UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE FÍSICA

Linux Para Comunidade

Átila Bohlke
Fabricio Ferrari
Leonardo Brunnet
Marlon Schmitz
Rodrigo Parizoto

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INSTALANDO & CONFIGURANDO

1.1 História ©2000 LEONARDO G. BRUNNET

Embora o Linux seja um produto dos anos 90, sua base, o sistema **Unix**, é bem mais antiga, sua origem datando dos anos 60 quando os Laboratórios Bell, a General Electric e o Massachusetts Institute of Technology (1965) procuravam criar um ambiente multi-usuário e multi-tarefa para ser usado em grandes computadores. Embora rodando em um computador GE 645, o assim chamado **Multics** foi abandonado pela Bell no final da década (1969) por não ter cumprido os objetivos do projeto original.

Muitos dos técnicos que participaram do projeto **Multics** envolveram-se com o desevolvimento do **Unix**. O **Unix** é visto como um produto intelectual de Ken Thompson e Dennis Ritchie, cientistas de computação que trabalhavam nos Laboratórios Bell e haviam trabalhado anteriormente no projeto **Multics**.

Inicialmente escrito em assembly, torna-se transportável quando Dennis Ritchie o escreve na linguagem **C** (1973) permitindo que ele seja instalado em uma máquina que possua um compilador C. A partir daí difundiu-se por algumas universidades e instituições comerciais, Berkeley fez o **BSD** (de domínio público). Outra versão, **Unix System V** foi desenvolvida pelos Laboratórios Bell e liberada em janeiro de 1983. Hoje há contribuições de variadas instituições, comerciais ou não, e roda em variados hardwares de PC's a supercomputadores contando com anos de depuração.

O que é Unix?

É um sistema operacional, um conjunto de programas projetados para controlar as interações das funções de baixo nível com as **aplicações** (programas) e com os usuários. O Unix controla os recursos do computador, distribui as tarefas entre os diferentes usuários, controla a ordem das tarefas. Sua estrutura pode ser visualizada pela figura.

Origem do Linux

O sistema original Linux surgiu da fusão de dois trabalhos: o projeto GNU e o KERNEL Linux. A idéia fundamental do projeto GNU esta expressa "Manifesto GNU"escrito em 1985 por Richard Stallman. GNU é uma abreviatura em inglês auto-referente que explicitada torna-se GNU is Not Unix. Abaixo algumas palavras de Stallmann expressando suas idéias a respeito de programas livres:

Why I Must Write GNU. *I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement. For years I worked within the Artificial Intelligence Lab to resist such tendencies and other inhospitalities, but eventually they had gone too far: I could not remain in an institution where such things are done for me against my will. So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. I have resigned from the AI lab to deny MIT any legal excuse to prevent me from giving GNU away.*

No período de 1991 a 1993 um estudante de informática, Linus Torvald, em um projeto paralelo à sua graduação, desenvolveu a partir do zero um KERNEL dentro do espírito Unix, um clone de Unix. Este kernel veio suprir a necessidade da comunidade de usuários de Unix, em geral constituída de pessoas que trabalham em grandes computadores e que não tinham a sua disposição um sistema operacional Unix para seus computadores pessoais, isto é, necessitavam de um "UNIX"que rodasse em pc's . Estabeleceu-se, então uma colaboração espontânea internacional, em particular envolvendo universidades, com o intuito de aprimorar o sistema desenvolvido por Torvald. É claro que, neste contexto, a fusão com o projeto GNU seria natural. Este KERNEL é o programa base que estabelece a relação entre o usuário, as tarefas do sistema e suas entradas e saídas, é este programa que possui as propriedades de multitarefa e multiusuário mencionadas. A união destes projetos é a base do sistema Linux.

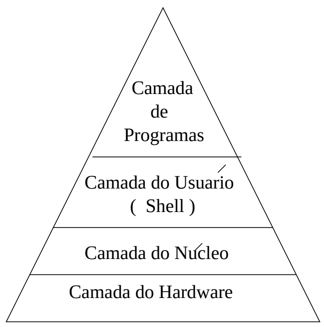


Fig. 1.1: Arquitetura em camadas do Unix

1.2 Defrag ©2000 Marlon Schmitz

O DEFRAG é um utilitário de desfragmentação de disco. Você irá utilizá-lo para mover **todos** os dados do disco para o início. Isto pois o programa FIPS apenas redivide uma partição já existente sem danificar seus dados (isto se você usar o DEFRAG).

1.3 FIPS e similares ©2000 Marlon Schmitz

Como já foi dito o FIPS é um programa que divide(split) uma partição já existente em duas sem danificar os dados já contidos no HD, isto se eles não estiverem no local onde será criada a nova partição.

Ao se dividir a partição deve-se ter o cuidado de que os dados (pelo menos os vitais) estejam alocados no início do disco, pois o FIPS, quando edita o setor de boot, cria uma nova FAT **vazia** para a nova partição. Com isto os dados que estejam alocados onde foi criada a nova partição serão perdidos. Contudo este programa faz um **backup** do setor de boot antigo possibilitando este ser recuperado.

Atenção!!! Isto não quer dizer que se você dividiu uma partição acidentalmente e com isso perdeu dados irá recuperá-los, o FIPS só lhe permite reconstituir o setor de boot e a FAT não os dados perdidos.

Atenção: FIPS não "splita" (divide) partições *LINUX* e também não trabalha com partições estendidas!

Se você usa algum utilitário contra formatação acidental deverá deletar os arquivos ocultos que este utilitário provavelmente irá colocar no final do seu HD. Não se preocupe estes arquivos serão criados novamente.

Vamos sonsiderar exemplo, utilizando um disco de $\approx$ 120Mb.

Primeiro o FIPS irá lhe mostrar algumas mensagens de abertura como as vistas abaixo:

```
FIPS version 1.5, Copyright (C) 1993/94 Arno Schaefer
     FAT32 Support Copyright (C) 1997 Gordon Chaffee

DO NOT use FIPS in a multitasking environment like Windows, OS/2, Desqview,
Novell Task manager or the Linux DOS emulator: boot from a DOS boot disk first.

If you use OS/2 or a disk compressor, read the relevant sections in FIPS.DOC.

FIPS comes with ABSOLUTELY NO WARRANTY, see file COPYING for details
This is free software, and you are welcome to redistribute it
under certain conditions; again see file COPYING for details.

Press any Key

WARNING: FIPS has detected that it is running under MS-Windows version 4.0
FIPS should not be used under a multitasking OS. If possible, boot from a DOS
disk and then run FIPS. Read FIPS.DOC for more information.

Do you want to proceed (y/n)? y
```

Após ele irá lhe perguntar qual o disco em que você vai trabalhar:

`Which Drive (1=0x80/2=0x81)?` *2*

Então ele irá lhe dar diversas informações sobre o setor de boot do disco que você escolheu e fará alguns testes:

```
Partition table:

     |        |      Start      |      |       End       | Start  |Number of|
Part.|bootable|Head Cyl. Sector|System|Head Cyl. Sector| Sector |Sectors  |  MB
-----+--------+----------------+------+----------------+--------+---------+----
1    |      no |   1    0       1|   06h|  13  1022      17|      17|   243457| 118
2    |      no |   0    0       0|   00h|   0     0       0|       0|        0|   0
3    |      no |   0    0       0|   00h|   0     0       0|       0|        0|   0
4    |      no |   0    0       0|   00h|   0     0       0|       0|        0|   0

Checking root sector ... OK

Press any Key

Boot sector:

Bytes per sector: 512
Sectors per cluster: 4
Reserved sectors: 1
Number of FATs: 2
Number of rootdirectory entries: 512
Number of sectors (short): 0
Media descriptor byte: F8h
Sectors per FAT: 238
Sectors per track: 17
Drive heads: 14
Hidden sectors: 17
Number of sectors (long): 243457
Physical drive number: 00h
Signature: 29h

Checking boot sector ... OK
Checking FAT ... OK
Searching for free space ... OK
```

Então ele criará um arquivo com definições para se recuperar o setor de boot(rootboot.xxx), onde "xxx" é um número de *000* até *999*, ou seja você pode ter vários arquivos de backups em um mesmo lugar.

```
Do you want to make a backup copy of your root and boot sector
before proceeding (y/n)?  y
```

Esteja certo de que o disco que está no drive `A:` é botável(disco):

```
 Do you have a bootable floppy disk in drive A: as described
in the documentation (y/n)?   y


    Writing file a:\rootboot.000
```

Assim ele irá lhe mostrar a tela de edição de partições:

```
Enter start cylinder for new partition (71 - 1022):

Use the cursor keys to choose the cylinder, <enter> to continue

Old partition      Cylinder       New Partition
    8.3 MB            71            110.6 MB
```

Com as setas você altera o tamanho das partições...

```
Use the cursor keys to choose the cylinder, <enter> to continue

Old partition      Cylinder       New Partition
   59.4 MB           511             59.5 MB
```

Quando satisfeito, pressione enter :

```
First Cluster: 30275
Last Cluster: 60738


Testing if empty ... OK

New partition table:

     |        |     Start      |      |      End       | Start  |Number of|
Part.|bootable|Head Cyl. Sector|System|Head Cyl. Sector| Sector |Sectors  |  MB
-----+--------+----------------+------+----------------+--------+---------+----
1    |     no |  1    0       1|   06h|  13  510     17|      17|   121601|  59
2    |     no |  0  511       1|   06h|  13 1022     17|  121618|   121856|  59
3    |     no |  0    0       0|   00h|   0    0      0|       0|        0|   0
4    |     no |  0    0       0|   00h|   0    0      0|       0|        0|   0

Checking root sector ... OK

Do you want to continue or reedit the partition table (c/r)? c

New boot sector:

Bytes per sector: 512
Sectors per cluster: 4
Reserved sectors: 1
Number of FATs: 2
Number of rootdirectory entries: 512
Number of sectors (short): 0
Media descriptor byte: F8h
Sectors per FAT: 238
Sectors per track: 17
Drive heads: 14
Hidden sectors: 17
Number of sectors (long): 121601
Physical drive number: 00h
Signature: 29h

Checking boot sector ... OK

Ready to write new partition scheme to disk

Do you want to proceed (y/n)? y
```

Então o setor de boot será reeditado e o seu HD está pronto.

Se por algum acaso você fizer alguma "besteira", é melhor não tentar reeditar para deixar como era antes, use o `restorrb.exe` para recuperar o backup do setor de boot antigo.

Este utilitário utiliza os arquivos de backup criados pelo FIPS para recuperar setores de boot editados de forma incorreta. Ele é extremamente simples de se usar basta executá-lo e ele lhe perguntará qual o arquivo que você quer utilizar para restaurar. Veja abaixo:

```
FIPS version 1.5, Copyright (C) 1993/94 Arno Schaefer
Module RESTORRB.EXE - Please read the file README.1ST
FIPS comes with ABSOLUTELY NO WARRANTY, see file COPYING for details
This is free software, and you are welcome to redistribute it
under certain conditions; again see file COPYING for details.

Found save file a:\rootboot.000
Found save file a:\rootboot.001
Found save file a:\rootboot.002
Found save file a:\rootboot.003

Which file do you want to restore (0/1/2/3)?
```

Assim você restaura o setor de boot editado acidentalmente.

Existe uma forma bem simples de iniciar o sistema base de instalação do *LINUX* sem precisar gerar o disco de inicialização, basta utilizar o `loadlin.exe`. Na realidade no mesmo lugar(CDROM) onde você consegue o FIPS e os demais programas existe um diretório chamado "autoboot", copie este diretório o loadlin.exe e o autoboot.bat para um disquete de boot para DOS. Basta dar boot com este disquete e depois rodar o autoboot.bat que o sistema básico de instalação do *LINUX* será inicializado.

```
a:\autoboot.bat
```

```
a:\loadlin autoboot\vmlinuz initrd=autoboot\initrd.img
```

1.4 Fontes ©2000 Marlon Schmitz

O *LINUX* pode ter seus arquivos de instalação vindos de vários lugares como CDROM, via FTP, HD, etc... Existem até algumas distribuições como a Slackware e a Debian que instalam o sistema base via disquetes e depois você decide quais os pacotes que quer instalar.

1.5 Hardware ©2000 Marlon Schmitz

Antes de se começar uma instalação, seja qual for o sistema operacional, devemos fazer três perguntas muito importantes:

1.5.1 *Qual é o meu HARDWARE?*

Devemos saber de antemão o máximo possível de especificações sobre a nossa máquina possíveis, para não termos surpresas na hora de instalar drivers e/ou configurar dispositivos. Apesar de que atualmente as distribuições mais conhecidas de *Linux*(RedHat e Mandrake) reconhecem quase todos os dispositivos do sistema(placa de rede, placa de vídeo, modens,gravadores de cd, etc ...). Posso aqui citar as partes do hardware mais importantes junto com as especificações, também, mais importantes dos mesmos:

- Discos: tipo, tamanho, o famoso *CHS*(números de Cilindros/Cabeças/Sectores);
- Monitor: freqüências(taxa de atualização), resolução;
- : placa de rede e/ou fax-modem;
- Placa de vídeo: modelo, freqüência, memória, suporte de cores;
- Placa de Som: modelo, número de irq, endereço dma, portas I/O ;
- Mouse (apesar de este ser um dispositivo dos mais generalizados possíveis é melhor garantir);
- Configurações de internet(provedor, ip number, etc...).

O último item possui um programa que pode lhe auxiliar na configuração chama-se Xisp, existem outros, mas na realidade o que muda basicamente é a interface gráfica que ele utiliza. Enfim podemos citar inúmeras informações que deveriam ser conhecidas antes de se começar a pensar em instalar um novo sistema operacional, mas ficaremos apenas com este número para que não se crie confusão desnecessária.

1.5.2 Qual é o tipo de instalação que devo fazer?

Para quem não sabe, o sistema UNIX é uma ferramenta muitíssimo poderosa para se trabalhar em rede, servindo área em disco, programas, periféricos, base de dados, etc.. Logo, é de certa forma muito lógico que os seus variantes também o façam e o *LINUX* não foge a regra.

Quando se pensa em instalar alguma das várias distribuições de *LINUX* que existem por aí, deve-se ter em mente o que o seu computador irá fazer. Se por exemplo, você está pensando em montar uma rede, seja ela de qual porte for, deve ter em mente que terá no mínimo dois tipos de instalação a fazer: Server e WorkStation.

Agora se você quer apenas ter um micro que irá servir só para se editar textos, criar planilhas, fazer programas, criar páginas em HTML, etc.. é melhor escolher um tipo de instalação que seja compatível com o seu trabalho. Não há propósito nenhum em se instalar um servidor de e-mail se você nem sequer está conectado ao mundo exterior.

Assim é bom se ter em mente, também, o que você pretende para o seu micro computador, para que não sejam instaladas coisas que na melhor das hipóteses so irão lhe ocupar espaço em disco. Por exemplo, citarei algumas coisas mais específicas em que você pode pensar antes de começar a instalação:

- Qual tipo de estação: servidor ou estação de trabalho ou doméstico;
- Serviço de e-mail: preciso que meu pc seja servidor de e-mail para que eu possa me comunicar com o mundo exterior visto que meu pc é possante e não preciso(ou tenho) um provedor, ou vou apenas mandar mensagens para eu mesmo da minha máquina para ela mesma ???
- NetWork: preciso instalar servidores de acesso remoto e rede em geral para que eu possa compartilhar impressoras discos, programas, ou só estou intalando um programa que vai ficar o tempo todo como um latifundiário de memória e/ou devorador de CPU ???
- Programas: preciso mesmo de "N+1" programas que executam a mesma terefa pois os que irão trabalhar no pc tem idéias muito diferentes ou só estou colocando porque irá me trazer alguma forma de "status" social no clube de bocha ???
- WindowManagers: tenho necessidade de ter trinta gerenciadores de área de trabalho pois pessoas de gostos muito variados trabalharão na minha máquina, ou somento porque eu aprendi a fazer um "script" que troca a minha interface a cada login???
- Etc... preciso instalá-los ou o redator deste texto esta apenas divertindo-se as minhas custas ???

Logicamente eu estou sendo espirituoso para que o texto não fique enfadonho, mas é extremamente relevante fazer estas perguntas para que não se perca espaço em disco e/ou tempo de CPU com programas que não preenchem nossas necessidades.

1.5.3 O que há no meu pc ?

Se nada, ótimo!!! O *LINUX* não encontra problema em trabalhar sozinho sem niguém para atrapalhar, basta você ter o *boot disk* e dizer de onde virão os arquivos de instalação do sistema, ou seja CDROM, rede, outro HD, 387 disquetes formatados em 1.68Mb, etc...

Em geral é possível butar via CDROM,mas se você, por algum motivo qualquer, não os tiver também não há problema(no máximo transtorno), basta criá-los em outra máquina *LINUX*, usado o DD, ou no DOS mesmo usando o RaWrite.

No CDROM de qualquer distribuição do *LINUX* existe um diretório chamado *images* que possui dois arquivos: boot.img e rescue.img, estes arquivos são “imagens” de discos, respectivamentes, de boot e de rescue(resgate). Existem outras imagens para outros tipos de instalaçã como por exemplo em laptop's(pcmcia.img).

Com o comando “dd” (assim mesmo em minúculas) você aplica esta imagem em um disco de destino. A respectiva linha de comando (no prompt do *LINUX*) está abaixo:

`dd if=`⟨arquivo de imagem⟩ `of=`⟨disco de destino⟩

no caso mais comumente usado você fará:

`dd if=boot.img of=/dev/fd0`

assim você cria um disco de inicialização para instalação do *LINUX*.

No MS-DOS (ou Windows 9X) use o rawrite (aí não faz diferença minusculas, maiúsculas, misturadas...):

`rawrite -f boot.img -d a:`

ou

`rawrite -h` para obter ajuda

Se já existe um outro sistema, seja ele qual for, é muitíssimo prudente que se faça backup dos arquivos de configurações principais (pelo memos), é também uma atitude de sabedoria fazer backup das partições e setores de boot.

1.6 Acentos ©2000 Rodrigo Parizotto

Se quisermos escrever um texto em português, é inevitável o uso de acentuação. Apesar do problema não estar completamente solucionado, existe solução para um poderoso editor, o **Emacs**. Deixaremos a disposição um arquivo que pode ser interpretado por este editor, permitindo a acentuação normal. A equipe do **servidor X** pensou que haveria mais versatilidade caso o aplicativo gerenciasse a composição de teclas como 'a. Como a maioria dos aplicativos foram feitos para a língua inglesa, é usual não conseguirmos acentuar em vários aplicativos. Foi desenvolvido um programa chamado **Xkeycaps** com a intenção de corrigir a configuração do teclado.

1.6.1 Xkeycaps ©2000 Átila Bohlke

Este programa nos permite configurar um teclado, tecla-a-tecla. Com ele é possível especificar exatamente cada uma das teclas de nosso teclado. Uma vez instalado o programa, iniciamos ele através do comando

`$ xkeycaps &` $\boxed{\text{enter}}$

dentro do X. Na tela que se abre, nos é dada a opotunidade de escolhermos qual teclado é o mais semelhante ao nosso.

Na verdade não importa muito que ele seja ligeiramente diferente, uma vez que podemos mesmo mudar toda sua configuração. O mais interessante é que ele seja o mais semelhante possivel. De maneira geral, o *pc 105 key* ou o *pc 104 key* é uma boa escolha nos computadores mais modernos. Na parte inferior dessa tela, existe um "desenho"do teclado que serve de guia.

Uma vez escolhido o teclado, somos apresentados com uma tela onde existe um desenho do teclado, com cada uma das teclas discriminadas. No canto superior direito, existe um curto menu, onde algumas opções são meio óbvias, como *quit* e *Select Keyboard*, e outras nem tanto. *Type at Window* nos permite testar uma determinada tecla. Quando clicamos ali a opção fica escurecida até escolhermos uma janela para a qual o resultado de uma determinada tecla será enviado. Escolhida a janela, clicando sobre uma determinada tecla, o seu resultado será redirecionado para a janela-teste. *Restore default map* recarrega o mapa padrão para aquele teclado e desfaz qualquer mudança que tenhamos efetuado. *Write Output* escreve as configurações feitas para um arquivo de configuração que pode ser lido pelo X server.

Configurando um acento "morto"

Vamos agora configurar uma tecla de acentuação dita "morta", isto é, quando ela é teclada, o computador fica esperando outra letra ser digitada para então apresentar a combinação de ambas. Escolha a tecla a ser alterada. Para ter certeza que é a tecla correta, utilize o recurso de direcionar a saída para uma janela, por exemplo um *xterm*.

Tendo escolhido a tecla correta, clique com o botão direito do mouse sobre a mesma, e abre-se um menu. Neste menu escoha *Edit Keysyms of Key* e abre-se a janela de edição da tecla. Nessa tela vamos redefinir o símbolo ' (apóstrofe) da respectiva tecla para ser um acento agudo morto, e não o apóstrofe.

Selecione o Keysym 1: apostrophe. Na coluna "Caracter Set"escolha **ISO**. Finalmente na coluna keysym escolha **dead_acute**

Pronto a tecla está redefinida como um acento agudo morto, ou seja, é preciso teclar outra letra depois dele para que tenhamos a combinação (se possível) de ambas.

Produzindo um ç

Mas dessa maneira, não podemos produzir um ç... como fazer então?

Bem, podemos atribuir à tecla "windows"uma nova função... e usá-la para produzir um ç. Primeiro vamos definir ela como um modificador, para o mod3, podendo assim acessar os keysym 3 e 4 das demais teclas.

Para tanto selecionamos *Mode_Switch* no *keysym1* dela e depois no modes, selecionamos *mod3*.

Agora vamos até a tecla c, onde vamos mudar os seus Keysym 3 e 4 para imprimirem um ç quando acionada junto com a tecla "windows"e um Ç quando acionada juntamente com as teclas "windows"e shift, respectivamente. O procedimento é semelhante ao do acento agudo feito anteriormente, mudando apenas o keysym a ser selecionado. Escolhido o keysym a ser alterado, escolhemos **Latin1** em Caracter Set e ccedilla em keysym ou Ccedilla para a produzirmos o símbolo minúsculo ou maiúsculo.

Usando o mapa de teclado no X

Agora vem a grande vantagem de se usar esse método aparentemente tão complicado. Vamos poder utilizar esse mapa de teclado sem precisar nos preocupar com ele no futuro.

Para tanto, vamos salvar o nosso arquivo alterado, na opção *Write Output*, e aceitemos o nome que o programa nos propõem. É um nome um tanto esdrúxulo, por isso vamos alterá-lo para `.Xmodmap` (arquivos que começam com . são arquivos oculto; este deve deve estar no nosso diretorio home) de forma que sempre que iniciarmos o ambiente X, este vai automaticamente ler este arquivo e configurar o teclado de forma adequada.

Na interface de terminal também há problema de acentuação, não se restringindo apenas ao processo, mas também à visualização. Quanto à visualização, podemos corrigir alterando no seu diretório o arquivo

```
.profile
```

acrescentando as seguintes linhas:

```
set meta-flag on
set convert-meta off
set output-meta on
export LANG=pt_BR
export LC_CTYPE=ISO-8859-1
export LESSCHARSET=latin1
```

Esta solução para a interface de terminal está baseada na versão Linux da Conectiva.

1.7 Consoles ©2000 Rodrigo Parizotto

O console é o que se conhece por conexão direta ao sistema via monitor e teclado. Como o sistema Unix é multiusuário, é possível que outros usuários estejam conectados através de terminais, mas aí não estariam via console.

É possível numa mesma máquina termos conexões de **login** diferentes usando um recurso que possibilita a existência de consoles virtuais. Podemos alternar de console virtual, pressionando-se as teclas

Alt e Fn (n é um número). Podemos ter até 8 VC funcionando simultaneamente.

1.8 Senhas

©2000 Rodrigo Parizotto

Para que um sistema multiusuário funcione bem é necessário que haja algum tipo de privacidade entre os usuários e principalmente restrições de acesso ao sistema. Imagine que um usuário apague por discuido um arquivo do sistema. O sistema corre o risco de não funcionar corretamente, prejudicando a todos os usuários. Além disso, escrevemos arquivos que não gostaríamos que outras pessoas alterassem ou apagassem. O sistema Linux funciona de tal maneira que os usuários são diferenciados por um **username**, que está associado a uma senha de acesso. Para nos conectarmos ao sistema, devemos informar ambos, e o sistema confere a autenticidade, permitindo ou não o acesso. Exemplo:

```
pclief7 login: aluno01
Password:
```

Existe um **username** reservado a quem tem acesso irrestrito ao sistema, chamado **root**. Mesmo que um usuário normal entre como **root** terá o poder de alterar qualquer coisa e por isso é preciso ter cuidado e evitar o uso desta conta. O comando usado para trocar a senha é

```
passwd
```

Escolher uma senha exige um cuidado especial. É preferivel usar combinações de letras e números, não usar palavras de dicionários, etc.

1.9 Desligando

©2000 Rodrigo Parizotto

Assim como devemos nos "logar" quando quisermos usar o sistema, devemos nos "deslogar" ao terminar. Isto se faz através do comando

```
logout
```

Antes de **desligar o computador** devemos avisar o Linux para que salve o necessário e organize o encerramento do que estiver pendente. Para isto usa-se o comando

```
shutdown -h now
```

ou

```
halt
```

É extremamente importante este procedimento, pois poderá haver sérios erros se a máquina for simplesmente desligada.

1.10 Partições e Sistemas de Arquivos ©2000 Fabricio Ferrari

1.10.1 O que são?

Partições são divisões lógicas numa mesma unidade de disco. Para que um disco seja utilizado, ele precisa primeiramente ser particionado e depois formatado, que é quando os espaços utilizáveis são alocados e endereçados. Embora fisicamente alocadas na mesma unidade, as partições comportam-se como discos rígidos independentes. Pode-se ter mais de uma partição no mesmo computador por questões de segurança, por economia de espaço em alguns Sistemas Operacionais (**SO**) ou para compartilhar dois SO num mesmo disco, como é o caso que deve interessar a muitos.

O primeiro setor do disco rígido contém o *master boot record* (**MBR**) e a tabela de partição. Ambos contém a localização no disco de quem irá inicializar o sistema e as informações de tamanhos e tipos das partições, respectivamente. As partições dividem-se em primárias, estendidas e lógicas. Por uma limitação de tamanho na tabela de partição, só são possíveis 4 primárias num disco rígido, por este motivo criaram-se as estendidas e lógicas, onde a estendida não guarda dados, mas serve como suporte para armazenar várias partições lógicas dentro de si.

Existem várias maneiras de se referir a uma partição, por exemplo, alguns sistemas operacionais utilizam letras para designar suas diferentes partições, como no caso do MS-DOS (`a:` `b:` `c:` `d:`). No mundo UNIX todo o conjunto de arquivos de um diretório pode estar associado a um **sistema de arquivos**, como é o caso de `/usr`, `/tmp` e assim por diante. Cada um destes sistemas de arquivos vive numa partição separada, portanto se quisermos colocar `/` e `/usr` em sistemas de arquivos separados, deveremos ter uma partição para cada um deles.

O mais importante destes é o **diretório raiz** (*root*) `/`, que está no nível mais alto da árvore de diretórios, e que é o mínimo indispensável para instalação e funcionamento o Linux. Além do raiz, muitas vezes é aconselhável criar uma partição de **swap**, que servirá como memória auxiliar em disco. Embora este também possa ser utilizado através de um arquivo, é mais seguro reservar uma partição específica para seu uso. Resumindo, a configuração mínima é a de uma partição para o diretório raiz, que conterá tanto os arquivos de sistema como os pessoais, e uma para o *swap*.

Antes de alterar suas partições, esteja ciente de que você estará sujeito a perder todo o conteúdo do seu disco, portanto **FAÇA CÓPIAS DE SEUS ARQUIVOS** antes que o pior aconteça. Exitem aplicativos que redimensionam partições sem alterar seu conteúdo (veja Seção 1.3 sobre o `fips`), mas sua eficácia não é absoluta, portanto **MODIFIQUE AS PARTIÇÕES SOB SEU PRÓPRIO RISCO**. Se você quiser evitar reparticionar seu disco, procure uma das distribuições do Linux que funcionam numa partição MS-DOS ou similares[1].

[1] Uma das distribuições testadas pelo autor e bastante completa é a PhatLinux v3.0

Deste modo, basta colocar os arquivos num diretório da árvore MS-DOS e executar um aplicativo para ler o *kernel* do sistema, entretanto não espere desempenho sequer comparável ao método usual explicado nesta seção.

Cada sistema operacional tem suas próprias ferramentas para criar e destruir partições. No caso do Linux e do MS-DOS, estas ferramentas possuem o mesmo nome: FDISK; por isso nos referiremos àquela do primeiro por `fdisk` e a do segundo por FDISK, respectivamente. Como regra geral, não modifique a partição de um sistema operacional com o fdisk de outro, mas se precisar fazê-lo, prefira o `fdisk`.

1.10.2 Criando e destruindo partições

A maneira normal de criar e destruir partições é através dos fidsk mencionados. No MS-DOS usualmente invoca-se:

`FDISK C:`

depois de ter inicializado o computador através de um disquete[2]. Nos limitaremos aqui a uma descrição generalizada do `fdisk`. Refira-se à ajuda do `fdsik` e FDISK para uma descrição mais detalhada.

Como já foi mencionado, as partições são referidas de maneira diferente em cada um dos sistemas operacionais; em especial, cada diretório está associado a uma partição no Linux. De modo geral, no Linux (e nos UNIX em geral), cada dispositivo do sistema está relacionado com um arquivo especial dentro do diretório `/dev` (de *devices*). Os discos rígidos IDE estão ali referidos como `/dev/hd`, seguido pelo letra que representa a unidade e o número da partição . Por exemplo, a primeira partição do primeiro disco IDE se chamará `/dev/hda1`, a segunda de `/dev/hda2`, e assim por diante. O segundo disco IDE será `/dev/hdb?`, etc. No caso de discos de interface SCSI, chamarão-se de `/dev/sd??` com a mesma sintaxe anterior.

Para criar as partições do Linux, num ponto da instalação você recairá no prompt do `fdisk`[3]. Se não for este o caso, comece chamando o `fdisk` seguido pelo nome da unidade:

```
# fdisk /dev/hda
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
```

(*www.phatlinux.com*, ≃ 180 Mb), com *kernel* 2.2.x e a maioria dos aplicativos

[2] Para criar um disquete de inicialização no MS-DOS, formate-o com a opção /s de incluir sistema: `FORMAT /S A:`

[3] No RedHat existe um aplicativo alternativo chamado DiskDruid. Discutiremos somtente o `fdisk` aqui porque seu uso está mais difundido e os conceitos são genéricos.

```
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

No caso de estar instalando num disco que já contém uma ou mais partições MS-DOS, você poderá examinar a tabela de partição existente pressionando `p`. Anote as informações para referência futura.

```
Command (m for help): p

Disk /dev/hda: 128 heads, 63 sectors, 787 cylinders
Units = cylinders of 8064 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       236    951520+   6  FAT16
```

Neste exemplo temos uma partição MS-DOS+W95 ocupando 236 dos 787 cilindros possíveis no disco rígido, restando-nos 551 cilindros para o Linux. Para começar, criaremos duas novas partições para o Linux, `/dev/hda2` e `/dev/hda3`, que serão usadas para o *swap* e para o diretório raiz, respectivamente. Cria-se uma nova partição com o comando `n`

```
Command (m for help):  n

Command action
   e   extended
   p   primary partition (1-4)

 p
```

responde-se `p`, pois neste caso, todas as partições são primárias. Em seguida, é perguntado o numero da partição a ser criada:

```
Partition number (1-4):  2
First cylinder (237-787, default 237):  237
Last cylinder or +size or +sizeM or +sizeK (237-787):  +70M
```

observe que devem estar entre 1 e 4, pois só 4 partições primárias são permitidas. Aqui acabamos de criar uma partição de 70 Mb que depois alocaremos para *swap*. O processo para criar a partição para o diretório raiz é análogo:

```
Command (m for help): n

Command action
   e   extended
   p   primary partition (1-4)
 p

Partition number (1-4):  3
First cylinder (248-787, default 237):  248
Last cylinder or +size or +sizeM or +sizeK (248-787): 787
```

Agora faz-se necessário definir o tipo de cada uma das partições criadas: `/dev/hda2` como *swap* e `/dev/hda3` como *Linux Native*, esta última será usada para armazenar os arquivos do sistema. Assim:

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): L

 0  Empty           16  Hidden FAT16    61  SpeedStor       a6  OpenBSD
 1  FAT12           17  Hidden HPFS/NTF 63  GNU HURD or Sys a7  NeXTSTEP
 2  XENIX root      18  AST Windows swa 64  Novell Netware  b7  BSDI fs
 3  XENIX usr       24  NEC DOS         65  Novell Netware  b8  BSDI swap
 4  FAT16 <32M      3c  PartitionMagic  70  DiskSecure Mult c1  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     75  PC/IX           c4  DRDOS/sec (FAT-
 6  FAT16           41  PPC PReP Boot   80  Old Minix       c6  DRDOS/sec (FAT-
 7  HPFS/NTFS       42  SFS             81  Minix / old Lin c7  Syrinx
 8  AIX             4d  QNX4.x          82  Linux swap      db  CP/M / CTOS / .
 9  AIX bootable    4e  QNX4.x 2nd part 83  Linux           e1  DOS access
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 84  OS/2 hidden C:  e3  DOS R/O
 b  Win95 FAT32     50  OnTrack DM      85  Linux extended  e4  SpeedStor
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 86  NTFS volume set eb  BeOS fs
 e  Win95 FAT16 (LB 52  CP/M            87  NTFS volume set f1  SpeedStor
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux 93  Amoeba          f4  SpeedStor
10  OPUS            54  OnTrackDM6      94  Amoeba BBT      f2  DOS secondary
11  Hidden FAT12    55  EZ-Drive        a0  IBM Thinkpad hi fe  LANstep
12  Compaq diagnost 56  Golden Bow      a5  BSD/386         ff  BBT
14  Hidden FAT16 <3 5c  Priam Edisk

Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 83

Command (m for help): p

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        10       171    e  Win95 FAT16 (LBA)
/dev/hda2            11        20       180   82  Linux swap
/dev/hda3            21        30       180   83  Linux
```

Observe na penúltima coluna à direita que o *Id* reflete o tipo da partição criada. Finalmente, seguro das mudanças, grave as alterações e reinicialize o sistema:

```
Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

Reboot your system to ensure the partition table is updated.
```

1.11 *Língua da Instalação* ©2000 Leonardo G. Brunnet

Dentre as várias distribuições de Linux existem duas específicas em português: a Conectiva que é uma versão em português do RedHat ou a Mandrake que é uma adaptação do RedHat ao ambiente gráfico do KDE.

1.11.1 Vídeo

Há um programa que controla as relações entre a placa de vídeo e o vídeo oferecendo padrões básicos de alta resolução: é o **servidor XFree86**. Ele possibilita que o (super) usuário configure o seu sistema de forma a obter o melhor desempenho do hardware existente. Documentação específica pode ser encontrada em: `http://sunsite.unc.edu/LDP`.

Toda a informação sobre a configuração do seu vídeo está colocada no arquivo **XF86Config**, usualmente localizado nos diretórios `/etc/` (um diretório geral de configuração do sistema) ou em `/etc/X11` ou ainda em `/usr/X11R6/lib/X11`.

As linhas gerais para configurar o seu vídeo:

- usar um programa como o `Xconfigurator` ou o `XF86Setup` para obter uma configuração básica;
- usar o programa `xvidtune` para fazer um ajuste fino e melhorar o aproveitamento do hardware.

Antes de prosseguir vale a pena chamar a atenção:

ULTRAPASSAR OS LIMITES TÉCNICOS ESPECIFICADOS PARA O SEU VÍDEO PODERÁ INUTILIZÁ-LO!

Quais são estes limites? Aqui cabe entender como funciona o vídeo: ele é um feixe de elétrons incidindo sobre uma tela fosforescente controlados eletromagneticamente por um sinal que sua placa de vídeo emite. O feixe varre a tela de esquerda para a direita com uma dada frequência (chamada frequência horizontal e da ordem de 30 KHz a 70KHz), após ter varrido uma linha, desce um ponto na vertical para varrer novamente uma linha horizontal da esquerda para a direita. No modo entrelaçado as linhas horizontais pares são varridas primeiro e as ímpares depois, de forma que apenas a metade da tela é iluminada de cada vez. A frequência com que o feixe volta à posição inicial no alto da tela é a chamada frequência vertical, medida em Hz, variando de 50Hz a 160Hz.

A partir desta breve descrição já podemos entender algumas especificações básicas a mais: a frequência de varredura da sua placa de vídeo (*dot clock*) que é a frequência máxima do cristal do seu adaptador, ou seja, o máximo de pontos-por-segundo que ele emite; e a largura de banda (*bandwidth*) do seu monitor que é a maior frequêcia com a qual você pode alimentar um vídeo e ainda discernir alguma coisa. Um bom monitor tem elevada largura de banda (e.g. 150MHz), mas é caro! Aqui você já pode entender outra advertência:

Os limites de frequência de sua placa de vídeo (*dot clock*) NÃO são os limites do seu vídeo (*bandwidth*), com certeza o *dot clock* é superior ao que seu vídeo suporta. Não tome um pelo outro!

Para configurar seu vídeo você deve encontrar um conjunto de frequências que se adapte ao seu conjunto. Estas **não** são necessariamente as máximas especificações técnicas do seu monitor, mas é necessário saber se ele suporta variar continuamente as frequências abaixo do limite (multisync) ou se ele sintoniza apenas para um conjunto discreto de frequências, as chamadas frequências de sincronização.

A idéia nesta seção é que você consiga obter o melhor do seu conjunto placa de rede + vídeo via software. O programa de instalação vai usar o **Xconfigurator** para tentar configurar seu ambiente gráfico, se você não obteve um bom resultado na instalação pode usá-lo depois. A ação deste programa pode ser separada em duas partes:

1. teste automático de hardware: para saber que tipo de placa de vídeo você tem permitindo a escolha do **driver**, se as coisas falharem aqui significa que a sua placa de rede não é suportada, ou a versão do **Xconfigurator** que você tem está desatualizada;

2. identificação do monitor: você especifica seu monitor pela marca, se você não encontrar seu monitor na lista apresentada não se desespere: se o seu monitor for multisync, escolha generic multisync, se você não souber nada, escolha generic monitor.

Com isto, mesmo sem informar as especificações técnicas do seu monitor você deve obter uma configuração mínima. Usualmente os monitores de PC no mundo Gates vem minimamente configurados, é por isso que não há grandes problemas associados a configuração de vídeo neste mundo, assim não há riscos de se ter prejuízos com a queima de monitores. Mas quem está interessado em monitores minimamente configurados ;) ?

Se ainda assim, você não conseguiu nada, não entre em pânico. Você pode fazer tudo manualmente e ainda obter o máximo de seu equipamento. O procedimento é o seguinte: primeiro verfique se o driver da placa que você está usando detectou-a corretamente, se ele detectar você vai obter muitas informações úteis a mais. Isto se faz lançando o servidor **X** com uma opção de teste:

`$ X -probeonly` ENTER

No meio de inúmeras outras informações sobre o sistema você deve encontrar uma linha que identifica o chip da sua placa de vídeo, no caso abaixo trata-se de uma placa Cirrus Logic[4]

```
(--) SVGA: PCI: Cirrus Logic GD5446 rev 0, Memory @ 0xe0000000
(--) SVGA: chipset:  clgd5446
(--) SVGA: videoram: 4096k
(--) SVGA: clocks:  25.23  28.32  41.16  36.08  31.50  39.99  45.08  49.87
(--) SVGA: clocks:  64.98  72.16  75.00  80.01  85.23
(**) SVGA: Using 16 bpp, Depth 16, Color weight: 565
(--) SVGA: Maximum allowed dot-clock: 85.500 MHz
(**) SVGA: Mode "800x600": mode clock =  50.000, clock used =  49.866
(**) SVGA: Virtual resolution set to 800x600
```

[4] O símbolo (-) indica que o hardware foi testado, se você encontrar (**) significa que o dado foi fornecido pelo administrador.

As linhas seguintes indicam o modelo do processador central da placa de vídeo (`cgld5446`) e a memória de vídeo (`4096 kb`). As duas linhas posteriores indicam as frequências (`clock`) possíveis nas quais a placa de vídeo pode enviar os sinais para o vídeo (em MHz).

Agora verifique a seção correspondente ao monitor (`Monitor section`) no arquivo de configuração produzido pelo `Xconfigurator` durante a instalação. Você deve encontrar vários pares de linhas no formato abaixo, a primeira é apenas um comentário sobre os dados da segunda.

```
# 800x600 @ 60 Hz, 37.8 kHz hsync
Modeline "800x600"     40     800  840  968 1056   600  601  605  628
+hsync +vsync
```

Neste exemplo, o terceiro campo indica o `clock` que está sendo usado (`40 Hz`), note que ele coincide com uma das frequências detectadas anteriormente (`39.99`). Os campos seguintes indicam como o feixe deve varrer o seu vídeo na horizontal e na vertical. São estes últimos campos que vamos modificar para fazer o ajuste fino com o programa `xvidtune`.

Antes de usar o `xvidtune` certifique-se que colocou corretamente os limites do seu vídeo no início da seção do monitor no arquivo (`XF86Config`) pois este programa pode ser executado por qualquer usuário da máquina e este poderia inadvertidamente exceder os limites do seu monitor, ao executar o programa você verá os avisos correspondentes. O objetivo ao usar o `xvidtune` é maximizar sua frequência vertical tambem conhecida como `refresh`, isto significa que a imagem no vídeo deve aumentar, para evitar que ela exceda os limites do quadro da tela minimize a imagem usando os botões do vídeo. Feito isto clique em [Auto] e [Apply] para que as modificações apareçam na tela. Com os botões [Left], [Right], [Wider], [Narrower], [Up], [Down], [Shorter], [Taller], tente aumentar sua imagem ao máximo cuidando as distorções nas bordas. Você ouvirá um "bip" quando os limites especificados no arquivo `XF86Config` forem atingidos. Finalmente quando estiver satisfeito com a imagem clique em [Show] e o `xvidtune` colocará a nova `Modeline` na janela donde ele foi lançado. Copie esta linha para o arquivo `XF86Config` e comente a linha original. Pronto `:)` ! Outras resoluções de tela compatíveis com sua memória virtual podem ser tentadas com a sequência [Ctrl]+[Alt]+[+] ou [Ctrl]+[Alt]+[-]

Toda esta configuração foi feita em 8 (planos de) bits de cor (256 cores). Existe no arquivo `XF86Config` seções para 8, 16, 24 e 32 bits de cor. Para configurar seu sistema para estes planos proceda exatamente da mesma maneira. A diferença está apenas nas opções do servidor `X` , por exemplo, para 16 bits você vai usar:

```
$ X -probeonly -bpp 16
```

e para lançar a tela gráfica,

```
$ startx -- -bpp 16
```

1.12 Modem

©2000 Leonardo G. Brunnet

Com um modem você faz sua conexão com o mundo por portas seriais. Como em Unix todos os dispositivos são mapeados em arquivos do tip `/dev`, você deve procurar por arquivos do tipo `/dev/cua*` ou `/dev/ttyS*`[5]. As correspondências com as portas DOS são as seguintes:

DOS	Linux
COM1	/dev/cua0, /dev/ttyS0
COM2	/dev/cua1, /dev/ttyS1
COM3	/dev/cua2, /dev/ttyS2
COM4	/dev/cua3, /dev/ttyS3

Para certificar-se que estes dispositivos estão prontos para serem usados procure informações sobre as portas seriais nas mensagens de inicialização do sistema que estão no arquivo `/var/log/dmesg`. Ele pode ser lido pelo comando homônimo `dmesg`:

```
...
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
...
```

Se você tem pressa e já sabe a qual porta está associado seu modem basta fazer um "link" com `/dev/modem` para que os programas que usam o modem possam acessá-lo, por exemplo, se o seu modem estiver em `/dev/cua1`:

```
$ ln -s /dev/ttyS1 /dev/modem
```

Se tudo estiver funcionando, há vários programas de comunicação interativos que vêm com qualquer distribuição Linux: o minicom, seyon efax, efix, fax.

Se as coisas não estão indo bem com o seu modem, você pode testá-lo; diretamente como administrador digite (se a porta do modem for /dev/cua1):

```
$ echo ``ATDTXXX-XXXX\n''>/dev/cua1
```

onde XXX-XXXX é o seu número de telefone. Você deve ouvir o tom de discagem, a discagem do seu número e após o sinal de ocupado. Há também dois programas que podem ajudá-lo a configurar o modem e completar conexões **PPP (Point-to-Point Protocol)**. Estas conexões, embora mais complexas, são fáceis de configurar e permitem, por exemplo, que você se conecte a um provedor para enviar mensagens (**mail**), usar seu "browser", ou ainda trocar arquivos. Para

[5] Nas versões mais recentes do Linux `/dev/cua*` deverá ser substituído por `/dev/ttyS*`

saber se existe suporte para PPP no seu “kernel” Linux, olhe novamente nas mensagens de inicialização do sistema:

`# dmesg` RETURN

```
PPP: version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
registered device ppp0
PPP: ppp line discipline successfully unregistered
CSLIP: code copyright 1989 Regents of the University of California
PPP: version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
registered device ppp0
PPP: ppp line discipline successfully unregistered
```

Se você não encontrar linhas como estas terá de recompilar seu kernel, pois não escolheu esta opção durante a instalação do sistema. Se você as encontrou poderá configurar a conexão usando os programas `kppp` e `xisp`.

2. O SHELL

2.1 O que é o Shell? ©2000 Fabricio Ferrari

O *shell* é a ligação entre o usuário e o sistema. É ele quem interpreta os comandos entrados para outros aplicativos ou diretamente em chamadas de sistema. Além disso, os recursos do *shell* são indispensáveis para lidar com muitos arquivos ao mesmo tempo, para realizar uma tarefa repetidamente ou para programar uma ação para determinada ocasião, entre outros recursos. Começamos apresentando os tipos de *shell* mais difundidos e depois definindo alguns conceitos que serão úteis na sua utilização, para então tratarmos de exemplos práticos.

2.2 Tipos de Shell ©2000 Fabricio Ferrari

O mais comum de todos é o *shell* Bourne (`sh`), desenvolvido a partir de *shell* original, escrito em 1975 por S. R. Bourne. Mais tarde, alguns estudantes da Universidade da Califórnia em Berkeley criaram o *shell* C (`csh`), util para quem é familiar com a sintaxe da linguagem de programação C. O terceiro dos mais difundidos é o *shell* Korn, criado por Dadid Korn da AT&T, que preserva a funcionalidade do Bourne incluindo as características poderosas do *shell* C. Atualmente o Linux e a maioria dos sistemas UNIX contam com o `bash` ou Bourne Again *shell*[1] e uma versão melhorada do *shell* C, o `tcsh`. Nossa discussão aqui está direcionada ao `tcsh`, mas os tópicos aplicam-se aos outros e as diferenças relevantes serão mencionadas. A escolha do *shell* depende principalmente das preferências pessoais de cada usuário, pois os principais recursos estão presentes em qualquer das alternativas.

2.3 Inicialização do shell ©2000 Fabricio Ferrari

Para que o usuário abra sua sessão, tanto local como remotamente, é preciso que ele digite o seu nome e senha para o programa `login`, que irá iniciar a sessão do usuário no sistema. O `login` examina se o nome informado consta na lista de usuários e se sua senha confere com aquela armazenada. Em caso de sucesso, o `login` chama o *shell* configurado para esse usuário. Esta informação

[1] Um trocadilho em inglês que pode significar "o *shell* Bourne novamente" como também "o *shell* nascido de novo".

está guardada no arquivo `/etc/passwd` na maioria dos sistemas[2], juntamente com o nome do usuário, sua senha criptografada, seu número de identificação, o número de identificação do seu grupo, uma descrição do usuário, seu diretório padrão (`HOME`) e o *shell* de seu uso, nesta ordem. Os campos estão separados por dois pontos (`:`).

Passados os processos de *login* em si, o *shell* executa os comandos nos arquivos de sistema `/etc/csh.cshrc` e `/etc/csh.login` e depois procura pelos arquivo de inicialização `.tcshrc`, no diretório `HOME` do usuário. Em caso de não existir este arquivo, o *shell* procura por `~/.cshrc`. finalmente o *shell* executa `~/.login` e `~/.history`, recaindo no *prompt*, quando está pronto para receber as entradas do usuário.

Os arquivos de inicialização em `/etc` servem para estabelecer configurações pertinentes às particularidades do sistema como um todo, enquanto os arquivos de inicialização de cada usuário permitem que cada um ajuste o comportamento do *shell* ao seu gosto. Estes arquivos podem contem quaisquer comandos que façam sentido ao *shell*, que os executará na ordem que os encontrar. As tarefas mais comuns realizadas a partir destes arquivos são gerenciamento básico de arquivos, como apagar arquivos temporários, definição de variáveis de ambiente (seção 2.10) de aplicativos e do próprio usuário e criação de apelidos de comandos (*alias*, seção 2.12). Um arquivo `.tcshrc` típico será incluído no apêndice C.

2.4 Descritores Padrão de Arquivos

©2000 Fabricio Ferrari

Os arquivos no ambiente UNIX possuem forma livre, consistindo apenas de uma seqüência de caracteres. As quebras de linha (nova linha) são delimitadas por caracteres de nova linha `\n`, enquanto o final de arquivo delimitado por \0 ou Ctrl+d, que representam EOF (*End Of File*). Cada arquivo pode ser lido caracter a caracter e gerado da mesma forma. Para se referir a um arquivo, o sistema usa os **descritores de arquivos**, que são palavras chave associadas a cada arquivo. Os sistemas de arquivos serão explicados em maior detalhe numa seção adiante, mas este conceito nos serve para entender um ponto chave do trabalho do *shell*.

O *shell* define três descritores de arquivos muito importantes:

- **a entrada padrão (*stdin*):** como o nome sugere, a entrada padrão é o descritor de arquivo de onde um aplicativo lerá a entrada de dados se não for informado um outro descritor de arquivos específico. Está normalmente associada ao terminal de entrada, normalmente o teclado.

[2] Há situações em que essas informações estão escondidas em outro arquivo (*shadow passwords*) ou em que um servidor informa tais parâmetros para todas os clientes de uma rede (NIS). O primeiro caso ocorre por motivos de segurança e o segundo para manter a base de dados de usuários e senhas comum e sincronizada.

Fig. 2.1: Um comando do *shell* como um filtro.

- **a saída padrão (*stdout*):** é o descritor onde será colocada a saída de qualquer aplicativo, se outro específico não for informado. Normalmente associada ao terminal de saída, o vídeo.

- **saída de erro padrão (*stderr*):** é onde são escritos os erros decorrentes do processamento. Está separada da saída padrão para que os possíveis erros ou avisos não contaminem os resultados em si. Também está associada ao terminal de saída.

Neste contexto, a maioria dos comandos do *shell* agem como filtros, que possuem uma única entrada (*stdin*) e duas saídas (*stdout* e *stderr*), por onde entra e sai um caracter por vez. O que cada comando faz é filtrar a entrada, transformando-a e escrevendo-a na saída padrão, enquanto as eventuais mensagens de erro vão para a saída de erros. A Fig. 2.1 ilustra este conceito.

2.5 *Redirecionamentos* ©2000 Fabricio Ferrari

Os recursos de redirecionamento de **entrada e saída** (E/S) são úteis para redefinir as entradas e saídas padrão em outros descritores de arquivos definidos pelo usuário. A tabela 2.1 mostra a função dos operadores de redirecionamento para os dois principais tipo de *shells*.

`bash/ksh`	`tcsh`	ação
< ⟨arquivo⟩		considera ⟨arquivo⟩ como *stdin*
> ⟨arquivo⟩		coloca a *stdout* no novo ⟨arquivo⟩
>> ⟨arquivo⟩		anexa *stdout* ao ⟨arquivo⟩ (ou cria-o)
<< ⟨delim⟩		toma como *stdin* até encontrar ⟨delim⟩
2 > &1	> &	reúne *stdout* com *stderr*
2 >> &1	>> &	anexa *stderr* a *stdout*
	>! ⟨arquivo⟩	sobrescreve *stdout* em ⟨arquivo⟩
$2 > \&1\vert$	$\vert\&$	reúne *stderr* e *stdout* no duto

Tab. 2.1: Os operadores de redireção de E/S.

Por exemplo, para criar um arquivo com a lista dos arquivos de um diretório, faríamos:

```
$ ls -l > lista_de_arquivos
```

se `lista_de_arquivos` já existisse, poderíamos sobreescrevê-lo[3]

`$ ls -l >! lista_de_arquivos`

ou acrescentar a nova listagem ao final do arquivo

`$ ls -l >> lista_de_arquivos.`

Em todos estes exemplos, ao invés de colocar a saída padrão na tela, todo o seu conteúdo vai para outros arquivos, isto é, estamos redefinindo a saída padrão. A operação análoga pode ser feita com a entrada padrão. Por exemplo, na situação em que quiséssemos mandar um arquivo como conteúdo de uma mensagem de correio eletrônico, simplesmente faríamos:

`$ mail` ⟨endereço⟩ < ⟨mensagem⟩

Por fim, podemos reunir ambos conceitos numa mesma situação em que determinado programa lê seus parâmetros iniciais de um arquivo e grava-os em outro. Para ordenar os nomes dentro de um dado arquivo e gravá-los em outro, simplesmente diríamos:

`$ sort < fora_de_ordem > ordenados`

Neste caso, como em muitos dos aplicativos UNIX, é suposto que o primeiro argumento trata-se da entrada padrão, de forma que poderíamos omitir o sinal de <. Outra aplicação dos redirecionadores é criar um arquivo usando o comando `cat`[4] simplesmente escrevendo

```
$ cat > astros
mercurio
venus
terra
marte
jupiter
```
Ctrl+d

Aqui o `cat` imprimirá a entrada padrão, que como não foi especificada permanece sendo o teclado, na saída padrão, que agora é o arquivo `astros`. Então podemos entrar a lista de planetas até que Ctrl+d (caracter de fim de arquivo - EOF) seja pressionado. Neste momento, o `cat` grava o nome dos planetas no arquivo.

O recurso dos delimitadores também é util quando o argumento de entrada tem mais de uma linha. Neste exemplo usamos os comando `cat`, que serve para imprimir a entrada padrão na saída padrão sem qualquer formatação, para imprimir as seguintes linhas

---

[3] Lembre-se que estamos usando a sintaxe do `tcsh`. Consulte a Tab. 2.1 se o seu *shell* preferido é outro.

[4] O `cat` foi concebido para concatenar arquivos, isto é

`$ cat` ⟨arq_1⟩ ⟨arq_2⟩ ⟨arq_3⟩ > ⟨arq_final⟩

faz com que ⟨arq_final⟩ contenha exatamente, exceto os marcadores de fim de arquivo, todo o conteúdo de ⟨arq_1⟩ e ⟨arq_2⟩ e ⟨arq_3⟩, nesta ordem. Para ler o manual de qualquer comando, digite `man` ⟨comando⟩.

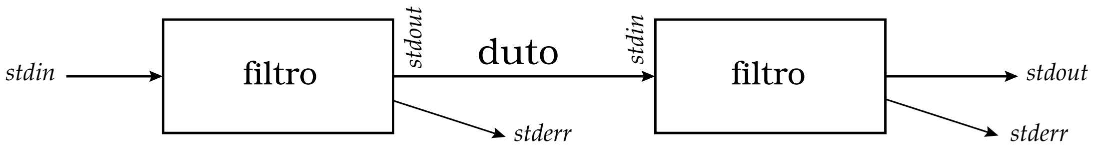


Fig. 2.2: Um duto entre dois filtros.

```
$ cat << ATE_AQUI
primeira linha
segunda
terceira
ATE_AQUI
```

imprimirá

```
primeira linha
segunda
terceira
```

na saída padrão.

2.6 Dutos

©2000 FABRICIO FERRARI

Os dutos (*pipes*) são elos de ligação entre a saída padrão de um programa e a entrada padrão de outro, conforme mostrado na figura 2.2. Este conceito simples está por trás desta ferramenta útil e eficiente. Suponha que você quisesse juntar (`cat`) o conteúdo de dois arquivos (`alunos` e `alunas`), ordená-los (`sort`) e gravar a saída num novo arquivo (`turma`). Para isto bastaria

```
$ cat alunos alunas | sort > turma
```

Desta forma o `cat` concatena `alunos` e `alunas` numa mesma saída padrão, que servirá como entrada padrão para o `sort`, que terá sua saída redirecionada para o arquivo `turma`. O comando anterior sem o uso de dutos seria escrito

```
$ cat alunos alunas > alunosealunas
$ sort alunosealunas > turma
$ rm alunosealunas
```

Na forma tradicional a quantidade de comandos é maior e há a necessidade de usar o arquivo temporário `alunosealunas`. Também é possível usar mais de um duto na mesma linha de comando. Para imprimir diretamente a saída do processo acima simplesmente acrescente o comando de impressão `lpr`

```
$ cat alunos alunas | sort | lpr
```

2.7 Caracteres Especias e Curingas

©2000 Fabricio Ferrari

Para facilitar o manuseio de arquivos o *shell* dá sentido especial a certos tipos de caracteres que mudam de valor conforme o uso que se dá a eles, por isso chamados de curingas (formalmente: metacaracteres). Estas regras tendem a reduzir a digitação e encorajar convenções na nomeação de arquivos. Os curingas são os seguintes:

- *****: poderá ser substituído por zero ou mais caracteres.
- **?**: poderá ser substituído por um caracter.
- **[...]**: poderá ser substituído por um dos caracteres entre colchetes.

Por exemplo, se tivéssemos os arquivos `dia1`, `dia2` e `dia3`, poderíamos concatená-los todos usando os curingas apropriados, sem mencionar explicitamente cada um dos nomes. Neste caso, existem três possibilidades de realizar o mesmo trabalho

```
$ cat dia* > dias
$ cat dia?  > dias
$ cat dia[123] > dias
```

Estes conceitos podem ser generalizados e superpostos, isto é, `?ida*` poderia coincidir com `cidade`, mas não `opacidade` nem `claridade`. Da mesma forma, `cor[ear][sle]` passaria por `cores`, `coral` e `corre` (analise as outras combinações possíveis). Além disso, você poderá especificar um intervalo de caracteres, ou seja, a expressão `[a-z]` significa qualquer caracter minúsculo e `[0-9]` qualquer número. Para complementar estas possibilidades, existe o caracter de negação ^, usado em associação com os colchetes `[...]`. Se escrevemos `aluno[^0-9]` estamos nos referindo aos nome que começam com a palavra `aluno` **não** seguida dos números de 0 a 9.

Exitem outros caracteres especiais para o *shell* além dos curingas. Um deles é o ponto-e-vírgula ; que serve para colocar multiplos comandos numa mesma linha. Assim escrevendo

```
$ cd fazenda; rm cavalos* ; cd ..
```

exterminaríamos todos os eqüinos de uma vez e voltaríamos ao diretório anterior.

2.8 Controle de Processos

©2000 Fabricio Ferrari

Outra característica de operação bastante útil está relacionada ao caracter &. Lembre-se que o Linux, sendo um *sabor* de UNIX, é multiusuário e multitarefa. Isso significa que vários usuários podem estar usando o mesmo computador e executando mais de uma tarefa ao mesmo tempo, com os recursos compartilhados. Assim, se desejássemos colocar um determinado programa para ser executado em segundo plano, anexaríamos o caracter & depois do comando. Imagine que um programa seu chamado `chuva` levará dias executando e você não quer esperar que ele termine ou precisa continuar trabalhando. Bastaria executá-lo como

```
$ chuva &
[1] 925
```

e o *shell* o colocará em segundo plano, informando a ordem dentre os seus processos ([1]) e a ordem geral do processo (925). O processamento em segundo plano é um recurso poderoso quando bem utilizado, permitindo que preencha-se o tempo ocioso dos computadores, entretanto lembre-se que se o programa não terminar por si próprio ficará rodando até que seja explicitamente morto, como no caso de uma reinicialização ou através do comando `kill`[5]. Uma maneira alternativa de colocar um programa já iniciado em segundo plano é através da seqüência de teclas Ctrl+z, ou

```
$ chuva
Ctrl+z
Suspended
$ bg
[1] chuva &
```

pois quando Ctrl+z é pressionado o programa é suspenso até que a ordem `bg` (de *background* ou segundo plano) faz com que sua execução continue em segundo plano, exatamente de onde tinha parado quando o *shell* suspendeu-o, sem nenhuma alteração de seus dados ou parâmetros.

Outra utilidade deste recurso consiste em suspender um aplicativo, realizar outra tarefa e depois continuar executando-o. No exemplo anterior

```
$ chuva
Ctrl+z
Suspended
$ sol
sol:  Command not found.
$ fg
chuva
```

aqui a ordem `fg` serve para recolocar o último aplicativo que foi para segundo plano (`bg`) novamente em primeiro plano (`fg` de *foreground*). Se há vários processos em segundo plano, é possível identificá-los antecedendo o número do processo por %, isto é `fg %1`, neste caso.

2.9 *Substituição da Saída Padrão* ©2000 Fabricio Ferrari

A substituição de *stdin* é outro poderoso recurso do *shell* usado através dos acentos graves \` \`. Os graves servem para que você coloque a saída da execução de um programa como argumento de outro, de uma maneira diferente daquela com

[5] Veja o manual do `kill` (`$ man kill`) para maiores detalhes de como matar processos, mas não torne-se um assassino.

dutos. Por exemplo, o comando `which` (qual) serve para mostrar aonde o *shell* está achando determinado aplicativo. Se fizemos

```
$ which cat
/bin/cat
```

somos informados que o *shell* executa o `cat` que está em `/bin`. Se quisermos saber as propriedades do arquivo `cat`, usamos o `ls`[6] em conjunto com os acentos graves

```
$ ls `which cat`
```

e a saída do `which cat` servirá de argumento para a execução do `ls`. Portanto, a linha anterior seria idêntica a escrever

```
$ ls /bin/cat
```

Novamente realizamos duas operações ao mesmo tempo: achamos o `cat` no sistema de arquivos e listamos suas propriedades. Este exemplo sintetiza a filosofia do UNIX: uma série de programas simples que agrupados proporcionam grande eficácia e flexibilidade. A importância do *shell* neste contexto é proporcionar comunicação dos aplicativos entre si e de você com os aplicativos.

2.10 Variáveis do shell

©2000 Fabricio Ferrari

O *shell* permite que se criem variáveis e que se lhes atribuam valores guardados durante a execução do *shell*. As variáveis podem ser **variáveis locais** ou **variáveis de ambiente** (globais). Os nomes das variáveis podem ser constituídos de quaisquer caracteres alfanuméricos. Para criar e ao mesmo tempo atribuir um valor à variável na mesma operação, usamos o comando `set` para variáveis locais ou `setenv` para as de ambiente[7]. No caso do `setenv` não é usado o sinal de = entre o nome e o valor da variável. O valor de uma variável é acessado através do nome da variável precedido pelo operador `$`[8], por exemplo

```
$ set estado= ‘‘Rio Grande do Sul’’
$ echo $estado
Rio Grande do Sul
```

Neste caso o `echo` mostra o conteúdo de `$estado` na tela. Note que o argumento passado para o `echo` é “Rio Grande do Sul” pois o *shell* já fez a interpolação da variável `$estado` na cadeia de caracteres correspondente. As variáveis podem ser usadas em qualquer comando no contexto do *shell*. Para referir-se ao nome de uma variável sem ambiguidade, inclue-se seu nome entre chaves { } , observe

```
$ set arquivos=‘‘/dados/curso’’
```

[6] O `ls` lista arquivos e suas propriedades. Voltaremos a falar dele mais tarde.

[7] No caso do `bash` é usado ⟨nome_da_variável⟩=⟨valor⟩, sem a necessidade do `set`. No caso de variáveis de ambiente, depois de definí-la, executa-se o `export` ⟨nome_da_variável⟩ para torná-la global.

[8] Até aqui indicamos o *prompt* do *shell* pelo mesmo símbolo. Não confuda o `$` no início da linha, indicando que se trata da linha de comando, com `$` antes de um nome de uma variável (`$nome`), que serve para que seu valor seja interpolado.

`$ cp ${arquivos}/aula2.tex /home/alunos`

assim o *shell* procura por uma variável `arquivo` e concatena o seu valor com o resto da cadeia `/aula2.tex` e então passa o argumento inteiro para o comando `cp`. Note que tanto a interpolação de variáveis em seus respectivos valores como a substituição dos curingas numa lista de arquivos (Seção 2.7) é feita pelo *shell* e o resultado desta operação (em caso de sucesso) é passada ao aplicativo. Nenhum dos aplicativos realmente recebe `$estado` ou `aluno?` como argumento, mas o significado que estas expressões tem para o *shell*. Se for necessário usar caracteres especiais do tipo `$, *, ?, [ ]` como argumento de aplicativos, usa-se o **caracter de fuga** \ precedendo os caracteres especiais. Isto evita que o *shell* interprete-os. Por exemplo, para lermos a página de ajuda do comando `less`, que serve para mostrar o conteúdo de um arquivo no terminal, usamos `$ less -\?` pois não queremos que o *shell* substitua `?` pelo nome de um arquivo com uma só letra (se `?` fosse um curinga) mas que o `less` receba o `?` como argumento.

O recurso de substituição da saída padrão (Seção 2.9) pode ser explorado para definir variáveis a partir da saída de aplicativos

```
$ set sistema=‘uname‘
$ echo $sistema
Linux
```

que pode ser extensivamente explorado na construção de scripts do *shell*.

Algumas variáveis são definidas internamente pelo *shell* e podem ser usadas durante sua execução. A tabela 2.2 relaciona parte das variáveis definidas pelo `tcsh` ao iniciar.

NOME DA VARIÁVEL	SIGNIFICADO
`user` e `USER`	nome do usuário
`home` e `HOME`	diretório padrão de `$user`
`path`	lista de diretórios onde o *shell* procura pelos aplicativos
`shell`	*shell* em uso
`tcsh`	a versão do `tcsh`
`cwd` ou `PWD`	diretório de trabalho atual
`HOST`	nome do computador
`HOSTTYPE`	arquitetura do computador

Tab. 2.2: Variáveis definidas pelo *shell* `tcsh`. Os nomes em minúsculas são locais e em MAIÚSCULAS são de ambiente.

A interpolação de variáveis ocorre sempre que seu nome for invocado diretamente ou entre aspas duplas. Qualquer texto entre aspas simples será interpretado literalmente, sem interpolações. Acompanhe o exemplo abaixo

```
$ set dia=sexta
$ set aviso=“Ontem foi $dia”
```

```
$ echo $aviso
Ontem foi sexta
$ set aviso=‘‘Ontem foi $dia’’
$ echo $aviso
Ontem foi $dia
```

desta forma, as aspas simples fornecem um mecanismo para que as interpolações de variáveis sejam ignoradas, enquanto as duplas permitem que as substituições sejam feitas, preservando a unidade da cadeia de caracteres.

2.11 História de Comandos ©2000 Fabricio Ferrari

A partir do momento em que o *shell* inicia, ele passa a guardar todos os comandos digitados pelo usuário, na seqüência em que são digitados. Estes comandos podem ser recuperados, editados e novamente executados pelo *shell*, simplificando tarefas repetitivas em arquivos distintos. Este recurso é chamado de **história** porque grava a seqüência de acontecimentos no âmbito do *shell*.

O número de comandos gravados pelo *shell* é determinado pela variável de ambiente `history`[9], enquanto que a variável `savehist` determina quantos comandos serão gravados no arquivo `.history` quando o usuário termina a sessão, para serem recuperados numa sessão futura.

A primeira maneira de examinar os comandos do *shell* é através do comando `history` que mostra em parte algo como

```
          ⋮
122   9:35   df
123   9:35   new
124   9:35   rusers
125   9:35   ls
126   9:35   sort Makefile
127   9:35   sort Makefile > lixo
128   9:35   sort Makefile > lixo2
129   9:35   ls
130   9:35   diff lixo lixo2
131   9:35   diff lixo Makefile
132   9:35   ls
133   9:35   clear
134   9:35   rm lixo*
135   9:35   make
          ⋮
```

onde vemos o número do comando na história, a hora da chamada e a sintaxe do comando. O **caracter de substituição da história** é o !, logo se quiséssemos executar o comando de ordem 133 (`clear`) novamente, simplesmente digitamos

[9] Por exemplo, set history=100 grava os 100 últimos comandos

```
$ !133
clear
```

que limparia a tela do terminal.

Operador	Efeito
!!	comando anterior
!n	n-ésimo comando
!-n	!n a partir do fim
!txt	comando mais recente que inicia com txt
!?txt?	comando mais recente que contém txt
!*	todos argumentos do comando anterior, exceto o argumento zero (o nome do comando)
!^	o primeiro argumento do comando anterior
!$	o último argumento do comando anterior
!{txt}	faz com que somente txt seja usado na interpretação da história

Tab. 2.3: Indicadores da história do tcsh.

Na maioria das implementações do tcsh é possível usar as setas do teclado $\boxed{\uparrow}$ $\boxed{\downarrow}$ para percorrer a história de comandos. Existem alguns modificadores da história que simplificam o reutilização dos comandos já digitados. Para repetir o último comando executado digitamos !!. Como no exemplo acima, o comando de ordem n é reexecutado através da sintaxe !n, onde n é seu número. Se quisermos contar do fim para o começo, isto é, para reexecutar o comando n a partir do atual, basta fazer $!-n$. Se o clear devesse ser executado novamente, poderíamos procurá-lo pelo seu nome, através da sintaxe !nome. Neste caso

```
$ !clear
clear
```

fazemos com que o *shell* procure pelo comando mais recente que comece com o texto *clear*, podendo abreviá-lo por !clea, !cle ou mesmo !c. O *shell* lê os comandos na ordem inversa daquela em que foram digitados, até encontrar o texto que coincida com aquele fornecido pelo usuário, quando executa o comando. Veja outros operadores de procura da história na tabela 2.3.

2.11.1 Modificadores e Indicadores

Embora todos estes recursos sejam valiosos para abreviar a quantidade de texto digitado pelo usuário, são pouco frequentes as situações em que uma seqüência de comandos é executado várias vezes da mesma maneira[10]. Normalmente deseja-se fazer pequenas modificações nestes comandos e então reexecutá-los. Este é o caso

[10] Para este fim tem-se o comando interno do *shell* repeat

de repetir-se o mesmo processo em vários arquivos, corrigir erros de digitação ou testar novos parâmetros dos aplicativos. Para esse fim o manipulador da história do *shell* inclui os **modificadores da história**. Os modificadores servem para que se repita os comandos da história alterando-os antes de sua execução. Os principais estão listados na tabela 2.4

OPERADOR	EFEITO
`:*`	todos os argumentos, exceto o zero
`:^`	o primeiro argumento
`:$`	o último argumento
$:n$	o n-ésimo argumento
$:p-q$	do argumento p ao q
$:-q$	abrevia $:0-q$
$:p*$	argumento p até o último
`:p`	imprime o comando mas não executa
`:s/l/r/`	substitui `l` por `r`
`:&`	repete a substituição anterior

Tab. 2.4: Modificadores e indicadores de argumentos da história do `tcsh`.

Os argumentos são nomeados a partir de zero (o nome do comando), conforme o esquema abaixo.

```
$ ls discos livros telas ...
  0    1      2      3    ...
```

Tomando este comando como exemplo, poderíamos usar os indicadores de argumentos da tabela 2.4 para selecionar cada um dos argumentos das linha de comando anterior. Por exemplo, !* seria substituído por `discos livros telas`. Então, usando !* reaproveitamos todos argumentos do `ls` numa nova operação.

```
$ rm !*
rm discos livros telas
```

agora, `rm discos livros telas` é o comando mais recente da história, de maneira que !! referir-se-ia a ele e não mais ao `ls ...`. Para selecionar um só argumento da linha de comando anterior, usamos os indicadores de argumentos. !^ seria substituído por `discos`, !`$` por `telas`, !`:2` por `livros`, !`:2-3` por `livros telas` e assim por diante. Novamente, explore a tabela 2.4 com criatividade.

Os modificadores e indicadores da história podem ser superpostos, desde que não sejam contraditórios. Além disso, podem referir-se a qualquer comando da história e não só ao último. Nos exemplos acima, bastaria incluir o número do comando logo após o ponto de exclamação ! para que o *shell* refira-se a outro comando que não o último. Se examinamos a história listada no início desta seção (página 35) poderíamos editar o arquivo `Makefile` recém criado, simplesmente digitando

```
$ vi !128:1
vi Makefile
```

se quiséssemos apenas fazer a substituição e colocá-la na história sem executar o comando em si, bastaria acrescentar o `:p` na linha substiuição acima

```
$ vi !128:1:p
vi Makefile
```

[COMANDO NÃO EXECUTADO]

e assim sucessivamente poderíamos empilhar modificadores e indicadores para se referir a comandos passados.

Em suma, o recurso de registro, edição e procura da história facilita o processo de edição de comandos, agilizando o processo de digitação e abreviando a tarefa de memorização de nomes de arquivos. Neste ponto fica difícil evitar a analogia com a vida real, onde também se aprende com os fatos passados. Sem atribuir ao *shell* propriedades da natureza humana, é o conhecimento da nossa história que nos permite evoluir.

2.12 Apelidos de Comandos

©2000 Fabricio Ferrari

Os apelidos de comandos (*alias*) servem para criar ou redefinir comandos a partir dos comandos do sistema, criando assim um *alias*. Criar um alias é bastante simples, basta usar

`$ alias` ⟨nome do apelido⟩ ⟨comandos⟩

de maneira que quando o ⟨nome do apelido⟩ for chamado, na verdade os comandos que o *shell* executará serão os ⟨comandos⟩. Por exemplo, você pode definir um comando `apaga` a partir do `rm -i`, onde o `-i` serve para que seja solicitada confirmação ao apagar cada arquivo. Veja

```
$ alias apaga 'rm -i'
```

assim cada vez que o usuário digitar `apaga` o que o *shell* estará executando `rm -i`. Pode-se usar o auxílio de colocar mais de um comando numa mesma linha para definir apelidos. Para criar um apelido `lista` que limpa a tela antes de executar o `ls` e depois mostra a data, bastaria escrever

```
$ alias lista 'clear; ls; date'
```

É aconselhável não definir um apelido com o mesmo nome de um comando já existente, pois podem existir outros apelidos que contenham tal comando na sua definição e desta forma sua definição estará pervertida. Em outras palavras, evite que um apelido seja definido a partir de outro. Acompanhe o exemplo

```
$ alias ls 'ls -laFL'
$ alias dir 'ls'
```

desta forma, `dir` executará `ls`, que por sua vez está definido como `ls -laFl` e não simplesmente `ls`, como era a intenção. Faça pequenas modificações no nome do apelido com relação ao nome do comando, como `ll`, `lls`, etc., para evitar tal efeito colateral

Se existe um apelido com mesmo nome que um comando do sistema, você poderá ignorar a definição do apelido usando o caracter de fuga \ antes do nome do comando

`$ alias ls 'ls -laFL'`

`$ \ls`

executará `ls` e não `ls -laFL`. Estas definições estão entre aspas simples para que o *shell* não tente interpretar qualquer um dos parâmetros entre aspas. Se você deseja que o *shell* faça as substituições necessárias, reescreve sua definição usando aspas simples “. Por exemplo o comando

`alias bomdia ``cat $HOME/mensagem''`

definirá `bomdia` como `cat /home/ferrari/mensagem` no meu caso.

O uso dos apelidos ajuda a recordar comandos extensos e complexos, além de resumir a quantidade de digitação. Em contrapartida, o abuso no uso dos apelidos faz com que o usuário perca ciência do que realmente está fazendo, um risco a longo prazo.

3. UNIX BÁSICO

3.1 *Documentação* ©2000 Rodrigo Parizotto

Embora o conhecimento do sistema não seja essencial no ambiente MS Windows, é extremamente importante em ambientes **Unix**. Quanto mais aprendemos sobre os comandos e sobre o sistema em si, maior a possibilidade de fazermos o que queremos com mais rapidez, mais confiabilidade, e com menos trabalho de manutenção. Além disso, entender a filosofia do sistema pode nos orientar para boas práticas de programać ao e nos facilitar para operações futuras. Felizmente existe um projeto de documentação sério sobre o **Linux**, chamado LDP (*Linux Documentation Project*), reunindo esforços de colaboradores no mundo inteiro e edificando uma sólida referência. Originalmente escrito em inglês, o LDP já está traduzido para algumas línguas e conta com o interesse de um projeto nacional para traduzi-lo para a língua portuguesa (informações sobre o andamento do projeto podem ser encontradas na Internet).

3.1.1 *Guias de Referência - LDP*

Até o momento da edição deste texto o LDP está dividido em:

- Linux Installation and Getting Started (install-guide)
- Linux Programmer's Guide (lpd)
- The Network Administrator's Guide (nag)
- Linux System Administrator's Guide (sag)

e os documentos **HOWTOs**. *Para todos os usuários a leitura é recomendada.* Na distribuição Red Hat Linux, os documentos estão abaixo de

```
/usr/doc
```

3.1.2 *HOW-TOs*

Integrante do projeto de documentação, os HOW-TOs são textos mais específicos que em geral respondem às necessidades de configuração de software ou hardware específicos. Suponhamos que o usuário queira saber detalhes da configuração de seu CD-ROM, então poderá ler o **CDROM-HOWTO**; se quiser saber como compilar o **Kernel** pode ler o **Kernel-HOWTO**. Para acelerar o processo de conhecimento, convido-os a navegar pelo diretório onde geralmente se encontram os **HOWTOs** e os **mini**:

```
/usr/doc/HOWTO
```

Na versão mais usual (arquivos ASCII puros), podem ser visualizados com qualquer exibidor, o **less** por exemplo. Se estiverem compactados, podem ser descompactados ou visualizados diretamente pelo **zless**. Se alguém preferir poderá encontrar a versão em HTML na Internet.

3.1.3 man

O comando `man` (de manual) formata e exibe páginas de manuais, em geral relacionadas com os os comandos. Por exemplo, para saber as descrições e opções do comando `rm` deve-se colocar na linha de comando:

```
$ man rm
```

Para informações sobre o man,

```
$ man man
```

O **xman** é uma ótima interface gráfica para o **man**, devendo ser usado com o servidor X. O **xman** inclui divisão por assunto, listagem de comandos referenciados e outras opções.

3.1.4 info

Info é um programa para leitura e navegação de documentos informativos, com fundamentos semelhantes ao programa **Ajuda** do MS Windows. Contém referências sobre utilitários para textos ASCII, shell, comandos úteis em Linux, etc. com exemplos, mostrando também a filosofia de uso e poderio. Informações acerca dos compiladores **assembler** e **C**, com bibliotecas e comentários de padrões também importantes para usuários que procuram aprofundamento. Para executar o navegador, basta digitar

```
$ info
```

na linha de comando, logo ao entrar você receberá instruções básicas de navegação pelo **info**, para informações detalhadas de navegação digite “h”. Os documentos ‘info’ também podem ser acessados via **Emacs**.

3.2 Utilidades Padrão DOS - mtools ©2000 Rodrigo Parizotto

Comentamos anteriormente que um dispositivo de hardware como o **disquete** precisa ser montado para acessá-lo e desmontado quando se quer tirá-lo. No entanto, se você não se sente a vontade montando e desmontando diretórios, isto pode ser evitado. Foi desenvolvido um pacote de ferramentas chamado **mtools** que contém os comandos de sintaxe semelhante aos usados no MS DOS, se você domina os comandos **DOS** apenas acrescente um **m** na frente do comando. Por exemplo, o comando

`$ mdir a:`

lista os arquivos do drive **a:** sem que haja necessidade de montá-lo. O comando **mcopy** é semelhante ao **copy** do MS DOS. Para informações adicionais:

`$ man mtools`

Outro recurso para acessar discos floppy é usar o **autofs** que faz montagem automática. Para detalhes procure ajuda sobre o **autofs** (pelo comando `man` por exemplo).

3.3 Árvore de diretórios e Caminhos ©2000 Átila Bohlke

Conceitualmente a árvore de diretórios, no Linux, é a mesma coisa que no MS-DOS/MS-Windows. Ou seja, uma expressão do tipo:

`$ /home/bohlke/programas`

tem no Linux o mesmo significado que no DOS, isto é, um diretório chamado *programas* dentro do diretório *bohlke* que está dentro do diretório *home* que está dentro do diretório /.

ATENÇÃO: Nesse caso temos 4 diretórios, porque no Linux não temos o C:, mas sim o diretório / (chamado de root ou raiz).

Um caminho (ou path) é uma sucessão de diretórios, como no exemplo acima. Neste caso temos um caminho **absoluto** pois ele inicia no diretório raiz (/). Um caminho **relativo**, inicia-se no atual diretório de trabalho. Exemplos:

`/home/bohlke/programas/testes/arquivo1.blah`

é um caminho absoluto para o arquivo. Pode-se alcançar o `arquivo1.blah` independendo do diretório atual.

`programas/teste/arquivo1.blah`

é um caminho relativo para o mesmo arquivo. Só se pode alcançá-lo se estivermos previamente em `/home/bohlke`

3.3.1 Diretórios importantes

˜ Diretório *home* de cada usuário, em geral /home/nome-do-usuário

. É o presente diretório, o diretório no qual estamos trabalhando

.. É o diretório imediatamente acima do qual estamos trabalhando

/bin Contém os comandos essenciais do unix, como o *ls*

/usr/bin Outros comandos não tão essenciais :-)

/usr/sbin Comandos de super-usuário

/sbin Outros comandos de super-usuário

/boot Onde está o kernel e outros arquivos utilizados na inicialização

/etc Maioria dos arquivos de configuração do sistema

/etc/X11 Idem ao anterior, porém relativos ao X-server

/var Arquivos administrativos, tais como *log-files*

/var/spool Arquivos temporários que estão na espera de alguma fila

/usr/lib Bibliotecas padrão

/usr/lib/X11 Arquivos do pacote X e suas bibliotecas

/usr/include Localização padrão dos arquivos de inclusão em C

/usr/src Localização dos fontes de vários programas, inclusive do kernel

/etc/skel Arquivos de inicialização padrão para serem colocados nos diretórios *home* de cada usuário

3.3.2 Comandos básicos

A seguir são apresentados alguns comandos básicos para manipulação de arquivos no Linux (para maiores detalhes, consulte a página **man** de cada um):

pwd Mostra o diretório de trabalho atual (ou o diretório .)

ls Lista os arquivos no corrente diretório. Com a opção `-color`, mostra os subdiretórios em azul e os arquivos com permissão de execução em verde. Se lhe for fornecido o nome de um diretório como argumento, ele mostra o conteúdo desse diretório. A opção `-a` mostra os arquivos ocultos (que no unix iniciam com .) e a opção `-l` mostra os detalhes dos arquivos, tais como permissões, tamanho, etc...

mkdir *caminho* Cria um diretório. Aceita caminhos relativos ou absolutos.

cd *caminho* Troca para um determinado diretório. Seu argumento pode ser um caminho relativo ou absoluto.

rmdir *caminho* Deleta um diretório que esteja vazio. Para deletar um diretório e todo seu conteúdo (inclusive outros diretórios) usa-se a opção `-r` **(use com cuidado!!)**.

rm *arquivo* Exclui um arquivo existente. Novamente aceita caminhos absolutos ou relativos.

mv *nome-original nome-novo* Copia um arquivo com um novo nome e deleta o antigo.

cp *nome-original nome-novo* O mesmo que o anterior, porém preserva o arquivo antigo.

3.4 Permissões de Acesso

©2000 Átila Bohlke

Ao utilizar o comando

```
$ ls -l
```

obtém-se algo semelhante a

```
drwxr-xr-x 3 bohlke users 1024 Jan 10 1999 public_html
```

onde o primeiro grupo de 10 caracteres indicam as permissões de acesso ao arquivo em questão (no caso um diretório de nome *public_html*). Em seguida vem o nome do usuário dono do arquivo e o grupo ao qual ele pertence. Depois temos o tamanho (em bytes) que é ocupado pelo arquivo, a data de sua última modificação e finalmente seu nome.

Os bits de permissão são divididos em 4 categorias: o mais a esquerda é o bit de status que pode ser um traço (-) para arquivos comuns, um *d* para diretórios dentre outros[1]. O grupo de 3 bits seguintes são de leitura, escrita e execução (respectivamente) para o usuário dono do arquivo. Os 3 seguintes são as mesmas coisas, mas aplicáveis ao grupo ao qual o usuário pertence. E por fim, os 3 últimos bits se referem a qualquer outro usuário.

3.4.1 Trocando as permissões de um arquivo

Os seguintes comandos:

```
$ chown <user>[:<user>] <file>
$ chgrp <group> <file>
```

trocam o dono e o grupo de um determinado arquivo, respectivamente. Sendo que o primeiro também pode trocar o grupo. Ambos só podem ser utilizados pelo super-usuário.

No entanto, qualquer usuário pode trocar as permissões de acesso de seus arquivos:

Utiliza-se o comando

[1] veja a página de manual para *ls*

`$ chmod <modo> <arquivo>`
onde ⟨*modo*⟩ pode ser de 2 formas distintas: forma relativa e forma absoluta (ou numérica).

Modos relativos

Tem a seguinte forma: <user>e/ou<group>e/ou<others><+>ou<-><bit_de_permisão>
Alguns exemplos deixarão isso mais claro:

`$ chmod u+x foo.txt` habilita o direito de execução de execução pelo dono do arquivo *foo.txt*.

`$ chmod ugo-w foo.txt` desabilita o direito de escrita por parte do dono, do grupo e de outras pessoas quaisquer no arquivo *foo.txt*.

Modos absolutos

Em vez de se utilizar letras usa-se uma numeração de potências de dois:

400 leitura para o usuário

200 escrita para o usuário

100 execução para o usuário

40 leitura para o grupo

20 escrita para o grupo

10 execução para o grupo

4 leitura por qualquer pessoa

2 escrita por qualquer pessoa

1 execução por qualquer pessoa

Quaisquer combinações dos ítens acima são válidas. Exemplo:

`$ chmod 777 foo.txt` libera todas as permissões de *foo.txt*

`$ chmod 111 foo.txt` libera a execução pelo usuário, pelo grupo e por outros do arquivo *foo.txt*

`$ chmod 711 foo.txt` libera a execução de *foo.txt* por qualquer um, mas só o dono do arquivo pode ler e escrever ele.

Comando **umask**

Esse comando nos permite indicar quais devem ser as permissões que arquivos novos devem ter no momento de sua criação. A sintaxe é:

```
$ umask <argumento>
```

onde o seu argumento é: 777 - <permissão desejada> . Exemplo:

```
$ umask 066
```

faz com que todos os novos arquivos tenham as permissões setadas para 711, como no exemplo acima. Ele pode ser utilizado a qualquer momento, mas é mais aconselhável colocá-lo em algum script de inicialização, como o .bashrc ou .tcshrc. Pode ser utilizado por qualquer usuário. É uma configuração estranha de permissões, sem dúvida, mas funciona :-)

4. APLICATIVOS I

4.1 Os pacotes do Linux ©2000 Átila Bohlke

No Linux existem basicamente 2 maneiras de instalar-se um determinado programa (ou conjunto desses). A maneira mais tradicional, é através dos conhecidos *tarballs*, ou arquivos `.tar`. Deste modo, estamos na verdade compilando o código fonte de um determinado programa e em seguida copiando-o para o seu local definitivo.

O segundo modo, desenvolvido pela equipe técnica da *RedHat* é através de códigos binários previamente compilados e empacotados num arquivo `.rpm`. Neste arquivo rpm está toda informação de pré-requisitos e localização para cada arquivo contido pelo pacote.

Para a primeira maneira de instalação, o único modo de lidar com os arquivos, é através da cópia, movimentação e deleção dos mesmos. Já para os pacotes RPM existem vários programas que se encarregam de fazer o "trabalho sujo"para o usuário.

4.1.1 GnoRPM, Kpackage & cia. lltda.

O GnoRPM é um dos programas que se encarrega de manipular os pacotes de binários. É totalmente gráfico e agrupa os pacotes por grupos e categorias. É desenvolvido em parceria entre a RedHat e o projeto GNOME. É desenvolvido para funcionar dentro da interface GNOME, mas pode ser utilizado em qualquer ambiente gráfico.

O Kpackage cumpre as mesmas funções que o GnoRPM de forma baste semelhante. É um gerenciador gráfico de pacotes rpm projetado para intregrar-se ao KDE (outro *window manager*).

E existe é claro o bom e velho comando de linha

`$ rpm` ⟨opções⟩ ⟨pacote⟩

onde as opções mais comumente usadas são:

- **-U** para atualizar um pacote previamente instalado
- **-i** para instalar um pacote
- **-e** para desinstalar um pacote

- **-h** para mostrar um barra de progresso na forma ###########
- **-v** para mostrar as mensagens de erro
- **-q** para verificar se um pacote esta instalado

4.1.2 Ted & Lyx

Um programa recente que com certeza irá facilitar a vida de muitas pessoas é o **Ted**, que edita arquivos de texto no formato rtf (Rich Text Format). Esse formato permite a inclusão de figuras de vários tipos, tabelas e outros "adornos". O programa é inteiramente gráfico e bastante intuitivo. Pode ser achado em `http://www.nllgg.nl/Ted`.

Outro programa de edição de textos que com certeza agrada a Gregos e Troianos é o **Lyx** ou a sua versão para o KDE, o **Klyx**. Esse programa tem o grande trunfo, se ser um aplicativo "WYSWYG"para edição de texto, mas que porém grava o texto em formato LaTeX.

4.2 GNU Plot ©2000 Marlon Schmitz

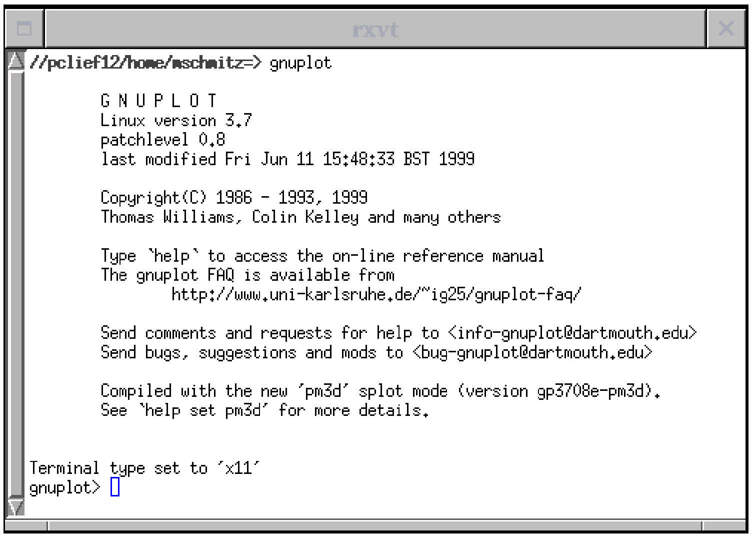


Fig. 4.1: GNUPlot

Uma dentre as muitas vantagens do sistema *LINUX* é que ele vem com diversos progamas para as mais variadas aplicações, desenvolvimento, editoração, científica, comunicação via rede, etc... Hoje falaremos um pouco sobre programas simples porém poderosos: gnuplot, xgfe, xmgr/xmgrace, elm e exmh.

O **GNUPlot** é um programa que se usa para plotar gráficos em duas e três dimensões, bem como fazer curvas de nível, sólidos, mapas degradé, etc...

Este program está atualmente no ranking dos plotadores gráficos mais potentes, junto com ele estão o Xmgr(*LINUX*), Origin e Sigma Plot (Windows).

O **GNUPlot** trabalha em um ambiente de linha de comando próprio(4.2), isto é você deve digitar comandos para plotar um gráfico nele.

Vejamos alguns dos mais usados.

4.2.1 plot

Este é o mais usado deles com ele se plota gráficos bidimensionais cartesianos e paramétricos. Pode ser usado para plotar funções ou dados.

`>plot ⟨função e/ou arquivo de dados⟩ [parâmetros adcionais]`

Esta é a forma mais básica que se pode usar, nesta caso ele irá adotar os parâmetros padrões para o seu gráfico. Porém você pode "complicar"o quanto queira de forma a obter um gráfico melhor apresentado ou apenas personalizado.

Exemplos simples de utilização do comando *plot*:

```
>plot [-0:10] exp(-0.3*x)*sin(x) title 'curva' with lines
>plot ‘‘dados.dat’’ using 1:2 title 'pontos' with points
```

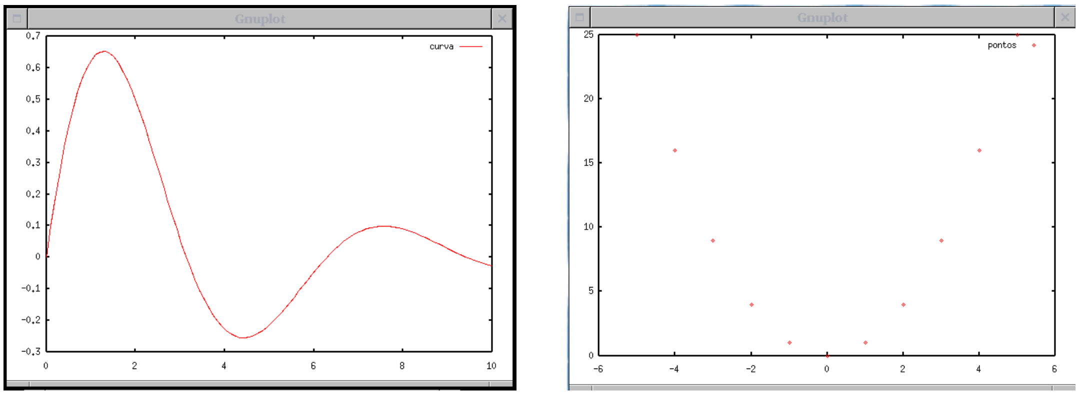


Fig. 4.2: Plot

O especificador de parâmetros *title* diz qual é o título da legenda do gráfico, pode ser qualquer texto sem formatação. O *with* diz como será plotado o gráfico, isto é com linhas, pontos, etc ... O *using* diz com quais colunas eu irei plotar os meus dados, sendo que ele adota o seguinte formato x:y:[z].

Exemplos complexos :

```
>plot exp(-0.3*x)*sin(x) title 'curva 1' with lines, \
exp(-0.1*x**2)*cos(x) title 'curva 2'with linespoints, \
‘‘dados1.dat’’ using 1:3 title ‘‘dados 1’’ with points, \
‘‘dados2.dat’’ using 2:5 title 'dados 2' with dots
```

As "\" são para que ele entenda tudo como uma única linha.

4.2.2 splot

Este comando é extremamente semelhante ao plot(4.2.1) porém usa-se para plotar gráficos em duas dimensões, sendo assim você deve especificar funções de três variáveis ou ter um arquivo de dados com três colunas no mínimo.

`>splot [-10:10] sin(2*x)*cosh(y/2)`

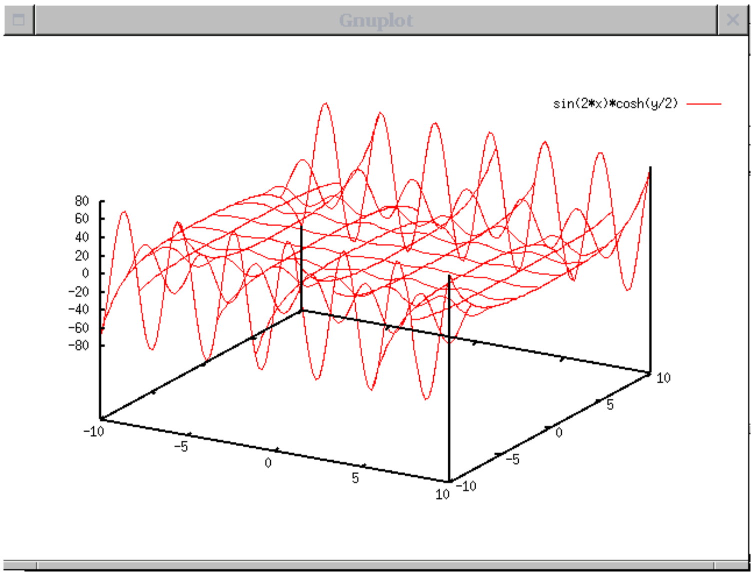


Fig. 4.3: SPlot

Este comando usa as mesmas opções do comando *plot* porém ele trabalha sempre ou com funções de duas variáveis ou arquivos com no mínimo três colunas.

4.2.3 *set*

Este comando muda os valores padrões de certas opções dentro do programa, como estilo de gráfico, título do gráfico, nome dos eixos,etc ...

`>set terminal x11` - *diz qual o terminal de saída, isto é como você vai visualizar o gráfico.*

`>set title ‘‘Gráfico 01’’` - *define um título para o gráfico*

`set output ‘‘grafico01.eps’’` - *define aonde aonde o gráficos será visualizado nste caso em um arquivo em formato “postscript”*

`>set xlabel ‘‘tempo’’` - *define o nome do eixo “x”*

`>set ylabel ‘‘posicao’’` - *idem acima para o eixo “y”*

`>set parametric` - *define o modo paramétrico*

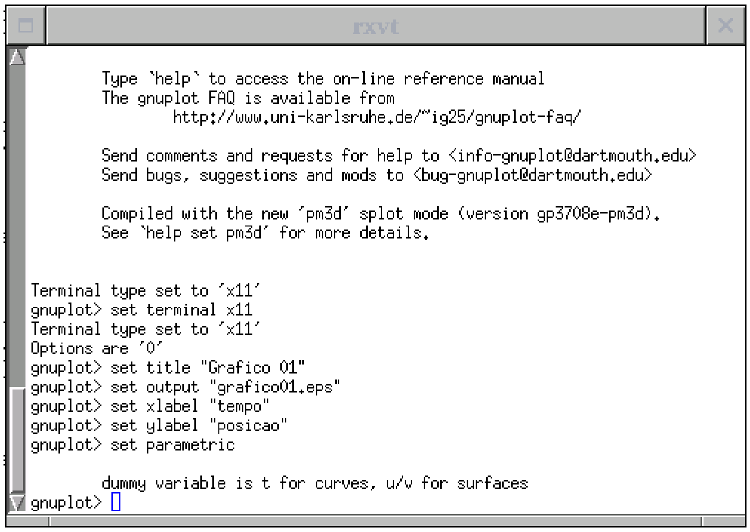


Fig. 4.4: SET

set terminal

Um terminal, neste caso, significa o tipo de dipositivo de exibição, pode ser o vídeo, impressora, ou um arquivo, nos casos mais comuns. O vídeo é o terminal padrão, mas se você quiser especificar um outro terminal como "GIF", "EPS", etc ... que podem ser usados para salvar o gráfico em uma imagem.

`>set terminal` ⟨*nome do terminal*⟩

set parametric

Define o modo de função paramétrica, isto é ele assume que você irá fornecer os valores de x e/ou y em função de parâmetros.

```
>set parametric
>plot 2*cos(t),2*sin(t)
>splot sin(v)*cos(u),sin(v)*sin(u),cos(v)
```

set output

Com este linha de comando você consegue salvar o seu gráfico em uma imagem ou imprimí-lo, para isto basta "setar"(definir) o terminal correto e a saída correta.

`>set output` ⟨*"nome da saida"*⟩

4.2.4 save

Este comando é sem duvida "uma mão na roda", com ele você salva todas as definições do gráfico que você criou em um arquivo, com isso você poderá **carregar**

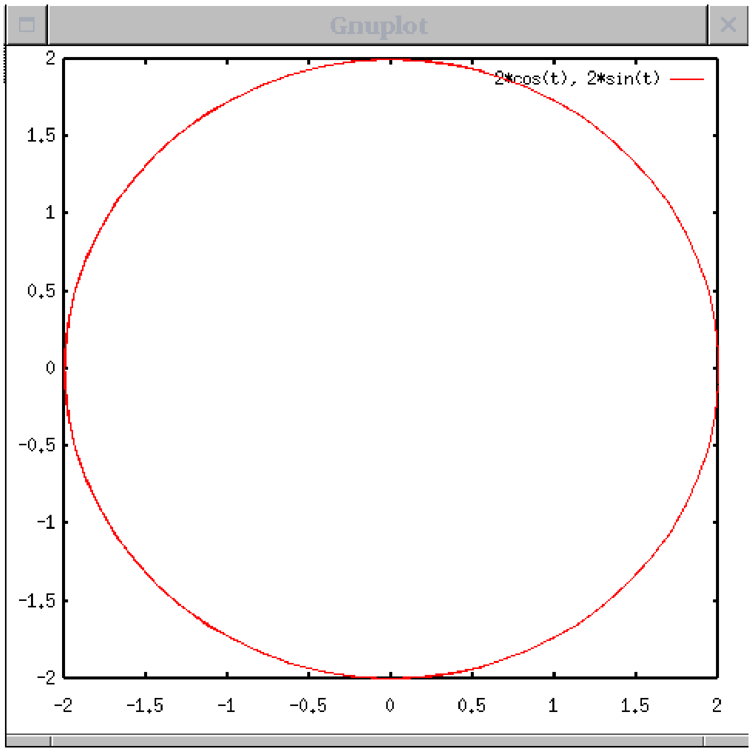


Fig. 4.5: Plot Parametric

este arquivo e o seu gráfico irá ser plotado em modo automático. Nada o impede de que você faça alterações e salve-as no arquivo novamente.

`>save` ⟨*"nome do arquivo"*⟩

4.2.5 load

Este comando carrega um arquivo salvo (4.2.4) para ser plotado.

`>load` ⟨*"nome do arquivo"*⟩

4.2.6 pm3d

Este é um recurso novo que não esta disponível nas versões atuais do **GNUPlot**, para que você tenha acesso a este recurso se faz necessário recompilar o programa adicionando este recurso.

Este recurso trás a possibilidade de se fazer gráficos em gradientes de cores muito útil para que trabalha com gráficos tridimensionais e mapas.

`> set pm3d`

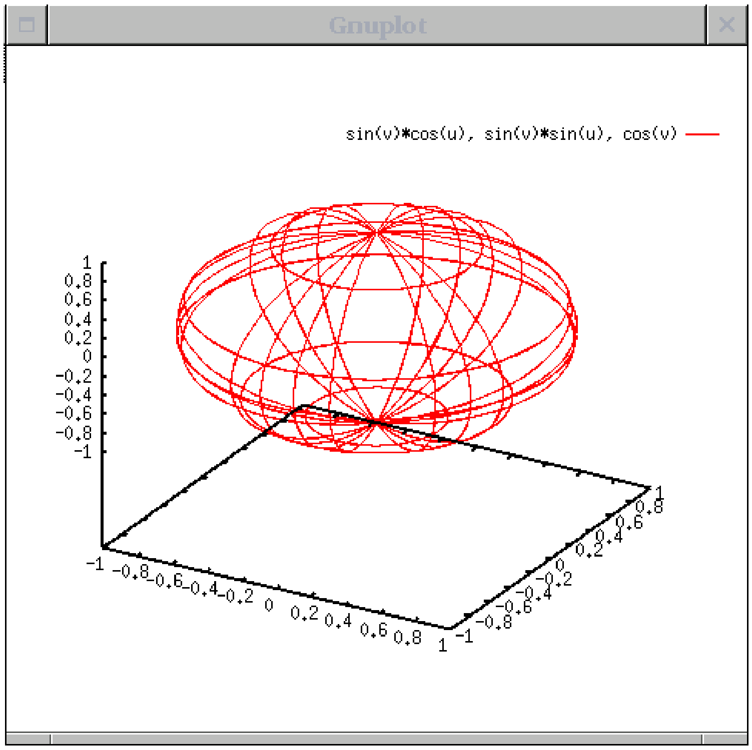


Fig. 4.6: SPlot Parametric

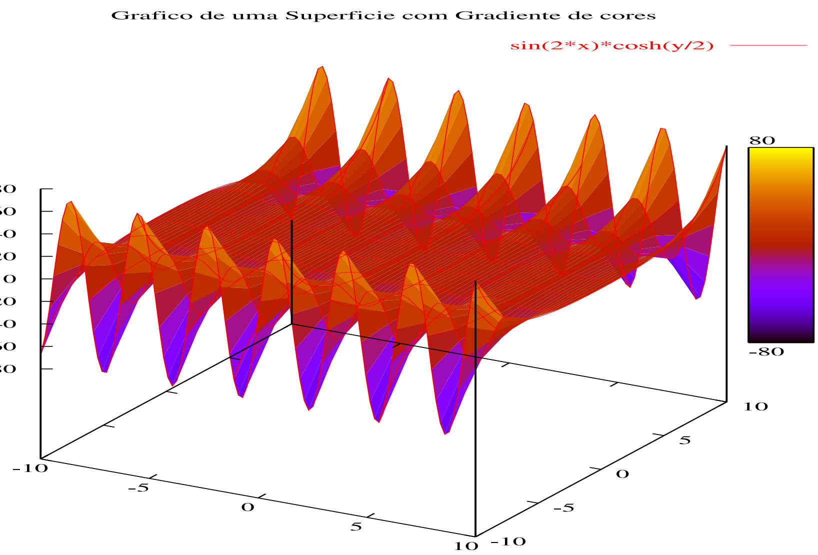


Fig. 4.7: PM3D

4.2.7 help

Outra coisa muito importante que n ao podemos deixar de falar é sobre comando `help`, basta digitar `help` na linha de comando do **GNUPlot** que ele lhe dará um help de **tudo** o que ele pode fazer.

`help`

4.3 XMgr/XMgrace ©2000 MARLON SCHMITZ

Este programas possuêm uma filosofia bem diferente do **GNUPlot**, uma vez que se baseiam em uma interface gráfica muito intuitiva que ajuda trabalhar, principalmente para os usuários que não se adaptam bem a linha de comando. Estes programas possuêm a limitação: de só pode fazerem gráficos em duas dimensões.

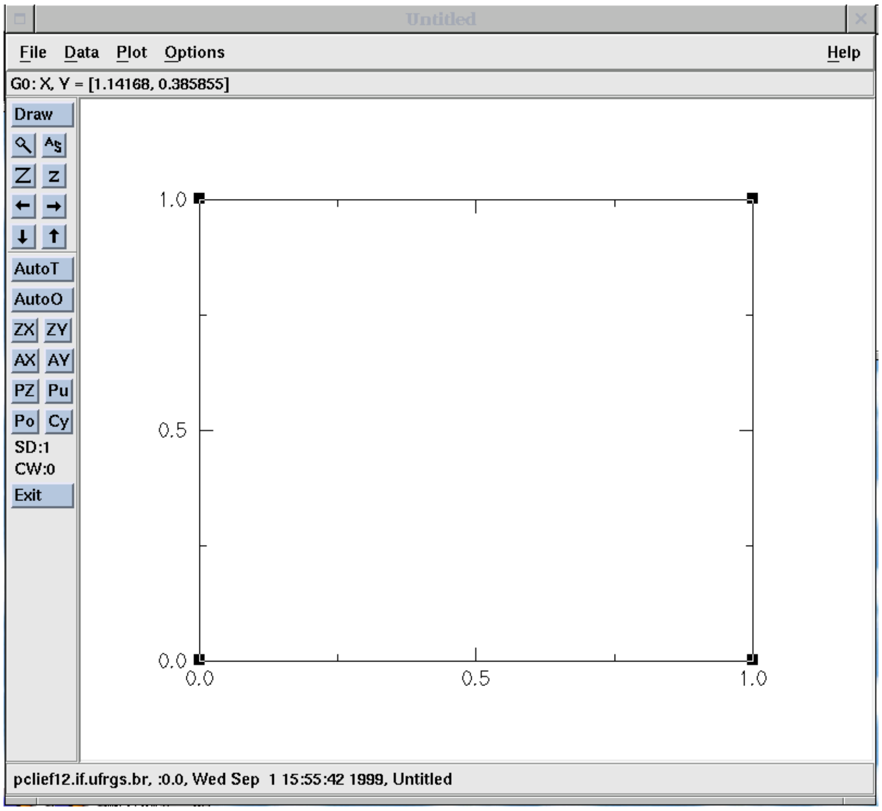


Fig. 4.8: XMGR

Não explanarei muito sobre estes programas aqui no material escrito, porém nos ateremos na parte prática onde mostra-se mais proveitoso.

Serei breve:

Para se ler um arquivo de dados via arquivo basta clicar em

FILE - **READ** - **BLOCK DATA**

Aparecerá então uma tela de abertura de arquivos que permite selecionar o arquivo a ser lido. Após ele irá perguntar como deve ser lido este arquivo, isto é será mostrado quantas colunas e quantos dados tem e cada coluna para que você possa escolher as colunas de dados que se quer plotar. Você pode escolher o tipo de gráfico, xy, xy com erro em x, y ou ambos, etc ...

Clicando duas vezes no fundo do gráfico ele irá lhe mostrar uma tela de formatação de gráfico e assim por diante clicando duas vezes nos eixos você formata-os , etc ...

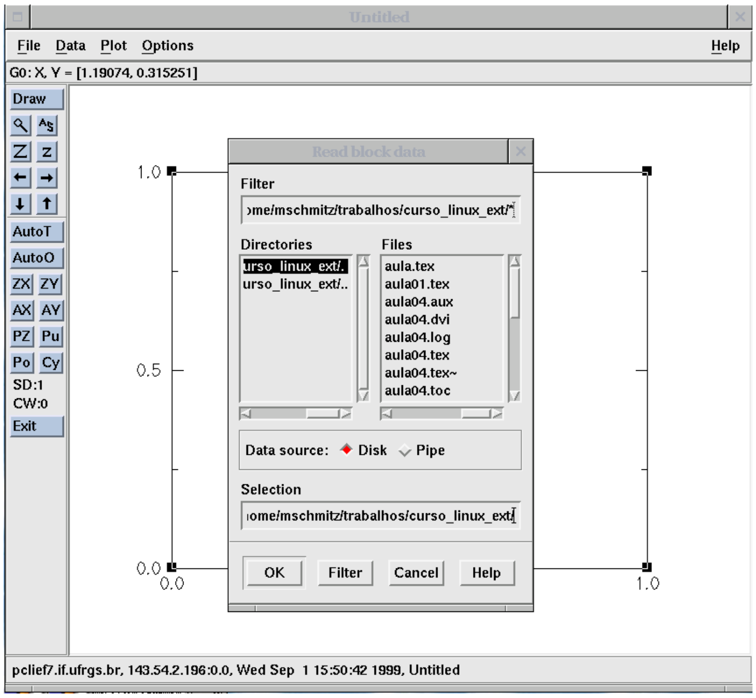


Fig. 4.9: XMGR - ler arquivo

O XMgrace, ou simplemente grace, é o programa que já substituiu o XMgr, porém ele ainda não vem como pacote padrão nas distribuições, mas é de relativa facilidade “baixar” este pacote.

4.4 XV

©2000 Leonardo Brunnet

O programa **xv** mostra imagens nos formatos GIF, JPEG, TIFF, PBM, PGM, PPM, X11 bitmap, Utah Raster Toolkit RLE, PDS/VICAR, Sun Rasterfile, BMP, PCX, IRIS RGB, XPM, Targa, XWD, possivelmente PostScript, e PM em sistema de janelas X11. A documentação detalhada é enorme e inadequada para uma página de `man`. Se você instalou os “docs” em seu computador deve encontrar

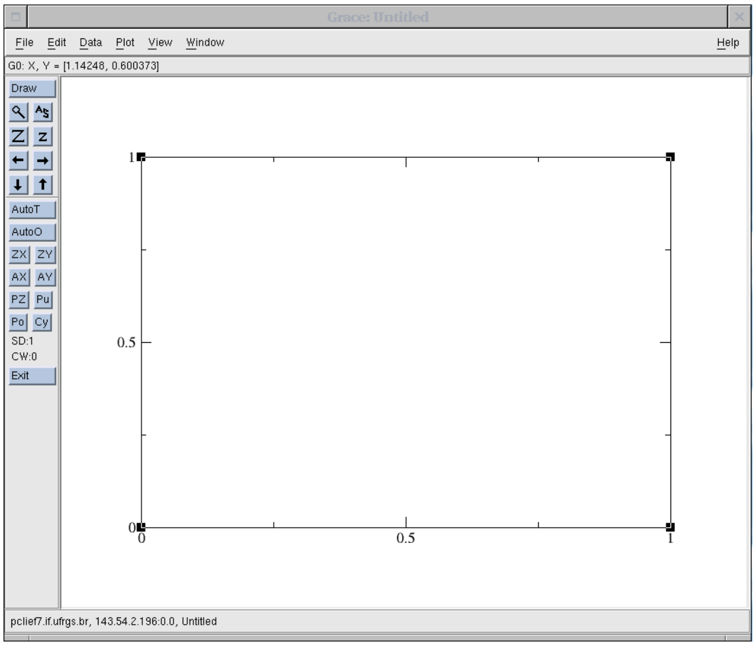


Fig. 4.10: XMGRace

arquivos em postscript em `/usr/doc/xv-3.10a`. Se você não tiver este arquivo pode encontrar a última versão da documentação em `ftp://ftp.cis.upenn.edu`.

Para ver a imagem `dontpanic.jpeg` digite apenas : `$ xv dontpanic.jpeg` e a imagem aparecerá. Para interagir com o programa através de uma janela você deve usar o botão direito do mouse sobre a imagem (veja figura 4.11). A partir da janela que surgir você poderá, por exemplo, salvar a figura com outro formato ou enviá-la para impressão.

Instalações recentes de Linux tem preterido o `xv` em detrimento de softwares semelhantes como: `ee, gqview` ou `kview`. Mas todos estes tem a mesma idéia básica: são aplicativos para visualização de imagens com a possibilidade de transformar de um tipo de arquivo para outro. Se você não tem o `xv` na sua máquina pode procurar pelas alternativas acima.

4.5 Xfig

©2000 Leonardo Brunnet

Xfig é uma ferramenta orientada por menu que permite ao usuário desenhar e manipular objetos no formato "fig" interativamente no sistema X11. Há uma guia de referência em HTML no menu de ajuda (*Help*) do programa ou ainda um guia básico "xfig.pdf" no diretório /usr/X11R6/lib/X11/xfig.

Usando o programa TransFig, Xfig pode exportar suas imagens para uma grande variedade de formatos.

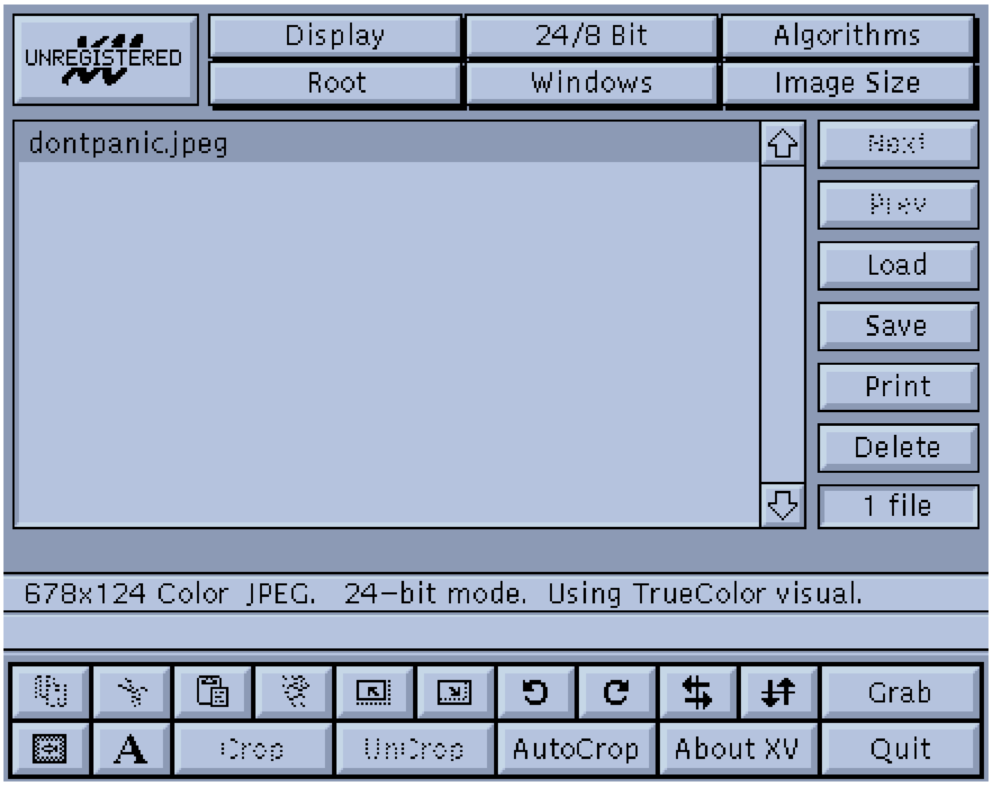


Fig. 4.11: Interface do usuário no `xv`

```
Nome        Linguagem
-------------------------------------------
box         LaTeX box (figure boundary)
latex       LaTeX picture
epic        LaTeX picture + epic macros
eepic       LaTeX picture + eepic macros
eepicemu    LaTeX picture + eepicemu macros
pictex      PiCTeX macros
ibmgl       IBMGL (or HPGL)
eps         Encapuslated PostScript
ps          PostScript
pstex       Combined PS/LaTeX (both PS and LaTeX parts)
textyl      Textyl special commands
tpic        TPIC
pic         PIC
mf          MF (MetaFont)
acad        ACAD (AutoCad slide)
pcx         PCX
png         PNG
gif         GIF*
```

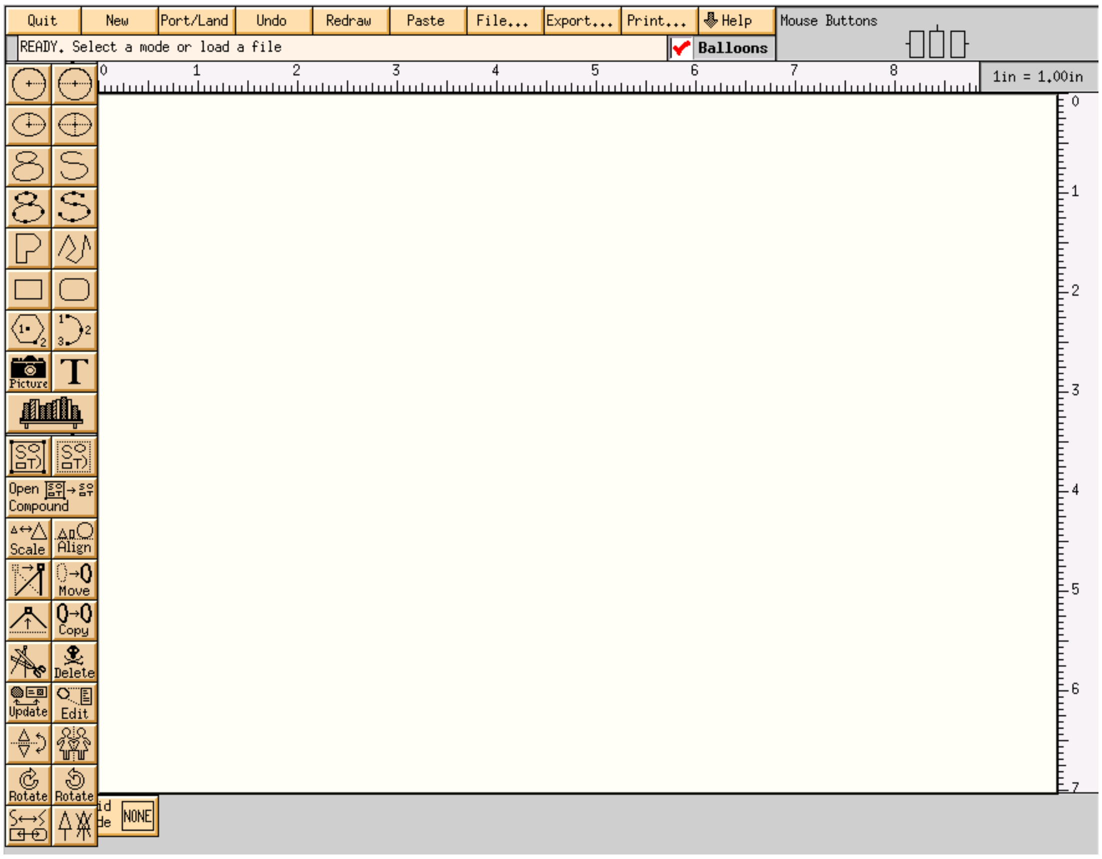


Fig. 4.12: Interface do usuário no `xfig`

`jpeg`	`JPEG (JFIF)`
`tiff`	`TIFF`
`tk`	`TK`
`ppm`	`PPM (portable pixmap package)`
`xbm`	`X11 Bitmap`
`xpm`	`X11 Pixmap (XPM3 package)`

Para abri-lo digite: `$ xfig` e você terá como interface a figura 4.12. Há também programas executados na linhas de comando que transformam o formato "fig"em outros: fig2dev, fig2ps2tex.

5. UNIX AVANÇADO

5.1 Expressões Regulares

©2000 Fabricio Ferrari

Uma **expressão regular** (ER ou *regexp*) é um padrão que denota um conjunto de caracteres. As expressões regulares são tradicionalente usadas para selecionar e alterar textos. A correspondência de uma ER com um determinado texto pode resultar em sucesso ou fracasso, que, dependendo das circunstâncias, é o resultado que se quer comparar.

Uma expressão regular especifica um conjunto de caracteres para comparação, algo como *qualquer linha que comece com* `a` *e termine com* `b` ou *qualquer palavra que contenha três ou quatro* `a`*'s juntos* e assim por diante. Estes padrões caracterizam a classe de comparação das expressões regulares. Veremos como representá-los por conjuntos de caracteres.

Expressões regulares são usadas por muitos aplicativos UNIX, como `grep, sed, awk, vi, emacs, perl` e outros. A maioria destes aplicativos possui o seu próprio conjunto de códigos para representar as ER, mas a maioria deles são comuns entre todos. A tabela 5.1 mostra as classes mais comuns de expressões regulares.

As expressões regulares formam entidades tão genéricas que devem ser previamente testadas para evitar efeitos colaterais. As ER representam um valioso recurso de trabalho. Além de sua generalidade, estão disponíveis em grande parte dos aplicativos do UNIX. Observe os exemplos que seguem.

Por exemplo, a ER `ab*c` significa `a` *seguido por zero ou mais (*`*`*)* `b`*'s e um c*. Coincidiria com `ac, abc, abbc, abbbc` e assim por diante. Já `ab+c` não representaria `ac`, mas `abc, abbc, abbbc` sim. Os caracteres `*, +, ., ?` são chamados de multiplicadores nas expressões regulares. Outra construção comum dos multiplicadores é `.*` que significa uma seqüência de qualquer tamanho e qualquer caracter.

Se colocamos o circunflexo ^ no início da ER, como em `^Âncora`, estamos coincidindo a ER com uma linha que começa com esta palavra, mas nenhuma outra. Já colocando um `$` no fim da linha, como em `fim da linha$`, restinge-se a RE a terminar com `fim da linha`. Assim, na ER `^tudo$` a linha só poderá conter o que tiver entre o ^ e `$`. Por restringir o início e fim da ER, ^ e `$` são chamados de âncoras.

Como as expressões regulares estão na maioria dos aplicativos de respeito no UNIX, convém conferir as pequenas variações entre os especificadores de um e

Especificador	Significado
`c`	Um caracter sozinho especifica ele mesmo
`.`	Coincide com qualquer caracter exceto `\n` (nova linha)
`*`	Zero ou mais ocorrências do caracter anterior
`+`	Uma ou mais ocorrências do caracter anterior
`?`	Zero ou uma ocorrência do caracter anterior
`|`	Alterna entre a classe anterior e posterior
`^`	Como o caracteres mais à esquerda, o circunflexo ancora a ER ao início da linha, ou seja, a linha deverá começar com o caracter imediatamente depois do ^
`$`	Como o caracteres mais à direita, o cifrão ancora a ER ao fim da linha, ou seja, a linha deve terminar com o caracter que imediatamente antecede o `$`.
$\{m\}$ $\{m,\}$ $\{,m\}$ $\{m,n\}$	Uma ER de um caracter seguido por $\{m\}$, $\{m,\}$, $\{,m\}$ ou $\{m,n\}$ é uma ER que coincide com um intervalo de ocorrências desta ER de um caracter. $\{m\}$ significa exatas m ocorrências; $\{m,\}$ para no mínimo m ocorrências, $\{,m\}$ no máximo m e $\{m,n\}$ entre m e n ocorrências. $\{0,\}$ corresponde a `*`, $\{1,\}$ a `+` e $\{0,1\}$ a ?.
`[...]`	Classes: qualquer caracter entre os colchetes.
`(...)`	Agrupamento
`\m`	$m = 1 - 9$ substitui a m-ésima ocorrência entre ()

Tab. 5.1: Especificadores de Expressões Regulares.

outro. Alguns aplicativos expandem a lista de especificadores enquanto outros apresentam pequenas variações. Além disso, existem situações em que os caracteres de classe das ER precisam ser mascarados, via caracter de fuga \, para que sejam corretamente interpretados pelo aplicativo. Consulte o manual do aplicativo em questão (*man page*) e a página 7 do manual *regex*, digitando `man 7 regex`.

5.2 Sed

©2000 Fabricio Ferrari

O `sed` é um editor de fluxo (**s**tream **ed**itor) que lê a entrada padrão uma linha por vez, executa uma série de comandos de edição e escreve na saída padrão. Estes conjuntos de operações podem ser gravados num arquivo (chamado *script*) contendo os vários comandos a ser executados. Se os comandos são poucos, também podem ser informados via linha de comando. No caso do *script*, deve-se informar o nome do arquivo que contém os comandos através da opção `-f`

`$ sed -f script.sed ⟨arquivo⟩`
onde `script.sed` contém a seqüência de comandos a ser aplicados no ⟨arquivo⟩. No caso do comando ser informado na linha de comando, utiliza-se a diretiva `-e`, conforme abaixo
`$ sed -e '`*comandos*`' ⟨arquivo⟩`

5.2.1 O Comando de Substituição

O comando de substituição serve para trocar uma cadeia de caracteres por outra num texto. O formato do comando é

[*endereço1*[,*endereço2*]]s/⟨procura⟩/⟨substituição⟩/{*modificadores*}

onde os modificadores podem ser os seguintes

n substitui a n-ésima ocorrência de ⟨procura⟩ por ⟨substituição⟩

g substitui todas as ocorrências de ⟨procura⟩ por ⟨substituição⟩

p escreve na saída padrão se alguma substituição é feita

w ⟨arq⟩ escreve a saída padrão para ⟨arq⟩ se alguma substituição é bem sucedida

Um endereço pode ser uma Expressão Regular (ER) entre barras `/ER/` ou um número de linha. Se dois endereços estão separados por vírgulas, a substiuição é aplicada para todas as linhas entre as duas especificadas, inclusive. Por exemplo, considere o arquivo `linhas`

```
$ cat linhas
linha 1 (um)
linha 2 (dois)
linha 3 (três)
```

Para aplicarmos a substituição de `linha` por `LINHA` nas linhas 1 e 2, escrevemos

```
$ sed -e '1,2s/linha/LINHA/' linhas
LINHA 1 (um)
LINHA 2 (dois)
linha 3 (três)
```

ou seja, a substituição de `linha` por`LINHA` só acontece nas linhas 1 e 2, conforme especificado no endereço `1,2s/...` . Ao invés de usar números de linha, podemos endereçar a substituição através de Expressões Regulares, conforme abaixo

```
sed -e '/um/s/linha/LINHA/' linhas
LINHA 1 (um)
linha 2 (dois)
linha 3 (três)
```

pois o `sed` somente executa a substituição nas linhas que contém a ER `/um/`. É uma outra forma de endereçamento.

5.2.2 O Comando de Apagar

A sintaxe do comando de apagar (*delete*) é bastante simples

[*endereço1*[,*endereço2*]]**d**

que apaga as linhas no intervalo de endereços especificado. Voltando ao arquivo `linhas`

```
$ sed -e '1,2d' linhas
linha 3 (três)
```

O endereço especificado poderia consistir de Expressões Regulares no lugar de números de linhas. O `sed` é muito usado em dutos, quando a operação acima seria escrita como

```
$ cat linhas | sed -e '1,2d'
linha 3 (três)
```

5.3 Awk ©2000 Leonardo Brunnet

`Awk` é uma linguagem de procura de padrões e processamento. Sua implementação no projeto GNU é o **gawk**. Uma referência completa pode ser encontrada em "The AWK Programming Language", de Aho, Kernighan, e Weinberger. A forma genérica do comando é:

`$ gawk [ opções ] -f ⟨arquivo-programa⟩ [-] arquivo`

ou

`$ gawk [ opções ] -file ⟨arquivo-programa⟩ [-] arquivo`

As opções podem ser em formato "POSIX"de letras precedidas por -"ou em formato longo "GNU"quando precedidas por "-". `Veja man awk para detalhes das opções.`

`Um programa` **awk** `consiste de uma seqüência do tipo` ***padrão-ação***`. A seguir listamos exemplos extremamente simples do uso do awk na linha de comando. Não esqueça de proteger o conjunto` ***padrão-ação***`do shell com aspas simples (' '). Construa o arquivo pinguim com o seguinte conteúdo:`

```
1 0
1 2
3 4
5 0
6 7
8 9
10 11
```

`Para imprimir na tela os elementos da primeira coluna do arquivo maiores que 4 digitamos:`

```
$ awk '{if($1>4 )print $1}' pinguim
5
6
8
10
```

Para imprimir na tela os elementos da segunda coluna do arquivo quando os elementos da primeira são maiores que 4 digitamos:

```
$ awk '{if($1>4 )print $2}' pinguim
0
7
9
11
```

Nestes dois exemplos está implícito que o separador dos campos das variáveis $1 e $2 é o espaço. Se quizermos explorar outros separadores (e.g. o ":" do arquivo /etc/passwd) usamos a opção -F. Suponha que nosso arquivo pinguim seja agora:

```
1:0:4
1:2:23
3:4:34
5:0:3
6:7:8
8:9:22
10:11:1
```

Se agora voce quiser selecionar os elementos da segunda coluna quando a terceira for maior que 20 digitará:

```
$ awk -F : '{if($3>20) print $2}' toto
2
4
9
```

Se você quer saber quais os usuários do sistema tem **UID** maior que 20:

```
$ awk -F : '{if($3>20) print $1}' /etc/passwd
nobody
alice
beto
leon
lucas
naty
sandra
```

Ou ao contrário, suponha que você queira saber o UID de um dado usuário. Neste caso você tem de usar aspas para indicar uma *string*. Vamos procurar o usuário nobody em /etc/passwd:

```
$ awk -F : '{if($1==''noboby'') print $3}' /etc/passwd
99
```

6.1 Mount, NFS

©2000 Leonardo Brunnet

O comando mount está na base da implementação da estrutura de diretórios do Unix. É através dele que se pode associar a um diretório desta estrutura dispositivos tão variados como **cdroms, disquetes, partições de diferentes discos rígidos** ou **diretórios de máquinas remotas**. Uma vez ''montado'', um dispositivo aparece para o usuário final apenas como um diretório na árvore de diretórios.

Em particular, é através da montagem de diretórios remotos fornecidos por um *servidor* que as estações *clientes* em uma rede podem compartilhar diretórios. É esta estrutura *servidor-cliente* montada através da rede que forma a base dos **sistema de arquivos da rede**, **NFS**. Assim, diretórios contendo programas (e.g. */usr/local*) e arquivos de usuários (e.g */var/spool/mail* ou */home*) comuns são armazenados e atualizados apenas no servidor aparecendo sempre os mesmos independentemente da máquina que o usuário esteja usando.

A forma genérica do comando mount é:

```
$ mount [-t tipo ] [dispositivo] [ponto de montagem]
```

Por exemplo se você quer montar o **cdrom** no subdiretório /cdrom, digite como **root**:

```
$ mount -t iso9660 /dev/hdc /cdrom
```

Iso9660 é o tipo de sistema de arquivos, /dev/hdc é o dispositivo associado ao **cdrom** e /cdrom é o ''ponto de montagem'', ou seja o diretório que aparecerá para o usuário final. Este diretório deve existir antes de se montar o dispositivo sobre ele.

Fstab

É no arquivo **fstab** que se sistematiza a estrutura da árvore de diretórios, todos os diretórios que são montados na inicialização do sistema ou ainda aqueles que um usuário final necessitaria montar como disquetes são colocados aí. Um exemplo:

```
/dev/hda1   /        ext2    defaults         1 1
```

```
/dev/hdb2   /back    ext2    defaults         1 2
/dev/hdb3   /home    ext2    defaults         1 2
/dev/hda3   /home1   ext2    defaults         1 2
/dev/hda2   /usr     ext2    defaults         1 2
/dev/hdb1    swap    swap    defaults         0 0
/dev/hdc    /cdrom   iso9660 noauto,ro,user   0 0
/dev/fd0    /dosf    vfat    noauto,user      0 0
none        /proc    proc    defaults         0 0
none        /dev/pts devpts  mode=0622        0 0
```

Para montar um diretório remoto, por exemplo /var/spool/mail fornecido pelo *servidor* com nome camboim no diretório /var/spool/mail em uma máquina qualquer da rede você digita nesta máquina:

$ mount camboim:/var/spool/mail /var/spool/mail

Neste caso é desnecessário usar o argumento -t. Mas é importante que o arquivo /var/spool/mail esteja localmente vazio, caso contrário seu conteúdo ficará oculto pelo, assim chamado, *volume* que virá pela rede.

Exports

É claro que para que você monte este *volume* ele terá de ser "exportado" pelo servidor. Os *volumes* que um servidor "exporta" você encontra no arquivo */etc/exports* do mesmo. Um arquivo *exports* típico tem no primeiro campo o *volume* a ser exportado e nos campos seguintes os enderecos das máquinas clientes.

```
#/etc/exports
/var/spool/mail    143.54.20.238 143.54.20.194
/home/ftp/pub (ro)
```

A primeira linha não comentada indica que o diretório /var/spool/mail está sendo exportado para as clientes 143.54.20.238 e 143.54.20.194. A segunda linha indica que o diretório /home/ftp/pub (ro) está sendo exportado para qualquer máquina na rede mundial com permissão de leitura apenas (read only).

Daemons

Para que uma montagem qualquer funcione é necessário que esteja rodando o "daemon" rpc.mountd. Para que uma montagem via rede funcione dois outros "daemons" devem estar funcionando: o portmap e o rpc.nfsd. O primeiro "mapeia" as chamadas de procedimentos remotos (RPC) em números de porta do protocolo DARPA. O segundo é o "daemon" que recebe/envia no servidor/cliente as solicitações para estruturação do sistema de arquivos pela rede.

Dicas

Para saber se uma máquina com um dado *ip* está conectada à rede use o comando ping + <ip>. Se a máquina está conectada você receberá sucessivas mensagens indicando o tempo de acesso a ela. Quando você interromper (Crtl+C) o programa fornece uma estatística dos tempos de acesso à máquina remota.

Se você tem acesso de "root" pode usar o comando showmount -e camboim e verificar quais diretórios e para quem a máquina camboim está exportando.

6.2 Automount

©2000 Leonardo Brunnet

Automount é um comando que permite que um dispositivo ou um volume da rede seja montado automaticamente no sistema de arquivos quando o diretório correspondente é acessado sem a necessidade de se acessar diretamente o comando mount. Após um intervalo de tempo pré-determinado o sistema desmonta o referido diretório automaticamente. Este comando é útil seja para acessarmos um dispositivo de mídia variável, como um cdrom, seja para possibilitar que um volume da rede pouco usado seja automaticamente desmontado minimizando o tráfego na mesma.

A configuração básica é definida por um arquivo, /etc/auto.master. Este arquivo define diretórios genéricos sob os quais serão feitas as montagens automáticas. Não é possível usar **automount** diretamente no diretório raiz, . Assim, para organizar seu sistema de arquivos você poderia definir um ponto genérico para a máquina local, misc, e um outro para volumes da rede exportados pela máquina timbauva. Neste caso, um auto.master exemplo seria:

```
# Sample auto.master file
# For details of the format look at autofs(8).
/misc    /etc/auto.misc --timeout 60
/timbauva /etc/auto.timbauva --timeout 60
```

O formato deste arquivo amostra é: ponto de montagem, tipo de mapa e opções. O ponto de montagem deve existir a priori. O tipo de mapa pode ser um arquivo normal ou um mapa da rede distribuído pelo sistema de informação da rede, NIS. Se debaixo de /misc quisermos montar cdrom e um disquete, podemos usar o arquivo /dev/auto.misc para indicar isto:

```
# This is an automounter map
```

```
dos   -fstype=vfat,user  :/dev/fd0
cdrom -fstype=iso9660,ro :/dev/cdrom
```

Se abaixo de /timbauva você quiser montar o home e o mail exportados pela timbauva:

```
# This is an automounter map
home   -rw,soft,intr      timbauva:/home
mail   -rw,soft,intr      timbauva:/var/spool/mail
```

Assim quando você digitar:

```
$ cd /misc/cdrom
```

ou

```
$ ls /misc/cdrom
```

ou qualquer comando que se refira a este diretório, ele será montado. Para tornar as coisas menos tediosas pode-se fazer um **link**:

$ ln -s /misc/cdrom /cdrom

Assim para acessar o cdrom bastaria referir-se ao diretório /cdrom. É claro que você poderia usar apenas um mapa como o misc (este é o default) para fazer todas as montagens automáticas.

Finalmente, para lançar, parar, relançar ou simplesmente verificar o estado deste serviço use o "script"

/etc/rc.d/init.d/autofs

com as opções

start,stop,reload,status.

6.3 NIS

©2000 Leonardo Brunnet

NIS é uma sigla para representar o sistema de informações da rede (Network File System). Através do NIS é possível que um usuário entre em qualquer estação do **domínio** NIS usando a mesma senha e o mesmo "shell". Em qualquer destas estações este mesmo usuário pode trocar sua senha ou seu "shell", as atualizações sendo automaticamente repassadas para o(s) **servidor(es)** de NIS.

A estrutura conceitual de um ambiente NIS é composta por um **domínio** NIS, por um ou mais **servidores** de NIS e pelas estações **clientes** NIS. O domínio é definido pelo conjunto das estações que compartilham o **nome de domínio (domainname)** estando este nome definido na configuração de cada máquina do domínio. Digite,

$ domainname
mataatlantica

para descobrir o nome de seu domínio NIS. Vale a pena salientar que as máquinas que compõem um domínio de **IP** da rede (e.g. abaixo de 143.54.2.) não pertencem necessariamente ao mesmo domínio NIS. A rede NIS é construída sobre a rede **IP**. Assim, nada impede que, por exemplo, uma estação pertencente a um domínio NIS sirva arquivos a uma estação fora deste domínio.

A estrutura **servidor-cliente** de um dado domínio NIS é realizada por dois "daemons" lançados durante a inicialização do sistema: **ypserv** que roda no servidor e o **ypbind** que roda nos clientes e no servidor. Se você tem um instalador **rpm** (distribuições RedHat, Mandrake) pode facilmente instalar estes programas encontrados sob os nomes: **ypserv-versão.rpm** e **ypbind-versão.rpm**.

Para que as coisas funcionem do lado do cliente basta que você informe qual o **IP** do servidor e rode o programa **ypbind** na inicialização do sistema. Isto tudo pode ser feito a partir do **linuxconf**. Não esqueça que o **ypbind** também deve rodar no servidor.

Do lado do servidor as coisas são um pouco mais complexas. Ao instalar o **ypserv-versão.rpm** será criada uma estrutura de diretórios: /var/yp. Abaixo desta estrutura você deve criar um diretório com o nome do seu domínio NIS:

```
$ mkdir /var/yp/mataatlantica
```

No subdiretório /var/yp você encontrará um arquivo, Makefile. Todas as informações que você quer difundir pelo seu domínio NIS são feitas através de mapas NIS. Estes mapas serão produzidos quando você digitar make neste diretório. Este comando seguirá a configuração da referida **Makefile**. Editando a **Makefile** você escolhe quais são os mapas que devem ser distribuídos no seu domínio.

6.4 Samba

©2000 Leonardo Brunnet

Samba é um conjunto de programas que permite estabelecer relações servidor-cliente primárias compatíveis com o protocolo *Lan-Manager*. Isto inclui MSCLIENT 3.0 para DOS, "Windows for Workgroups", Windows 95/98, Windows NT, OS/2, DAVE para Macintosh e *smbfs* para Linux. Serviços tais como espaço em disco e de impressão são fornecidos pelo programa smbd, acesso a máquinas remotas pelo programa smbclient e montagem de sistema de arquivos pela rede usando smbmount. Outros programas relacionados ao conjunto: testprns que testa se uma dada impressora é válida para uso via samba, testparm que testa a correção do arquivo de configuração do servidor smbd e nmbd, o servidor de nomes de NetBIOS (*Network Basic Input/Output System)* que os clientes usam para procurar servidores.

6.4.1 Smbd

O servidor smbd funciona criando uma sessão cada vez que um cliente solicita um serviço (e.g. de impressão ou de espaço em arquivos). Cada cliente obtém uma cópia do servidor para cada sessão. Esta cópia serve todas as conexões feitas pelo cliente durante a seção. Quando todas as conexões do cliente são fechadas a sessão é terminada.

O servidor *smbd* é lançado quando o sistema é ligado, neste caso o programa é lançado com a opção -D passando a rodar imediatamente como "daemon"em "background". Esta *não* é a opção default. Veja detalhes em man smbd.

A configuração dos serviços do smbd é feita no arquivo /etc/smb.conf. Olhando-o você pode ter uma noção dos inúmeros serviços possíveis. Este arquivo está no Apêndice D, sendo bastante auto-explicativo. Seu formato consiste de seções e parametros. As seções começam com colchetes e continuam até encontrar os colchetes da seção seguinte. As seções contém parâmetros do tipo: nome=valor. Linhas comentadas são indicadas por "#" ou ";". Sempre que você fizer uma modificação pode testar seu funcionamento com o programa testparm mencionado acima.

6.4.2 Smbclient

É um cliente do tipo ftp que permite acessar recursos de arquivos e impressão em servidores. Digitando smbclient diretamente você obtém na tela as opções do comando:

```
$ smbclient
Added interface ip=143.54.2.83 bcast=143.54.2.255 nmask=255.255.255.0
Usage: smbclient service <password> [options]
Version 2.0.3
  -s smb.conf           pathname to smb.conf file
  -B IP addr            broadcast IP address to use
  -O socket_options     socket options to use
  -R name resolve order use these name resolution services only
  -M host               send a winpopup message to the host
  -i scope              use this NetBIOS scope
  -N                    don't ask for a password
  -n netbios name.      Use this name as my netbios name
  -d debuglevel         set the debuglevel
  -P                    connect to service as a printer
  -p port               connect to the specified port
  -l log basename.      Basename for log/debug files
  -h                    Print this help message.
  -I dest IP            use this IP to connect to
  -E                    write messages to stderr instead of stdout
  -U username           set the network username
  -L host               get a list of shares available on a host
```

```
  -t terminal code      terminal i/o code {sjis|euc|jis7|jis8|junet|hex}
  -m max protocol       set the max protocol level
  -W workgroup          set the workgroup name
  -T<c|x>IXFqgbNan      command line tar
  -D directory          start from directory
  -c command string     execute semicolon separated commands
```

Se você quer saber que tipo de serviços o PC *pcdpfis01* pode fornecer digite:

```
$ smbclient -L pcdpfis01
Added interface ip=143.54.2.83 bcast=143.54.2.255 nmask=255.255.255.0
Got a positive name query response from 143.54.2.68 ( 143.54.2.68 )
Password:

 Sharename      Type      Comment
 ---------      ----      -------
 ADMIN$         Disk
 WIN98          Disk
 PUB            Disk
 WWW            Disk
 PRINTER$       Disk
 HP5L           Printer
 WPS            Disk      HP LaserJet 5L (PCL)
 IPC$           IPC       Remote Inter Process Communication
```

A saída informa o nome de compartilhamento e o que está compartilhado na rede. Isto não significa que você tem acesso direto pois sempre há solicitação de senha. Tentemos o diretório WWW como usuário (-U) anônimo, aqui você *deve* incluir a opção do IP (-I).

```
$ smbclient //PCDPFIS01/WWW -U  -I 143.54.2.68
Added interface ip=143.54.2.83 bcast=143.54.2.255 nmask=255.255.255.0
Got a positive name query response from 143.54.2.68 ( 143.54.2.68 )
Password:
```

Caso não existam restrições digite diretamente enter e veja os arquivos à disposição com ls ou dir:

```
smb: \> ls
  .             D       0  Wed Dec  9 13:22:54 1998
  ..            D       0  Wed Dec  9 13:22:54 1998
  Prof.html     A    3506  Wed Sep 23 07:11:14 1998
  Disc98_1.html A    2586  Wed Mar  4 12:57:50 1998
```

Para trazer ou levar arquivos use os mesmos comandos do ftp, por exemplo busquemos Prof.html e aproveitemos para aprender os comandos disponíveis:

```
smb: \> get Prof.html
getting file Prof.html of size 3506 as Prof.html (37.2155 kb/s)
(average 37.2155 kb/s)
smb: \>help
ls             dir            du             lcd            cd
pwd            get            mget           put            mput
rename         more           mask           del            open
rm             mkdir          md             rmdir          rd
prompt         recurse        translate      lowercase      print
printmode      queue          cancel         quit           q
exit           newer          archive        tar            blocksize
tarmode        setmode        help           ?              !
smb: \> help ?
HELP ?:
        [command] give help on a command

smb: \> help !
          ![command] executes linux shell command
smb: \> !lpq
no entries
smb: \> quit
```

6.4.3 smbmount

Smbmount é uma versão do programa smbclient usada para montar arquivos SMB compartilhados diretamente na sua máquina. Ele implementa o comando mount que por sua vez chama diretamente o programa smbmnt que faz a montagem real. No exemplo abaixo "montamos" o diretório WWW compartilhado por PCDPFIS01 no ponto de montagem mnt para o usuário (-u) 500 do grupo (-g) 500.

```
$ smbmount "//PCDPFIS01/WWW" -c   'mount /mnt -u 500 -g 500' -I 143.54.2.68
Added interface ip=143.54.2.83 bcast=143.54.2.255 nmask=255.255.255.0
Server time is Fri Sep 24 21:27:52 1999
Timezone is UTC-3.0
security=share
```

Veja agora o que aparece "abaixo"de /mnt.

```
$ cd /mnt
$ ls
Disc98_1.html  Prof.html
```

Desta forma o diretório remoto de uma máquina rodando Win95, Win98, NT, OS2 ou mesmo Mac pode se tornar parte de sua árvore de diretórios local.

6.5 Conectando ao Provedor ©2000 Atila Bohlke Vasconcelos

6.5.1 Introdução

Antes de iniciarmos, gostaria de esclarecer alguns pontos. Existem diversas maneiras de alguém se conectar ao seu provedor: aqui eu descreverei a mais simples e imediata delas. Fiz esse documento pensando no usuário doméstico, que não está procurando nenhuma maneira mirabolante de fazer piruetas na Internet (apesar de isso ser possível :-), mas se você quiser ou precisar disso, deve consultar outras fontes. Usuário doméstico, entende-se como o sujeito que está no recôncavo do seu lar e precisa de uma maneira rápida e simples de usufruir os benefícios da Internet. Neste texto não será discutido a instalção de um modem, apesar de ficar o lembrete: FUJA DE WINMODEMS, eles não são modems, são apenas meio-modems.

O programa que utilizaremos para efetuar a conexão, será o KPPP, parte integrante dos pacotes do KDE. Existem outros programas (como o xisp) que efetuam o serviço de forma muito boa também, mas não serão abordados aqui.

6.5.2 O que é preciso saber?

Antes de iniciarmos, uma lista das informações necessárias para configurar o KPPP.

- O número do telefone do seu provedor (para dados)
- O DNS do seu provedor
- O tipo de autenticação utilizado pelo provedor (não é vital)
- se o KPPP está corretamente instalado

KPPP Instalado

Obviamente o programa KPPP deve estar instalado para podermos utilizá-lo. Se você já tem o KDE instalado, com certeza já tem o KPPP instalado, bastando apenas configurá-lo.

Tipo de autenticação

Essa informação não é vital para nós, pois sempre podemos descobrir isso pelo método da "tentativa e erro", mas sendo conhecido, isso nos poupa um monte de trabalho. Se você usa um "grande"provedor, é bem provável que o tipo de autenticação seja o PAP.

O DNS

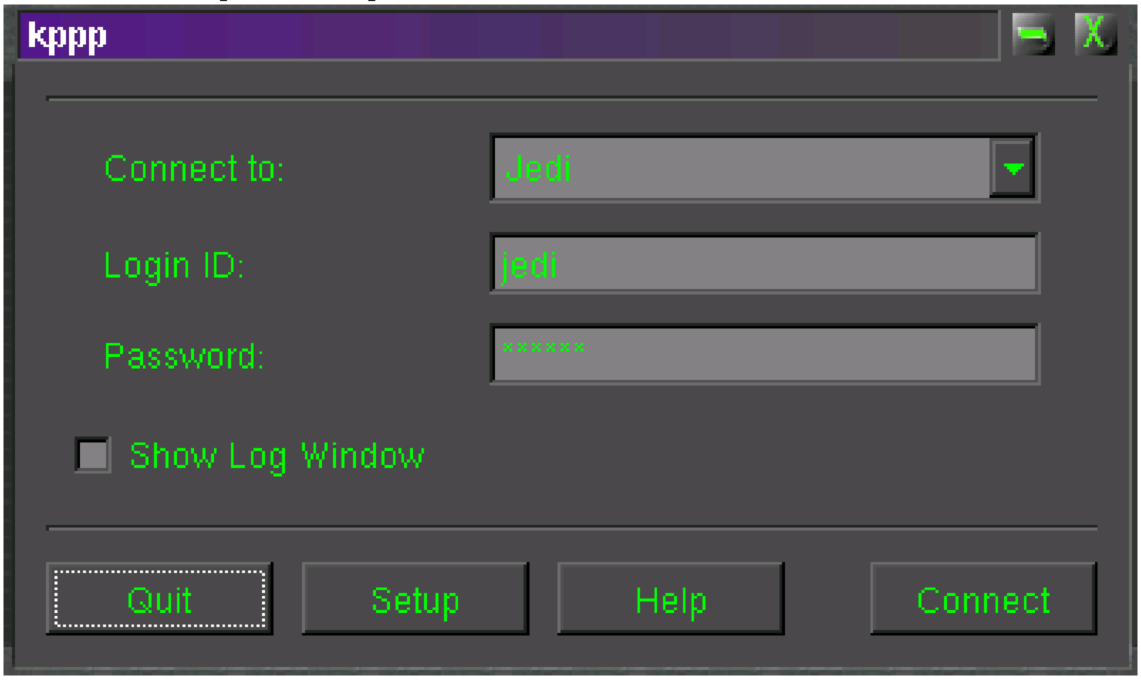
Se você não souber o DNS do seu provedor, talvez possa utilizar algum outro DNS, mas nem sempre isso é possível. Se não tiver acesso a nenhum DNS, então você não estará apto a usar nomes como *www.if.ufrgs.br* e terá que utilizar o respectivo número IP em seu lugar (no caso 143.54.2.210).

Número do telefone

Por motivos óbvios, se você não souber o número do telefone para qual discar, não poderá utilizar os serviços de seu provedor.

6.5.3 Colocando a mão na massa

Quando você iniciar o KPPP, surgirá uma tela semelhante a figura abaixo. Nela escolha a opção *Setup*



Escolhida a guia *Device* você pode configurar o *modem* que será usado. Como mencionado acima, pressupõem-se aqui que o seu modem esteja corretamente instalado e exista um link de /dev/modem para o seu dispositivo de modem. Se não existir, é bem possível que o seu modem esteja em /dev/ttySn onde n vai de 0 até 3 correspondendo, respectivamente, as portas COM1-COM4 do DOS.

Em *Flow Control* escolha CRTSCTS. Em *Line Termination* escolha CR e coloque *Conection Speed* no máximo.

Na guia *Modem*, é possível instruir o KPPP que comandos enviar para o modem, tais como volume e também é possível até mesmo fazer um autoteste do modem com a opção *Query Modem*.

Configurando as Contas

Com o KPPP é possível manter num mesmo computador várias configurações simultâneas para vários provedores diferentes ou contas diferentes num mesmo provedor ou contas diferentes em provedores diferentes. Não que isso faça muita diferença, pois geralmente os e-mails podem ser acessados independentemente do usuário com o qual realizamos a conexão.

Pois bem, vamos examinar a janela *Accounts*. O principal quadro na janela é uma lista de contas já configuradas. Nesse exemplo já temos uma configurada, mas para você esse quadro deve aparecer vazio.

Isso não é problema... vamos preenchê-lo agora escolhendo o botão *New...*. Nos será aberta a janela *New Acount:...* com a guia *Dial* já selecionada.

Nessa janela colocamos um nome qualquer pelo qual essa conta será conhecida, o número a ser discado e o método de autenticação. Aqui também podemos definir comandos, scripts ou programas a serem executados depois de efetuada a conexão, antes ou depois de desconectar.

Nota sobre autenticação

Apesar de o método PAP funcionar na maioria das vezes, ainda existem provedores com métodos de autenticação estranhos ou proprietários. Para isso existe o modo *script*, onde o usuário define um script que será executado para negociar com o computador do outro lado da linha a conexão.

Isso pode ser efetuado na guia *Login Script*. Como produzir tais scripts varia de caso para caso, isso fugiria a discução desse texto. Portanto apresento aqui um script típico que seria algo do tipo:

Expect:
ogin:
Send:
<seu-login-aqui>
Expect:
assword:
Send:
<xxxxxxxxx>

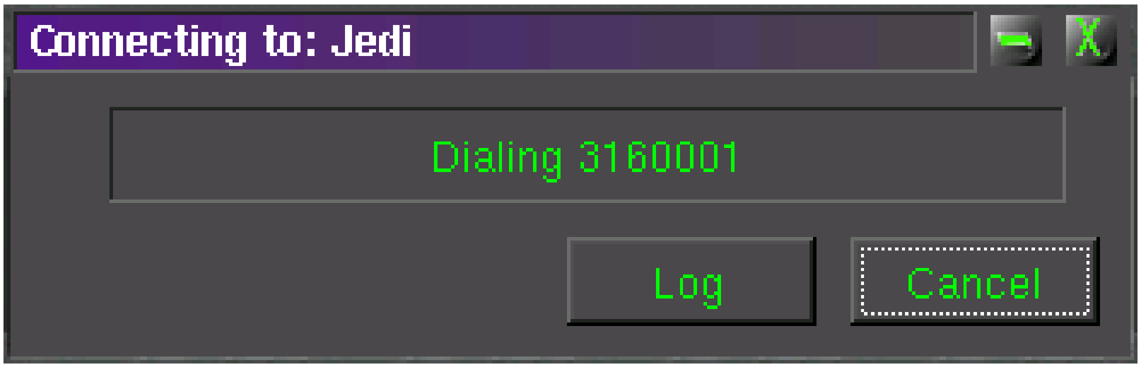
Na guia seguinte, podemos configurar o funcionamento do mecanismo de *IP*. No caso de usuários domésticos, apenas selecionamos *Dynamic IP Adress*.

Finalmente na guia *DNS*, deixamos tudo em branco, pois o mecanismo PAP se encarrega de administrar as coisas. Novamente o lembrete: se você não está usando PAP ou CHAP, então você deveria definir esses valores (de DNS) a mão (Por isso é importante saber o DNS do provedor).

Na guia *Gateway* escolhe-se o default.

6.5.4 Conectando

Finalmente, estando tudo devidamente configurado, podemos então nos conectar.



E se depois de conectado, você usar o botão *Details*, será presenteado com uma janela com informações detalhadas sobre a sua conexão.

Para desconectar, basta clicar em *Disconnect*

7.1 O Núcleo ©2000 Fabricio Ferrari

7.1.1 O Linux é o Núcleo

O núcleo é o Linux em si. Qualquer máquina que esteja rodando o núcleo do Linux é um sistema Linux, mesmo que o restante dos aplicativos e bibliotecas tenham sido alterados. É o núcleo quem controla os dispositivos do computador de acordo com as necessidades dos aplicativos, determinadas pelo usuário. O núcleo foi de onde o Linux nasceu realmente e a partir no núcleo é que os outros aplicativos são construídos. O núcleo continua sendo mantido pelo próprio Linus Torvalds, o criador original do Linux.

As fontes do núcleo podem ser encontradas em

http://www.kernel.org

7.1.2 As Interfaces de Configuracão

Nas instalações RedHat o núcleo é instalado em /usr/src/linux. Ali estão todos os arquivos necessários para recompilar o núcleo. Como **root**, você deve ir até este diretório[1] e configurar as opções de compilação. Primeiramente, faça um limpeza dos antigos arquivos-objeto (*.o) digitando

$ make mrproper

A seguir, passe à configuração dos módulos e diretivas de compilação. Basicamente exitem três interfaces para configurar estas opções. A primeira e mais simples é através de *scripts* do *shell*. Para configurar de tal modo, faça make config, conforme abaixo

```
ROOT /usr/src/linux \% make config
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in arch/i386/defconfig
#
*
* Code maturity level options
```

[1] Se você não instalou as fontes e os cabeçalhos do núcleo, pode lê-los de www.kernel.org.

```
*
Prompt for development and/or incomplete code/drivers
(CONFIG_EXPERIMENTAL) [Y/n/?]
*
* Processor type and features
*
Processor family
(386,486/Cx486,586/K5/5x86/6x86,Pentium/K6/TSC,PPro/6x86MX) [386] 586
  defined CONFIG_M586
Math emulation (CONFIG_MATH_EMULATION) [Y/n/?] n
MTRR (Memory Type Range Register) support (CONFIG_MTRR) [Y/n/?]
```

No exemplo acima o usuário terá de responder a uma série de perguntas para configurar a compilação do núcleo. Observe que entre colchetes [ ] está a resposta padrão indicada pelo nome ou pela letra maiúscula (Y/n), que será usada teclando-se enter.

O método a seguir é um pouco mais intuitivo e faz uso dos menus do console, conforme a Fig.7.1. O modo de menus é escolhido digitando

$ make menuconfig

Selecionado uma opção apresentada na tela da Fig.7.1, o usuário é levado às subseções específicas de cada conjunto de drivers, como por exemplo da escolha do tipo de processador, mostrado na Fig.7.2

A última das interfaces de configuração é a interface gráfica. Esta interface é mais amigável que as anteriores e por isso é preferida na maioria das vezes. Entretanto, requer que o modo gráfico esteja funcionando, o que nem sempre acontece num servidor, ou quando se está compilando o núcleo remotamente, por exemplo. Ela é chamada por

$ make xconfig

quando aparece a janela da Fig. 7.3. Um dos submenus existentes está mostrado na Fig. 7.4

Depois de ter passados por todas as janelas de configuração e gravado-as, pode-se partir para a compilação em si. Para tal, comece digitando

$ make dep

que vai averiguar as dependências entre os arquivos. Se tudo correr bem, compile o núcleo digitando

make zImage

para fazer um núcleo comprimido (gzip) ou

make bzImage

para comprimir ainda mais (bzip2) caso o anterior não seja suficiente[2]. Depois de terminado ($\sim 1^{\mathrm{h}}30^{\mathrm{m}}$ num 486), o novo núcleo estará em

/usr/src/linux/arch/i386/boot/zImage ou bzImage

[2] Por uma limitação da máquina do PC, todo o código do núcleo deve estar nos primeiros 640 kb, a chamada memória baixa.

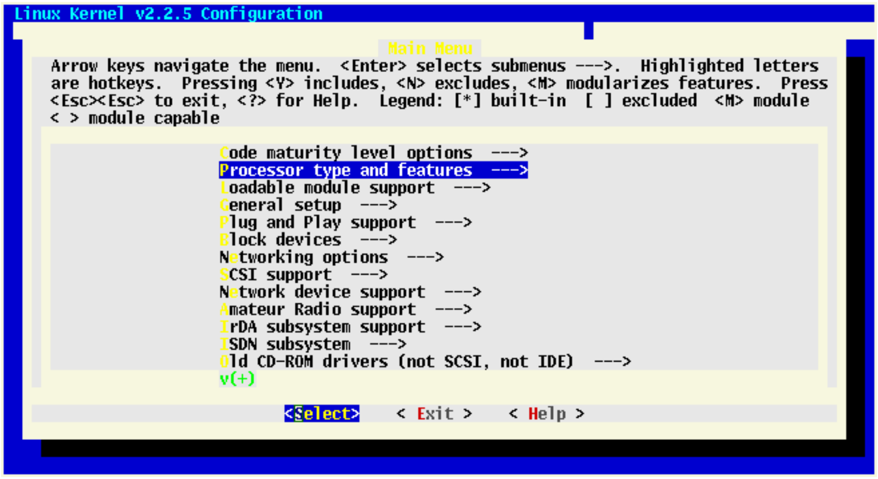


Fig. 7.1: A aparência do `menuconfig`

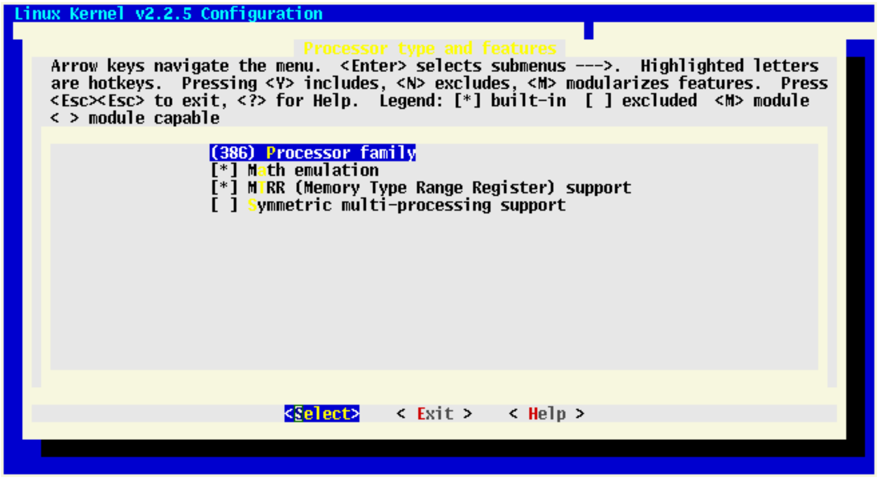


Fig. 7.2: O submenu escolhido na Fig. 7.1

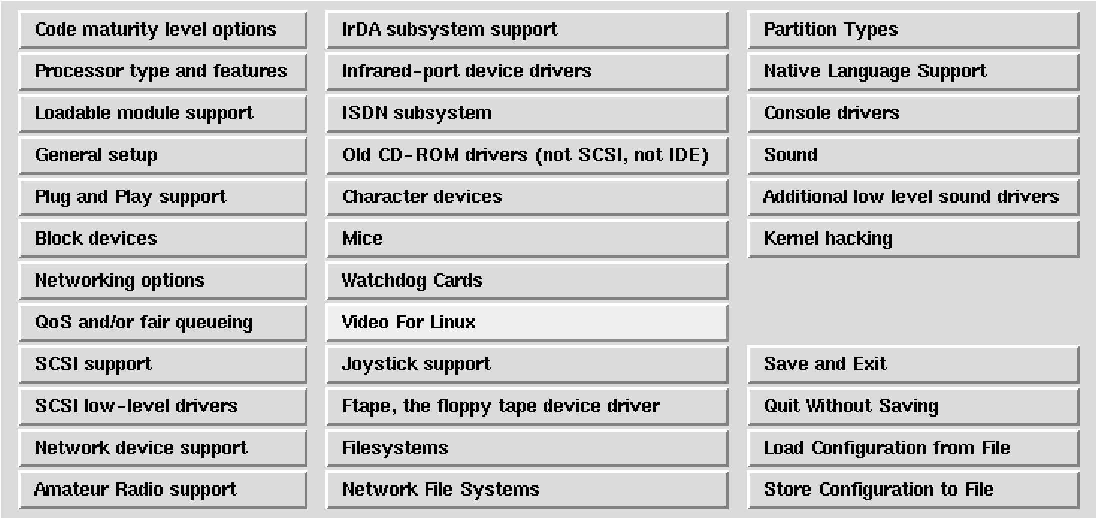


Fig. 7.3: A aparência do `xconfig`

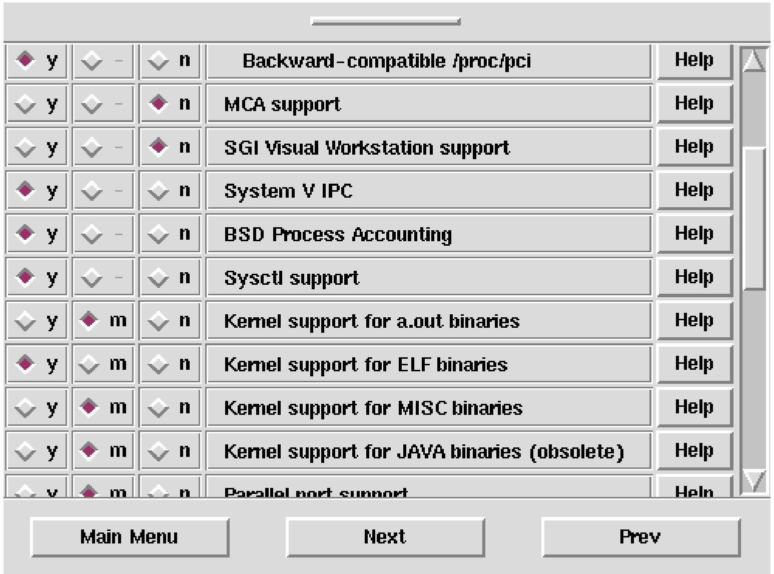


Fig. 7.4: O submenu `general setup` do `xconfig`

que deverá ser copiado para /boot com um nome que o identifique, por exemplo vmlinuz_set99. Este novo núcleo deverá ser identificado na configuração do LILO para que possa ser lido como *boot*.

Abaixo está um lilo.conf de exemplo, mostrando o núcleo original, que neste caso se chama vmlinuz-2.2.5-15 e o recém compilado, arbitrariamente nomeado como vmlinuz_set99. É sempre boa idéia manter o núcleo original **que funciona** (ou você não estaria tão longe) para o caso de algo acontecer errado. Se o novo núcleo funcionar, coloque-o em primeiro lugar no /etc/lilo.conf para que seja o padrão.

```
boot = /dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=600

image = /boot/vmlinuz-2.2.5-15
  label = linux.orig
  root  = /dev/hdb3

image = /boot/vmlinuz_set99
  label = linux.new
  root  = /dev/hdb3
```

7.1.3 Os módulos

Observe que muitas das opções ou acionadores de dispositivos[3] podem ser compilados como módulos. Um módulo é uma porção de instruções binárias em código de máquina para realizar tarefas específicas, armazenados em arquivos separados do arquivo do núcleo. Um módulo contém as rotinas específicas para tratar de determinado dispositivo que serão usadas somente quando necessário. Assim, o código binário fica disponível para ser usado quando for necessário, mas até então não compromete o tamanho, e logo o desempenho, do núcleo. Enquanto você não estiver seguro do que estiver fazendo, coloque todos os acionadores como módulos. Mantenha no núcleo somente o código que estiver sendo efetivamente usado em tempo integral.

Para compilar os módulos, basta fazer

$ make modules

em /usr/src/linux. Depois de compilados, execute

$ make modules_install

para instalá-los em /lib/modules/$x.y.z{-}w$ onde x, y, z, w representa a versão do núcleo em questão. Se você não quiser sobrescrever os módulos

[3] *device drivers*

originais do sistema com os que está compilando, remova o diretório de módulos para um lugar seguro antes de fazer make modules_install, por exemplo

mv /lib/modules/2.2.5-15 /lib/modules/2.2.5-15.orig

assim sua compilação vai criar um novo conjunto de módulos onde não estarão módulos desnecessários que você não selecionou.

É possível ver os módulos que estão sendo usados com o comando lsmod, removê-los com rmmod e inseri-los com insmod. Para testar se um módulo funciona com determinado dispositivo usa-se o modprobe. Veja o manual destes aplicativos[4] para detalhes atualizados.

ATENÇÃO: Execute o `/sbin/lilo` no final da instalação para atualizar os mapas do *boot* correspondente !!!

[4] man ⟨aplicativo⟩

APÊNDICE

A. CARTÃO DE REFERÊNCIA DO EMACS

B. CARTÃO DE REFERÊNCIA DO VI

Comandos `vi` Básicos

Iniciando `vi`	
`vi` *arquivo*	abre ou cria arquivo
`vi` +18 *arquivo*	abre arquivo na linha 18
`vi` +/"foo bar"*arquivo*	abre arquivo na ocorrência de "foo bar"
`vi -r` *arquivo*	restaura arquivo perdido
`view` *arquivo*	abre arquivo para leitura
Comandos do Cursor	
`h`	move para esquerda
`j`	move para baixo
`k`	move para cima
`l`	move para direita
`w`	avança uma palavra
`W`	avança uma palavra, passa pontuação
`b`	retrocede uma palavra
`B`	retrocede uma palavra, passa pontuação
`H`	move para o topo da tela
`M`	move para o meio da tela
`L`	move para o pé da tela
`$`	move para o fim da linha
`0`	move para o início da linha
Enter	avança uma linha
BackSpace	retrocede um caracter
Espaço	avança um caracter
Ctrl+f	rola uma tela adiante
Ctrl+d	rola meia tela adiante
Ctrl+b	rola uma tela para atrás
Ctrl+u	rola meia tela adiante
Copiando e Movendo Texto	
`a`	insere à direita
`A`	insere à direita, no fim da linha
`i`	insere à esquerda
`I`	insere à esquerda, no início da linha
`o`	insere linha abaixo
`O`	insere linha acima

Comandos `vi` Básicos

Editando Texto	
`cw`	muda palavra (ou parte da palavra à direita)
`cc`	muda linha
`C`	muda parte da linha à direita do cursor
`s`	substitui texto por caracter sob o cursor
`r`	substitui caracter sob o cursor
`r-Enter`	quebra linha
`J`	junta a próxima linha com a atual
`xp`	transpõe caracter com o da direita
`~`	muda caso da letra, maiúscula ou minúscula
`u`	desfaz o comando prévio
`U`	desfaz todos comandos na linha
`:u`	desfaz o último comando de linha
Apagando Texto	
`x`	apaga caracter
`dw`	apaga palavra
`dd`	apaga linha
`D`	apaga parte da linha à direita
`:5,10 d`	apaga linha 5 a 10
Copiando e Movendo Texto	
`yy`	copia linha
`Y`	copia linha
`p`	cola linha copiada abaixo
`P`	cola linha copiada acima
Procurando e Substituindo	
/*texto*/	acha *texto*
?*texto*?	acha texto para trás
`n`	encontra próxima ocorrência (ou anterior)
`:/`*velho*`/s//`*novo*`/gc`	procura e substitui, c/ consulta
Salvando e Saindo	
`:w`	salva
`:w` *arquivo*	salva como *arquivo*
`:wq`	salva e sai
`ZZ`	salva e sai
`:q!`	sai sem salvar
Vários	
`:r` *arquivo*	insere *arquivo*
`:set nu`	mostra números de linha
`:set nonu`	esconde números de linha
`:!`*comando*	executa comando no *shell*
Ctrl+L	redesenha tela

C. EXEMPLO DE .TCSHRC

```
########################################################
#         Arquivo  .tcshrc                             #
#        #
#      tcsh  6.07.02                             #
#        #
#        ferrari@if.ufrgs.br  1999                     #
########################################################

# Comentarios comecam com '#'
# Para continuar uma linha na proxima, termine-a com  '\'

# ajusta permissoes de criacao para  rwx------
umask 077



# ++++++ Verifica a arquitetura   ++++++
# ++ e ajusta as variaveis de acordo +++

if ($?ARCH) unset ARCH
set ARCH = `uname`

if ($ARCH == Linux ) then
    setenv X11HOME /usr/X11
    setenv TEXHOME /usr/lib/texmf
else if ($ARCH == SunOS) then
    setenv X11HOME /usr/local/X11
    setenv TEXHOME /usr/local/tex
endif




# +++++    PATH's      ++++++

setenv MANPATH .:$HOME/util/man:/astrodsk/local/man:/usr/man:\
/usr/local/man:/usr/local/tex/man:$X11HOME/man:/usr/local/f77/man:\
/usr/openwin/share/man

setenv HPATH $MANPATH

set lpath = ( /bin              \
      /usr/bin          \
      /usr/local/bin    \
```

```
      /sbin             \
              /usr/sbin          \
      $X11HOME/bin      \
      /opt/kde/bin      )

set mypath = ( $HOME/util/bin      \
       $TEXHOME/bin       )

set path = (. $mypath     \
      $lpath      \
      /usr/ucb    \
      /usr/etc    )




#+++++       ajustes para TCSH       +++++
#++ veja o manual do tcsh para detalhes ++
#++             (man tcsh)              ++

set prompt="%U%m%u %~ %% "
set prompt3="------> %R "
set correct=all
set autologout=0
set listjobs
set printexitvalue
set rmstar
set matchbeep=never
set ignore_symlinks
set pushdtohome
set watch=(2 any any)
set history=100
set savehist=500
set ignoreeof
set filec=1
set fignore=(.log .aux .o .toc .tof .lot  .log )
set noclobber

set tperiod=20
alias periodic w

setenv EDITOR 'vi'
setenv PRINTER psl205
setenv PAGER 'less'
```

```
#+++++ ALIASES +++++


alias ff 'find . -name \!* -print'

alias ^L             clear
alias l             'ls -lLF'
alias la            'ls -A'
alias ll            'ls -f'
alias qual          'ls `which \!*`'

alias cp            'cp -i'
alias mv            'mv -i'
alias rm            'rm -i'
alias h             'history \!* | tail  -39 | more'

alias bye            logout




# Fim do arquivo
```

D. EXEMPLO DE SMB.CONF

`samba.conf`