                (let ((buf (create-file-buffer (downcase buffer-file-name))))
                  (setq buffer-new-name (buffer-name buf))
                  (kill-buffer buf))
                (rename-buffer buffer-new-name)))
          ;; If buffer has no file name, ask user for one.
          (or buffer-file-name
              (let ((filename
                     (expand-file-name
                      (read-file-name "Salvar arquivo: ") nil)))
                (and (file-exists-p filename)
                     (or (y-or-n-p (format "J  existe um arquivo chamado %s;
                         deseja substitu -lo? "
                                           filename))
                         (error "Cancelado")))
                (set-visited-file-name filename)))
          (or (verify-visited-file-modtime (current-buffer))
              (not (file-exists-p buffer-file-name))
              (yes-or-no-p
               (format "Deseja salvar altera  es no documento %s? "
                       (file-name-nondirectory buffer-file-name)))
              (error "Salvar n o confirmado"))
          (save-restriction
            (widen)
            (save-excursion
              (and (> (point-max) 1)
                   (not find-file-literally)
                   (/= (char-after (1- (point-max))) ?\n)
                   (not (and (eq selective-display t)
                             (= (char-after (1- (point-max))) ?\r)))
                   (or (eq require-final-newline t)
```

```
                    (and require-final-newline
                         (y-or-n-p
                          (format "Arquivo %s n o termina em nova linha.  Deseja
                              adicionar uma? "
                                  (buffer-name)))))
                (save-excursion
                  (goto-char (point-max))
                  (insert ?\n))))
          ;; Support VC version backups.
          (vc-before-save)
          (or (run-hook-with-args-until-success 'write-contents-hooks)
              (run-hook-with-args-until-success 'local-write-file-hooks)
              (run-hook-with-args-until-success 'write-file-hooks)
              ;; If a hook returned t, file is already "written".
              ;; Otherwise, write it the usual way now.
              (setq setmodes (basic-save-buffer-1)))
          ;; Now we have saved the current buffer.  Let's make sure
          ;; that buffer-file-coding-system is fixed to what
          ;; actually used for saving by binding it locally.
          (if save-buffer-coding-system
              (setq save-buffer-coding-system last-coding-system-used)
            (setq buffer-file-coding-system last-coding-system-used))
          (setq buffer-file-number
                (nthcdr 10 (file-attributes buffer-file-name)))
          (if setmodes
              (condition-case ()
                  (set-file-modes buffer-file-name (car setmodes))
                (error nil))))
        ;; If the auto-save file was recent before this command,
        ;; delete it now.
        (delete-auto-save-file-if-necessary recent-save)
        ;; Support VC `implicit' locking.
        (vc-after-save)
        (run-hooks 'after-save-hook))
      (message "(Nenhuma altera  o precisa ser salva)")))

;;* (files.el @ 2789 @) Save file: -> Salvar arquivo: *************************
(defun save-some-buffers (&optional arg pred)
  "Save some modified file-visiting buffers.  Asks user about each one.
Optional argument (the prefix) non-nil means save all with no questions.
Optional second argument PRED determines which buffers are considered:
If PRED is nil, all the file-visiting buffers are considered.
If PRED is t, then certain non-file buffers will also be considered.
If PRED is a zero-argument function, it indicates for each buffer whether
to consider it or not when called with that buffer current."
  (interactive "P")
  (save-window-excursion
    (let* ((queried nil)
           (files-done
            (map-y-or-n-p
             (function
              (lambda (buffer)
                (and (buffer-modified-p buffer)
```

```
                    (not (buffer-base-buffer buffer))
                    (or
                     (buffer-file-name buffer)
                     (and pred
                          (progn
                            (set-buffer buffer)
                            (and buffer-offer-save (> (buffer-size) 0)))))
                    (or (not (functionp pred))
                        (with-current-buffer buffer (funcall pred)))
                    (if arg
                        t
                      (setq queried t)
                      (if (buffer-file-name buffer)
                          (format "Salvar arquivo %s? "
                                  (buffer-file-name buffer))
                        (format "Salvar janela %s? "
                                (buffer-name buffer)))))))
             (function
              (lambda (buffer)
                (set-buffer buffer)
                (save-buffer)))
             (buffer-list)
             '("buffer" "buffers" "save")
             (list (list ?\C-r (lambda (buf)
                                 (view-buffer buf
                                              (function
                                               (lambda (ignore)
                                                 (exit-recursive-edit))))
                                 (recursive-edit)
                                 ;; Return nil to ask about BUF again.
                                 nil)
                         "display the current buffer"))))
           (abbrevs-done
            (and save-abbrevs abbrevs-changed
                 (progn
                   (if (or arg
                           (eq save-abbrevs 'silently)
                           (y-or-n-p (format "Save abbrevs in %s? "
                                             abbrev-file-name)))
                       (write-abbrev-file nil))
                   ;; Don't keep bothering user if he says no.
                   (setq abbrevs-changed nil)
                   t))))
       (or queried (> files-done 0) abbrevs-done
           (message "(Nenhum arquivo precisa ser salvo)")))))

;;* (files.el @ 3741 @) Kill Emacs: -> Encerrar Emacs: ************************
(defun save-buffers-kill-emacs (&optional buffer &optional arg)
  "Offer to save each buffer, then kill this Emacs process.
With prefix arg, silently save all file-visiting buffers, then kill."
  (interactive "P") (if (not (active-minibuffer-window)) (list (read-buffer
      "Pressione Enter, para confirmar o encerramento do Ambiente Emacs. Para
      cancelar pressione Escape." nil)) (abort-recursive-edit))
```

```
  (save-some-buffers arg t)
  (and (or (not (memq t (mapcar (function
                                 (lambda (buf) (and (buffer-file-name buf)
                                                    (buffer-modified-p buf))))
                                (buffer-list))))
           (y-or-n-p "Existem janelas modificadas; Mesmo assim deseja sair? "))
       (or (not (fboundp 'process-list))
           ;; process-list is not defined on VMS.
           (let ((processes (process-list))
                 active)
             (while processes
               (and (memq (process-status (car processes)) '(run stop open))
                    (let ((val (process-kill-without-query (car processes))))
                      (process-kill-without-query (car processes) val)
                      val)
                    (setq active t))
               (setq processes (cdr processes)))
             (or (not active)
                 (list-processes)
                 (y-or-n-p "Existem processos ativos; Mesmo assim deseja
                     encerr -los e sair? "))))
       ;; Query the user for other things, perhaps.
       (run-hook-with-args-until-failure 'kill-emacs-query-functions)
       (or (null confirm-kill-emacs)
           (funcall confirm-kill-emacs "Deseja realmente encerrar o Emacs? "))
       (progn (if (equal window-system nil) (shell-command (format "sudo loadkeys
           %s" (expand-file-name "console_restore.kmap" emacspeak-etc-directory))))
           (kill-emacs))))

;;* (isearch.el @ 1534 @) I-search -> Localizar ********************************
(defun isearch-message-prefix (&optional c-q-hack ellipsis nonincremental)
  ;; If about to search, and previous search regexp was invalid,
  ;; check that it still is.  If it is valid now,
  ;; let the message we display while searching say that it is valid.
  (and isearch-invalid-regexp ellipsis
       (condition-case ()
           (progn (re-search-forward isearch-string (point) t)
                  (setq isearch-invalid-regexp nil
                        isearch-within-brackets nil))
         (error nil)))
  ;; If currently failing, display no ellipsis.
  (or isearch-success (setq ellipsis nil))
  (let ((m (concat (if isearch-success "" "(n o sucedido) ")
                   (if (and isearch-wrapped
                            (if isearch-forward
                                (> (point) isearch-opoint)
                              (< (point) isearch-opoint)))
                       "over")
                   (if isearch-wrapped "wrapped ")
                   (if isearch-word "word " "")
                   (if isearch-regexp "regexp " "")
                   (if nonincremental "Localizar" "Localizar")
                   (if isearch-forward "" " atr s")
```

```
                  (if current-input-method
                      (concat " [" current-input-method-title "]: ")
                    ": ")
                  )))
    (concat (upcase (substring m 0 1)) (substring m 1))))

;;* (map-ynp.el @ 38 @) (y, n, !, ...) -> (s ou n) ****************************
(defun map-y-or-n-p (prompter actor list &optional help action-alist
                              no-cursor-in-echo-area)
  "Ask a series of boolean questions.
Takes args PROMPTER ACTOR LIST, and optional args HELP and ACTION-ALIST.

LIST is a list of objects, or a function of no arguments to return the next
object or nil.

If PROMPTER is a string, the prompt is \(format PROMPTER OBJECT\).  If not
a string, PROMPTER is a function of one arg (an object from LIST), which
returns a string to be used as the prompt for that object.  If the return
value is not a string, it may be nil to ignore the object or non-nil to act
on the object without asking the user.

ACTOR is a function of one arg (an object from LIST),
which gets called with each object that the user answers ‘yes’ for.

If HELP is given, it is a list (OBJECT OBJECTS ACTION),
where OBJECT is a string giving the singular noun for an elt of LIST;
OBJECTS is the plural noun for elts of LIST, and ACTION is a transitive
verb describing ACTOR.  The default is \(\"object\" \"objects\" \"act on\"\).

At the prompts, the user may enter y, Y, or SPC to act on that object;
n, N, or DEL to skip that object; ! to act on all following objects;
ESC or q to exit (skip all following objects); . (period) to act on the
current object and then exit; or \\[help-command] to get help.

If ACTION-ALIST is given, it is an alist (KEY FUNCTION HELP) of extra keys
that will be accepted.  KEY is a character; FUNCTION is a function of one
arg (an object from LIST); HELP is a string.  When the user hits KEY,
FUNCTION is called.  If it returns non-nil, the object is considered
\"acted upon\", and the next object from LIST is processed.  If it returns
nil, the prompt is repeated for the same object.

Final optional argument NO-CURSOR-IN-ECHO-AREA non-nil says not to set
‘cursor-in-echo-area’ while prompting.

This function uses ‘query-replace-map’ to define the standard responses,
but not all of the responses which ‘query-replace’ understands
are meaningful here.

Returns the number of actions taken."
  (let* ((actions 0)
         user-keys mouse-event map prompt char elt tail def
         ;; Non-nil means we should use mouse menus to ask.
         use-menus
```

```
      delayed-switch-frame
      (next (if (or (and list (symbolp list))
                    (subrp list)
                    (byte-code-function-p list)
                    (and (consp list)
                         (eq (car list) 'lambda)))
                (function (lambda ()
                            (setq elt (funcall list))))
              (function (lambda ()
                          (if list
                              (progn
                                (setq elt (car list)
                                      list (cdr list))
                                t)
                            nil))))))
(if (and (listp last-nonmenu-event)
         use-dialog-box)
    ;; Make a list describing a dialog box.
    (let ((object (if help (capitalize (nth 0 help))))
          (objects (if help (capitalize (nth 1 help))))
          (action (if help (capitalize (nth 2 help)))))
      (setq map `(("Yes" . act) ("No" . skip) ("Quit" . exit)
                  (,(if help (concat action " " object " And Quit")
                      "Do it and Quit") . act-and-exit)
                  (,(if help (concat action " All " objects)
                      "Do All") . automatic)
                  ,@(mapcar (lambda (elt)
                              (cons (capitalize (nth 2 elt))
                                    (vector (nth 1 elt))))
                            action-alist))
            use-menus t
            mouse-event last-nonmenu-event))
  (setq user-keys (if action-alist
                      (concat (mapconcat (function
                                          (lambda (elt)
                                            (key-description
                                             (char-to-string (car elt)))))
                                         action-alist ", ")
                              " ")
                    ""))
        ;; Make a map that defines each user key as a vector containing
        ;; its definition.
        map (cons 'keymap
                  (append (mapcar (lambda (elt)
                                    (cons (car elt) (vector (nth 1 elt))))
                                  action-alist)
                          query-replace-map))))
(unwind-protect
    (progn
      (if (stringp prompter)
          (setq prompter `(lambda (object)
                            (format ,prompter object))))
      (while (funcall next)
```

```
(setq prompt (funcall prompter elt))
(cond ((stringp prompt)
       ;; Prompt the user about this object.
       (setq quit-flag nil)
       (if use-menus
           (setq def (or (x-popup-dialog (or mouse-event use-menus)
                                         (cons prompt map))
                         'quit))
         ;; Prompt in the echo area.
         (let ((cursor-in-echo-area (not no-cursor-in-echo-area))
               (message-log-max nil))
           (message "%s (s ou n) "
                    prompt)
           (if minibuffer-auto-raise
               (raise-frame (window-frame (minibuffer-window))))
           (while (progn
                    (setq char (read-event))
                    ;; If we get -1, from end of keyboard
                    ;; macro, try again.
                    (equal char -1)))
           ;; Show the answer to the question.
           (message "%s (s ou n)"
                    prompt))
         (setq def (lookup-key map (vector char))))
       (cond ((eq def 'exit)
              (setq next (function (lambda () nil))))
             ((eq def 'act)
              ;; Act on the object.
              (funcall actor elt)
              (setq actions (1+ actions)))
             ((eq def 'skip)
              ;; Skip the object.
              )
             ((eq def 'act-and-exit)
              ;; Act on the object and then exit.
              (funcall actor elt)
              (setq actions (1+ actions)
                    next (function (lambda () nil))))
             ((or (eq def 'quit) (eq def 'exit-prefix))
              (setq quit-flag t)
              (setq next `(lambda ()
                            (setq next ',next)
                            ',elt)))
             ((eq def 'automatic)
              ;; Act on this and all following objects.
              (if (funcall prompter elt)
                  (progn
                    (funcall actor elt)
                    (setq actions (1+ actions))))
              (while (funcall next)
                (if (funcall prompter elt)
                    (progn
                      (funcall actor elt)
```

```
                                (setq actions (1+ actions))))))
                           ((eq def 'help)
                            (with-output-to-temp-buffer "*Help*"
                              (princ
                               (let ((object (if help (nth 0 help) "object"))
                                     (objects (if help (nth 1 help) "objects"))
                                     (action (if help (nth 2 help) "act on")))
                                 (concat
                                  (format "Type SPC or `y' to %s the current %s;
DEL or `n' to skip the current %s;
RET or `q' to exit (skip all remaining %s);
! to %s all remaining %s;
ESC or `q' to exit;\n"
                                          action object object objects action
                                          objects)
                                  (mapconcat (function
                                              (lambda (elt)
                                                (format "%c to %s"
                                                        (nth 0 elt)
                                                        (nth 2 elt))))
                                             action-alist
                                             ";\n")
                                  (if action-alist ";\n")
                                  (format "or . (period) to %s \
the current %s and exit."
                                          action object))))
                              (save-excursion
                                (set-buffer standard-output)
                                (help-mode)))
                            (setq next `(lambda ()
                                          (setq next ',next)
                                          ',elt)))
                           ((vectorp def)
                            ;; A user-defined key.
                            (if (funcall (aref def 0) elt) ;Call its function.
                                ;; The function has eaten this object.
                                (setq actions (1+ actions))
                              ;; Regurgitated; try again.
                              (setq next `(lambda ()
                                            (setq next ',next)
                                            ',elt))))
                           ((and (consp char)
                                 (eq (car char) 'switch-frame))
                            ;; switch-frame event.  Put it off until we're done.
                            (setq delayed-switch-frame char)
                            (setq next `(lambda ()
                                          (setq next ',next)
                                          ',elt)))
                           (t
                            ;; Random char.
                            (message "Type %s for help."
                                     (key-description (vector help-char)))
                            (beep)
```

```
                        (sit-for 1)
                        (setq next `(lambda ()
                                      (setq next ',next)
                                      ',elt)))))
                 (prompt
                  (funcall actor elt)
                  (setq actions (1+ actions))))))
      (if delayed-switch-frame
          (setq unread-command-events
                (cons delayed-switch-frame unread-command-events))))
    ;; Clear the last prompt from the minibuffer.
    (let ((message-log-max nil))
      (message ""))
    ;; Return the number of actions that were taken.
    actions))

;;* (replace.el @ 800 @) (y == act | Y == act) -> (s == act | S == act) ********
(define-key query-replace-map "s" 'act)
(define-key query-replace-map "S" 'act)

;;* (simple.el @ 831 @) Beginning of history -> Come o do hist rico ************
(defun next-history-element (n)
  "Insert the next element of the minibuffer history into the minibuffer."
  (interactive "p")
  (or (zerop n)
      (let ((narg (- minibuffer-history-position n))
            (minimum (if minibuffer-default -1 0))
            elt minibuffer-returned-to-present)
        (if (and (zerop minibuffer-history-position)
                 (null minibuffer-text-before-history))
            (setq minibuffer-text-before-history (field-string (point-max))))
        (if (< narg minimum)
            (if minibuffer-default
                (error "Fim do hist rico")
              (error "Fim do hist rico")))
        (if (> narg (length (symbol-value minibuffer-history-variable)))
            (error "Come o do hist rico"))
        (unless (or (eq last-command 'next-history-element)
                    (eq last-command 'previous-history-element))
          (let ((prompt-end (field-beginning (point-max))))
            (set (make-local-variable 'minibuffer-temporary-goal-position)
                 (cond ((<= (point) prompt-end) prompt-end)
                       ((eobp) nil)
                       (t (point))))))
        (goto-char (point-max))
        (delete-field)
        (setq minibuffer-history-position narg)
        (cond ((= narg -1)
               (setq elt minibuffer-default))
              ((= narg 0)
               (setq elt (or minibuffer-text-before-history ""))
               (setq minibuffer-returned-to-present t)
               (setq minibuffer-text-before-history nil))
```

```
             (t (setq elt (nth (1- minibuffer-history-position)
                               (symbol-value minibuffer-history-variable)))))
        (insert
         (if (and (eq minibuffer-history-sexp-flag (minibuffer-depth))
                  (not minibuffer-returned-to-present))
             (let ((print-level nil))
               (prin1-to-string elt))
           elt))
        (goto-char (or minibuffer-temporary-goal-position (point-max))))))

;;* (simple.el @ 934 @) Line -> Linha ****************************************
(defun what-line ()
  "Print the current buffer line number and narrowed line number of point."
  (interactive)
  (let ((opoint (point)) start)
    (save-excursion
      (save-restriction
        (goto-char (point-min))
        (widen)
        (forward-line 0)
        (setq start (point))
        (goto-char opoint)
        (forward-line 0)
        (if (/= start 1)
            (message "Linha %d (linha reduzida %d)"
                     (1+ (count-lines 1 (point)))
                     (1+ (count-lines start (point))))
          (message "Linha %d" (1+ (count-lines 1 (point)))))))))

;;* (simple.el @ 934 @) Undo! -> Desfeito *************************************
(defun undo (&optional arg)
  "Undo some previous changes.
Repeat this command to undo more changes.
A numeric argument serves as a repeat count.

In Transient Mark mode when the mark is active, only undo changes within
the current region.  Similarly, when not in Transient Mark mode, just C-u
as an argument limits undo to changes within the current region."
  (interactive "*P")
  ;; If we don't get all the way thru, make last-command indicate that
  ;; for the following command.
  (setq this-command t)
  (let ((modified (buffer-modified-p))
        (recent-save (recent-auto-save-p)))
    (or (eq (selected-window) (minibuffer-window))
        (message "Desfeito"))
    (unless (eq last-command 'undo)
      (if (if transient-mark-mode mark-active (and arg (not (numberp arg))))
          (undo-start (region-beginning) (region-end))
        (undo-start))
      ;; get rid of initial undo boundary
      (undo-more 1))
    (undo-more
```

```
     (if (or transient-mark-mode (numberp arg))
         (prefix-numeric-value arg)
       1))
    ;; Don't specify a position in the undo record for the undo command.
    ;; Instead, undoing this should move point to where the change is.
    (let ((tail buffer-undo-list)
          (prev nil))
      (while (car tail)
        (when (integerp (car tail))
          (let ((pos (car tail)))
            (if (null prev)
                (setq buffer-undo-list (cdr tail))
              (setcdr prev (cdr tail)))
            (setq tail (cdr tail))
            (while (car tail)
              (if (eq pos (car tail))
                  (if prev
                      (setcdr prev (cdr tail))
                    (setq buffer-undo-list (cdr tail)))
                (setq prev tail))
              (setq tail (cdr tail)))
            (setq tail nil)))
        (setq prev tail tail (cdr tail))))

    (and modified (not (buffer-modified-p))
         (delete-auto-save-file-if-necessary recent-save)))
  ;; If we do get all the way thru, make this-command indicate that.
  (setq this-command 'undo))

;;* (simple.el @ 1370 @) Shell command %sed with no output -> Comando executado
;;    com sucesso ************************************************************
(defun shell-command-on-region (start end command
                                      &optional output-buffer replace
                                      error-buffer)
  "Execute string COMMAND in inferior shell with region as input.
Normally display output (if any) in temp buffer `*Shell Command Output*';
Prefix arg means replace the region with it.  Return the exit code of COMMAND.

To specify a coding system for converting non-ASCII characters
in the input and output to the shell command, use
    \\[universal-coding-system-argument]
before this command.  By default, the input (from the current buffer)
is encoded in the same coding system that will be used to save the file,
`buffer-file-coding-system'.  If the output is going to replace the region,
then it is decoded from that same coding system.

The noninteractive arguments are START, END, COMMAND, OUTPUT-BUFFER,
REPLACE, ERROR-BUFFER.  Noninteractive callers can specify coding
systems by binding `coding-system-for-read' and
`coding-system-for-write'.

If the output is short enough to display in the echo area (which is
determined by the variable `max-mini-window-height' if
```

```
‘resize-mini-windows’ is non-nil), it is shown there, but it is
nonetheless available in buffer ‘*Shell Command Output*’ even though
that buffer is not automatically displayed.  If there is no output, or
if output is inserted in the current buffer, then ‘*Shell Command
Output*’ is deleted.

If the optional fourth argument OUTPUT-BUFFER is non-nil,
that says to put the output in some other buffer.
If OUTPUT-BUFFER is a buffer or buffer name, put the output there.
If OUTPUT-BUFFER is not a buffer and not nil,
insert output in the current buffer.
In either case, the output is inserted after point (leaving mark after it).

If REPLACE, the optional fifth argument, is non-nil, that means insert
the output in place of text from START to END, putting point and mark
around it.

If optional sixth argument ERROR-BUFFER is non-nil, it is a buffer
or buffer name to which to direct the command's standard error output.
If it is nil, error output is mingled with regular output.
In an interactive call, the variable ‘shell-command-default-error-buffer’
specifies the value of ERROR-BUFFER."
  (interactive (let ((string
                      ;; Do this before calling region-beginning
                      ;; and region-end, in case subprocess output
                      ;; relocates them while we are in the minibuffer.
                      (read-from-minibuffer "Shell command on region: "
                                            nil nil nil
                                            'shell-command-history)))
                 ;; call-interactively recognizes region-beginning and
                 ;; region-end specially, leaving them in the history.
                 (list (region-beginning) (region-end)
                       string
                       current-prefix-arg
                       current-prefix-arg
                       shell-command-default-error-buffer)))
  (let ((error-file
         (if error-buffer
             (make-temp-file
              (expand-file-name "scor"
                                (or small-temporary-file-directory
                                    temporary-file-directory)))
           nil))
        exit-status)
    (if (or replace
            (and output-buffer
                 (not (or (bufferp output-buffer) (stringp output-buffer)))))
        ;; Replace specified region with output from command.
        (let ((swap (and replace (< start end))))
          ;; Don't muck with mark unless REPLACE says we should.
          (goto-char start)
          (and replace (push-mark (point) 'nomsg))
          (setq exit-status
```

```
              (call-process-region start end shell-file-name t
                                   (if error-file
                                       (list t error-file) t)
                                   nil shell-command-switch command))
;;; It is rude to delete a buffer which the command is not using.
;;;          (let ((shell-buffer (get-buffer "*Shell Command Output*")))
;;;            (and shell-buffer (not (eq shell-buffer (current-buffer)))
;;;                 (kill-buffer shell-buffer)))
          ;; Don't muck with mark unless REPLACE says we should.
          (and replace swap (exchange-point-and-mark)))
      ;; No prefix argument: put the output in a temp buffer,
      ;; replacing its entire contents.
      (let ((buffer (get-buffer-create
                     (or output-buffer "*Shell Command Output*")))
            (success nil))
        (unwind-protect
            (if (eq buffer (current-buffer))
                ;; If the input is the same buffer as the output,
                ;; delete everything but the specified region,
                ;; then replace that region with the output.
                (progn (setq buffer-read-only nil)
                       (delete-region (max start end) (point-max))
                       (delete-region (point-min) (min start end))
                       (setq exit-status
                             (call-process-region (point-min) (point-max)
                                                  shell-file-name t
                                                  (if error-file
                                                      (list t error-file)
                                                    t)
                                                  nil shell-command-switch
                                                  command)))
              ;; Clear the output buffer, then run the command with
              ;; output there.
              (let ((directory default-directory))
                (save-excursion
                  (set-buffer buffer)
                  (setq buffer-read-only nil)
                  (if (not output-buffer)
                      (setq default-directory directory))
                  (erase-buffer)))
              (setq exit-status
                    (call-process-region start end shell-file-name nil
                                         (if error-file
                                             (list buffer error-file)
                                           buffer)
                                         nil shell-command-switch command)))
            (setq success (and exit-status (equal 0 exit-status)))
            ;; Report the amount of output.
            (if (with-current-buffer buffer (> (point-max) (point-min)))
                ;; There's some output, display it
                (display-message-or-buffer buffer)
              ;; No output; error?
              (message (if (and error-file
```

```
                            (< 0 (nth 7 (file-attributes error-file))))
                       "(A execu  o do aplicativo foi interrompida)"
                     "(Aplicativo executado com sucesso)"))))))
    (when (and error-file (file-exists-p error-file))
      (if (< 0 (nth 7 (file-attributes error-file)))
          (with-current-buffer (get-buffer-create error-buffer)
            (let ((pos-from-end (- (point-max) (point))))
              (or (bobp)
                  (insert "\f\n"))
              ;; Do no formatting while reading error file,
              ;; because that can run a shell command, and we
              ;; don't want that to cause an infinite recursion.
              (format-insert-file error-file nil)
              ;; Put point after the inserted errors.
              (goto-char (- (point-max) pos-from-end)))
            (display-buffer (current-buffer))))
      (delete-file error-file))
    exit-status))

;;* (simple.el @ 2427 @) next-line -> emacspeakbrasil-forward-line *************
;; (replaced all)
(defun next-line (arg)
    "Forward-char redefined to speak char moved to. "
    (interactive "p")
    (cond
      ((<= (+ arg (point)) (point-max)) (forward-line arg) (if (equal 0
          (current-column)) (when (interactive-p) (emacspeak-speak-line))))
      (t (ding) (dtk-stop) (dtk-tone 500 75) (message "Fim da janela"))))

;;* (simple.el @ 2462 @) previous-line -> emacspeakbrasil-backward-line ********
;;; (replaced all)
(defun previous-line (arg)
    "Backward-char redefined to speak char moved to. "
    (interactive "p")
    (cond
      ((>= (- (point) arg) (point-min)) (forward-line (- arg)) (when
          (interactive-p) (emacspeak-speak-line)))
      (t (ding) (dtk-stop) (dtk-tone 500 75) (message "Come o da janela"))))

;;* (w3m-favicon.el @ 238 @) Reading -> Lendo ***********************************
(defun w3m-favicon-retrieve (url type target)
  "Retrieve favicon from URL and convert it to image as TYPE in TARGET.
TYPE is a symbol like 'ico' and TARGET is a buffer where the image is
stored in the 'w3m-favicon-image' buffer-local variable."
  (if (and (w3m-favicon-cache-p url)
           (or (null w3m-favicon-cache-expire-wait)
               (< (- (w3m-float-time)
                     (w3m-float-time (w3m-favicon-cache-retrieved url)))
                  w3m-favicon-cache-expire-wait)))
      (with-current-buffer target
        (w3m-favicon-set-image (w3m-favicon-cache-favicon url)))
    (lexical-let ((url url)
                  (type type)
```

```
                (target target))
      (w3m-process-with-null-handler
        (w3m-process-do-with-temp-buffer
            (ok (w3m-retrieve url 'raw nil nil nil handler))
          (let (idata image)
            (if (and ok
                     ;; Some broken servers provides empty contents.
                     (>= (buffer-size) 4))
                (setq idata (buffer-string)
                      image (w3m-favicon-convert idata type))
              (w3m-message "Conclu do"))
            (with-current-buffer target
              (w3m-favicon-set-image image)
              (push (list url idata (current-time) w3m-favicon-image)
                    w3m-favicon-cache-data)))))))
  (w3m-static-when (boundp 'header-line-format)
    ;; Emacs frame needs to be redisplayed to make favicon come out.
    (run-at-time 1 nil
                 (lambda (buffer)
                   (if (and (buffer-live-p buffer)
                            (eq (get-buffer-window buffer t)
                                (selected-window)))
                       (w3m-force-window-update)))
                 target)))

;;* (w3m.el @ 3278 @) Fontifying... -> Renderizando ****************************
(defun w3m-fontify ()
  "Fontify the current buffer."
  (let ((case-fold-search t)
        (buffer-read-only))
    (run-hooks 'w3m-fontify-before-hook)
    (w3m-message "Renderizando")
    ;; Delete <?xml ... ?> tag
    (goto-char (point-min))
    (if (search-forward "<?xml" nil t)
        (let ((start (match-beginning 0)))
          (search-forward "?>" nil t)
          (delete-region start (match-end 0))))
    ;; Delete extra title tag.
    (goto-char (point-min))
    (let (start)
      (and (search-forward "<title>" nil t)
           (setq start (match-beginning 0))
           (search-forward "</title>" nil t)
           (delete-region start (match-end 0))))
    (w3m-fontify-bold)
    (w3m-fontify-strike-through)
    (w3m-fontify-underline)
    (when w3m-use-symbol
      (w3m-replace-symbol))
    (w3m-fontify-anchors)
    (when w3m-use-form
      (w3m-fontify-forms))
```

```
    (w3m-fontify-images)
    ;; Remove other markups.
    (goto-char (point-min))
    (while (re-search-forward "</?[A-Za-z_][^>]*>" nil t)
      (let ((fid (get-text-property (match-beginning 0) 'w3m-form-field-id)))
        (if (and fid (string-match "/type=textarea/" fid))
            (goto-char (match-end 0))
          (delete-region (match-beginning 0) (match-end 0)))))
    ;; Decode escaped characters (entities).
    (w3m-decode-entities 'reserve-prop)
    (when w3m-use-form
      (w3m-fontify-textareas))
    (goto-char (point-min))
    (when w3m-delete-duplicated-empty-lines
      (while (re-search-forward "^[ \t]*\n\\([ \t]*\n\\)+" nil t)
        (delete-region (match-beginning 0) (1- (match-end 0)))))
    (w3m-message "Conclu do")
    (w3m-header-line-insert)
    (run-hooks 'w3m-fontify-after-hook)))

;;* (w3m.el @ 3469 @) URL (default HOME) -> Digite o endere o e pressione
;;    Enter: ************************************************************************
(defun w3m-input-url (&optional prompt initial default quick-start)
  "Read a url from the minibuffer, prompting with string PROMPT."
  (let (url)
    (w3m-arrived-setup)
    (unless default
      (setq default w3m-home-page))
    (unless initial
      (setq initial (w3m-active-region-or-url-at-point)))
    (if (and quick-start
             default
             (not initial))
        default
      (setq url (let ((minibuffer-setup-hook
                       (append minibuffer-setup-hook '(beginning-of-line))))
                  (completing-read
                   (or prompt
                       (if default
                           (format "Digite o endere o e pressione Enter: ")
                         "Digite o endere o e pressione Enter: "))
                   'w3m-input-url-history)))
      (when (string= "" url)
        (setq url default))
      (if (stringp url)
          (progn
            ;; remove duplication
            (setq w3m-input-url-history
                  (cons url (delete url w3m-input-url-history)))
            (w3m-canonicalize-url url))
        ;; It may be `popup'.
        url))))
```

```
;;* (w3m.el @ 3659 @) Save to (%s): -> Salvar como: ***************************
(defun w3m-read-file-name (&optional prompt dir default existing)
  (when default
    (setq default (file-name-nondirectory (w3m-url-strip-query default))))
  (unless prompt
    (setq prompt (if default
                     "Salvar como: "
                   "Salvar como: ")))
  (setq dir (file-name-as-directory (or dir w3m-default-save-directory)))
  (let ((default-directory dir)
        (file (read-file-name prompt dir nil existing default)))
    (if (not (file-directory-p file))
        (setq w3m-default-save-directory
              (or (file-name-directory file) w3m-default-save-directory))
      (setq w3m-default-save-directory file)
      (if default
          (setq file (expand-file-name default file))))
    (expand-file-name file)))

;;* (w3m.el @ 4024 @) Request sent, waiting for response -> Requisi  o enviada,
;;    aguardando resposta ***************************************************
(defun w3m-w3m-dump-head (url handler)
  "Return the header string of URL."
  (lexical-let ((url url))
    (w3m-message "Requisi  o enviada, aguardando resposta")
    (w3m-process-do-with-temp-buffer
        (success (progn
                   (setq w3m-current-url url
                         url (w3m-url-strip-authinfo url))
                   (w3m-process-start handler
                                      w3m-command
                                      (append w3m-command-arguments
                                              (list "-o" "follow_redirection=0"
                                                    "-dump_head" url)))))
      (w3m-message "Conclu do")
      (when success
        (buffer-string)))))

;;* (w3m.el @ 4115 @) Reading -> Lendo ************************************
(defun w3m-w3m-dump-extra (url handler)
  "Retrive headers and contents pointed to by URL"
  (lexical-let ((url url))
    (setq w3m-current-url url
          url (w3m-url-strip-authinfo url))
    (w3m-message "Lendo %s" url)
    (w3m-process-do
        (success
         (w3m-process-start handler
                            w3m-command
                            (append w3m-command-arguments
                                    (w3m-w3m-expand-arguments
                                     w3m-dump-head-source-command-arguments)
                                    (list url))))
```

```
      (w3m-message "Conclu do")
      (when success
        (goto-char (point-min))
        (let ((case-fold-search t))
          (when (and (re-search-forward "^w3m-current-url:" nil t)
                     (progn
                       (delete-region (point-min) (match-beginning 0))
                       (search-forward "\n\n" nil t)))
            ;; Asahi-shimbun sometimes says gif as jpeg mistakenly, for
            ;; example.  So, we cannot help trusting the data itself.
            (when (prog2
                      (setq case-fold-search nil)
                      (looking-at "\\(GIF8\\)\\|\\(\377\330\\)\\|\211PNG\r\n")
                    (setq case-fold-search t))
              (let ((type (cond ((match-beginning 1) "gif")
                                ((match-beginning 2) "jpeg")
                                (t "png"))))
                (save-excursion
                  (when (re-search-backward "^content-type: image/\\(.+\\)$"
                                            nil t)
                    (delete-region (goto-char (match-beginning 1))
                                   (match-end 1))
                    (insert type)))))
            (let ((header (buffer-substring (point-min) (point))))
              (when w3m-use-cookies
                (w3m-cookie-set url (point-min) (point)))
              (delete-region (point-min) (point))
              (w3m-cache-header url header)
              (w3m-cache-contents url (current-buffer))
              (w3m-w3m-parse-header url header)))))))

;;* (w3m.el @ 4506 @) Download %s to -> Download %s para ************************
(defun w3m-download (url &optional filename no-cache handler)
  (interactive
   (let* ((url (w3m-input-url "Download endere o: "
                              (when (stringp w3m-current-url)
                                (if (string-match
                                     "\\`about://\\(header\\|source\\)/"
                                     w3m-current-url)
                                    (substring w3m-current-url (match-end 0))
                                  w3m-current-url))
                              w3m-home-page))
          (basename (file-name-nondirectory (w3m-url-strip-query url))))
     (if (string-match "^[\t ]*$" basename)
         (list url
               (w3m-read-file-name (format "Download %s para: " url)
                                   w3m-default-save-directory "index.html")
               current-prefix-arg)
       (list url
             (w3m-read-file-name (format "Download %s para: " basename)
                                 w3m-default-save-directory basename)
             current-prefix-arg))))
  (if (and w3m-use-ange-ftp (string-match "\\`ftp://" url))
```

```
      (w3m-goto-ftp-url url filename)
    (lexical-let ((url url)
                  (filename (or filename (w3m-read-file-name nil nil url))))
      (w3m-process-do-with-temp-buffer
          (type (progn
                  (w3m-clear-local-variables)
                  (setq w3m-current-url url)
                  (w3m-retrieve url t no-cache nil nil handler)))
        (if type
            (let ((buffer-file-coding-system 'binary)
                  (file-coding-system 'binary)
                  (coding-system-for-write 'binary)
                  jka-compr-compression-info-list
                  jam-zcat-filename-list
                  format-alist)
              (when (or (not (file-exists-p filename))
                        (y-or-n-p
                         (format "J  existe um arquivo chamado %s; deseja
                             substitu -lo? "
                                 filename)))
                (write-region (point-min) (point-max) filename)
                (w3m-touch-file filename (w3m-last-modified url))
                t))
          (ding)
          (message "N o foi poss vel acessar o endere o: %s" url)
          nil)))))

;;* (w3m.el @ 4750 @) Rendering -> Renderizando ********************************
(defun w3m-rendering-buffer (&optional charset)
  "Do rendering of contents in the currenr buffer as HTML and return title."
  (w3m-message "Renderizando")
  (w3m-remove-comments)
  (w3m-check-header-tags)
  (w3m-check-refresh-attribute)
  (unless (eq w3m-type 'w3m-m17n)
    (w3m-remove-meta-charset-tags))
  (w3m-rendering-half-dump charset)
  (w3m-message "Conclu do")
  (w3m-rendering-extract-title))

;;* (w3m.el @ 4762 @) Cannot retrieve URL -> N o foi porr vel acessar o
;;    endere o ******************************************************************
(defun w3m-retrieve-and-render (url &optional no-cache charset
                                    post-data referer handler)
  "Retrieve contents of URL and render them in the current buffer.
It returns a `w3m-process' object and comes to an end immediately.
The HANDLER function will be called when rendering is complete.  When
a new content is retrieved in the buffer, the HANDLER function will be
called with t as an argument.  Otherwise, it will be called with nil."
  (unless (and w3m-current-ssl
               (not (string-match "\\`\\(ht\\|f\\)tps://" url))
               (not (y-or-n-p "Voc  est  saindo de uma p gina segura.  Deseja
                   continuar? ")))
```

```
    (lexical-let ((url (w3m-url-strip-fragment url))
                  (charset charset)
                  (page-buffer (current-buffer))
                  (arrival-time (current-time)))
      (w3m-process-do-with-temp-buffer
          (type (progn
                  (w3m-clear-local-variables)
                  (w3m-retrieve url nil no-cache post-data referer handler)))
        (when (buffer-live-p page-buffer)
          (setq url (w3m-url-strip-authinfo url))
          (if type
              (let ((modified-time (w3m-last-modified url)))
                (w3m-arrived-add url nil modified-time arrival-time)
                (unless modified-time
                  (setf (w3m-arrived-last-modified url) nil))
                (let ((real (w3m-real-url url)))
                  (unless (string= url real)
                    (w3m-arrived-add url nil nil arrival-time)
                    (setf (w3m-arrived-title real)
                          (w3m-arrived-title url))
                    (setf (w3m-arrived-last-modified real)
                          (w3m-arrived-last-modified url))
                    (setq url real)))
                (prog1 (w3m-create-page url
                                        (or (w3m-arrived-content-type url)
                                            type)
                                        (or charset
                                            (w3m-arrived-content-charset url)
                                            (w3m-content-charset url))
                                        page-buffer)
                  (unless (get-buffer-window page-buffer)
                    (w3m-message "O conte do (%s) foi renderizado em %s"
                                 url (buffer-name page-buffer)))))
            (ding)
            (when (eq (car w3m-current-forms) t)
              (setq w3m-current-forms (cdr w3m-current-forms)))
            (prog1 (when (and w3m-show-error-information
                              (not (or (w3m-url-local-p url)
                                       (string-match "\\`about:" url))))
                     (w3m-show-error-information url charset page-buffer))
              (w3m-message "N o foi poss vel acessar o endere o: %s" url)))))))

;;* (w3m.el @ 4819 @) Cannot retrieve URL -> Lendo ****************************
(defun w3m-show-error-information (url charset page-buffer)
  "Create and prepare the error information."
  (or (when (w3m-cache-request-contents url)
        (w3m-decode-encoded-contents (w3m-content-encoding url))
        t) ; Even if decoding is failed, use the cached contents.
      (let ((case-fold-search t)
            (header (w3m-cache-request-header url))
            (errmsg (format "\n<br><h1>N o foi poss vel acessar o endere o:
                %s</h1>"
                            (format "<a href=\"%s\">%s</a>" url url))))
```

```
       (if (or (null header)
               (string-match "\\‘w3m: Can’t load " header))
           (progn
             (erase-buffer)
             (insert
              errmsg
              (format "<br><br><b>%s</b> n o p de ser encontrado."
                      (w3m-get-server-hostname url))))
         (goto-char (point-min))
         (when (or (re-search-forward "<body>" nil t)
                   (re-search-forward "<html>" nil t))
           (goto-char (match-end 0)))
         (insert errmsg "<br><br><hr><br><br>")
         (when (or (re-search-forward "</body>" nil t)
                   (re-search-forward "</html>" nil 'max))
           (goto-char (match-end 0)))
         (insert "\n<br><br><hr><br><br><h2>Header information</h2><br>\n<pre>"
                 header "</pre>\n"))))
  (w3m-create-page url "text/html" charset page-buffer) (dtk-speak
      "N ofoiposs veuacessaroender  o.")
  nil)

;;* (w3m.el @ 4908 @) Input %s’s content type -> Pressione enter ***************
(defun w3m-create-page (url type charset page-buffer)
  ;; Select a content type.
  (unless (and (stringp type)
               (assoc type w3m-content-type-alist))
    (save-window-excursion
      (pop-to-buffer (current-buffer))
      (delete-other-windows)
      (ding)
      (setq type
            (completing-read
             (format "Pressione Enter, para confirmar o download de %s. Para
                 cancelar pressione Escape. "
                     (file-name-nondirectory url))
             w3m-content-type-alist nil t))
      (setf (w3m-arrived-content-type url) type)))
  (setq type (w3m-prepare-content url type charset))
  (w3m-safe-decode-buffer url charset type)
  (setq charset (or charset w3m-current-content-charset))
  (when w3m-use-filter (w3m-filter url))
  (w3m-relationship-estimate url)
  ;; Create pages.
  (cond
   ((string-match "\\‘text/" type)
    (w3m-create-text-page url type charset page-buffer))
   ((string-match "\\‘image/" type)
    (w3m-create-image-page url type charset page-buffer))
   (t
    (with-current-buffer page-buffer
      (w3m-external-view url)
      'external-view))))
```

```
;;* (w3m.el @ 5052 @) There's no more history -> Endere o inicial **************
(defun w3m-view-previous-page (&optional count)
  "Move back COUNT pages in the history.
If COUNT is a positive integer, move backward COUNT times in the
history.  If COUNT is a negative integer, moving forward is performed.
COUNT is treated as 1 by default if it is omitted."
  (interactive "p")
  (unless w3m-current-url
    ;; This page may have not been registered in the history since an
    ;; accident has probably occurred, so we should skip the page.
    (if (integerp count)
        (when (> count 0)
          (decf count))
      (setq count 0)))
  (let ((hist ;; Cons of a new history element and position pointers.
         (if (integerp count)
             (w3m-history-backward count)
           (w3m-history-backward)))
        ;; Inhibit sprouting of a new history.
        (w3m-history-reuse-history-elements t)
        (w3m-use-refresh nil))
    (if hist
        (let ((w3m-prefer-cache t))
          (w3m-goto-url (caar hist) nil nil
                        (w3m-history-plist-get :post-data)
                        (w3m-history-plist-get :referer)
                        nil
                        (w3m-history-element (caddr hist) t))
          ;; Set the position pointers in the history.
          (setcar w3m-history (cdr hist))
          ;; Restore last position.
          (w3m-history-restore-position))
      (message "Endere o inicial"))))

;;* (w3m.el @ 5457 @) No URL at point -> Nenhum endere o sob o cursor **********
(defun w3m-download-this-url ()
  "Download the file or the page pointed to by the link under point."
  (interactive)
  (let ((url (or (w3m-anchor) (w3m-image))))
    (if (w3m-url-valid url)
        (lexical-let ((pos (point-marker))
                      (curl w3m-current-url))
          (w3m-process-with-null-handler
            (w3m-process-do
                (success (w3m-download url nil nil nil handler))
              (and success
                   (buffer-name (marker-buffer pos))
                   (save-excursion
                     (set-buffer (marker-buffer pos))
                     (when (equal curl w3m-current-url)
                       (goto-char pos)
                       (w3m-refontify-anchor)))))))))
```

```
      (w3m-message "Nenhum endere o sob o cursor"))))

;;* (w3m.el @ 5483 @) There is no url -> Nenhum endere o sob o cursor **********
(defun w3m-print-this-url (&optional interactive-p)
  "Display the url under point in the echo area and put it into ‘kill-ring’."
  (interactive (list t))
  (let ((url (if interactive-p
                 (or (w3m-anchor) (w3m-image))
               (or (w3m-anchor (point)) (w3m-image (point))))))
    (when (or url interactive-p)
      (and url interactive-p (kill-new url))
      (w3m-message "%s"
                   (or url
                       (and (w3m-action) "Formul rio")
                       "Nenhum endere o sob o cursor")))))
```

dtk-speak.el:

```
#Se  o localizada nas linhas 214 e 224:

    (aset  table 0 "")
    (aset  table 10 "n val nha")

#Se  o localizada entre as linha 246 e 310:

    (aset table 32 "esp  o")
    (aset table 33 "exclama  o")
    (aset table 34 " spas")
    (aset table 35  "n mber")
    (aset table  36  "d lar")
    (aset table 37  "porc nto" )
    (aset table 38  " comerci l")
    (aset table 39  "ap strofo" )
    (aset table 40  "abrepar ntese" )
    (aset table 41 "f chapar ntese" )
    (aset table  42   "aster sco")
    (aset table 43  "s ma")
    (aset   table 44 "v rgula")
    (aset table  45  " fem")
    (aset table 46  "p nto")
    (aset table 47  "b rra")
    (aset table 48  "zero")
    (aset table 49 "um")
    (aset table 50 "dois")
    (aset table 51 "tr s")
    (aset  table 52  "qu tro")
    (aset  table 53 "c nco")
    (aset  table 54 "s is")
    (aset  table 55 "s te")
```

```
    (aset  table 56 " ito")
    (aset  table 57 "n ve")
    (aset table 58 "doisp ntos" )
    (aset table 59 "p ntoev rgula")
    1(aset table 60 "men r")
    (aset  table 61 "igu l")
    (aset  table 62  "mai r")
    (aset  table 63 "interroga  o")
    (aset  table 64 "arr ba")
    (aset  table 65 "A")
    (aset  table 66 "B")
    (aset  table 67 "C")
    (aset  table 68 "D")
    (aset  table 69 "E")
    (aset  table 70 "F")
    (aset  table 71 "G")
    (aset  table 72 "H")
    (aset  table 73 "I")
    (aset  table 74 "J")
    (aset  table 75 "K")
    (aset  table 76 "L")
    (aset  table 77 "M")
    (aset  table 78 "N")
    (aset  table 79 "O")
    (aset  table 80 "P")
    (aset  table 81 "Q")
    (aset  table 82 "R")
    (aset  table 83 "S")
    (aset  table 84 "T")
    (aset  table 85 "U")
    (aset  table 86 "V")
    (aset  table 87 "W")
    (aset  table 88 "X")
    (aset  table 89 "Y")
    (aset  table 90 "Z")
    (aset  table 91 "abrecolch te")
    (aset  table 92  "barrainvert da")
    (aset  table 93 "f chacolch te")
    (aset  table 94  "circunfl xo")
    (aset  table 95  "sublinh do")
    (aset  table 96 "cr se")

#Se  o localizada entre as linha 337 e 340:

    (aset  table 123 "abrech ve")
    (aset  table 124 "barravertic l")
    (aset  table 125 "f chach ve")
    (aset  table 126 "t l")

#Se  o localizada entre as linha 407 e 467:

    (aset  table 192  "A comacentocr se")
    (aset  table 193  "A comacentoag do")
```

```
    (aset  table 194  "A comacentocircunfl xo")
    (aset  table 195  "A comacentot u")
    (aset  table 199  "C cid lha")
    (aset  table 201  "E comacentoag do")
    (aset  table 202  "E comacentocircunfl xo")
    (aset  table 205  "I comacentoag do")
    (aset  table 211  "O comacentoag do")
    (aset  table 212  "O comacentocircunfl xo")
    (aset  table 213  "O comacentot u")
    (aset  table 218  "U comacentoag do")
    (aset  table 220  "U comacentotr ma")
    (aset  table 224  "a comacentocr se")
    (aset  table 225  "a comacentoag do")
    (aset  table 226  "a comacentocircunfl xo")
    (aset  table 227  "a comacentot u")
    (aset  table 231  "c cid lha")
    (aset  table 233  "e comacentoag do")
    (aset  table 234  "e comacentocircunfl xo")
    (aset  table 237  "i comacentoag do")
    (aset  table 243  "o comacentoag do")
    (aset  table 244  "o comacentocircunfl xo")
    (aset  table 245  "o comacentot u")
    (aset  table 250  "u comacentoag do")
    (aset  table 252  "u comacentotr ma")

#Se  o localizada na linha 214:

  (if (> char 255 )
```

dtk-tcl.el:

```
#Se  o localizada entre as linha 259 e 279:

;        (cond
;         ((= ?| (char-after (match-beginning 0 )))
;          (replace-match " pipe "))
;         ((= ?< (char-after (match-beginning 0 )))
;          (replace-match " less than "))
;         ((= ?> (char-after (match-beginning 0 )))
;          (replace-match " greater than "))
;         ((= ?{ (char-after (match-beginning 0 )))
;          (replace-match " left brace "))
;         ((= ?} (char-after (match-beginning 0 )))
;          (replace-match " right brace "))
;         ((=  ?\] (char-after (match-beginning 0)))
;          (replace-match " right bracket "))
;         ((= ?\[ (char-after  (match-beginning 0)))
;          (replace-match " left bracket "))
;         ((= ?\\ (char-after (match-beginning 0 )))
```

```
;          (replace-match " backslash "))
;         ((= ?# (char-after (match-beginning 0 )))
;          (replace-match " pound "))
;         ((= ?‘ (char-after (match-beginning 0 )))
;          (replace-match " backquote ")))
```

emacspeak-advice.el:

```
#Se  o localizada entre as linha 68 e 92:

;(defadvice next-line (before emacspeak pre act com)
;  "Produce auditory icon  if we cant move."
;  (when (and (interactive-p)
;             (save-excursion
;               (end-of-line)
;               (eobp)))
;    (emacspeak-auditory-icon 'warn-user)))
;(defadvice previous-line (before emacspeak pre act com)
;  "Produce auditory icon  if we cant move."
;  (when (and (interactive-p)
;             (save-excursion
;               (beginning-of-line)
;               (bobp)))
;    (emacspeak-auditory-icon 'warn-user)))
;(defadvice next-line (after emacspeak pre act)
;  "Speak line that you just moved to."
;  (when (interactive-p)
;    (emacspeak-speak-line  )))
;(defadvice previous-line (after emacspeak pre act)
;  "Speak line that you just moved to."
;  (when (interactive-p)
;    (emacspeak-speak-line  )))

#Se  o localizada nas linhas 649, 773, 780, 790, 797, 799, 1628 e 1641:

        (dtk-stop))
    (dtk-stop))
    (dtk-say "s"))
    (dtk-stop))
    (dtk-say "s"))
      (dtk-say "n" )))
      (message "Regi o recortada"))); count)))
      (message "Regi o recortada"))); count)))

#Se  o localizada entre as linha 1596 e 1606:

;  (declare (special emacspeak-last-message))
;  (cond
;   ((interactive-p)
```

```
;    (setq emacspeak-last-message nil)
;    ad-do-it
;    (emacspeak-auditory-icon 'save-object)
;    (or emacspeak-last-message
;        (message "Wrote %s"
;                 (buffer-file-name))))
;   (t ad-do-it))
 ad-do-it (message (format "Arquivo salvo %s" (buffer-file-name))) ad-return-value)

#Se  o localizada entre as linha 1650 e 1653:

    (message "Regi o copiada")))
;             (count-lines (region-beginning)
;                          (region-end)))))

#Se  o localizada nas linhas 1904, 1909, 2351, 2358, 2365 e 2372:

  (tts-restart) (dtk-tone 500 75) (error "");  (dtk-stop)
  (dtk-stop)
    (dtk-stop)))
    (dtk-stop)))
    (dtk-stop)))
    (dtk-stop)))

#Se  o localizada entre as linha 2656 e 2670:

;(defadvice occur-prev (after emacspeak pre act comp)
;  "Provide spoken feedback."
;  (when (interactive-p)
;    (emacspeak-speak-line)
;    (emacspeak-auditory-icon 'large-movement)))
;(defadvice occur-next (after emacspeak pre act comp)
;  "Provide spoken feedback."
;  (when (interactive-p)
;    (emacspeak-speak-line)
;    (emacspeak-auditory-icon 'large-movement)))
;(defadvice occur-mode-goto-occurrence (after emacspeak pre act comp)
;  "Provide auditory feedback."
;  (when (interactive-p)
;    (emacspeak-auditory-icon 'large-movement)
;    (emacspeak-speak-line)))
```

emacspeak-buff-menu.el:

```
#Se  o localizada entre as linha 65 e 69:

    (let*((buffer (Buffer-menu-buffer nil)))
      (if (Buffer-menu-buffer nil) (progn (get-buffer buffer)
          (dtk-speak (buffer-name  buffer)))
```

```
        (error "Nenhuma janela nessa linha"))))
   (t (error "Esse comando pode ser usado somente na lista de janelas"))))

#Se  o localizada nas linhas 122 e 128:

  (emacspeak-list-buffers-speak-buffer-name))
  (emacspeak-list-buffers-speak-buffer-name))

#Se  o localizada entre as linha 140 e 146:

    (define-key Buffer-menu-mode-map "n" 'emacspeak-list-buffers-next-line)
        (define-key Buffer-menu-mode-map [down] 'emacspeak-list-buffers-next-line)
    (define-key Buffer-menu-mode-map "p" 'emacspeak-list-buffers-previous-line)
        (define-key Buffer-menu-mode-map [up]
        'emacspeak-list-buffers-previous-line)
    (emacspeak-list-buffers-speak-buffer-name)
```

emacspeak-dired.el:

```
#Se  o localizada nas linhas 122, 123 e 128:

      (progn (dtk-stop) (when (file-directory-p filename) (dtk-tone 500 75))
          (dtk-speak filename))
      (message "Tamanho do arquivo %s bytes"
```

emacspeak-keymap.el:

```
#Se  o localizada nas linhas 324, 325, 327 e 328:

;(global-set-key '[(control left)] 'emacspeak-previous-frame)
;(global-set-key '[(control right)] 'emacspeak-next-frame)
;(global-set-key '[(control down)] 'emacspeak-mark-forward-mark)
;(global-set-key '[(control up)] 'emacspeak-mark-backward-mark)
```

emacspeak-redefine.el:

```
#Se  o localizada nas linhas 130, 141, 157 e 161:
```

```
     (dtk-tone 500 75) (message "Fim da janela"))))
      (dtk-tone 500 75) (message "Come o da janela"))))
    (read-buffer "Pressione Enter, para confirmar o encerramento da janela: "
  (message "Janela encerrada. Pressione f1, para obter ajuda."))
```

emacspeak-speak.el:

```
#Se  o localizada na linha 596:

(defcustom emacspeak-speak-maximum-line-length  1024

#Se  o localizada entre as linha 906 e 918:

          do (dtk-letter (char-to-string char)))))
;          (setq char-string (format "%c " char))
;          (when (and (<= ?A char)
;                     (<= char ?Z))
;            (if voice-lock-mode
;                (put-text-property 0 1
;                                   'personality 'paul-animated
;                                   char-string)
;              (setq char-string (format "cap %s " char-string))))
;          (setq result
;                (concat result
;                        char-string)))
;    (dtk-speak result)))

#Se  o localizada nas linhas 922, 923, 932, 942 e 944:

  (interactive) (skip-chars-forward "^\n\t ") (setq end (point))
      (skip-chars-backward "^\n\t ") (setq start (point))
  (emacspeak-speak-spell-word (buffer-substring start end)))
      (skip-chars-forward "^\n\t ");      (forward-word 1)
      (skip-chars-backward "^\n\t ");      (backward-word 1)

#Se  o localizada entre as linha 951 e 956:

;      (cond
;       ((and (interactive-p)
;             (eq emacspeak-speak-last-spoken-word-position orig))
;        (setq speaker 'emacspeak-speak-spell-word)
;        (setq emacspeak-speak-last-spoken-word-position nil))
;       (t (setq  emacspeak-speak-last-spoken-word-position orig)))

#Se  o localizada entre as linha 1686 e 1701:

(defun emacspeak-speak-buffer-filename (buffer)
(interactive (list (if (or (equal (buffer-name) "*scratch*") (equal (buffer-name)
```

```
    "*Messages*") (equal (buffer-name) "*Compile-Log*")) (message "Janela de
    sistema %s" (buffer-name)) (message "%s" (buffer-name))))))
;  (let ((location (or (buffer-file-name)
;                      default-directory)))
;    (when filename
;      (setq location
;            (file-name-nondirectory location)))
;    (kill-new location)
;    (dtk-speak
;     location)))

#Se  o localizada nas linhas 2240 e 2708:

  (message "Coluna %d" (current-column )))
;(emacspeak-use-customized-blink-paren)
```

emacspeak-w3m.el:

```
#Se  o localizada nas linhas 112, 113 e 131:

  (let ((emacspeak-speak-messages nil)) (dtk-speak (emacspeak-w3m-anchor-text "link
      simb lico"))))
(defun emacspeak-w3m-speak-form-input (form id name type width maxlength value)

#Se  o localizada entre as linha 133 e 134:

   (format "entrada de texto %s  %s. pressione enter para editar, e enter para
       concluir";           type

#Se  o localizada nas linhas 140, 143, 144 e 149:

(defun emacspeak-w3m-speak-form-input-password (form id name)
   (format "entrada de senha %s  %s. pressione enter para editar, e enter para
       concluir"
(defun emacspeak-w3m-speak-form-submit (form id name value)

#Se  o localizada nas linha 152, 153 e 158:

       "bot o de envio"
     (format "bot o %s"
(defun emacspeak-w3m-speak-form-input-radio (form id name value)


#Se  o localizada nas linhas 169, 171, 173 e 177:

          (format "bot o r dio %s n o marcado. pressione enter para marcar" name)
       (format "%s do bot o r dio %s"
                (concat "op  o " value)
(defun emacspeak-w3m-speak-form-input-select (form id name)
```

```
#Se  o localizada nas linhas 179, 187, 188, 195 e 197:

   (format "bot o de sele  o %s %s. pressione enter e use as setas para selecionar"
   (format " rea de texto %s  %s. pressione enter para iniciar a edi  o, e
       Control-c, Control-c, para finalizar"
   (format "bot o %s"
            "bot o res t"
```

emacspeak.el:

```
#Se  o localizada nas linhas 291, 292 e 300:

  (require 'emacspeak-keymap) (require 'emacspeak-keymap-brasil)
      (delete-other-windows)
;  (emacspeak-sounds-define-theme-if-necessary emacspeak-sounds-default-theme)

#Se  o localizada entre as linha 309 e 313:

  (let ((emacspeak-speak-messages nil)) (dtk-speak "Ol «« O ambi nteem cs
      est ativ do«« pressi ne  feu m« para obt raj da") (message
    (format "Ol . O ambiente Emacs est  ativado. \
Pressione f1, para obter ajuda."
           ))))
```

Apêndice C - Tutorial

1 Introdução

O "GNU Editor Macros", ou simplesmente "Emacs", é muito mais do que um simples editor de textos. Seu desenvolvimento teve início no ano de 1976 pelo patrono do software livre "Richard Stallman" e desde então, por se tratar de um software de livre utilização com código fonte aberto, sofisticações e melhorias vem sendo implementadas com a ajuda de milhares de programadores voluntários ao redor do mundo (http://www.emacs.org). Devido a sua constante evolução, podemos dizer que o "Emacs" se tornou um poderoso e completo ambiente de trabalho. Algumas das atividades que podemos realizar dentro desse ambiente são:

- Edição de texto;
- Acesso à *internet*;
- Gerenciamento de cartas eletrônicas (*e-mails*);
- Gerenciamento de arquivos;
- Execução de músicas (rádio, MP3 ou CD);
- Execução de vídeos (AVI, MPEG ou DVD);
- Produção de textos com formatação (LaTeX);
- Produção de músicas e partituras (MusiXTeX);
- Instalação de novos aplicativos e até mesmo o desenvolvimento de novos recursos.

Se você está ouvindo esse texto dentro do ambiente "Emacs", significa que está utilizando o aplicativo "Emacspeak" (ou Emacs Falado), que é o responsável pela transformação das informações presentes na tela do monitor de vídeo em informações faladas, ou seja, é o leitor de telas do ambiente (http://emacspeak.sf.net). Este Tutorial relaciona as principais funções do ambiente "Emacs/Emacspeak" implementado na sua versão para o idioma português do Brasil (http://www.decom.fee.unicamp.br/~samer). Para ler o texto com maior flexibilidade de avanço e retrocesso, pressione a tecla "`Escape`" (para interromper a leitura contínua) e utilize as setas "`Abaixo`" ou "`Acima`" (para avançar ou retroceder respectivamente na leitura das linhas).

O ambiente Emacs é a ferramenta preferida de milhões de usuários e programadores ao redor do mundo (sejam eles ou não deficientes visuais) e um dos fatores que contribuíram para essa notável popularidade é que praticamente todas as suas funções podem ser rapidamente acessadas por intermédio de teclas de atalho (seqüência ou combinação de teclas). Devido ao grande número de funcionalidades, veremos a seguir que esses atalhos são geralmente compostos por uma seqüência de duas e às vezes até três diferentes combinações de teclas, por exemplo `Control-h F5` , que é o comando utilizado para acessar a qualquer momento

esse tutorial. Observe que nesse tipo de representação, o caractere "hífen" não deve ser pressionado, sua representação é apenas indicativa para que as teclas Control e h sejam simultaneamente pressionadas. Observe também que a ausência do caractere "hífen" após a representação Control-h , significa que essa combinação de teclas deve ser liberada antes que seja pressionada a tecla F5 .

2 Inicialização do Ambiente

O ambiente "Emacs/Emacspeak" pode ser configurado para que seja automaticamente ativado durante a inicialização do sistema ou durante o pressionamento simultâneo de uma combinação de teclas qualquer, por exemplo `Control-Alt-E` , de modo que como alvo seja executado o aplicativo "emacspeakbrasil". Quando ativado, o ambiente informa o usuário que já está pronto para ser utilizado com a seguinte mensagem de saudação:

```
Ol .  O ambiente Emacs est  ativado.  Pressione <F1> para obter ajuda
```

A partir deste ponto, todos os aplicativos e funções disponíveis no sistema podem ser imediatamente acessados por intermédio das teclas de atalho apresentadas e documentadas no decorrer deste tutorial.

3 Funções Básicas

Escape localizada no canto superior esquerdo do teclado, essa tecla quando pressionada emite um pequeno *beep* sinalizador, cancela qualquer operação e silencia imediatamente o sintetizador de voz. Pode ser utilizada a qualquer momento e em qualquer situação.

Dica: *como atalho alternativo para execução dessa mesma função, utilize a combinação de teclas "*`Control-g`*". Essa opção evita que a mão esquerda seja a todo momento deslocada para o canto superior esquerdo do teclado.*

Dica: *quando a intenção for apenas silenciar o sintetizador de voz sem que qualquer operação seja cancelada, utilize o atalho "*`Control-z`*".*

F1 localizada logo à direita da tecla "Escape", essa tecla quando pressionada exibe o "menu de ajuda interativa", que nada mais é do que uma lista com as principais funções disponíveis no aplicativo em execução no momento da solicitação.

Dica: *durante a leitura do menu, pequenos* beeps *com diferentes tonalidades serão emitidos para diferenciar as combinações de teclas.*

F2 localizada logo à direita da tecla "F1", essa tecla apresenta uma continuação do "menu de ajuda interativa", ou seja, quando pressionada exibe uma lista com funções complementares disponíveis no aplicativo em execução no momento da solicitação.

Control-x Shift-c esse é o atalho utilizado para desligar o computador. Sempre que acionada essa função, a tecla "Enter" será solicitada para confirmar a operação.

Dica: *para cancelar essa, assim como qualquer outra operação, pressione "Escape".*

Dica: *quando a intenção for apenas encerrar a execução do ambiente, sem que o computador seja desligado, pressione "Control-x Control-c".*

4 Aplicativos

F5 localizada cinco posições à direita da tecla "Escape", essa é a tecla utilizada para iniciar o treino de teclado. Esse aplicativo foi desenvolvido para que o usuário possa fazer com segurança um reconhecimento integral do posicionamento das teclas, que durante a execução do treinamento são desabilitadas de suas funções originais e respondidas pelo sistema de forma auditiva, até que seja pressionada a tecla "Escape" para encerrar a execução do utilitário.

Dica: *esse treino, assim como qualquer outro aplicativo, pode também ser solicitado a qualquer momento e em qualquer situação, mesmo quando interagindo com outros aplicativos.*

Dica: *para que as teclas modificadoras "Control", "Shift" e "Alt" sejam sonorizadas, devem ser pressionadas em conjunto com qualquer outra alfabética.*

F6 localizada logo à direita da tecla "F5", essa é a tecla utilizada para iniciar o editor de textos. Para mais informações consulte a seção "8 Edição de Texto".

F7 localizada logo à direita da tecla "F6", essa é a tecla utilizada para iniciar o gerenciador de arquivos. Para mais informações consulte a seção "9 Gerenciamento de Arquivos".

F8 localizada logo à direita da tecla "F7", essa é a tecla utilizada para iniciar o navegador de *internet*. Para mais informações consulte a seção "10 Acesso à *Internet*".

F9 localizada logo à direita da tecla "F8", essa é a tecla utilizada para iniciar o gerenciador de cartas eletrônicas (*e-mails*). Para mais informações consulte a seção "11 Gerenciamento de *e-mails*".

Alt-x as teclas “F5”, “F6”, “F7”, “F8” e “F9”, são atalhos que foram atribuídos aos aplicativos utilizados com maior freqüência dentro do ambiente “Emacs”, mas além desses, existe uma infinidade de outros utilitários que podem também ser acessados. Para isso, basta pressionar a combinação de teclas “Alt-x”, digitar o nome do utilitário e pressionar a tecla “Enter” para confirmar a execução.

Dica: *existe também a possibilidade de acessar aplicativos não pertencentes ao ambiente “Emacs” (aplicativos externos). Para isso, basta pressionar a combinação de teclas “Alt-Shift-x”, digitar o nome do utilitário e pressionar a tecla “Enter”. Nesse caso, uma nova janela do sistema será criada - “Shell Command Output” - para informar a resposta do aplicativo externo. Para mais informações consulte a seção “12 Perguntas e respostas”.*

5 Gerenciamento de Aplicativos (Janelas)

O “Emacs” é um ambiente multitarefa, ou seja, é um sistema capaz de executar vários aplicativos ao mesmo tempo. Cada um desses aplicativos é representado no ambiente por uma janela do sistema, que pode ser alternada entre as demais. Estando num ambiente multitarefa, para que o usuário possa interagir com uma determinada aplicação é preciso que seja ativada a janela correspondente. As funções para gerenciamento de janelas são:

Control-e f para informar o nome da janela ativada.

Control-e m para ler a barra de estado da janela.

Control-f para localizar de forma incremental no conteúdo da janela.

Control-x Control-p para imprimir o conteúdo da janela.

Control-x Control-w para salvar o conteúdo da janela.

Control-x Control-b para exibir uma lista com as janelas iniciadas. Em seguida use as setas “Abaixo” ou “Acima” para selecionar e pressione “Enter” para alternar.

Dica: *mesmo quando não inicializado nenhum aplicativo, três janelas do sistema estarão sempre listadas nessa relação - “scratch”, “Messages” e “Compile-Log”.*

F4 para encerrar qualquer janela. Localizada quatro posições à direita da tecla “Escape”, essa é a tecla utilizada para encerrar a execução de aplicativos dentro do ambiente. Sempre que acionada essa função, a tecla “Enter” será solicitada para confirmar a operação.

Dica: durante a utilização do ambiente, para relembrar essa lista de funções referente ao gerenciamento de janelas, utilize o atalho "`Control-h F1`".

6 Leitura de Janelas

Em geral as janelas são compostas por informações puramente textuais, independente de qual seja o aplicativo em execução. O processo de leitura desse conteúdo é feito sempre tomando como base o posicionamento do cursor, que é o ponto de referência na janela ativa onde são inseridas ou removidas as informações enviadas pelo teclado. Portanto, as funções para leitura de janelas são:

Seta `Abaixo` **ou** `Acima` para ler a linha seguinte ou a anterior.

`Control-Abaixo` **ou** `Control-Acima` para ler o parágrafo seguinte ou o anterior.

Seta `Direita` **ou** `Esquerda` para ler o caractere seguinte ou o anterior.

`Control-Direita` **ou** `Control-Esquerda` para ler a palavra seguinte ou a anterior.

> ***Dica:*** *como atalho alternativo para o avanço ou retrocesso na leitura de palavras, utilize a tecla "`F3`" ou "`Control-F3`" respectivamente.*
>
> ***Dica:*** *o avanço do final de uma linha para o começo da próxima é feito de forma automática e sinalizado com a emissão de um pequeno* beep.
>
> ***Dica:*** *para posicionar rapidamente o cursor no início ou no final da linha, utilize a tecla "`Home`" ou "`End`", respectivamente.*
>
> ***Dica:*** *para posicionar rapidamente o cursor no início ou no final da janela, utilize a combinação de teclas "`Control-Home`" ou "`Control-End`", respectivamente.*

`Control-e l` para ler a linha sob o cursor.

`Control-e w` para ler a palavra sob o cursor.

`Control-e Shift-w` para soletrar a palavra sob o cursor.

`Control-e c` para ler o caractere sob o cursor.

`Control-e b` para ler a janela toda, mantendo o cursor em sua posição original.

`Control-e n` para ler a janela toda a partir da posição do cursor, também mantendo-o em sua posição original.

Control-e p para pausar a leitura.

Control-e Espa o para continuar a leitura após um comando de pausa.

Dica: durante a utilização do ambiente, para relembrar essa lista de funções referente à leitura de janelas, utilize os atalhos "Control-h F2" e "Control-h F3".

7 Configuração do Leitor

Depois de adquirir prática com a utilização do ambiente e costume com a síntese de voz do sintetizador, algumas características do sistema de leitura podem ser modificadas para agilizar a realização de tarefas. As funções para configuração do leitor são:

Alt-Acima **ou** **Alt-Abaixo** para aumentar ou reduzir o volume em 5 porcento.

Alt-Direita **ou** **Alt-Esquerda** para aumentar ou reduzir a velocidade em 5 porcento.

F12 para alternar entre os leitores português e inglês.

Control-e d k para ativar ou desativar o retorno auditivo de teclas.

Control-e d w para ativar ou desativar o retorno auditivo de palavras.

Control-e d l para ativar ou desativar o retorno auditivo de linhas.

Control-e d s para ativar ou desativar a pronúncia soletrada de letras capitalizadas.

Control-e d , (vírgula) para setar a leitura de pontuação para "nada", ou seja, nenhum caractere de pontuação será informado durante a leitura, além dos pontos de exclamação e interrogação.

Control-e d . (ponto) para setar a leitura de pontuação para "algo", ou seja, apenas alguns caracteres de pontuação serão informados durante a leitura (todos com exceção dos caracteres "ponto", "vírgula" e "ponto e vírgula").

Control-e d ; (ponto e vírgula) para setar a leitura de pontuação para "tudo", ou seja, todos os caracteres de pontuação serão informados durante a leitura.

Dica: durante a utilização do ambiente, para relembrar essa lista de funções referente à configuração do leitor, utilize o atalho "Control-h F4".

8 Edição de Texto

Como visto anteriormente na seção "4 Aplicativos", para iniciar o aplicativo editor de textos basta pressionar a tecla "F6". Por padrão, sempre que iniciado esse aplicativo no ambiente "Emacs", um arquivo novo e vazio é automaticamente carregado para que possa ser editado. As funções para edição de texto são:

Control-x Control-f para abrir um arquivo.

Control-x Control-s para salvar o arquivo.

Control-x Control-w para salvar o arquivo com nome diferente.

Control-x Control-p para imprimir o arquivo.

Control-f para localizar de forma incremental no conteúdo do arquivo.

Control-s para localizar e substituir no conteúdo do arquivo.

Control-Espa o para definir a posição do cursor como o início de uma seleção e o final será limitado pela posição atual do cursor.

Control-e r para ler a região selecionada.

Control-x c para copiar a região selecionada.

Control-x x para recortar a região selecionada.

Control-x v para colar a região copiada.

Control-x u para desfazer a última edição.

Control-e Control-l para ler o número da linha atual.

Control-e = (igual) para ler o número da coluna atual.

Dica: durante a utilização do editor de textos, para relembrar essa lista de funções utilize os atalhos "F1" e "F2".

Dica: a inserção e remoção de informações no conteúdo do texto é feita obviamente através do teclado (que funciona exatamente como uma "máquina de escrever"), sendo a inserção realizada por intermédio das teclas "alfanuméricas" e a remoção por intermédio das teclas "Delete" (que apaga o caractere exatamente sob o cursor), "backspace" (que

apaga o caractere que antecede o cursor) ou pela combinação "Control-Backspace" (que apaga a palavra que antecede o cursor). Para um melhor controle do usuário, sempre que apagada uma determinada informação, essa será também pronunciada.

9 Gerenciamento de Arquivos

Como visto anteriormente na seção "4 Aplicativos", para iniciar o aplicativo gerenciador de arquivos basta pressionar a tecla "F7". Por padrão, sempre que iniciado esse aplicativo no ambiente "Emacs", o diretório padrão do usuário será carregado para que seus arquivos e diretórios possam ser gerenciados. As funções para gerenciamento de arquivos são:

Seta `Abaixo` **ou** `Acima` para avançar ou retroceder o cursor pelos arquivos e diretórios. Um bip será sempre emitido antes de anunciar os diretórios para diferenciá-los dos arquivos.

`PageDown` **ou** `PageUp` para avançar ou retroceder o cursor em 10 posições.

`Enter` para acessar o diretório ou editar o arquivo sob o cursor.

`Backspace` para sair do diretório e explorar um nível anterior.

`g` para informar o caminho completo e atualizar o diretório em exploração.

`Shift-c` para copiar o arquivo sob o cursor.

`Shift-d` para deletar o arquivo sob o cursor.

`Shift-p` para imprimir o arquivo sob o cursor.

`Shift-r` para renomear o arquivo ou diretório sob o cursor.

`+` (soma) para criar um diretório.

`Shift-x` para executar um aplicativo com o arquivo sob o cursor e durante a execução, pressione Control-g para encerrar.

`z` para informar o tamanho do arquivo ou diretório sob o cursor.

`Shift-z` para compactar ou descompactar o arquivo no formato "gz".

Dica: durante a utilização do gerenciador de arquivos, para relembrar essa lista de funções utilize os atalhos "F1" e "F2".

Dica: o diretório padrão do usuário é a pasta "`/home/nome-do-usuario/`" que também pode ser representado pela abreviação " `~/` ". Portanto, o caminho completo de um arquivo armazenado nessa pasta pode ser representado tanto por "`/home/nome-do-usuario/arquivo`" quanto por " `~/arquivo` ".

10 Acesso à *internet*

Como visto anteriormente na seção "4 Aplicativos", para iniciar o aplicativo navegador de *internet* basta pressionar a tecla "`F8`". As funções para acesso à *internet* são:

`g` para acessar um novo endereço.

`w` para salvar o conteúdo do endereço atual.

`Tab` **ou** `Shift-Tab` para posicionar o cursor no link seguinte ou no anterior.

`d` para fazer download do endereço do link sob o cursor.

`Enter` para acessar o endereço do link sob o cursor.

`Shift-b` para voltar ao endereço anterior.

`c` para informar e copiar na área de transferência o endereço atual.

`u` para informar e copiar na área de transferência o endereço do link sob o cursor.

`Shift-r` para recarregar o conteúdo do endereço atual.

`]` **ou** `[` (fecha colchete ou abre colchete) para posicionar o cursor no campo de formulário seguinte ou no anterior.

`Control-c Control-c` para enviar o formulário.

`a` para adicionar o endereço atual à lista de favoritos.

`Alt-a` para adicionar o endereço do link sob o cursor à lista de favoritos.

`v` para acessar a lista de favoritos. Na lista de favoritos pressione "`Enter`" para acessar ou "`Control-k`" para excluir o registro sob o cursor.

Dica: durante a utilização do navegador de *internet*, para relembrar essa lista de funções utilize os atalhos "`F1`" e "`F2`".

11 Gerenciamento de *e-mails*

Como visto anteriormente na seção "4 Aplicativos", para iniciar o aplicativo gerenciador de *e-mails* basta pressionar a tecla "F9". Por padrão, sempre que iniciado esse aplicativo no ambiente "Emacs", o conteúdo do primeiro *e-mail* é exibido para que possa ser gerenciado. As funções para gerenciamento de *e-mails* são:

m para enviar um novo email.

g para receber novos emails.

n para ler o próximo email.

p para ler o email anterior.

r para responder o email selecionado.

f para encaminhar o email selecionado.

d para deletar o email selecionado.

Dica: durante a utilização do gerenciador de *e-mails*, para relembrar essa lista de funções utilize o atalho "F1".

Dica: para mais informações sobre como configurar o gerenciador de *e-mails* consulte a seção "12 Perguntas e respostas".

12 Perguntas e Respostas

12.1 Como adicionar (instalar) novos aplicativos?

A instalação de novos aplicativos, quando utilizando o sistema operacional "Linux Debian" ou derivados, é feita através da *internet* com o auxílio do aplicativo externo "apt-get" acompanhado da opção "install" e do "nome do aplicativo" a ser instalado, da seguinte forma:

- Pressione a combinação de teclas "Alt-Shift-x" para solicitar a execução do aplicativo externo;
- Digite "apt-get install nome-do-aplicativo" e pressione a tecla "Enter" para confirmar.

- ***Dica:*** *para acompanhar o resultado da solicitação, analise o conteúdo da janela do sistema "`Shell Command Output`", que é automaticamente criada no momento da solicitação.*

12.1 Como remover (desinstalar) aplicativos?

A desinstalação de aplicativos, quando utilizando o sistema operacional "Linux Debian" ou derivados, é feita também com o auxílio do aplicativo externo "`apt-get`", acompanhado da opção "`remove`" e do "`nome do aplicativo`" a ser removido, da seguinte forma:

- Pressione a combinação de teclas "`Alt-Shift-x`" para solicitar a execução do aplicativo externo;
- Digite "`apt-get remove nome-do-aplicativo`" e pressione a tecla "`Enter`" para confirmar.

12.1 Como acessar arquivos no formato "PDF"?

Um arquivo no formato "PDF" é na realidade uma seqüência de imagens. Para que o conteúdo desse arquivo possa ser acessado é necessário que seja submetido a uma conversão para o formato texto com auxílio do aplicativo externo "`pstotext`" (ps to text), da seguinte forma:

- Pressione a combinação de teclas "`Alt-Shift-x`" para solicitar a execução do aplicativo externo;
- Digite "`pstotext arquivo.pdf > arquivo.txt`" e pressione a tecla "`Enter`" para confirmar;
- Acesse o conteúdo já convertido para texto, abrindo com o editor de textos o recém criado arquivo.txt.

12.1 Como produzir textos com formatação (LaTeX)?

O processo de produção de texto com formatação é exatamente o inverso do descrito na resposta da pergunta anterior, ou seja, é uma conversão de um arquivo de formato texto para o formato "PDF". Para que essa conversão seja realizada o arquivo de texto deve ser produzido de acordo com as regras de conversão "LaTeX" (http://www.latex-project.org) e em seguida seja convertido com o auxílio dos aplicativos externos "latex" e "dvipdf", da seguinte forma:

- Produza com o editor de textos um arquivo.txt de acordo com as regras de conversão "LaTeX";
- Pressione a combinação de teclas "`Alt-Shift-x`" para solicitar a execução do aplicativo externo;
- Digite "`latex arquivo.txt && dvipdf arquivo.dvi arquivo.pdf`" e pressione a tecla "`Enter`" para confirmar;

12.1 Como produzir músicas e partituras (MusiXTeX)?

O processo de produção de músicas e partituras é semelhante ao de produção de textos com formatação, ou seja, inicialmente deve ser produzido um arquivo de texto de acordo com regras específicas de conversão, nesse caso de acordo com regras"MusiXTeX" (http://icking-music-archive.org/software/indexmt6.html) e em seguida, a conversão deve ser realizada com o auxílio dos aplicativos externos "latex" e "dvipdf".

- ***Dica:*** *para que conversões "MusiXTeX" sejam realizadas é necessário que o aplicativo "*`musixtex`*" seja inicialmente instalado.*

12.1 Como executar músicas (MP3) ou vídeos (AVI ou MPEG)?

A execução de qualquer tipo de arquivo multimídia, seja ele de áudio ou de vídeo, pode ser realizada através do gerenciador de arquivos com o auxílio do aplicativo externo "`mplayer`", da seguinte forma:

- Pressione a tecla "`F7`" para iniciar o gerenciador de arquivos e posicione o cursor sob o arquivo que deseja executar;
- Pressione a combinação de teclas "`Shift-x`" para solicitar a execução de um aplicativo com o arquivo sob o cursor;
- Digite "`mplayer`" e pressione a tecla "`Enter`" para confirmar. Durante a execução pressione "`Control-g`" para encerrar.

12.1 Como tocar um CD ou um DVD?

Assim como os arquivos multimídia, os CDs e DVDs também podem ser tocados com o auxílio do aplicativo externo "`mplayer`", mas nesse caso não precisam ser executados a partir do gerenciador de arquivos, pois podem ser solicitados a qualquer momento, da seguinte forma:

- Pressione a combinação de teclas "`Alt-Shift-x`" para solicitar a execução do aplicativo externo;
- Digite "`mplayer cdda://`" ou "`mplayer dvd://1`" para tocar um CD ou um DVD respectivamente e pressione a tecla "`Enter`" para confirmar. Durante a execução pressione "`Control-g`" para encerrar.

12.1 Como efetuar uma conexão discada para *internet*?

A conexão discada para *internet* pode ser realizada com o aplicativo externo "`wvdial`", da seguinte forma:

- Pressione a combinação de teclas "`Alt-Shift-x`" para solicitar a execução do aplicativo externo;
- Digite "`wvdial &`" e pressione a tecla "`Enter`" para confirmar;
- ***Dica:*** *para finalizar a conexão basta encerrar a janela do sistema "`Async Shell Command`", que é automaticamente criada no momento da solicitação.*