

PROJETO DE REDES

www.projetoderedes.com.br

Apostila Básica de Unix

Marcelo Palmieri Martins
(Analista de Sistemas – Imp. Eletr).

Introdução ao Unix

Histórico

A primeira versão do sistema operacional Unix foi desenvolvida em 1969 por Ken Thompson do grupo de pesquisas da Bell (braço subsidiário da AT&T). Várias versões do Unix foram desenvolvidas a partir daí, até que em 1978 a Universidade de Berkeley lançou uma versão do Unix para computadores VAX, além de incluir diversas facilidades no S.O. Vários fabricantes passaram então a se interessar pelo produto (que já possuía boa receptividade no meio acadêmico) desenvolvendo suas próprias versões.

Características Básicas

O sistema operacional Unix permite que várias pessoas o utilizem simultaneamente (multiusuário), arbitrando as várias solicitações para distribuir os recursos do computador justa e eficazmente. Diversos programas podem “rodar” simultaneamente (multiprogramação).

O sistema parcela o tempo do computador em uma série de partes e os aloca entre os vários usuários. O objetivo desta técnica, denominada “tempo compartilhado”, é dar a cada usuário a ilusão de uso exclusivo da máquina. Cada tarefa a ser executada pelo computador (programas, editoração, etc...) recebe uma fatia de tempo da CPU da máquina. Portanto, quanto mais tarefas, menor o tempo de CPU que cada uma recebe.

Todos estes conceitos demonstram que o sistema Unix é um sistema operacional complexo e que necessita da figura de um administrador denominado pelo sistema de “super-usuário”. Este tem privilégios que os demais usuários do sistema não possuem.

O sistema operacional Unix controla os recursos do computador, faz sua distribuição entre os vários usuários concorrentes, executa o escalonamento de tarefas (processos), controla os dispositivos periféricos conectados ao sistema, fornece funções de gerenciamento do sistema, e de um modo geral oculta do usuário final a arquitetura interna da máquina. Isso é realizado através de uma arquitetura que usa camadas de software projetadas para diferentes finalidades.

Shell

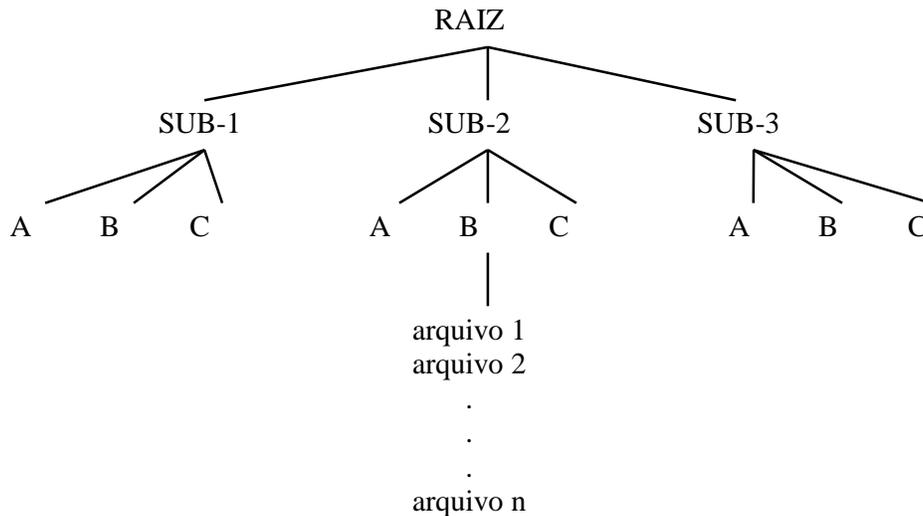
O “shell” é um interpretador de comandos (interpreta os comandos inseridos pelo teclado), ou seja, provê a interface entre o usuário e o sistema operacional. Em muitos sistemas o interpretador de comandos é uma parte da estrutura interna do S.O.. No Unix porém, o “shell” é um programa como outro qualquer, além de ser uma linguagem de programação sofisticada (scripts em shell), que será visto mais adiante. Existem diversos tipos de “shell” como o /bin/sh (“Bourne Shell”), /bin/ksh (“Korn Shell”) e o /bin/csh (“C Shell”).

OBS: No Unix espaços em branco, letras maiúsculas e minúsculas são extremamente importantes.

Ex.: o comando “echo hello” é diferente de “echohello” e de “ECHOhello” (os dois últimos errados).

O Sistema de Arquivos

O sistema de arquivos do Unix é um importante aspecto do sistema operacional. O sistema de arquivos é o local em que são armazenados os arquivos do sistema e dos usuários. Ele é organizado como uma estrutura de arquivo que se parece com uma árvore invertida, estando a raiz no topo, e ramificando-se para baixo.



O diretório raiz pode conter arquivos, elos, ou outros diretórios, chamados subdiretórios. Os subdiretórios algumas vezes são chamados **nós ramos**, enquanto que os arquivos em diretórios algumas vezes são chamados **nós folhas**. *Não há limite imposto à quantidade de níveis, exceto aqueles impostos pelo hardware, embora haja certas implicações de desempenho quando há uma quantidade excessiva de níveis.*

Inodos

Os arquivos do Unix são armazenados em meios auxiliares, como disco rígido ou disquete. Cada arquivo do disco tem um **inodo** único, ou **nodo de informação**. Um inodo contém informações usadas pelos processos, como informações de acesso que incluem permissões a arquivo, tamanho do arquivo em caracteres, a informação de propriedade do arquivo, e a sua localização (*endereço como disco, cilindro, trilha e setor*) da área de dados do arquivo no sistema de arquivos do Unix.

Vários campos compõem um inodo, tais como: de Propriedade do Arquivo, de Tipo de Arquivo, de Permissões do Arquivo, de Datas e Horas do Arquivo, de Ligações, de TOC de Tamanho de Arquivo.

Diretórios

Um diretório no sistema Unix é um tipo especial de arquivo que contém como dados números de inodos e nomes dos arquivos contidos no diretório. Cada registro do diretório tem 16bytes de tamanho; 2 bytes para o número do inodo e 14 bytes para o nome do arquivo. (*No sistema Unix os nomes de arquivos estão limitados a um máximo de 14bytes e podem ter um mínimo de 1 byte*).

Se o número do inodo de um diretório for 0, indicará que a entrada está vazia; não existe arquivo em tal diretório. O núcleo do Unix manipula diretórios de forma muito semelhante ao modo como manipula arquivos, usando inodos. Para garantir modificação e construção correta de diretórios, os processos podem lê-lo, mas somente o núcleo pode gravar em diretórios.

As permissões de acesso de diretórios são similares às permissões de acesso de arquivos, elas têm os mesmos três níveis de acesso - dono, grupo e outros. O atributo de leitura permite que o conteúdo do diretório seja lido; o atributo de gravação permite que o conteúdo do diretório seja modificado (*remover diretórios e arquivos, criar novos subdiretórios e arquivos*); e as permissões de execução simplesmente permitem que os processos procurem um nome de arquivo ou subdiretório no diretório. (Não é possível executar um diretório).

Ligar e Desligar Arquivos

Uma ligação de arquivos é criada através do comando **ln** do Unix ou chamada de sistema **link**. Uma ligação é um modo de criar novos nomes para arquivos do Unix. O resultado é um novo nome que, quando referenciado, afeta o arquivo real ao qual está ligado. É possível ligar diretórios, mas somente o superusuário pode fazer isso. É preciso tomar muito cuidado ao ligar diretórios, pois é possível criar ligações que criem um laço infinito.

Se um arquivo tiver uma ligação para si, então qualquer alteração feita nele afetará igualmente aos usuários de outras ligações. Da mesma forma, se qualquer usuário modificar um nome de arquivo que seja uma ligação, o arquivo original será afetado, e o mesmo acontece com todos os outros usuários do arquivo e suas outras ligações.

É possível remover uma ligação de um arquivo ou diretório usando o comando ou chamada do sistema **unlink**. Os comandos e chamadas de sistema ln/link/unlink são documentados no Unix System V User Reference e Unix System V Programmer's Reference. Lembre-se, é preciso muito cuidado ao ligar e desligar arquivos.

NFS

O Network File System (NFS - Sistema de Arquivos em Rede) foi originalmente implementado pela Sun Microsystems, Inc., e foi endossado pela IBM, Hewlett-Packard, Apollo Computer, Apple Computer, e muitos outros fornecedores de sistemas Unix, exceto a AT&T. Em resumo, a mais ampla implementação dos esquemas de compartilhamento de arquivos em rede é o NFS.

O NFS é um serviço independente de sistema operacional, que permite que todo um sistema de arquivo seja montado em redes. Assim, esses sistemas de arquivo podem ser tratados como parte do sistema de arquivos local. Parte importante do projeto do NFS foi remover todas as dependências do sistemas operacionais, juntamente com todas as dependências de hardware. O NFS oferece também facilidades para recuperação em caso de queda, alto desempenho, e acesso transparente - independentemente de rede ou sistema operacional.

Um exemplo disso é um produto disponível para Pc's baseados em DOS: PC-NFS. O PC-NFS comunica-se com instalações NFS baseadas em Unix. Para o PC, todas as partes do sistema de arquivos Unix aparecem como se fossem discos DOS locais. O PC-NFS ainda não oferece facilidades de servidor, somente serviços de cliente NFS.

Em sistemas baseados em Unix, o NFS é integrado ao núcleo por razões de eficiência, embora a integração não seja obrigatória. De importância para os desenvolvedores de aplicações é por que e como o NFS pode afetar suas aplicações, e como implementar esforços de programação para oferecer compatibilidade com o NFS.

A primeira parte da questão pode ser respondida considerando-se como uma aplicação é projetada. As partes mais sujeitas a serem afetadas são, quase certamente, as funções de I/O; mais provavelmente, I/O baseada em um sistema de arquivos. Se uma aplicação for projetada para trabalhar com certas rotinas de I/O de arquivo é possível que essas rotinas de I/O possam fazer com que a aplicação se comporte de maneira imprópria quando executada usando um sistema de arquivos remoto.

Scripts do Shell

O shell do Unix suporta o conceito de scripts. Os comandos do Unix podem ser colocados em um arquivo, e executados entrando-se com o nome do arquivo no prompt do shell. Os scripts do shell do Unix são análogos aos arquivos de execução em lote do DOS (BAT). Para que um script do shell possa ser executado, não é necessário seguir nenhuma convenção especial de nomes, como acontece no DOS; em vez disso, o arquivo com o script deve receber permissões de execução. Há uma vaga convenção de nomes para os scripts do Unix.

Para indicar que um comando é um script do shell do Unix, acrescenta-se a extensão.sh. Isto não é obrigatório, e o shell do Unix não trata os arquivos que usam esta convenção de nomes de forma diferente do que qualquer outro arquivo comum. É simplesmente uma convenção que torna mais fácil para o usuário reconhecer que um comando é, na realidade, um script do shell e não um executável binário. Observe, ainda, que as convenções de nomes do Unix não impõem padrões específicos para indicar executáveis binários, como acontece no DOS.

Programadores e usuários recém-chegados ao Unix devem compreender que o shell do Unix é tão versátil, e os recursos da programação em shell são tão poderosos, que frequentemente acontece de novas aplicações poderem ser escritas em linguagem shell, e não em C, mais difícil e mais pesada. Isto é particularmente verdadeiro para ferramentas de programação personalizada. Em muitos sistemas Unix provavelmente serão encontradas ferramentas personalizadas já disponíveis. De fato, é uma bênção e uma maldição no ambiente Unix.

O Cron

Talvez um dos recursos mais práticos do Unix seja o *cron* (cronógrafo), que oferece a possibilidade de executar comandos Unix e scripts do shell em tempos predeterminados. Um recurso como este seria extremamente útil no ambiente DOS.

O cron pode ser usado para executar funções simples como verificar periodicamente a correspondência, evitar lembretes para um destinatário qualquer, despachar recados quando do acontecimento de um evento qualquer, e muitas outras funções limitadas apenas pela imaginação do usuário.

O cron é extensivamente usado pela arquitetura do Unix para ajudar no controle automatizado do sistema como um todo, como na automação das cópias de segurança (backup) do sistema. Também o Subsistema de Comunicação UUCP usa o cron extensivamente.

Conexão (Abertura de Sessão)

Para usar os serviços do Unix, o usuário precisa ter uma conta; isto é, uma ID (identificação) criada pelo administrador (a pessoa geralmente encarregada de criar contas de usuários). Uma ID de usuário é necessária porque o Unix é projetado para suportar múltiplos usuários concorrentemente e, por conseguinte, precisa de uma forma para diferenciar um usuário do outro. O DOS foi projetado para ser usado por um único usuário de cada vez e não suporta um ambiente multiusuário, daí a falta de segurança intrínseca.

Uma vez que o usuário tenha recebido sua ID - geralmente as iniciais de seu pré-nome, nome e sobrenome - ele poderá entrar no Unix. Normalmente é necessário ter uma senha, mas em alguns casos um usuário que esteja entrando pela primeira vez pode não tê-la. Isto não é problema pois os usuários podem criar ou modificar suas senhas em qualquer momento. Imaginaremos que a ID de nosso usuário seja tsm e que sua senha seja go4it.

A primeira coisa que precisa ser feita é localizar o terminal a ser usado na sessão. O Unix não requer uma console ativa enquanto o sistema está rodando, como acontece com certos computadores de grande porte que constantemente exibem informações em uma console especial do operador.

Na tela do monitor deve aparecer um prompt como o seguinte:

```
login:
```

O prompt login significa que o Unix está esperando que alguém digite sua ID no teclado. Nossa resposta será tsm, que aparece como segue:

```
login: tsm
```

Os caracteres "tsm" são seguidos pela tecla enter ou return, dependendo da tecla usada para emitir um retorno do carro/avanço de linha (CRLF). Em PCs esta é chamada tecla <ENTRA> (enter). Certos terminais têm uma ou outra tecla, algumas vezes ambas. Após entrar a ID do usuário, o prompt seguinte pede a senha da conta, e aparece como segue:

```
login: tsm
Password:
```

Neste ponto entramos com a senha “go4it”, como segue:

```
login: tsm
Password: go4it
```

Quando a senha está sendo realmente digitada, os caracteres *não* são ecoados na tela por motivos de segurança. No exemplo anterior ela aparece somente para dar mais clareza ao exemplo; em situações reais ela não apareceria. Após entrar com a senha, ela é validada contra um registro em um arquivo especial do sistema. Se houver sucesso na validação então será permitido o acesso ao sistema; caso contrário, será exibida uma mensagem de erro (“Login incorrect”) e reaparecerá o prompt.

Quando o acesso é permitido, o usuário entra em contato com outro componente do Unix: o shell. O shell é uma camada de software entre o usuário e o sistema operacional. Ele foi projetado para interpretar os comandos digitados pelo usuário. É o sistema operacional que realmente providencia a execução dos comandos de usuário; o shell é o mecanismo usado para coletar as instruções.

Se a conexão for bem-sucedida, será exibido o caracter de prompt padrão do shell (o caracter \$), e a tela fica assim:

```
login: tsm
Password: *****
$
```

Em outros sistema podem aparecer outras coisas antes que o prompt apareça, como boas-vindas, mensagem do dia, novidades, ou outras informações. No DOS há um arquivo especial chamado AUTOEXEC.BAT que é automaticamente executado durante a partida do sistema. No Unix, há um arquivo com função similar, mas a execução ocorre a cada vez que o usuário abre uma sessão no sistema. No Unix System V, esse arquivo chama-se *.profile*.

Conexão Remota

A principal forma de se criar conexão com um computador remoto na rede é através do programa *telnet*. Este programa permite ao usuário utilizar os recursos de um computador distante até milhares de quilômetros.

<i>telnet</i> <hostname.subdomínio.domínio> especificado. computador.	→	Conecta o usuário ao computador O usuário precisa estar cadastrado no remoto para a conexão ser estabelecida.
---	---	---

Ex.: Exemplo de uma sessão *telnet* para o computador ADSRJ01:

```
$ telnet ADSRJ01
Trying 192.9.250.1 ...
Connected to ADSRJ01.
Escape character is '^]'.

UNIX System V Release 3.2 (ADSRJ01.ads) (tty0)

login:
```

Para terminar a sessão com o computador remoto basta digitar *exit*. O programa *rlogin* também realiza a conexão remota tal como o programa *telnet*, porém com algumas facilidades, quando configurado o sistema do computador remoto e do local. O arquivo `/etc/hosts.equiv` contém uma lista dos computadores remotos que compartilham as contas dos usuários, fazendo com que não seja necessário dar a password, quando o programa *rlogin* é invocado com o mesmo usuário em um computador ligado a mesma rede de sistema remoto. Existem também listas privadas dos computadores remotos no arquivo `.rhosts` no diretório de entrada de cada usuário.

A linha de comando deste programa é a seguinte:

rlogin <nome ou endereço do computador remoto>

Ex.: Exemplo de uma abertura de sessão da ADSRJ01 para ADSRJ02:

```
$ rlogin ADSRJ02
Last login: Tue Aug 22 10:52:51 on console
SunOS Release 4.1.3 (GEN_SYBASE) #1: Mon Jun 21 10:20:27 EST 1993

$
```

Correio Eletrônico (mail)

O objetivo do correio eletrônico é permitir que usuários se comuniquem uns com os outros, utilizando a rede como um meio de transmissão da mesma forma como se envia uma carta pelo correio. O programa *mail* é utilizado para ler correspondência que nos é enviada, ou para enviar correspondência a outros usuários dentro de uma determinada rede de computadores. Cada usuário do sistema possui a sua “caixa de correio” (mailbox) que é um arquivo pessoal que armazena as mensagens que chegaram e não foram lidas. Existe uma outra “caixa de correio” que fica no diretório pessoal de cada usuário e que armazena as mensagens já lidas (geralmente o arquivo `mbox`).

Examinando o conteúdo da sua “caixa de correio” (mailbox)

Quando um usuário entra no sistema, caso alguém lhe tenha enviado uma mensagem, aparecerá na tela a seguinte mensagem (ou similar):

“you have mail”

Para listar as mensagens recebidas digite:

```
$ mail
```

Você entrou no programa *mail* (cujo “prompt” inclusive é diferente) e uma saída similar a esta será apresentada:

```
SCO System V Mail (version 3.2) Type ? for help.
"/usr/spool/mail/marcelo": 1 message 1 new
>N 1 produc      Thu Aug 17 15:48 13/369 teste
&
```

Cada linha da tela anterior tem os seguintes campos:

```
status: >      → mensagem corrente
         N      → mensagem nova
         R      → mensagem lida
         U      → mensagem não lida
```

número: indica a numeração da mensagem.

remetente: identifica o endereço da pessoa que enviou a mensagem.

data: identifica a data de recebimento da mensagem.

tamanho: identifica o número de linhas e caracteres da mensagem (incluindo o cabeçalho).

assunto: identifica o assunto da mensagem (opcional).

Para ler qualquer uma das mensagens basta digitar o número correspondente no “prompt” do *mail*.

Para apagar, basta digitar o comando “d”:

```
Message 1:
From produc Thu Aug 17 15:48:26 1995
Received: by ADSRJ01.ads.COM (5.65/25-eef)
        id AA14313; Thu, 17 Aug 95 15:48:26 +0300
Return-Path: <produc>
Message-Id: <9508171248.AA14313@ADSRJ01.ads.COM>
From: produc@ADSRJ01.ads (Master for Producao)
X-Mailer: SCO System V Mail (version 3.2)
To: marcelo
Subject: teste
Date: Thu, 17 Aug 95 15:48:25 BRA
Status: R
```

Estou testando o mail

```
& d
&
```

Para terminar o programa *mail* digite *Ctrl-d* ou *q* (volta ao “prompt” do sistema). As mensagens que tiverem sido lidas serão guardadas na “caixa de correio” do seu diretório. Para verificar o conteúdo desta, digite:

```
$ mail -f      → lista as mensagens já lidas com seus respectivos números correspondentes e entra
no           no programa mail.
```

Dentro do programa *mail*, além de poder apagar mensagens (i.é.,d <número da mensagem>), pode-se copiar estas para arquivos comuns no diretório pessoal, mantendo-as ou não na “caixa de correio”.

\$ *copy* <número mail> <nome arquivo> → copia a mensagem indicada pelo número, para o arquivo indicado (criado na hora) e a mantém no mailbox.

\$ *save* <número mail> <nome arquivo> → remove a mensagem para o arquivo indicado (remove da mailbox).

ENVIADO UMA MENSAGEM:

Ao enviar uma mensagem você, além de conhecer o e-mail da outra pessoa deve informar o assunto da mesma de forma resumida (2 ou 3 palavras). Embora isto seja opcional é quase um padrão no serviço de correio eletrônico. Para enviar uma mensagem para um usuário qualquer digite:

\$ *mail* **username@computador.subdomínio.domínio** (Sistema Remoto)

\$ *mail* **username** (Sistema Local)

Após isto, o que você deve informar é o assunto, teclar ENTER e começar a escrever a mensagem. Concluindo a mensagem, é só digitar **Ctrl-d** em uma linha em branco para enviá-la. Suponha que você está enviando uma mensagem para o usuário marcelo:

```
$ mail marcelo
```

```
Subject: Teste
```

```
Marcelo,
```

```
Isto e somente um teste do mail!
```

```
Obrigado,
```

```
<Ctrl+d>
```

```
EOT
```

```
$
```

<Ctrl+d> → envia a mensagem e sai do programa *mail*.

Para interromper uma mensagem durante sua elaboração digite duas vezes **Ctrl+c**.

O programa *mail* não permite, por exemplo retroceder linhas para correções. Portanto, se você pretende escrever mensagem mais longas, onde não pode haver erros de editoração, é possível escrevê-las utilizando o editor de textos (vi), salvar seu conteúdo num arquivo e depois enviar este arquivo através do *mail*.

Ex.: Você digitou o arquivo “carta” no editor de textos vi. Para enviá-lo para outra pessoa, por exemplo marcelo, digite:

```
$ mail marcelo < carta
```

Este comando redireciona a entrada do *mail* para o arquivo carta através do caracter especial “<”.

Conteúdo do Diretório

O sistema de arquivos do DOS armazena informações como o nome do arquivo, tamanho do arquivo em bytes, data de criação e última alteração, hora de criação e última alteração, bem como outros atributos (leitura, gravação arquivo/diretório) de cada entrada no diretório. O sistema de arquivos do Unix armazena os atributos da entrada, quantidade de ligações, identificação do dono, identificação do grupo, tamanho do arquivo em bytes, data e hora da criação e última alteração, e nome do arquivo/diretório.

É evidente que a quantidade de informações sobre o arquivo que o Unix mantém é consideravelmente maior do que no DOS. Isso acontece porque o sistema de arquivos do Unix foi pensado para permitir acesso multiusuário concorrente. A saída do comando *ls -l* exibe informações detalhadas sobre as entradas no diretório. O primeiro campo dessa saída é usado para determinar o tipo de entrada e suas permissões de acesso. A seguir temos um exemplo desse campo:

```
drwxrwxrwx  
0123456789
```

Tipo e Permissões de Acesso

A segunda linha do exemplo anterior não é parte normal da saída, mas foi colocada como ilustração para indicar a posição dos caracteres na linha acima. Se o primeiro caracter, na posição 0, for um “d”, isso significa que a entrada é um diretório. Se for um “-”, significa que é um arquivo comum. Se esse caracter for um “b”, “c” ou “p”, significa que se trata de um arquivo especial usado pelo Unix para executar operações de I/O.

O grupo seguinte de caracteres, nas posições de 1 a 3, indica as permissões do dono do arquivo; os caracteres *rwx* são usados para indicar as permissões de *leitura (read)*, *gravação (write)* e *execução (execution)*. Se aparecer um “-” em qualquer dessas posições, isso indicará ausência da respectiva permissão. A permissão de acesso *x* indica que o arquivo é executável podendo ser um comando binário executável ou um script do shell. Arquivos que não contêm essa permissão não podem ser executados, independentemente de serem realmente programas binários executáveis ou scripts do shell. O DOS, por outro lado, usa o próprio nome do arquivo como mecanismo para determinar se o arquivo é executável e como será tratado.

As três posições, de 4 a 6, indicam as permissões de acesso no nível do grupo. Os últimos três caracteres, nas posições de 7 a 9, indicam as permissões de acesso de todos os outros usuários do sistema. Os três níveis de acesso aqui indicados são chamados, respectivamente, nível de permissão de *usuário*, de *grupo* e de *outros*. Os tipos de permissões de acesso para os três grupos são os mesmos. O sistema de arquivos do DOS não tem nada correspondente a essa funcionalidade, pois já foi projetado como sistema monousuário, as permissões de acesso não receberam consideração.

Segurança do Sistema de Arquivos

A implementação da segurança de sistema de arquivos começa com a compreensão de como é dada permissão aos usuários para acessar os arquivos.

O comando *chmod* é usado para mudar os valores de *modo de acesso* associados a um arquivo. O valor de modo de acesso de um arquivo define, entre outras coisas, as permissões de acesso dadas ao dono do arquivo, ao grupo e ao resto do mundo (chamado outros). As permissões de acesso são de *leitura, gravação e execução*, indicadas por *r*, *w* e *x*.

Os valores do modo de acesso aplicam-se a diretórios e subdiretórios, bem como a arquivos, portanto a termo “arquivo” mencionado nesse tópico vale para os dois. (Note que os valores do modo de acesso não se aplicam a superusuários!).

Basicamente, há três tipos de usuários que podem acessar um arquivo: o dono do arquivo, os usuários pertencentes ao grupo ao qual o arquivo está associado, e todos os outros usuários. Dentro de cada uma destas categorias de usuário, é possível definir como o arquivo poderá ser acessado: para leitura, gravação ou execução. Quando o usuário lista as informações detalhadas sobre um arquivo, usando a opção **-l** do comando **ls**, a informação retornada inclui as permissões de modo de acesso, representadas da seguinte forma:

rwxrwxrwx

Essa tríade de caracteres rwx vai da esquerda para a direita. O primeiro grupo refere-se ao dono, o seguinte refere-se ao acesso do grupo, e o último refere-se aos outros (o resto do mundo). Podem ser fornecidas outras informações como uma indicação de que a entrada é um diretório ou um arquivo. O exemplo a seguir ilustra como os grupos são associados aos modos de acesso:

<i>dono</i>	<i>grupo</i>	<i>outros</i>
<i>rw</i>	<i>rx</i>	<i>rx</i>

Cada modo de acesso dentro de cada grupo tem um valor numérico associado a ele, como indica a tabela abaixo:

MODO	VALOR	DESCRIÇÃO
-	0	Nenhuma permissão
x	1	Permissão de execução
w	2	Permissão de gravação
wx	3	Permissão de gravação e execução
r	4	Permissão de leitura
rx	5	Permissão de leitura e execução
rw	6	Permissão de leitura e gravação
rwx	7	Permissão de leitura/gravação/execução

Esses valores numéricos podem ser usados para ajustar as permissões de acesso que cada grupo tem. O comando **chmod** usa valores numéricos para cada uma das tríades; por exemplo, se as permissões de dono, grupo e outros fossem ler, gravar e executar, o valor numérico a especificar com o comando **chmod** seria 777. Se, em lugar de permitir acesso pleno ao grupo e outros, fosse necessário dar acesso pleno apenas ao dono, sem qualquer outra permissão de acesso, o valor numérico seria 700.

Como pode ser visto, o número octal de três dígitos tem uma relação posicional correspondente à tríade rwx. O primeiro número (à esquerda) corresponde ao campo do dono, o segundo número (no meio) corresponde ao campo do grupo, e o terceiro número (à direita) corresponde ao campo reservado para outros. Isso pode ser confuso para alguns usuários do Unix, por isso há um modo diferente de realizar a mesma coisa, mas que é um pouco mais fácil de entender.

O método alternativo de especificar o valor do modo no comando **chmod** é usando caracteres alfabéticos. Os caracteres alfa r, w e x podem ser usados em combinação com os caracteres u, g, o e a. O caracter u refere-se ao usuário que possui o arquivo, o caracter g refere-se ao grupo, o caracter o refere-se aos outros (todos os outros usuários), e a (all=todos) aplica as mudanças aos três grupos (u,g e o).

Os caracteres + (mais), -(menos) e = (igual) são também usados no processo de atribuição. O caracter + significa *acrescentar* o(s) valor(es) de permissão(ões) à direita dele, o caracter - significa *remover* o(s) valor(es) de permissão(ões) à direita dele, o caracter = significa *retirar* (nenhuma permissão) todos os valores de permissão para os usuários especificados à esquerda do caracter. A seguir temos alguns exemplos:

chmod a+rw nome_do_arquivo

```
chmod uo-x nome_do_arquivo
chmod a= nome_do_arquivo
```

O primeiro exemplo atribui permissões de leitura e gravação a todos os usuários que acessam nome_do_arquivo. Na segunda linha são retiradas as permissões de execução para o dono e outros, mas não para o grupo. A última linha remove todos os atributos (nenhuma permissão é concedida) do dono, grupo e outros. A fim de manipular esse arquivo, o dono deve antes usar o **chmod** para dar-lhe as permissões apropriadas.

Quando um arquivo é criado, ele recebe certos valores de propriedade, como a uid do dono (normalmente o criador do arquivo), o valor de propriedade do grupo (geralmente o grupo a que o dono pertence quando o arquivo é criado), e outros valores de permissão de acesso derivados dos valores associados ao valor *umask* do dono, no momento da criação.

Pode haver muitas situações em que se torna necessário mudar um ou outro desses valores; o comando **chgrp** é usado para atribuir uma nova propriedade do grupo. O grupo deve existir, caso contrário será retornada uma mensagem de erro. O valor de propriedade de grupo não se aplica ao superusuário. A fim de mudar as propriedades de grupo, o usuário deve ter autorização; deve ser membro do grupo ao qual o arquivo dá permissões de modificação.

O novo grupo ao qual a propriedade está sendo associada pode ser qualquer grupo, desde que já existia (geralmente mantido pelo administrador do sistema). Uma vez reatribuída a propriedade de grupo, o acesso no nível dogrupo fica restrito aos membros do novo grupo. As permissões de acesso de grupo anteriormente ajustadas continuam valendo.

O comando **chown** é usado para trocar a propriedade de um arquivo, passando-a de um usuário para outro. A fim de mudar o valor de propriedade de um arquivo, o usuário deve ser autorizado, deve ser dono do arquivo ou ter privilégios de superusuário. Para usuários autorizados não superusuário, uma vez mudada a propriedade do arquivo, eles passam a sujeitar-se às permissões de acesso definidas para o arquivo.

Se um usuário acidentalmente mudar a propriedade para outro usuário, então deve pedir ao novo dono para devolvê-la, o que também poderá ser feito por um superusuário (geralmente é feito pelo administrador do sistema).

Serviço de Transferência de Arquivos:

Este serviço é comumente conhecido como **ftp** (“file transfer protocol”). Havendo permissão, é possível copiar ou enviar arquivos entre equipamentos localizados em diferentes continentes por exemplo.

```
ftp <hostname.subdomínio.domínio> → conecta o usuário ao computador
especificado                          (hostname) se ele tiver autorização para
tal.
```

Ex.: \$ ftp ADSRJ01 (neste exemplo só utilizado o hostname)

O programa **ftp** sempre pede ao usuário sua identificação (username e senha) para permitir o acesso à máquina requisitada. Uma vez dado este acesso é como se você estivesse de fato trabalhando naquele computador (portanto é preciso que você esteja cadastrado nele).

Os subcomandos mais utilizados dentro do programa *ftp* (cujo prompt é *ftp>*) são:

?	lista os subcomandos disponíveis.
(ascii)	transfere arquivos do tipo ASCII.
binary	transfere arquivos binários.
dir <diretório remoto>	lista o conteúdo do diretório remoto especificado.
get <arquivo remoto> <arquivo local>	copia o arquivo remoto específico para o arquivo local..
mget <arquivos remotos>	copia a série de arquivos remotos especificados para o diretório corrente no computador local.
mput <arquivos remotos>	copia os arquivos listados para o computador remoto (é preciso ter permissão de escrita no diretório remoto que vai abrigar este arquivos).
open <hostname>	estabelece uma conexão ftp com o computador especificado.
put <arquivo local> <arquivo remoto>	copia o arquivo local para o arquivo remoto listado.
pwd	apresenta o diretório corrente no computador remoto.
quit ou bye	fecha a conexão entre os computadores e sai do programa ftp (volta ao "prompt do sistema).
Crtl+c	interrompe a transferência de qualquer arquivo.

Ex.: Exemplo de uma sessão de ftp marcelo para ADSRJ01:

```
$ ftp ADSRJ01
Connected to ADSRJ01.
220 ADSRJ01.ads FTP server (Version 5.60 #1) ready.
Name (ADSRJ01:marcelo):
331 Password required for marcelo.
Password:
230 User marcelo logged in.
ftp> pwd
257 "/w1/USERS/marcelo" is current directory.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls (0 bytes).
total 1876
-rw-r--r--  1 marcelo  desenv    822 Jul 06 15:13 .cshrc
-r--r--r--  1 marcelo  desenv    53 Sep 09 1994 .cueprofile
-rw-r--r--  1 marcelo  desenv   371 Jan 26 1995 .exerc
-r-----  1 marcelo  auth       0 Aug 01 16:01 .lastlogin
-rw-r--r--  1 marcelo  desenv   385 Sep 09 1994 .login
-rw-r--r--  1 marcelo  desenv  2166 Jul 19 12:18 .profile
drwxr-xr-x  2 marcelo  other     64 Jul 07 12:36 DOS
drwxr-xr-x  2 marcelo  other    128 Apr 04 23:36 amherj
drwxr-xr-x  2 marcelo  other    128 Apr 04 23:37 backup
drwxr-xr-x  2 marcelo  other     48 Apr 04 23:37 bin
-rw-----  1 marcelo  desenv   190 Feb 03 1995 cctj1
-rw-----  1 marcelo  desenv  4751 Feb 24 09:50 cshrc
-rwxr-xr-x  1 marcelo  desenv  6614 Jan 30 1995 envmer.doc
drwxr-xr-x  2 marcelo  other     80 Apr 04 23:37 fax
226 Transfer complete.
1553 bytes received in 0.23 seconds (6.5 Kbytes/s)
```

```
ftp> cd fax
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> get sqlv10.sun
200 PORT command successful.
150 Opening ASCII mode data connection for sqlv10.sun (12079 bytes).
226 Transfer complete.
local: sqlv10.sun remote: sqlv10.sun
12079 bytes received in 0.06 seconds (2e+02 Kbytes/s)
ftp> quit
221 Goodbye.
$
```

Quando se realiza uma transferência de arquivos de um lugar para outro, está-se aumentando o tráfego de informações na rede de comunicação (principalmente se o arquivo for grande). Portanto, é recomendável que sessões de ftp, de modo geral, sejam feitas fora do horário comercial, em horários de baixo tráfego na rede.

Editor de Texto VI

O *vi* é o editor de textos mais utilizado em máquinas Unix (outros são os editores *ex* e *ed*). Este editor opera em, basicamente, dois modos, onde em cada um deles existem comandos específicos.

modo texto → É o modo utilizado para inserir um texto, o que é possível após a inserção de subcomandos específicos. Para voltar ao modo comando basta pressionar a tecla **ESC**.

modo comando → É o modo em que o editor se encontra logo após ser invocado. Para cancelar um subcomando específico deste modo basta pressionar a tecla **ESC**.

O editor pode ser invocado através dos comandos:

vi <arquivo>

ou

vi

(neste último caso o nome do arquivo a ser editado deverá ser fornecido no momento em que for salvo).

MODO TEXTO

Subcomandos de inserção de texto:

<i>i</i>	insere texto antes do cursor
<i>r</i>	insere texto no início da linha onde se encontra o cursor
<i>a</i>	insere texto depois do cursor
<i>A</i>	insere texto no fim da linha onde se encontra o cursor
<i>o</i>	adiciona linha abaixo da linha corrente
<i>O</i>	adiciona linha acima da linha corrente
<i>Ctrl + h</i>	apaga último carácter
<i>Ctrl + w</i>	apaga última palavra minúscula
<i>Esc</i>	passa para o modo comando

MODO COMANDO:

Subcomandos para Movimentação pelo Texto:

<i>Ctrl+f</i>	passa para a tela seguinte.
<i>Ctrl+b</i>	passa para a tela anterior.
<i>H</i>	move o cursor para a primeira linha da tela.
<i>M</i>	move o cursor para o meio da tela.
<i>L</i>	move o cursor para a última linha da tela.
<i>k</i>	move o cursor para linha acima.
<i>j</i>	move cursor para linha abaixo.
<i>h</i>	move cursor para carácter a esquerda.
<i>l</i>	move cursor para carácter a direita.
<i>w</i>	move cursor para início da próxima palavra (Ignora pontuação).
<i>W</i>	move cursor para início da próxima palavra (Não ignora pontuação).
<i>b</i>	move cursor para início da palavra anterior (Ignora pontuação).
<i>B</i>	move cursor para início da palavra anterior (Não ignora pontuação).

0 (zero)	move cursor para início da linha corrente.
^	move cursor para o primeiro caracter não branco da linha.
\$	move cursor para o fim da linha corrente.
nG	move para a linha n.
G	move para a última linha do arquivo.

Subcomandos para Localização de Texto:

<i>/palavra</i>	move para a próxima ocorrência da palavra procurada (para repetir a busca usar n).
<i>?palavra</i>	move para a ocorrência anterior da palavra(para repetir a busca usar n).
Ctrl+g	mostra o nome do arquivo corrente, o número da linha corrente e o número total de linhas.

Subcomandos para Alteração de Texto:

rx	substitui o caracter sob o cursor pelo especificado x (é opcional indicar o caracter).
Rtexto	substitui o texto corrente pelo texto indicado (opcional indicar o texto adicionado).
cw	substitui a palavra corrente. Pode-se indicar a nova palavra inserida ou não.
cc	substitui a linha corrente. Pode-se inserir o novo conteúdo da linha automaticamente.
C	substitui restante da linha corrente. Pode-se inserir o conteúdo restante da logo após o comando.
u	desfaz a última modificação.
U	desfaz todas as modificações feitas na linha (se o cursor não mudou de linha desde a última mudança).
J	une a linha corrente a próxima.
s:/velho/novo	substitui a primeira ocorrência de “velho” por “novo”.

Subcomandos para Salvar o Texto:

:wq	salvar as mudanças feitas no arquivo e sai do editor.
:w <nome-arq>	salva o arquivo corrente com o nome especificado. Continua edição. Não é necessário fornecer o nome do arquivo se este já tiver um.
:w! <nome-arq>	salva (de modo forçado) o arquivo corrente no arquivo especificado (cujo conteúdo original é destruído!).
:q	sai do editor. Se mudanças não foram salvas é apresentada mensagem de advertência.
:q!	sai do editor sem salvar as mudanças realizadas.

Praticando

```

$ pwd
/w2
$ cd /w1/WORKADS
$ pwd
/w1/WORKADS
$ ls -l
total 18
drwxrwxrwx 2 produc produc 32 Apr 05 00:33 arq
drwxrwxrwx 3 produc produc 48 Apr 04 23:47 fax
drwxrwxrwx 9 produc produc 144 Apr 05 00:33 implanta
drwxrwxrwx 2 produc produc 128 Jun 22 16:27 keep
drwxrwxrwx 5 produc produc 80 Apr 05 00:33 libera
drwxrwxrwx 5 produc produc 80 Apr 05 00:33 log
drwxrwxrwx 6 produc produc 112 Jun 20 18:20 receive
drwxrwxrwx 6 produc produc 176 Aug 01 08:53 send
drwxrwxrwx 11 produc produc 208 Jun 06 16:02 ship
$ cd teste
$ cp /w2/SCCD/arq/*.dat .

$ ls -l
total 22
-rwxrwxrwx 1 produc produc 708 Aug 03 12:38 dtt326.dat
-rwxrwxr-x 1 produc produc 1746 Aug 03 12:38 hln0414.dat
-rwxrwxr-x 1 produc produc 1830 Aug 03 12:38 hln0807.dat
-rwxrwxr-x 1 produc produc 1072 Aug 03 12:38 hln0858.dat
-rw-r----- 1 produc produc 1542 Aug 03 12:38 mv031031.dat
-rwxrwxrwx 1 produc produc 1620 Aug 03 12:38 rngd1406.dat
$ rm *.dat
$ ls -l
total 0
$ cd ..
$ rmdir teste
$ who
produc ttyp0 Aug 03 12:36
$ finger
Login Name TTY Idle When Office
produc Master for Producao p0 Thu 12:36
$ lp manual
request id is emilia_unix-3109 (1 file)
$ lpstat
emilia_unix-3109 marcelo 43253 Aug 4 12:20 on
emilia_unix
$ cancel emilia_unix-3109
request "emilia_unix-3109" cancelled
$ chmod 755 teste
$ chgrp desenv teste
$ chown produc teste
$ compress teste
$ uncompress teste.Z
$ tar cvf fontes *.cbl.Z
a lf0303v1.cbl.Z 41 tape blocks
a lf0309v1.cbl.Z 49 tape blocks
a lf0310v1.cbl.Z 78 tape blocks
$ ps -ef
  UID  PID  PPID  C  STIME  TTY  TIME  COMMAND
  root   0   0  0  Aug 4  ?    0:00  sched
  root   1   0  0  Aug 4  ?    2:25  /etc/init
  root   2   0  0  Aug 4  ?    0:00  vhand

```

- Mostra o diretório corrente
- Caminha para o diretório /w1/WORKADS
- Verifica o diretório corrente
- Lista o conteúdo do diretório corrente
- Caminha para o diretório teste
- copia todos os arquivos *.dat do diretório /w2/SCCD/arq para diretório corrente
- Lista o conteúdo do diretório corrente
- Remove todos os arquivos *.dat do diretório corrente
- Lista o conteúdo do diretório corrente
- Retorna ao diretório pai
- Remove o diretório teste
- Identifica os usuários conectados
- Identifica com mais detalhes os usuários conectados
- Envia o arquivo manual para a fila de impressão (o processo da impressão é de nº 3109)
- Mostra o estado dos processos na fila de impressão
- Cancela o processo emilia_unix-3109 da fila de impressão
- Muda a permissão do arquivo teste para 775
- Muda o grupo do arquivo teste para desenv
- Muda o dono do arquivo teste para produc
- Comprime o arquivo teste que passa a ser teste.Z
- Descomprime o arquivo teste.Z
- Compacta todos os arquivos com extensão "cbl.Z" no arquivo fontes
- Fornece todos os processos ativos na máquina

```

root 3 0 0 Aug 4 ? 0:02 bdf flush
marconde 5937 1 0 11:37:13 01 0:02 -sh
root 6039 1 0 12:03:57 02 0:01 /etc/getty tty02 sc_m
root 143 1 0 Aug 4 ? 0:00 /etc/slattach +c tty1a
192.9.12 0.3 192.9.120.1 38400
root 58 1 0 Aug 4 ? 0:00 /etc/logger /dev/error
/usr/adm/messages /usr/adm/hwconfig
root 111 1 0 Aug 4 ? 0:07 /etc/cron
root 135 1 0 Aug 4 ? 0:00 cpd
root 117 1 0 Aug 4 ? 0:04 /usr/lib/lpsched
root 137 1 0 Aug 4 ? 0:00 slink
root 144 1 0 Aug 4 ? 0:00 /etc/slattach tty1b
192.9.50.6 192.9.50.8 2400
root 145 1 0 Aug 4 ? 0:00 /etc/slattach tty1c
192.9.50.7 192.9.50.19 19200
root 146 1 0 Aug 4 ? 0:00 strerr
root 881 1 0 Aug 4 03 0:01 /etc/getty tty03 sc_m
root 574 1 0 Aug 4 ? 0:00 pcnfsd
mmdf 555 1 0 Aug 4 ? 0:05 /usr/mmdf/bin/deliver -b
root 573 1 0 Aug 4 ? 0:01 portmap
root 581 1 0 Aug 4 ? 0:02 mountd
root 611 1 3 Aug 4 ? 61:42 /usr/sybase/bin/dataserver -
d/usr/sybase/master.dat -e/usr/sybase/install/error
root 618 1 0 Aug 4 07 0:00 /etc/getty tty07 sc_m
root 619 1 0 Aug 4 08 0:00 /etc/getty tty08 sc_m
root 620 1 0 Aug 4 09 0:00 /etc/getty tty09 sc_m
root 621 1 0 Aug 4 10 0:01 /etc/getty tty10 sc_m
root 622 1 0 Aug 4 11 0:01 /etc/getty tty11 sc_m
root 6004 157 0 12:02:19 ? 0:04 telnetd
root 5778 1 0 11:04:32 12 0:00 /etc/getty tty12 sc_m
lazarini 6005 6004 0 12:02:20 p0 0:02 -sh
root 6058 157 15 12:09:28 ? 0:01 telnetd
root 627 1 0 Aug 4 ? 0:00 /tcb/files/no_luid/sdd
marcelo 6059 6058 0 12:09:29 p1 0:02 -sh
marcelo 6080 6060 42 12:10:08 p1 0:00 ps -ef
$ kill -9 6059

$ ps -ef | grep marcelo
marcelo 6059 6058 0 12:09:29 p1 0:02 -sh
marcelo 6079 6059 42 12:10:08 p1 0:00 grep marcelo
marcelo 6080 6060 42 12:10:08 p1 0:00 ps -ef
$ ls lf0303v1.cbl | cut -f2 -d.
cbl
$ exit

```

- Mata o processo 6059, eliminando a sessão do usuário marcelo
- Mostra as linhas que contêm a palavra marcelo através do resultado do comando ps -ef
- Recorta o segundo campo delimitado pelo ponto
- Desconecta-se da sessão

Comandos do Unix

Sintaxe de Comando	Descrição
at <i>time</i> [<i>data</i>] [<i>incremento</i>]	Enfileira comandos para a execução em um momento posterior. Ex.: at 5 pm Friday next week < backup.sh
ar <i>opção nome arquivos</i>	Agrupa arquivos em um único arquivo de acervamento (semelhante ao comando tar). Ex.: ar -rv fontes *.c (compacta os arquivos *.c no arquivo fontes)
banner <i>caracter(es)</i>	Converte caracteres normais em garrafais para a saída padrão. Ex.: banner BOM DIA
cancel <i>ID number</i>	Cancela a impressão de processo <i>ID number</i> . Este é dado pelo comando lpstat -p <i>file</i> . Ex.: cancel emilia-195
cat <i>file1 file2 ...</i>	Concatena o <i>file1, file2 ...</i> e mostra na tela. Ex.: cat .profile
cd <i>pathname</i>	Muda o diretório de trabalho para <i>pathname</i> (caminho do diretório desejado). Ex.: cd /w1/WORKADS
cd	Muda para o diretório <i>home</i> (inicial).
chgrp <i>grupo file1 ... file n</i>	Muda os arquivos <i>file1</i> até <i>file n</i> para um determinado <i>grupo</i> . Ex.: chgrp produc .profile
chmod <i>número, + -, rwx file ou diretório</i>	Muda a permissão de um arquivo ou diretório.(4-leitura, 2-gravação e 1-execução) Ex.: chmod 640 .profile
chown <i>proprietário file1 ... file n</i>	Muda o <i>proprietário</i> dos arquivos <i>file1 ... file n</i> . Ex.: chown produc .login
clear	Limpa a tela.
compress <i>file1</i>	Comprime o <i>file1</i> , gerando um <i>file1.Z</i> . Ex.: compress lf0303v1.cbl
cp <i>file1 file2</i>	Copia <i>file1</i> para <i>file2</i> (copia por cima do <i>file2</i>). Ex.: cp /w1/WORKADS/teste /w2/USERS
cut [-d " <i>delimitador</i> "] [-f(<i>numero do campo</i>)]	Extrai informações selecionadas de um arquivo de entrada. Ex.: cat /etc/passwd cut -d":" -f1,5
date	Mostra a data e hora corrente.
df	Informa o número de blocos livres em disco.
dircmp <i>diretório1 diretório2</i>	Compara diretórios entre si.
du	Fornecer um resumo da utilização do disco.
echo <i>message</i>	Mostra <i>message</i> na tela. Ex.: echo Estou aqui
env <i>variável=valor</i>	Classifica variável de ambiente.
exit	Sai do sistema.
find <i>pathname -name file1 -print</i>	Mostra a localização do <i>file1</i> procurando a partir do <i>pathname</i> . Ex.: find /w1/USERS -name "marcelo" -print
finger [<i>opções</i>]	Obtem informações detalhadas dos usuários conectados.
ftp <i>nome_da_máquina</i>	Abre uma sessão para transferência de arquivos. (Ver lista de comandos do ftp). Ex.: ftp -i ADSRJ02
grep <i>opções expressão file1</i>	Mostra as linhas em que ocorre a <i>expressão</i> dentro do <i>file1</i> . Ex.: grep -i 'echo' .profile
head - <i>número</i> <i>arquivo</i>	Exibe as linhas iniciais(<i>número</i>) do arquivo na saída padrão. Ex. head -20 .login
id	Exibe informações sobre as ID's de usuário e grupo do usuário.
kill - <i>nu_sinal</i> <i>PID</i>	Encerra processos ativos no sistema Unix, onde <i>nu_sinal</i> é o número do sinal e <i>PID</i> é o número do processo a ser eliminado. Ex.: kill -9 135
logname	Fornecer o nome da conta do usuário.
lp <i>file{1...n}</i>	Imprime do <i>file1</i> até o <i>file n</i> . Ex.: lp .login
ls	Lista o conteúdo do diretório corrente.
ls -a	Lista todo o conteúdo do diretório corrente.
ls -l	Lista o conteúdo do diretório corrente, indicando tamanho e permissões.
mail	Envia correspondência aos usuários ou as lê. No caso de envio, finaliza-se com <Control+D>.
man <i>comando</i>	Mostra na tela a pagina do manual referente ao <i>comando</i> .
mesg [<i>opção</i>]	Permite ou impede a recepção de mensagens. Ex.: mesg n
mkdir <i>diretório novo</i>	Cria um <i>diretório novo</i> . Ex.: mkdir ADS
more <i>file1</i>	Mostra o conteúdo de <i>file1</i> .(q - quit). Ex.: more .profile
mv <i>file1 file2</i>	Move <i>file1</i> para <i>file2</i> . Ex.: mv .login .login.old
nohup <i>comando argumento</i>	Executa um comando imune à desconexão e aos sinais de quit.
pack <i>arquivo(s)</i>	Comprime arquivos, similar ao comando compress. Ex.: pack *.c

passwd	Cria ou muda uma password para o usuário.
pcat <i>arquivo.z</i>	Exibe o conteúdo do arquivo comprimido com o comando pack na saída padrão.
pr <i>arquivo</i>	Quebra o arquivo em páginas, onde a saída é formatada com um cabeçalho em cada página, com o nome do arquivo a data e a hora, e o número de página.
ps [<i>-opção</i>]	Reporta a situação de processos ativos no sistemas
pwd	Mostra o diretório corrente.
rm <i>file1</i>	Remove o <i>file1</i> .
rmdir <i>diretório</i>	Remove o <i>diretório</i> (este diretório deve estar sem arquivos).
rlogin <i>nome_da_máquina</i>	Abre uma conexão com <i>nome_da_máquina</i> . Ex.: rlogin ADSSP01
sleep <i>segundos</i>	Suspende a execução por um determinado de tempo.
sort <i>file1</i>	Ordena o <i>file1</i> e mostra na tela. OBS: não grava a alteração do <i>file1</i> .
su [-] <i>usuário</i>	Abre uma nova sessão com a conta do <i>usuário</i> mencionado. A opção “-” faz herdar o ambiente do <i>usuário</i> desejado. Ex.: su - root
tail [<i>opção</i>] <i>file1</i>	Mostra na tela as 10 últimas linhas do <i>file1</i> . Este valor pode ser ajustado com as opções +número(a partir do início) -número(a partir do fim). Ex.: tail +15 .login
tar {cvf, xvf e tvf} <i>nome file{1...n}</i>	Compacta <i>file1</i> até <i>file n</i> dentro de um arquivo chamado <i>nome</i> . Ex.: tar cvf arquivos.tar .login .profile (cria arquivos.tar compactado) tar xvf /dev/rst0 * (extrai todos os arq. contidos em fita streamer) tar tvf /dev/rst0 * (verifica o conteúdo em fita streamer)
tee	Este comando encaminha a entrada para o terminal e para o(s) arquivo(s) nomeado(s). Ex.: ls tee listagem sort
telnet <i>nome_da_máquina</i>	Abre uma conexão com <i>nome_da_máquina</i> . Ex.: telnet ADSRJ01
tty	Exibe o terminal que está sendo usado.
uncompress <i>arquivo(s)</i>	Descomprime arquivos (*.Z). Ex.: uncompress *.Z
unpack <i>arquivo(s)</i>	Descomprime arquivos (*.z). Ex.: unpack *.z
wall	Envia mensagem para todos os usuários logados. Para finalizar CTRL+D.
wc <i>file1</i>	Realiza a contagem das palavras do arquivo <i>file1</i> . Ex.: wc .profile
who	Mostra os usuários correntes no sistema.
write <i>usuário mensagem</i>	Envia <i>mensagem</i> para o <i>usuário</i> . Ex.: write produc fim de implantação
zcat <i>arquivo.Z</i>	Exibe na tela o conteúdo do arquivo comprimido pelo comando compress.

Caracteres Especiais

<i>file</i> > <i>file1</i>	Redireciona saída do comando especificado.
<i>file</i> < <i>file1</i>	Redireciona a entrada para o comando especificado
<i>file</i> >> <i>file1</i>	Redireciona saída do comando para o fim do <i>file1</i> .
<i>comando1</i> <i>comando2</i>	Utiliza a saída do <i>comando1</i> como entrada do <i>comando2</i> .
<i>\$variável</i>	Fornece o conteúdo da <i>variável</i> de ambiente.
<i>\metacaracter</i>	Anula a função do metacaracter.
*	Equivale a qualquer caracter.
<i>comando</i> &	Faz com que um processo seja executado em background.

Bibliografia:

- Unix para Usuários e Programadores do MS-DOS
Steven Mikes - Editora Campus
- Unix System V Manual do Usuário e Guia do Programador
Mitchell Waite, Don Martin e Stephen Prata
- Apostila de Administração do SunOS I
Medidata
- Guia de Referência do Sistema Operacional Unix
PUC
- Shells User's Guide
Hewlett Packard