INTRODUÇÃO

AO

SISTEMA OPERACIONAL

UNIX

Autoras:
Cristiana Munhoz Eugênio
Daniela Regina Barbetti

Universidade Estadual de Campinas
Centro de Computação
Versão: 4 - Mai/97

Histórico do UNIX

- Projeto MULTICS: trabalho da MIT, AT&T, BELL LABs e GE (Década de 60).
  - Sistema operacional experimental
  - Específico para o computador GE635
  - Criado para ser flexível e interativo

- KEN THOMPSON E DENIS RITCHE
  - Alteraram o complexo sistema operacional
  - Criaram um sistema de arquivos simples
  - Denominaram de sistema UNIX

- 1ª .Versão, BELL LABs, PDP-11 (1970)

- Versão 4, reescrita em *C*, tornando-se portável para outras máquinas e de fácil manutenção (1973)

- Aquisição por Universidades para ensino de "Projetos de Sistemas Operacionais" (1973)

- Sucesso relacionado ao desenvolvimento do sistema em linguagem de alto nível.

- Versão 7, 1ª . versão oficial da AT&T (1978)

- System V da AT&T e 4.2 BSD Berkeley

NOTAS:

Projetado em 1969, o sistema Unix tinha originalmente a intenção de propiciar um ambiente no qual os programadores pudessem criar programas. Logo ficou evidente que o Unix também propiciava um ambiente no qual usuários da área comercial, científica e industrial pudessem executar programas para ajudá-los em seu trabalho.

O sistema Unix é executado em tantos computadores e usado de maneiras tão diferentes que o sistema operacional básico gerou dezenas de implementações. Uma implementação é uma versão adaptada do sistema, para um computador específico.

O objetivo de todos os sistemas operacionais é mais ou menos o mesmo: controlar as atividades de um computador. Os sistemas operacionais diferem na maneira como eles fazem seu trabalho e nas caracterísitcas adicionais que oferecem. O Unix é único em seu desenho modular, que permite aos usuários acrescentar ou remover partes para adaptá-lo às suas necessidades específicas. Os programas em Unix são como peças de um quebra-cabeça; os módulos se encaixam como conexões-padrão. Você pode tirar um módulo e substituí-lo por um outro ou expandir o sistema acrescentando vários módulos. De uma certa maneira, o sistema Unix de cada pessoa é único. Muitos usuários acrescentam ou eliminam módulos sempre que preciso, adaptando suas implementações às suas necessidades. Se você não precisa de um módulo, pode geralmente removê-lo sem prejudicar a operação do resto do sistema. Essa característica é especialmente útil nas implementações de microcomputadores, onde as unidades de disco têm capacidade limitada; a remoção de programas desnecessários abre espaço para mais arquivos de dados.

Características do UNIX

- Multitarefa (Tempo Compartilhado)
  - executar programas
  - controlar periféricos
  - gerenciar performance
  - compilar programas
  - editar arquivos
- Multiusuário
- Ambiente Shell
- Sistema de arquivos
- Comunicação interativa e correio eletrônico
- Transportabilidade

NOTAS:

- **Multitarefa** significa executar mais que uma tarefa ao mesmo tempo, por exemplo, compilar um programa ao mesmo tempo que editar um arquivo.
  A **multitarefa** em um computador permite que você execute simultaneamente tarefas que anteriormente teriam que ser executadas sequencialmente. O conjunto de tarefas não só é executado mais rapidamente como também você e o computador ficam livres para fazer outras coisas economizando tempo.
- Um sistema **multiusuário** permite que vários usuários usem o computador simultaneamente. Mais de um terminal pode ser conectado a um computador e os usuários de todos os terminais podem executar programas, acessar arquivos e imprimir documentos de uma só vez. O sistema operacional gerencia os pedidos que os vários usuários fazem ao computador, evita que um interfira no outro e atribui prioridades quando duas ou mais pessoas querem usar o mesmo arquivo ou a mesma impressora simultaneamente.
- O sistema Unix em si é muito **portátil.** Isto é, é mais fácil modificar o código do sistema Unix para implementá-lo em um novo computador do que reescrever um novo sistema operacional. Essa facilidade (portabilidade) do Unix tem sido uma das razões principais de sua aceitação.

DOS X UNIX

- Semelhanças:
  - Estrutura hierárquica de diretórios

- Diferenças:
  - Estrutura de comandos
  - Proteção de Arquivos
  - Processamento
  - Nº de usuários
  - Comunicação e correio eletrônico

NOTAS:

As partes do sistema Unix podem ser funcionalmente classificadas em três níveis: o *kernel* (núcleo), o *shell* (casca) e as ferramentas e aplicativos.

- O *kernel* planeja as tarefas e administra o armazenamento de dados.
- O *shell* é um programa que conecta e interpreta os comandos digitados por um usuário. Ele interpreta os pedidos do usuário, chama programas na memória e executa-os individualmente ou em uma sequencia chamada *pipe*.
- As ferramentas e aplicações incorporam capacidades especiais ao sistema operacional.

O sistema Unix de arquivos hierárquicos permite que você prepare índices eletrônicos para o grande número de arquivos de dados que geralmente acumulamos em nossos computadores. Ele também funciona como a estrutura básica através da qual você se desloca de uma área de trabalho para outra.

Shell

- Interpreta e executa comandos
- Alias (Substituir nome de arquivos ou comandos)
- Redirecionamento de entrada e saída
- Filtros (Pipeline ou conduto)
- Path (Caminho de diretórios)
- Linguagem de programação
- Ativado através do login pelos arquivos de inicialização: .cshrc e .login
- Shells mais conhecidas: Bourne e C

NOTAS:

O *shell* fornece uma conexão fácil entre você e o computador. Como os intérpretes que ficam entre pessoas que falam línguas diferentes, o *shell* fica entre você e o *kernel*. O programa shell interpreta os comandos que o *kernel* compreende. Ele diz ao *kernel* para fazer o trabalho que você solicitou, eliminando a necessidade de você ter de falar com o *kernel* diretamente em uma linguagem que ele entenda.

Contém, também, o recurso de encadeamento de comandos, ou *pipeline* (canalização). Um único comando de encadeamento pode fazer com que um arquivo de dados seja processado por vários programas sequencialmente. A saída de um programa flui pelo *pipe* e se torna a entrada do programa seguinte.

É uma linguagem de programação completa. Ele tem: variáveis, construções condicionais e interativas e ambiente adaptável ao usuário.

Existem dois shells comumente usados: o Shell Bourne desenvolvido pela Bell Laboratories e o Shell C desenvolvido na Univeridade da Califórnia, em Berkeley.

É permitido o uso de metacaracteres. O Shell oferece ao usuário carcateres especiais (* ? \ [...]) para permitir a substituição automática de caracteres em nomes de arquivos. Suas principais vantagens são: reduz a quantidade de digitação necessária, encoraja boas convenções de nomeação e simplifica a programação do Shell.

O shell permite o uso de arquivos que começam com ponto. São chamandos de arquivos escondidos. Por exemplo: .cshrc, .login, .logout, etc.

C Shell e Bourne Shell

- **C SHELL (%)**
  - Implementado na Universidade da Califórnia;
  - Processamento background e foreground;
  - Mostra status dos jobs;
  - History;
  - Alias;
  - Programação parecida com “C”.

- **BOURNE SHELL ($)**
  - Implementado pela Bells Laboratories;
  - Interpretador de comandos padrão em quase todas as implementações do Unix;
  - É menor porém mais eficiente na maior parte do processamento de Shell.

NOTAS:

- A *C Shell* tem a vantagem sobre a *Bourne Shell* de possuir a facilidade do history e controle de jobs, fora isso a *Bourne Shell* possui a maioria das características externas do *C shell**.***

- **Background**: coloca-se um programa para ser executado e o prompt é liberado.

- **History***:* acompanha os comandos à medida que você os digita, permitindo voltar e executá-los sem a reentrada do comando.

- **Alias*:*** sinônimos para comandos. Permite criar apelidos para comandos e arquivos.

- **Path*:*** indica os caminhos a serem percorridos na busca de um arquivo executável.

- ***.cshrc e .login:*** executam os comandos de customização no momento da execução do login.

Sistema de Arquivos

==> / (root)
/boot (arquivo de inicialização do s.o)
/unix (Kernel do sistema operacional)
==> /bin (executáveis principais)
/dev (devices, arquivos de entrada/saída)
==> /etc (executáveis de gerenciamento, mapas)
==> /usr (linguagens, aplicativos)
/usr/bin (outros executáveis)
/tmp (arquivos temporários)
==> /var (arquivos que mudam frequentemente)
==> /home (diretórios de usuários)

NOTAS:

Um dos principais componentes de um Sistema Operacional é o ***Sistema de Arquivos.*** Um Sistema de Arquivos descreve o tipo e a organização dos dados gravados em um disco.

Os arquivos do Unix residem em um Sistema de Arquivo hierárquico, ou árvore invertida (semelhante a um organograma). Para implementar essa estrutura, o Unix usa um arquivo especial conhecido como *diretório* .Você pode pensar em um diretório como uma pasta ou gabinete de arquivo. Cada diretório é uma bifurcação em uma hierarquia, da qual outros ramos podem crescer.

Os diretórios e os arquivos de dados não são os únicos tipos de arquivos que o Unix oferece. Existem outros “arquivos” especiais que na verdade não são realmente arquivos, mas dispositivos, como manipuladores de terminal, drives de disco, drives de fita e assim por diante.

Acesso ao sistema

- Iniciar uma sessão

  login: *<username>*
  password: <senha>

- Alterar senha

  %passwd

  old password:
  new password
  re-type new password

- Finalizar uma sessão

  %logout

  ou

  %exit

NOTAS:

- O Unix por ser um sistema operacional multiusuário, possui controle de acesso.
- Todo usuário possui uma quota de disco onde armazena dados e executa tarefas.

  Passwords:
  - nunca usar o seu "login name" ou algo parecido
  - exige um mínimo de 6 e um máximo de 8 caracteres
  - deve conter no mínimo duas letras

- Caso você cometa um erro ao digitar o login ou a password, basta dar <CTRL><U> para limpar o campo.
- Após efetuar o seu login corretamente, aparecerá para você um símbolo de prompt do sistema, que pode ser:

  - **$ -** usuário comum (bourne shell)
  - **% -** usuário comum (c shell)
  - **# -** administrador do sistema (superuser)

- Se você estiver utilizando o Unix através de um micro e o winchester estiver particionado, por exemplo um micro com Windows 95 e FreeBSD, ao sair do sistema, quando estiver novamente no login, a máquina não pode ser desligada. Deve ser dado um <CTRL><ALT><DEL> para que seja feito um sincronismo entre o disco e a memória.

Arquivos de inicialização do SHELL

- Arquivos comuns contendo uma lista de uma ou mais linhas de comando do shell.
- Comandos utilizados para personalizar ou alterar seu ambiente de trabalho.
- Devem estar localizados no diretório-base para serem executados "automaticamente" no ato do login.
- Arquivo .login: executado pelo shell somente ao iniciar uma sessão. Esse arquivo deve conter comandos que precisam ser executados somente uma vez durante o login. Exemplo: stty
- Arquivo .cshrc: executado toda vez que uma nova cópia do shell for chamada (subshell).
- Ao iniciar uma sessão o arquivo .cshrc é executado antes do arquivo .login

NOTAS:

Ao alterar o arquivo .cshrc ou .login você deverá validar suas alterações utilizando o comando "source".

Exemplo:

%source .cshrc

Lembre-se que esses arquivos são muitos importantes, então antes de alterá-los faça uma cópia de segurança.

Alguns comandos

- Configuração da conta:

  %set

- Configurar a tecla “Backspace”:

  %stty erase <tecla_backspace>

- Sistema Operacional que está sendo utilizado:

  %uname -a

- Identificação do usuário:

  %id ou %whoami

- Verificar os usuários que estão utilizando a máquina:

  %who

- Limpar a tela:

  %clear

NOTAS:

O Unix permite que vários comandos sejam executados na mesma linha. Para isso basta utilizar ; (ponto e vírgula).

Todo comando no UNIX possui a seguinte característica:

%comando (-)opções argumentos

Help

- Ajuda sobre um determinado comando:

  %man <comando>

  - <ENTER>: linha a linha
  - <Barra_de_Espaço>: página seguinte
  - <CTRL B>: tela anterior
  - <CTRL F>: tela posterior
  - <CTRL C>: sai
  - /<string>: procura uma palavra
  - n (next): continua a consulta

- Listar todos os comandos que possuem a *string* que está sendo passada como parâmetro:

  %man -k <string>

NOTAS:

Alias (Apelidos)

- Permite ao usuário criar nomes simbólicos para nomes de comandos.

- Criar alias

  %alias <novo_comando> <velho_comando>

  Exemplos:
  %alias cls clear
  %alias u “who | more”
  %alias m more

- Exibir todos os alias definidos

  %alias

- Remover alias

  %unalias <nome_do_alias>

  Exemplos:
  %unalias cls
  %unalias u

NOTAS:

O nome do alias de ser uma palavra. Não são aceitos espaços em branco.

Devem ser criados no máximo 20 aliases para que a performance não seja afetada.

History

- Acompanha os comandos à medida que são digitados, permitindo voltar a executá-los sem a reentrada do comando.

- Ativar mecanismo de histórico

  %set history = tamanho

  Onde <tamanho> especifica o número máximo de linhas de comando a serem guardadas.

- Salvar o conteúdo do histórico

  %set savehist = tamanho

- Examinar o conteúdo da lista

  %history

NOTAS:

History (Cont.)

- Recuperar um evento pelo número do comando

  %!<número_do_comando>

- Recuperar um evento pelo nome parcial do comando

  %!<nome_parcial>

- Reexecutar o último comando digitado

  %!!

- Corrigir o último comando digitado

  %^<texto_velho>^<texto_novo>^

- Corrigir qualquer comando digitado

  %!<num_comando>:s/<texto_velho>/<texto_novo>

NOTAS:

O ponto de exclamação (!) é empregado como ponteiro de evento. O ponteiro de evento é combinado com números ou caracteres para indicar um evento específico do histórico.

Caso você queira zerar o seu “history” basta digitar:
%unset history

Arquivos

- Tipos de arquivos:

-	arquivo
d	diretório
l	link
b,c,s	especiais

- Tipos de acesso:

r	leitura
w	escrita e deleção
x	executável
-	sem permissão

- Nome dos arquivos

  Exemplos:
  - compra_De_Hoje
  - INFORMAÇÃO.da.bolsa
  - Diversos_arqs.FIM
  - tese.txt
  - semana_passada.textos
  - programa_pascal.pas
  - programa_c.c
  - programa_fortran.for

NOTAS:

- Tudo no Unix é um arquivo (programas, dados, diretórios, disco, teclado, impressora, etc). Os arquivos estão organizados num sistema de diretórios e subdiretórios que se subdividem como os galhos de uma árvore.

- Tipos de arquivos especiais:
  - b ==> blocos, /dev ==> discos
  - c ==> monitor, saída serial, console
  - s ==> socket ==> arquivo de comunicação, memória

- Nome dos arquivos:
  - O Unix considera a diferença entre maiúsculas e minúsculas no nome dos arquivos
  - Normalmente os arquivos são criados em minúsculas
  - O tamanho dos arquivos é livre
  - Normalmente não é colocada extensão no nome de arquivos; os compiladores exigem nome de arquivo fonte com extensão. Fora isso, se o arquivo tiver extensão, ela será utilizada apenas como identificação.
  - Nao é recomendado utilizar caracteres especiais nos nomes dos arquivos (!, #, $, ^, &, *, etc)

Manipulação de Diretórios

- Mostrar o diretório corrente
  %pwd

- Criar diretório
  %mkdir <nome_do_diretório>

- Mudar de diretório

  1. qualquer diretório
     %cd pathname/<nome_do_diretório>
  2. abaixo do corrente
     %cd <nome_do_diretório>
  3. acima do corrente
     %cd . .
  4. home
     %cd ~

- Remover diretório
  %rmdir <nome_do_diretório>

NOTAS:

- Utilizando o comando “rmdir” você só poderá remover um diretório vazio, isto é, antes você terá que remover todos os arquivos abaixo dele.

- Exemplos do comando *cd* **:**
  %cd /usr/lib
  %cd Mail
  %cd trabs/quimica
  %cd ../curso2

Manipulação de Arquivos

- Listar arquivos

  1. visíveis
     %ls
  2. todos (inclusive os invisíveis)
     %ls -a
  3. informação completa (list long)
     %ls -l
  4. mostrando o tipo dos arquivos
     %ls -F
  5. mostrando recursivamente os subdiretórios
     %ls * ou ls -R

- Criar um arquivo

  %cat > <nome_do_arquivo>

  Opções:
  - >: abre o arquivo como entrada
  - <: abre o arquivo como saída
  - >>: acrescenta dados em um arquivo já existente

NOTAS:

- Exemplos do comando ls**:**
  %ls /etc
  %ls *.txt
  %ls -la teste*
  %ls *
  %ls - F**:**
  / ==> indica os diretórios,
  ******* ==> indica os arquivos executáveis,
  @ ==> indica os links

- % cat > <*nome_do_arquivo*>***:***
  - O Unix moverá o cursor para a linha seguinte,
  - Escreva o texto, terminando cada linha com um *<enter>,*
  - Você só poderá fazer correções na linha com a *tecla Backspace*,
  - Para finalizar, *tecle <enter>*,
  - Pressione <*CTRL* D> para sair e salvar ou <CTRL C> para cancelar.

  Ao dar o comando "cat" em um arquivo já existente, o sistema irá lhe enviar uma mensagem de aviso e não abrirá o arquivo. Para que isso realmente aconteça é necessário que esteja setado no seu arquivo .cshrc o seguinte comando: set noclobber.

  Se essa variável não estiver setada, o UNIX simplesmente irá gravar o arquivo sem pedir confirmação, ou seja, você perderá os dados anteriores.

- O comando "cat" também pode ser utilizado para fazer cópia de arquivos:

  %cat < <arquivo_origem> > <arquivo_destino>

Manipulação de Arquivos (Cont.)

- Mostrar o conteúdo do arquivo na tela

  % more <nome_do_arquivo>

  % cat [opção] <nome_do_arquivo>

  Opções:
  - -n: numera todas as linhas do arquivo
  - -b: numera as linhas que não estiverem em branco

- Copiar arquivos

  %cp [opção] <arquivo_origem> <arquivo_destino>

  Opções:
  - -i: solicita confirmação para efetuar a cópia quando o arquivo destino já existir
  - -p: preserva todas as características iniciais referentes a proteção do arquivo

NOTAS:

A diferença entre usar o comando cat e o comando more, é que o cat não oferece a facilidade de vir o conteúdo página a página.

- Exemplos do comando cp:
  %cp arq arqold
  %cp dir1/arq1 arq2
  %cp dir1/arq1 dir2/arq2

Manipulação de Arquivos (Cont.)

- Remover arquivos

  % rm [opção] <nome_do_arquivo>

  Opções:
  - -i: solicita confirmação para remoção;
  - -r: recursivamente, deleta o conteúdo do diretório, seus subdiretórios e ele mesmo.

- Mover ou renomear arquivos

  % mv [opção] <arquivo_origem> <arquivo_destino>

  Opção:
  - -i: quando o arquivo destino já existir, pede confirmação para apagá-lo.

- Concatenar arquivos

  % cat <origem1> <origem2> > <destino>

NOTAS:

- Exemplos do comando rm:
  %rm -i meuarq
  %rm arq2

- Exemplos do comando mv:
  %mv arqold arqnew (renomeando)
  %mv dir1/arq1 . (movendo)
  %mv dir1/arq1 dir2/arq1 (movendo)

OBS: Cuidado ao usar os comandos cp e mv sem a opção “i”, pois se os arquivos destinos existirem você perderá o conteúdo original dos mesmos.

Permissões e Propriedades dos Arquivos

- Tipos de permissões:

r	leitura (copiar, imprimir, visualizar)
w	escrita (mover, apagar, modificar)
x	executabilidade
-	sem permissão

- Níveis de permissão:

u	dono do arquivo
g	grupo de usuários que o dono do arquivo pertence
o	todos usuários que possuem conta na máquina
a	todas as permissões anteriores (u+g+o)

NOTAS:

- Todo arquivo tem um proprietário. O *superusuário* pode alterar a posse individual de um arquivo, se necessário.
- O proprietário tem total controle sobre a restrição ou permissão de acesso ao arquivo a qualquer hora.
- Um usuário que não for o proprietário do arquivo pode ter acesso a ele se pertencer ao grupo de usuários que têm permissão para isso. Porém, esse usuário não pode restringir ou permitir acesso ao arquivo; apenas o proprietário pode fazer isso.
- Permissões para diretórios

  Leitura ***(r)***: permite que você liste o conteúdo dos mesmos.

  Escrita ***(w)***: permite que você crie, altere e apague arquivos.

  Execução ***(x)***: permite que voce efetue buscas no diretório.

Alterando as permissões dos arquivos

- Método simbólico

  %chmod [quem]_operação_acesso <arquivo>

  - quem: u (usuário), g (grupo), o (outros), a (todos)
  - operação: + (atribuir), - (remover)
  - acesso: r, w, x, –

- Formato absoluto

  %chmod valor_octal <arquivo>

  - valor octal:
    4 : leitura
    2 : gravação
    1 : execução

NOTAS:

- Exemplo de alteracao de permissão de arquivo utilizando o Método simbólico:

usuário	grupo	outros
r _ x	_ w x	r _ _

  %chmod u+w,g+r,o+x nome_do_arquivo

  Ficará da seguinte forma:

usuário	grupo	outros
r w x	r w x	r _ x

- O segundo método para alterar a permissão de um arquivo é chamado formato absoluto. Ele é baseado em números octais que incluem os dígitos de 0 a 7, que fornecem um código de um único dígito para especificar as condições dos três bits.

  Para expressar os vários modos de permissão possíveis para um determinado arquivo, simplesmente some os valores octais que correspondam às permissões individuais (isto é, leitura, gravação e execução)

  Exemplo:

usuário	grupo	outros
r w x	r w x	r _ _
(4+2+1)	(4+2+1)	(4+0+0)
7	7	4

  %chmod 656 nome_do_arquivo

  Ficará da seguinte forma:

usuário	grupo	outros
r w _	r _ x	r w _

Outros comandos

- Procurar arquivos por nome

  %find path [opção]

  Onde:
  - -name: especifica que será informado nome de um arquivo
  - -print: exibe na tela a rota dos arquivos que satisfazem os critérios
  - -type: especifica que será informado o tipo do arquivo

- Procurar uma expressão em um arquivo
  %grep [opção] “expressão” <arquivo>

  Opções:
  - -i: ignora a diferença entre maiúscula e minúscula
  - -c: mostra no. de vezes que a expressão foi encontrada
  - -l: lista somente o nome dos arquivos que contém a expressão procurada
  - -n: numera cada linha que contém a expressão procurada

NOTAS:

- Exemplos do comando *find:*

  %find . -name arqstese -print
  %find . -name “arq*.dat” -print

  Obs: Podem ser utilizados metacaracteres: ***[ ], ?, * .***

- O utilitário ***grep*** é útil para localizar as linhas do texto que contenham uma ocorrência específica. Ele lê as linhas do texto do arquivo de entrada padrão e grava no arquivo de saída padrão apenas as linhas que contenham a ocorrência ou quando não for especificado um arquivo de saída o resultado é mostrado na tela.

  Exemplos:

  %grep -i gravida *
  %grep “bom dia” sas/*.c

Outros Comandos (Cont.)

- Contar linhas, palavras e caracteres de um arquivo

  %wc [opção] <arquivo>

  Opções:
  - -l: conta o número de linhas
  - -w: conta o número de palavras
  - -c: conta o número de caracteres

- Classificar arquivos

  %sort [opção] <arquivo>

  Opções:
  - -n: classifica pelo 1° campo numérico
  - -r: inverte a ordem de classificação
  - -f: ignora distinção entre maiúscula e minúscula

- Diferença entre dois arquivos, linha a linha

  % diff <arquivo1> <arquivo2>

NOTAS:

- Exemplos do comando *wc:*

  %wc arq1
  resultado: 110 270 1790 arq1
  linhas palavras bytes

  %wc -l c1.c

- **Exemplos do comando *sort:***

  %sort -r listanome
  %sort -n arq1

Outros Comandos (Cont.)

- Mostrar espaço livre no disco (file system)

  %df

- Mostrar o número de blocos usados por diretórios

  %du [opção] <diretório>

  Opções:
  - -s: relata apenas o número de blocos
  - -a: informa o tamanho de cada arquivo

- Mostrar a quota em disco do usuário

  %quota

NOTAS:

Pipes e Redirecionamento

- Pipeline
  % <comando1> | <comando 2> | <comando3>

O resultado da execução de um comando pode ser utilizado como parâmetro de entrada na execução de outro comando. ***Pipe*** (filtro) é o nome dado para esse tipo de operação. É representado pelo caracter "**|".**

- Redirecionamento ("< ", "> ")
  %comando1 < fonte > destino

NOTAS:

A saída padrão de *comando1* alimenta a entrada padrão do *comando2*, que por sua vez, alimenta a entrada padrão do *comando3.*

Exemplos:
```
% who | grep curso
% ls | grep gravida | wc -l
% cat /etc/passwd | grep doris
```

Normalmente o Unix assume o teclado como entrada de dados padrão e a tela do terminal como saída. Utilizando os caracteres ***"> "e "< "***, pode-se mudar esse padrão.

Exemplos:
```
% cat < gravida | grep -i to | wc -l > estou.num
% cat >> arq1
% mail kikinha@cesar.unicamp.br < meuarq.txt
% cat > texto
```

Processos

- Após cada comando interpretado pelo shell, um processo independente, com um número de identificação (PID) é criado para executá-lo.

- O sistema utiliza o PID para acompanhar o status de cada processo.

- Status dos processos:
  - executando
  - bloqueado
  - parado esperando uma requisição de e/s
  - suspenso

- Processamento em background
  - Terminar a linha de comando com o símbolo “&”
  - Teclar <CTRL><Z>

NOTAS:

Se você estiver executando um programa que irá demorar muito para terminar e você quer começar a trabalhar em outra tarefa, você pode chamar seu programa para ser executado no que é conhecido como *o segundo plano (background).*

Exemplos da utilização de processos em background:

```
% find / -name “*.c” -print&
% du / > du.all&
```

Outros comandos relacionados a processos:

%jobs ==> lista os jobs em background

%fg *[job]* ==> leva o job p/ foreground

%bg *[job]* ==> leva o job para background

Processos (Cont.)

- Informações sobre o status dos processos

  %ps [opção]

  Opções:
  - -a: exibe os processos associados a um terminal
  - -l: informações completas sobre os processos
  - aux: exibe todos os processos que estão sendo executados na máquina.

- Encerrar um processo

  %kill -9 <pid>

- Executar um processo num determinado momento

  %at hora [mes dia] [dia da semana] [week] <arq>

- Mostrar os jobs que estão na fila para serem executados:

  %at -l

NOTAS:

O comado ***ps*** mostra os processos que estão rodando.

Exemplo:

```
%ps
PID     TT          STAT      TIME        COMMAND
 23     Co           R        0:23            ps
```

PID ==> num. de identificação do processo
TT ==> Terminal
STAT ==> Status corrente (Running, Sleepping, Stopped(T), etc)
TIME ==> Tempo de CPU
COMMAND ==> Nome do comando

Exemplo do comando "**at**":

```
%at 8am <arquivo>
%at 2130 tue <arquivo>
%at 2pm apr 3 <arquivo>
```

Editor de Texto - vi

- Editor de texto padrão do Unix - virtual editor

  % vi [-r] <arquivo>

  Opção:
  - -r: recupera o arquivo se houver queda do sistema

- Três modos de funcionamento:
  - Modo de digitação *(“a“ ou “i”)*
  - Modo comando interno *(<esc>)*
  - Comando na última linha *(<esc>“:”)*

NOTAS:

Alguns sistemas operacionais Unix, como por exemplo o FreeBSD, possuem um help on-line do editor de textos vi.

Para obter ajuda sobre o “modo de comando interno” digite <ESC> : e escreva “visuage”.

Para obter ajuda sobre os “comandos de última linha” digite <ESC> : e escreva “exusage”.

Comandos Internos - vi

- Teclar <ESC> e a letra correspondente:

h	move o cursor para esquerda
l	move o cursor para direita
j	move o cursor para baixo
k	move o cursor para cima
^f	move uma tela para frente
^b	move uma tela para trás
a	insere caracter à direita do cursor
A	insere caracter no final da linha
i	insere caracter à esquerda do cursor
I	insere caracter no início da linha [letra i maiuscula]
O	insere linha acima do cursor
o	insere linha abaixo do cursor
x	apaga caracter(es)
dw	apaga palavra(s)
dd	apaga linha(s)
s	substitui caracter
cw	substitui palavra

/*string* ==> procura string
n ==> procura nova ocorrência

ZZ ==> sai do editor e salva o arquivo

NOTAS:

Comandos da Última Linha - vi

- Teclar <ESC>: e o comando correspondente

set num numera o texto
set nonu retira numeração do texto

5,10 d apaga da linha 5 até a linha 10
1,2 co 4 copia linhas 1 e 2 para depois da linha 4
4,5 m 6 move linhas 4 e 5 para depois da linha 6

g/string_procurada/s//string_substituta/gc
substitui palavras

1 posiciona o cursor na primeira linha do texto
$ posiciona o cursor no final do texto
r insere arquivo na posição do cursor
w salva o arquivo e continua editando
q! sai do editor de textos, sem gravar o arquivo
x sai do editor de textos e salva o arquivo

set ignorecase: não diferencia letras maiúsculas e minúsculas na busca de uma palavra

NOTAS:

Editor de texto - pico

- Acessar o editor:
  %pico <arquivo>

- Comandos:

  <ctrl><g>: help
  <ctrl><o>: salva o arquivo
  <ctrl><r>: inclui um arquivo na posição do cursor
  <ctrl><y>: página anterior
  <ctrl><v>: página posterior
  <ctrl><k>: apaga linhas
  <ctrl><u>: volta as linhas que foram apagadas
  <ctrl><w>: procura uma palavra
  <ctrl><c>: mostra a posição do cursor
  <ctrl><x>: sai

NOTAS:

Ao digitar <CTRL><X> o editor sempre irá perguntar se você quer gravar em um buffer. Sua resposta deverá ser sempre YES senão o arquivo não será gravado e você perderá todas as alterações.

MAIL

- Comando “mail” sem definir usuário: listará as correspondências recebidas.

- Ler mensagem: digite o seu número e <ENTER>.

- Principais comandos:

?	help
h	mostra o cabeçalho das mensagens
m	envia uma mensagem
r	responde a mensagem corrente
d	marca mensagem para ser apagada
s	salva a mensagem corrente em um arquivo
~v	edita a mensagem utilizando o editor vi
~f	adiciona a mensagem corrente ao responder uma mensagem
~r	inclui um arquivo na mensagem corrente
x	sai do utilitário Mail, não envia as mensagens para o arquivo “mbox” e não apagas as mensagens marcadas para deleção
q	sai do utilitário Mail, envia as mensagens para o arquivo “mbox” e apaga as mensagens marcadas para deleção

NOTAS:

- Exemplo do comando mail quando temos mensagens no mbox:

```
%mail
   > 1 sc11@apoio.cmp.unicamp.br    Teste do Curso
     2 sc12@apoio.cmp.unicamp.br     Curso Novo

&more 2
```

- Para enviar uma mensagem:

```
%mail usuário@subdomínio.subdomínio.domínio
Subject:: <assunto_da_mensagem>
```

  A partir desse momento você deverá começar a escrever o texto. Para enviar a mensagem basta digitar <CTRL><D>.

- Para ler uma mensagem nova você deverá sair e entrar no utilitário novamente.

ELM

- Iniciar o programa
  %elm

  Aparecerá um índice com os cabeçalhos das mensagens e um mini-menu.

- Ler uma mensagem

  1. Posicionar a barra de seleção sobre a mensagem e teclar <ENTER>

     ou

  2. Informar o número da mensagem na linha de comando e teclar <ENTER>

- Enviar uma mensagem

  Teclar "**m**" na linha de comando.

NOTAS:

- Status das mensagens:
  N)ew: mensagem nova
  O)ld: mensagem antiga mas ainda não lida
  D)elete: mensagem marcada para deleção

- Para chamar os comandos que aparecem no menu (parte inferior da tela), digite a primeira letra do comando (com letra minúsucla).

- Outros comandos úteis para movimentação no menu:
  +: mostra a próxima página de cabeçalhos
  -: mostra a primeira mensagem da lista
  =: vai para a primeira mensagem da lista
  *: vai para a última mensagem da lista

- Teclando "i" você volta ao menu principal.

- Ao digitar "m" para enviar uma mensagem será solicitado:

  To: <endereço_do_destinatário>
  Subject: <assunto_da_mensagem>
  Copies to: <outros_endereços> ou <ENTER>

  O Elm invocará um editor de texto para que você digite sua mensagem.

  Ao concluir a edição, para sair do editor, digite:
  vi: <ESC>:wq
  pico: <CTRL><x>

  Para enviar a mensagem selecione a opção **s**)end

ELM (Cont.)

- Responder uma mensagem

  Posicione a barra de seleção sobre a mensagem e tecle "**r**" (reply to message)

- Reenviar uma mensagem recebida

  Posicione a barra de seleção sobre a mensagem e tecle "**f** " (forward).

- Guardando mensagens em arquivos

  Para salvar, posicione a barra de seleção sobre a mensagem e tecle "**s**" (save).

- Alterando a opção do editor a ser invocado pelo Elm

  Tecle "**o**" (options), na linha de comando.

  Aparecerá um menu onde você deverá alterar a opção "Editor (primary)". Onde estiver vi, você poderá alterar para pico e vice-versa.

NOTAS:

- Ao digitar "r" para responder uma mensagem será solicitado:

  Copy message (y/n):
  y(es): se você quiser que a mensagem original seja inserida na sua resposta, antes da edição da mesma**.**

  Subject: Re: subject_original
  Você poderá, se quiser, entrar com um novo subject, ou apenas tecle <ENTER> mantendo-o assim**.**

- Ao digitar "f" para reenviar uma mensagem será solicitado:

  Forward Edit outgoing message? (y/n):
  y(es): se você quiser alterar o conteúdo da mensagem antes de enviá-la
  n(o): se você quiser reenviar a mensagem exatamente como a recebeu.

- Ao digitar "s" para salvar uma mensagem será solicitado o nome do folder. Por default, o Elm sugere o username do emitente.

  Se voce digitar: =<nome_folder> você irá salvar essa mensagem em um folder.

  Se voce digitar somente <nome_arquivo>, sem o sinal de igual, você irá salvar essa mensagem em um arquivo.

ELM (Cont.)

- Saindo do Elm

  Tecle “**q**” (quit), na linha de comando.

NOTAS:

- Ao digitar “q” para sair do Elm será solicitado:

  1. **Você realmente quer deletar as mensagens marcadas para tal ? (y/n)**

  2. **Você quer enviar as mensagnes já lidas para o folder “received”? (y/n)**

- **Alguns Conceitos:**

- Workstation:
  Supermicro + Terminal+ Software, microprocessador poderoso, tela de alta resolução gráfica.

  - Tipos:
    Stand-Alone: Não necessita de suporte de outra máquina, pois possui capacidade de disco suficiente para funcionar isoladamente.

    Server: Provê recursos para outras estações, como disco, CPU, sistema operacional e software em geral.

    Diskless: Estação que não possui disco rígido. Depende de uma servidora e por isso só funciona ligada à rede.

    Dataless: Possui um disco com o sistema operacional e área de trabalho (Swap). Necessita de servidora para software e área de armazenamento em disco,. Tem que estar ligada à rede.

    X-Terminal: Utiliza todos os recursos a partir da servidora.

NOTAS:

Alguns Conceitos (Cont.):

- Rede: é um conjunto de máquinas conectadas que se comunicam através de um programa chamado protocolo.
- Protocolo: é um software que permite a intercomunicação entre as máquinas da rede, mesmo quando elas possuem sistemas operacionais diferentes. O protocolo instalado na rede Unicamp é o TCP/IP que fornece os programas de telnet e ftp.
- Telnet:: permite aos usuários conectar-se com outras máquinas ligadas na rede, independentemente do tipo de sistema operacional.
- FTP: utilizado na transferência de arquivos entre as máquinas ligadas na rede.
- Domínio: é o nome dado ao conjunto de máquinas administradas por uma única entidade. Por exemplo, o domínio da Universidade Estaudual de Campinas é "unicamp.br".

NOTAS:

- Exemplo de "ftp" anonimo e nterativo para pegar dados no repositório da Unicamp (por exemplo, esta apostila):
  - Entrar em uma máquina onde você possui conta: cesar, obelix, turing, ou qualquer outro equipamento ligado na rede.
  - No *prompt* digitar:
    C:>ftp ftp.unicamp.br
  - Será solicitado um *username*. Você deverá digitar "anonymous"
    User (obelix.unicamp.br (none)): anonymous
  - Será solicitado uma password. Você deverá digitar seu endereço eletrônico
    Password: alguem@teste.algum.lugar.com
  - Devemos sempre fazer a transferência binária
    ftp>bin
  - Vamos mudar de diretório utilizando do comando "cd"
    ftp> cd /pub/CCUEC/cursos/unix
  - Para pegar o arquivo você deverá utilizar o comando "get"
    ftp> get curso_intr.ps
  - Você notará que o arquivo "curso_intr.ps" estará gravado no diretório onde foi dado o comando "ftp ftp.unicamp.br".
  - Comandos básicos para utilização do *ftp:*
    1. *bin*: transferência binária dos dados
    2. *get*: copiar arquivo
    3. *mget*: copiar mais de um arquivo
    4. *put*: gravar um arquivo
    5. *mput*:: gravar mais de um arquivo
    6. *cd*: mudar de diretório
    7. *ls*: listar o conteúdo do diretório