Centro Universitário
Fundação Santo André

Sistema Operacional

Unix

Parte I

Centro Universitário Fundação Santo André

Prof. Ms. Lázaro Aparecido da Silva Pinto
Fevereiro de 2016

Sistema Operacional UNIX

- O que é UNIX?
- Funções executadas pelo UNIX

O que é UNIX?

- É um sistema operacional aberto
- Multiusuário
- Tempo compartilhado (time-sharing)
- Interativo

Sistema Operacional Unix

As seguintes características descrevem o sistema operacional UNIX:

- O sistema operacional Unix é um programa que suporta e utiliza o equipamento de seu computador. Isto é, o equipamento poderia ser parcial ou totalmente substituída por peças de outros fabricantes, sem que isso implicasse nenhum tipo de reaprendizado de sua parte.
- Unix é tempo compartilhado (time-sharing), sistema de multi-usuário. Ou seja, ele pode atender a vários usuários simultaneamente dividindo seu tempo entre eles. Grosso modo, tempo compartilhado funciona da seguinte maneira: a qualquer momento que você parar para pensar, UNIX fará um "bico" e trabalhará para outro usuário; normalmente o UNIX é tão rápido que você nem notará sua ausência.
- O UNIX é um sistema operacional interativo. Cada usuário pode comunicar diretamente com o computador através de um terminal, simplesmente entrando com comandos e obtendo respostas.

Funções realizadas pelo Unix

- Aloca recursos do sistema
- Agenda tarefas
- Processa solicitações do usuário
- Realiza funções administrativas
- Realiza funções de manutenção

- Como qualquer sistema operacional, o UNIX aloca recursos do sistema, agenda tarefas, processa pedidos do usuário e realiza funções administrativas e de manutenção do sistema.

Características do Unix

- Disponibilidade de vários utilitários
- Shell
- Arquitetura

Características do Unix

As seguintes características e capacidades estão incluídas no sistema operacional UNIX:

- Centenas de utilitários para realizar uma vasta variedade de funções tais como criar, editar e manipular arquivos e textos, processar comandos e jobs, comunicar com outros usuários, manter o sistema e escrever programas.
- A *shell*, que atua como uma interface para o usuário, é uma ferramenta flexível que habilita os usuários a realizar seu trabalho enquanto fornece uma estrutura para proteger e separar cada usuário individualmente e seus ambientes do sistema operacional.
- Sistema de arquivos e de entrada e saída são flexíveis e simples, onde cada arquivo, comando, programa e periférico de E/S é

percebida pelo sistema operacional como um arquivo que contém um conjunto de caracteres, permitindo cada um realizar sua função unicamente e também conectá-los para realizar tarefas específicas. UNIX é portável de forma que pode ser implementado facilmente em várias plataformas diferentes. Como o UNIX é escrito na linguagem de programação C, pode-se facilmente substituir partes do código por outros que melhor se adaptam as necessidades do seu sistema ou aplicações.

Estrutura do Unix

Podem ser agrupadas em duas categorias:

- Nível do usuário
- Nível do sistema

Nível do Usuário

- Shell
- Sistema de arquivos e diretórios
- Comandos, utilitários e compiladores
- Shell scripts
- Aplicações

Nível do Usuário

As seguintes características fazem parte da categoria *nível do usuário* :

- A shell
- Sistema de arquivos e diretórios
- Comandos, utilitários e compiladores
- Shell scripts ou programas em shell
- Aplicações

Nível do Sistema

- Gerenciamento de recursos
- Gerenciamento de processos
- Gerenciamento de arquivos
- Gerenciamento de E/S

Nível do Sistema

As seguintes características fazem parte da categoria *nível do sistema*:

- Gerenciamento de recursos
- Gerenciamento de processos
- Gerenciamento de arquivos
- Gerenciamento de Entrada e Saída

Níveis do Unix

- Kernel
- Shell

Kernel

- Controla uso da CPU
- Gerenciamento de memória
- Gerenciamento de E/S
- Gerenciamento de processos
- Gerenciamento de arquivos.

O kernel

O *kernel* é a parte do sistema operacional residente na memória. A maioria das características são externas ao kernel para deixá-lo pequeno e eficiente. As funções do sistema operacional são acessadas por utilitários e aplicações através de chamadas ao sistema (system calls). O kernel realiza as seguintes funções:

- Agenda da CPU (scheduling)
- Gerenciamento de memória
- Gerenciamento de Entrada e Saída
- Gerenciamento de processos
- Gerenciamento de arquivos

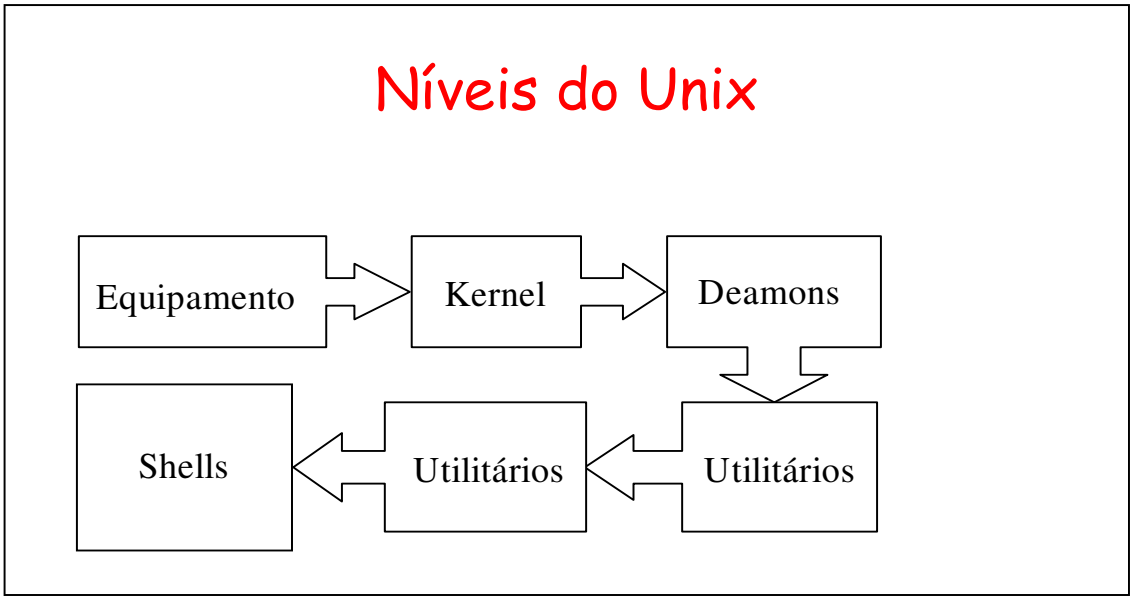
Shell

- Interface para o usuário
- Interpretador de comandos
- Linguagem de programação

A Shell

A *shell* é um processo que executa continuamente do login ao logout e interage com o kernel para realizar os pedidos dos usuários. A shell está instalada como um programa no diretório /bin. A shell realiza as seguintes funções:

- Interface do usuário
- Interpretador de comandos
- Linguagem de programação



Níveis do Unix

- Equipamento
- Kernel (CPU Sched, Mem mgt, E/S mgt, Proc mgt, file mgt)
- Deamons (lpd, init, ...)
- Utilitários (cp, rm, man, vi, date, ...)
- Aplicações (cc, libs)
- Shells (sh, ksh, csh)

Evolução do Unix

Versões do Unix

1975	1978	1982	1983
V6	V7	System III	System V
Primeira Comunicação Versão em C interprocessos	Portabilidade	vi Programadores	Suporte a redes Sistema de Arquivos mais rápidos Melhoramento no vi

História do Unix

As seguintes subseções discutem a história do UNIX.

AT&T System V

O sistema operacional Unix foi desenvolvido em 1969 pelo Laboratório Bell da AT&T em Murray Hill, New Jersey, e foi originalmente desenhado para uso comercial. Ele foi escrito por Ken Thompson em assembler para um computador DEC PDP-7 para fornecer um ambiente de programação mais flexível e poderoso para a equipe de desenvolvimento de programas (e para permiti-los jogar o "Space Travel"). Devido ao fato dele ter sido originalmente desenhado para somente um usuário por vez, ele foi chamado UNIX por Brian Kernighan, outro "pai" do UNIX, um trocadilho com o sistema operacional multi-usuário (batch) já existente, chamado MULTICS.

Thompson também escreveu o editor de texto ed e a linguagem de programação B. Dennis Ritchie expandiu a linguagem B adicionando controle de fluxo do Pascal renomeou-o para C. Thompson e Ritchie então rescreveram UNIX em C (10.000 linhas dos quais 80% é em C) Desde então, muitas pessoas tem contribuído para o seu desenvolvimento levando a mais recente release, o System V.

História do UNIX

UNIX evoluiu em um sistema de tempo compartilhado que permite mais que uma pessoa possa usar o computador simultaneamente (multi-usuário) e permite você comunicar diretamente com o computador através de um terminal (interativamente).

A equipe que desenvolveu o UNIX tirou vantagens de outros sistemas operacionais multi-usuários quando o desenvolveram. Pesquisadores desenvolveram UNIX de forma que pessoas que estão trabalhando juntos em um projeto podem compartilhar dados e programas, enquanto ao mesmo tempo, protegem seus dados de outros usuários.

Linha de vida do UNIX

Ano	Versão	Descrição
1969	Primeira versão	Escrita em B
1971	Primeira Edição	Escrita em assembler
1972	Segunda Edição	Escrita em assembler
1973	Quarta Edição	Shell e kernel escritos em C
1974	Quinta Edição	Disponível para Universidades
1975	Sexta Edição	Licenciada pela AT&T
1977	1BSD	Primeira Edição Berkeley
1979	Sétima Edição	Portado para VAX
1981	BSD 4.1	Release principal de Berkeley
1983	System III	Melhoram. versão de Berkeley
1985	U N I C O S	Sistemas Cray
1987	BSD 4.3	Última release de Berkeley
1988	System V R4	Última release principal da AT&T

A influência de Berkeley

Quando o UNIX se tornou disponível em 1974 e 1975, o laboratório Bell da AT&T ofereceu-o para instituições educacionais a um custo mínimo. Eles o usaram em seus computadores científicos onde se tornou bastante popular entre os estudantes de graduação, que então trouxeram suas experiências com o UNIX para o mundo comercial.

O Departamento de Ciências Computacional da Universidade de Berkeley na Califórnia, que foi o ponto principal por muitos anos para os desenvolvedores do UNIX, fez adições e modificações importantes ao UNIX. Por que eles fizeram tantas mudanças, nasceu uma versão separada do UNIX, que é chamada Berkeley Software Distribution (BSD). System V da AT&T adotou muitas das características desenvolvidas em Berkeley.

A versão Berkeley do UNIX caracteriza uma interface de usuário, ou shell, que é inteiramente diferente da shell da AT&T. É chamada de C shell porque suas capacidades de programação são similares a linguagem de programação C.

Sistema Operacional

Unix

Parte II

O usuário

Objetivos:

- Log on em um sistema UNIX
- Mudar sua senha (passwd)
- Reconhecer os vários prompts do sistema
- Ver quem está no sistema (who) (who i am)
- Sair do sistema (exit) (logout) (ctrl-d)

Conectando ao Sistema

Antes de conectar-se ao sistema:

Digital UNIX (nome_da_máquina) (ttyp3)
login:

Conectando-se ao Sistema

Para começar, você deve obter um login ID, uma senha, e possivelmente, informações de como se conectar ao sistema UNIX com o seu instrutor de classe ou administrador de sistema. Quando você tiver acesso a um terminal, pressione a tecla RETURN (ou ENTER) até você ver o prompt do login em sua tela.

UNIX aceita tanto caracteres em letras maiúsculas quanto minúsculas. Mas, como a maioria dos comandos em UNIX são reconhecidos somente em minúsculas, é recomendado que você entre com os comandos em letra minúsculas na sua seção de login. Por exemplo, se seu login ID for *mit*, e você digitar Mit, o sistema não aceitará a entrada.

Na primeira vez que se conectar ao sistema UNIX, a tela irá mostrar alguma coisa como o exemplo da transparência, parando no prompt do login.

Conectando ao Sistema

Enquanto a conexão está sendo efetuada:

Digital UNIX (nome_da_máquina) (ttyp3)
login: **mit**
Password: **xxxxxxx**
Last login: Wed May 27 08:16:10 from : 0.0
You have mail.
$

Para conectar-se ao sistema, entre com o seu login ID (por exemplo, mit) em resposta ao prompt **login:**. O sistema então irá te pedir uma senha (Password). Por razões de segurança, sua senha não será mostrada na tela. Se você entrar no login ID ou senha incorreta, o sistema irá dar uma mensagem de

Login : incorret

Depois que o sistema verificou seu login ID e sua senha e você tiver conseguido se conectar com sucesso, você poderá ver uma mensagem do sistema para os usuários, chamada "message of the day" (motd), uma notificação de mail para ler (You have mail.), se há alguma novidade no sistema, que você ainda não leu (news: news_items).

O sistema então produzirá um *caracter de prompt do sistema*, que freqüentemente é o nome do computador ao qual você está conectado, seguindo por um sinal de dólar ($) ou percent (%). Algumas vezes, somente o sinal de dólar ou percent é usando para o prompt do sistema. O prompt do sistema indica que você está conectado ao seu sistema UNIX e que a shell está pronta para receber um comando. Os prompts do sistema serão discutidos em mais detalhes nas subseções subsequentes.

Prompt do Sistema

Prompt	Descrição
$	Você está usando o **Korn** ou **Bourne Shell**
%	Você está usando o **C Shell**
#	Cuidado! Você está conectado com o usuário **root**, o todo poderoso.

Prompt do Sistema

Depois que você conectou com sucesso ao sistema UNIX, no canto inferior esquerdo de sua tela irá aparecer um caracter de prompt do sistema. O *prompt do sistema* é um símbolo que diz a você que o sistema está pronto para receber comandos. Cada vez que você entrar um comando e pressionar a tecla RETURN (ENTER), o sistema executará o comando e retornará um novo prompt, que indica que ele está pronto para receber outro comando.

A primeira vez que você se conectar, e até você explicitamente mudá-lo, o caracter de prompt será sempre o mesmo, pois ele foi definido pelo seu administrador de sistema.
Os exemplos nas páginas seguintes mostram os prompts mais comuns no UNIX. Cada prompt tem um significado específico que indica o tipo de login ID que você está usando e o tipo de shell que é a sua interface de usuário para o login ID.

Neste curso, iremos usar sempre o caracter ($) como prompt default do sistema, que representa o Korn shell, a menos que outro seja indicado.

Mudando a Senha

```
$ passwd
Changing password for mit
Old password: xxxxxxxx
New password: xxxxxxxx
Re-enter new password: xxxxxxxx
$
```

Mudando sua Senha:
Para mudar a senha que você se conectou ao sistema, use o comando *passwd*. No exemplo da transparência, Mit está conectada com seu login ID, mit. Para mudar sua senha, ela deve saber a senha atual (ou velha). Se Mit não entrar com a senha atual corretamente, o comando responderá com *sorry* ou algo parecido e sairá. A senha, por si mesma, quando digitada, não é mostrada na tela (representada no exemplo por uma séria de x's). Depois de digitar a senha corretamente, será pedido a mit que entre a nova senha duas vezes para garantir que não foi cometido nenhum erro de digitação na primeira vez.

Somente ao login ID *root* é permitido mudar a senha de outros login ID. Todos os usuários podem mudar somente sua própria senha.

Por razões de segurança, as senha do UNIX devem satisfazer alguns dos seguintes requisitos, dependendo de seu sistema em particular:

- Cada senha deve ter entre 6 e 8 caracteres.
- As senhas devem conter pelo menos 2 caracteres alfabéticos e 1 caracter especial.
- Senhas devem ser diferentes de seu login ID e qualquer combinação reserva ou circular do login ID.
- Senhas novas devem diferir das antigas por pelo menos 3 caracteres.

Verificando quem está usando o sistema

```
$ who
root        console        Jul 28 11:36
rei2        tty19          Jul 28 11:54
mimi        tty20          Jul 28 12:03
mit         tty84          Jul 28 12:15

$ who am I
mit         tty84          Jul 28 12:15
```

Determinando quem está conectado ao sistema

Se você saber quem está conectado ao sistema, entre com o comando *who* no prompt do sistema, como no exemplo. Uma lista de usuários correntemente conectados no seu sistema será mostrada, com informações adicionais sobre casa usuário.

A primeira coluna lista os login lds de todos os usuários correntemente conectados no sistema. A coluna do meio mostra o terminal específico que cada usuário está usando. A última coluna mostra a data e hora que cada usuário se conectou.

Outras opções estão disponíveis para o comando *who*, incluindo *am I* que lista informações sobre seu próprio login ID. Um exemplo de *who am I*, mostrada na transparência. O comando who, assim como suas opções será discutido em mais detalhes no tópico "Comandos e Utilitários" .

Saindo do Sistema

$ **logout** (csh)
$ **exit**
$ **Ctrl-D**

Digital UNIX (nome_da_máquina) (ttyp3)
login:

Saindo do Sistema:

Para sair do sistema UNIX, entre com o comando *exit* ou o comando *logout* no prompt do sistema, seguido de RETURN (ENTER). Alternativamente, você pode pressionar CONTROL-D (pressione as teclas CONTROL e D simultaneamente) para sair dos sistema.

Independentemente da forma como você sai do sistema, você deve ver o prompt login: novamente após você ter saído. A seção de login nas transparências conclui com uma seqüência de logout.

Se você está conectado remotamente ao seu sistema UNIX de um outro sistema UNIX, CONTROL-D irá fechar a sua seção remota, mas não a seção do sistema local. Neste ponto, você poderá continuar entrando com comandos no sistema local. Para sair do sistema local, pressione CONTROL-D novamente.

Exercícios

1. Obtenha uma conta e conecte-se ao sistema. Observe qual o tipo de prompt que o seu login ID tem.
2. Verifique quem está conectado ao sistema.
3. Liste na tela as informações sobre seu próprio login ID
4. Mude a sua senha para alguma coisa segura, mas que seja fácil de lembrar.
5. Feche a conexão usando ***exit***. Conecte-se novamente e tente o comando ***logout***. Conecte-se novamente e tente ***Control-d***. *Observe a diferença entre os três métodos.*

Comandos Básicos do UNIX

Comandos são:

- **Utilitários**
- **Aplicações**

Comandos e Utilitários

Definições de Comandos e Utilitários

As subseções seguintes definem comandos e utilitários e introduz o formato do comando e introduz alguns comandos básicos.

Os termos *comandos* e *utilitários* são usados alternadamente. Quando esses termos são usados neste curso, eles se referem a definições descritas nas subseções seguintes.

Comandos

Comandos UNIX são caracterizados como:

- Utilitários – Comandos que são instalados com o sistema operacional. Os utilitários UNIX formam um conjunto poderoso de ferramentas que realizam muitas tarefas sem ter que escrever programas especiais. Esses utilitários são usados em diferentes combinações, permitindo a você customiza-los para um usá-lo em um propósito específico.

- Aplicações – Qualquer outro comando tais como programas de terceiros, programas locais e seus próprios programas e shell scripts.

Utilitários

O sistema operacional UNIX inclui mais de 200 utilitários, que podem ser divididos nos seguintes tipos:

- Operações gerais
- Administração do sistema
- Processamento de texto
- Desenvolvimento de aplicações
- Manutenção de arquivos
- Correio Eletrônico

O curso usa o termo comando para referir-se a maioria dos utilitários nos grupos listados acima. Quando um comando tem seu próprio subconjunto de comandos, tais como os comandos *mail* e *ftp*, será usado o termo utilitário.

Man Page

- **Seções Padrão nas man page do UNIX**

NAME	SYNOPSIS
IMPLEMENTATION	STANDARDS
DESCRIPTION	NOTES
CAUTIONS	WARNINGS
ENVIRONMENT VARIABLES	RETURN VALUES
EXIT STATUS	MESSAGES
BUGS	EXAMPLES
FILES	SEE ALSO

- **Cabeçalhos podem diferir em versões diferentes do UNIX**

Comandos Básicos do UNIX

As seguintes subseções introduzem alguns comandos UNIX de uso geral e que são úteis aos usuários que estão começando. Cada comando é descrito, incluindo o formato do comando e um exemplo de como usá-lo está na transparência. Somente as opções e usos mais comuns serão discutidos neste curso. Você é fortemente encorajado a procurar outras opções e informações sobre cada comando do manual online de cada comando. O comando man, que é usado para ver esta informação online sobre cada comando está descrito abaixo.

O comando man

Os manuais de referência do UNIX são armazenados online. Informações detalhadas sobre cada comando, incluindo suas opções, podem ser acessadas usando o comando man (manual). Cada sistema UNIX pode ser diferente da mesma entrada em outro sistema UNIX.

(Se um comando executa diferentemente daquela especificada neste manual, veja a man page no seu sistema para informações mais específicas). Para usar o comando man, digite man seguido do comando que você está investigando. A informação sobre o comando será mostrada em sua tela. Por exemplo, para ler

sobre o comando who, você pode entrar com man who. Para obter informações sobre o próprio comando man¸ você deve digitar man man.
O formato do comando man é:

man[opções] command_name

Ele pode demorar um pouco para formatar man page depois de digitar o comando man. O comando man geralmente para automaticamente no fim de cada tela. Para ver a próxima tela, pressione a tecla de espaço. Para ver a próxima linha, pressione a tecla RETURN. Isso pode ser revertido dependendo da instalação do UNIX no seu sistema. Para sair do display de uma man page, você deve digitar a caracter q (quit).

Entrando com comandos UNIX

- **O Shell interpreta e executa o comando especificado**

Formato do comando:

Nome_do_comando *argumentos*

Entrando com Comandos UNIX

A shell executa um comando quando você entra com o nome do comando no prompt do sistema e pressiona e tecla RETURN. A shell interpreta e executa o comando especificado.

Formato do Comando

O formato de um comando UNIX é da forma:

Nome_do_comando *argumentos*

O Nome_do_comando pode ser qualquer comando válido do UNIX. Argumentos do comando são usados quando informações adicionais são necessárias para executar o comando ou para fornecer opções para o tipo de saída que o comando gera. Um argumento é uma seqüência de caracteres separados por espaços, tabulações ou nova linha. Você pode entrar um ou mais argumentos na linha de comando.

Um argumento está freqüentemente na forma de um hífen seguido por um caracter particular (por exemplo, -c, onde c é o caracter). Este caracter é chamado um *opção* ou *flag*. As opções são usadas para executar variantes de um comando. Alguns comandos não tem opções, enquanto outros tem vários. Algumas opções tem opções. Várias opções podem ser usadas por um comando agrupando opções de letras simples na forma –*abi* (onde ambas opções *a* e *b* são usadas).

Os exemplos das transparências ilustram o uso de argumentos e opções com os comandos *echo*, *ls*, e *cal*. Os comandos echo e cal serão discutidos mais tarde nessa seção, e o comando ls será discutido em uma seção posterior. O comando echo mostra o texto seguido do comando echo na tela; ele imprime os argumentos do comando echo na tela. O comando cal imprime um calendário e o comando ls imprime uma lista de arquivos e diretórios. Os exemplos nas transparências mostram como os argumentos dos comandos variam de comando para comando.

O comando date

$ date
Fri Dec 20 14:19:45 CST 1991

$date `+%m/%d/%y`
09/10/98

O comando date

O comando *date* imprime a data e hora corrente de acordo com o relógio do sistema. É também usado pelos administradores de sistema (com poderes de super-usuário) de acerta a hora correta do sistema. Muitas opções existem para o comando date, na maioria controlando o formato de saída do comando. Para maiores informações, veja a man page de *date*.

O formato para o comando date é:

date [+formato]

O comando echo

$ echo um dois três
um dois três

$ echo Favor entrar com seu nome e pressione a tecla return
Favor entrar com seu nome e pressione a tecla return

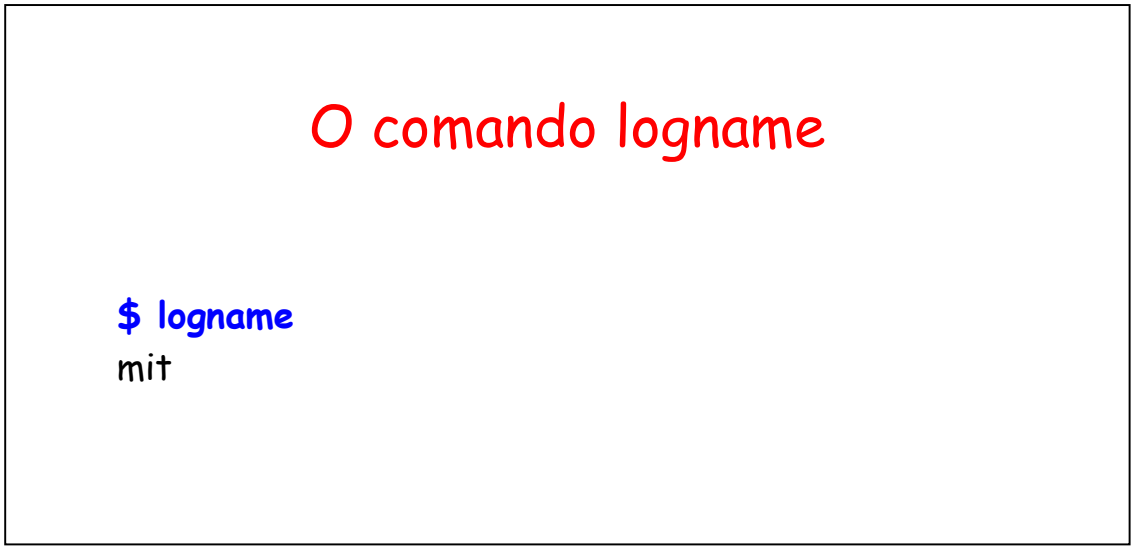
$ echo O programa esta sendo executado
O programa esta sendo executado

O comando echo

O comando *echo* ecoa na tela seus argumentos. O formato do comando echo é:

echo [opções] mensagem

Os exemplos do comando echo na próxima página mostram como você pode usar este comando em um programa (ou shell script) para fornecer informações da pessoa rodando o programa ou para documentar o progresso de um programa.



O comando logname

O comando *logname* imprime o login ID do usuário que executou o comando. O formato do comando logname é:

logname

Não há opções para o comando logname.

O comando tty

```
$ tty
/dev/tty79
```

O comando *tty* imprime o caminho completo do arquivo especial que representa seu terminal. Cada vez que você se conectar a um terminal, um arquivo especial é associado com aquele terminal. Este arquivo irá variar com cada seção de login.

O formato para o comando *tty* é:

tty [opção]

O comando news

$ news -n

Ethernet Printers asa

$ news Printer

As impressoras que estão conectadas ao sistema são as seguintes:

:

:

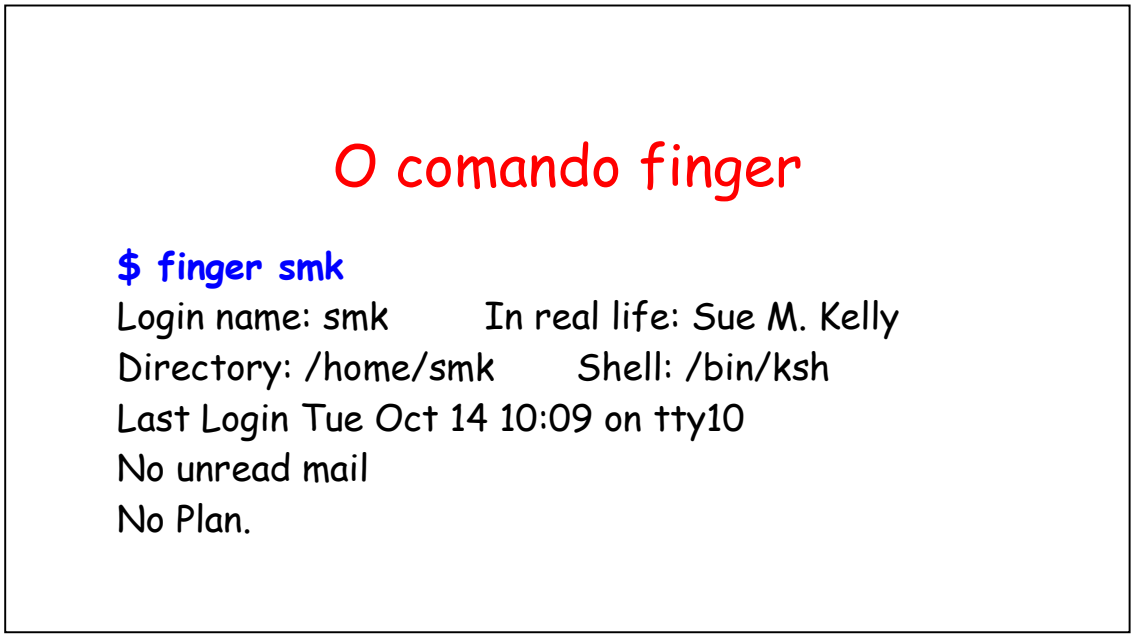
O comando news

O comando *news* mostra informações sobre eventos correntes no sistema. Esses eventos são descritos em arquivos que são geralmente criados pelo administrador de sistemas no diretório */usr/news*. Entrando com o comando *news* sem argumentos todos os itens de news será impresso, sendo que a mais recente será mostrada primeiro. Para obter uma listagem dos eventos correntes que são cobertas em news, use a opção -n . Você pode selecionar a notícia, entre o nome como argumento para comando news.

O formato para o comando news é:

news [opções] news_itens

Se o seu sistema é baseado no UNIX BSD, o comando news pode não estar disponível, como no AT&T.



O comando finger

O comando *finger* fornece informações sobre um usuário ou usuários no seu sistema. O formato para o comando *finger* é como segue:

finger [opções] [name...]

O argumento name para o comando finger é ou o login ID, ou primeiro nome, ou último nome. O arquivo */etc/passwd* é lido para procurar o usuário específico e as informações sobre o usuário é mostrada.

O comando *finger* sem argumentos mostra o login ID, nome completo, home directory, shell, nome do terminal, status do mail, idle time e a última vez que o usuário se conectou. Essas informações são para cada usuário que está conectado ao seu sistema.

O comando *finger* trabalha com os arquivos .plan e .project. Se algum desses arquivos existir em seu home directory, o conteúdo do arquivo será impresso junto com as outras informações que o comando *finger* fornece.

O comando cal

O comando *cal* mostra um calendário para a data que você especificou. O comando cal sem argumentos mostra um calendário para o mês corrente. Se o mês é omitido, o comando cal mostra um calendário para o ano inteiro que você especificou.

O formato para o comando cal é:

cal [mês] [ano]

O comando cal

$ cal 12 1988

December 1998

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Exercícios

1. Imprima na sua tela a data e hora corrente.
2. Imprima em sua tela o seguinte:
   Sessão de Comandos Básicos do UNIX
3. Leia sobre o comando ***date*** no manual online (man pages). Liste a data corrente usando o seguinte formato:
   Mm/dd/yy
4. Leia sobre outro comandos desta sessão no manual online.
5. Liste o seu login ID na sua tela.
6. Veja em que terminal você está conectado (arquivo terminal). Dê logout e login novamente. Veja se você continua usando o mesmo terminal de antes.
7. Use o comando ***finger*** para obter informações sobre o login ID root.
8. Use o comando ***finger*** para obter informações sobre o seu login ID.
9. Imprima um calendário para a data de hoje. Imprima um calendário para o mês e ano que você nasceu e observe que dia da semana você nasceu.
10. Imprima a data e hora m sua tela usando o seguinte formato:
    DATA: mm/dd/yy
    TIME: hh:mm:ss
11. Imprima na tela as seguintes informações sobre cada usuário conectado ao sistema:
    - Login ID
    - Nome
    - Diretório home
    - Login shell
    - Terminal
    - Mail Status
    - Idle Time
    - Data do último login.
12. Imprima na tela o calendário para Setembro de 1752.
13. Determine que dia da semana cairá o primeiro dia do ano 2000.

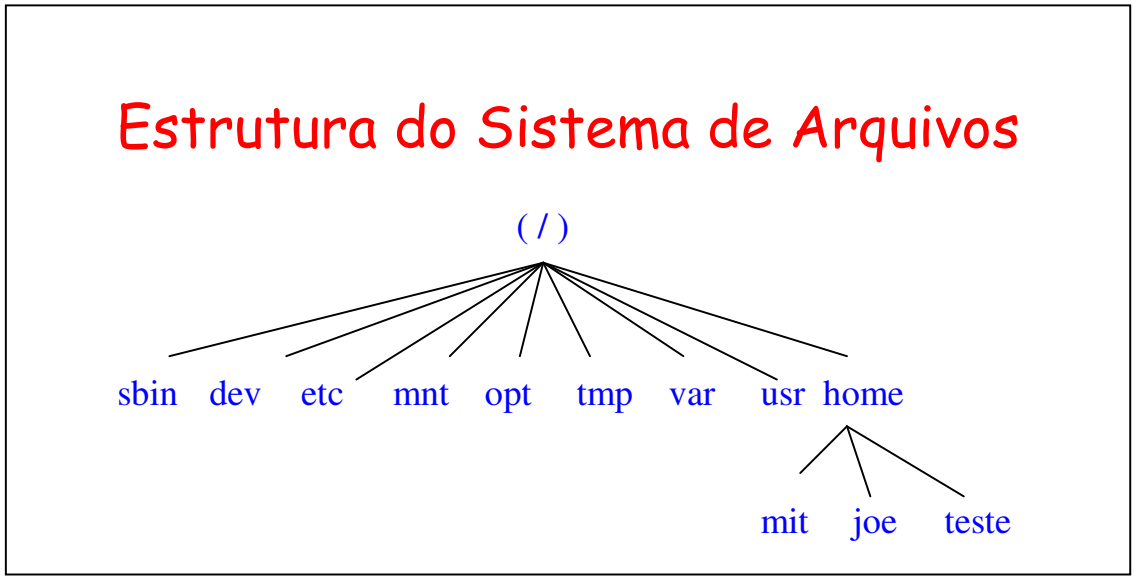
Respostas:

1. $ date
2. $ echo Laboratorio para sessão de Comandos Basicos do UNIX
3. $ man date
   $ date '+%m/%d/%y'
4. $ man echo
   $ man man
   .
   .
5. $ logname
6. $ tty
7. $ finger root
8. $ finger *seu_loginid*
9. $ cal
   $ cal *mês_aniversário ano_nascimento*
10. $ data '+DATE: %m/%d/%y%nTIME: %H:%M:%S:'
11. $ finger
12. $ cal 09 1752
13. $ cal 01 2000

Sistema Operacional

Unix

Parte III



Estrutura do Sistema de Arquivos

O sistema operacional UNIX organiza os arquivos em uma estrutura hierárquica do tipo árvore. Isto é feito agrupando arquivos relacionados entre si em diretórios. Um *diretório* ou *arquivo de diretório* é um arquivo que contém informações sobre os arquivos agrupados imediatamente abaixo da árvore. Arquivos de diretório não contém dados, somente nomes de arquivos.

O sistema de arquivos UNIX pode ser imaginado como uma árvore invertida com a raiz em cima, *arquivos de diretórios* como galhos e os *arquivos regulares* como sendo folhas. Com esta analogia de árvore invertida, os galhos podem dividir-se em outros galhos de forma de cada diretório pode conter outros diretórios.

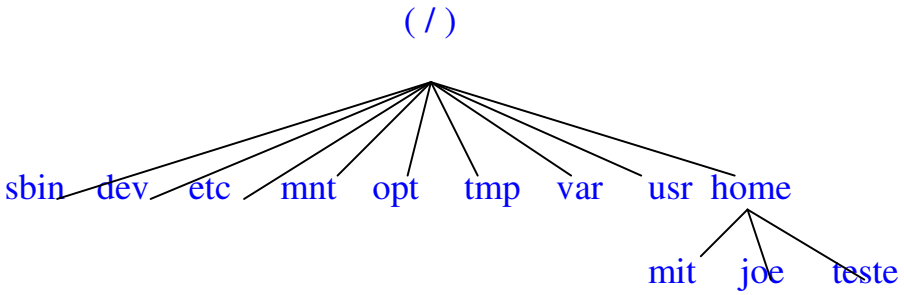
Um *diretório pai* (parent directory) é o diretório imediatamente acima de cada diretório na estrutura de árvore. Cada diretório tem um diretório pai, exceto para o diretório root, porque ele está no topo da hierarquia do diretório.

Os exemplos nas páginas adjacentes ilustram uma estrutura de arquivos simplificada. Todos os rótulos (por exemplo, root, bin e assim por diante) representam arquivos de diretórios. Observe que o diretório root é representado por uma barra (/). Cada diretório mostrado provavelmente terá arquivos e diretórios adicionais abaixo dele que não estão incluídos na ilustração simplificada.

Para localizar um arquivo ou diretório em particular, você deve rastrear o caminho a partir do diretório root através de cada diretório até que aquele que você está procurando seja encontrado. Este caminho é chamado de *full path name (caminha completo)* do arquivo ou diretório.

Diretórios Padrão do UNIX

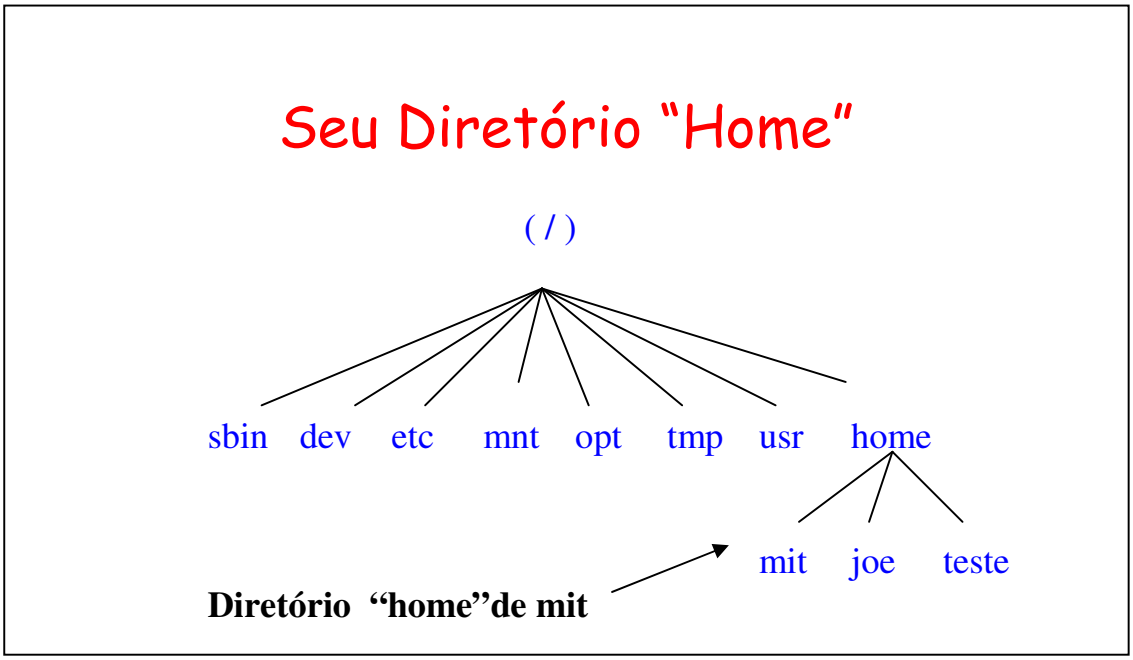
O diretório *root*, representado por uma /, está presente em todos os sistemas UNIX e é o ancestral para todos os arquivos e diretórios abaixo dele:



Diretórios Padrão UNIX

Os diretórios que mapeam os sistemas de arquivos no seu sistema UNIX (e seus conteúdos) podem variar dependendo da instalação do UNIX que você está rodando e como os sistemas de arquivos foram construídos originalmente.
O topo do sistema de arquivos no UNIX sempre será o diretório root, indicado por uma (/). Devido ao fato dele estar no topo do sistema de arquivos, ele é o único diretório que não tem um diretório pai. O diretório root está presente em todos os sistemas UNIX e é ancestral para todos arquivos e diretórios abaixo dele.
Abaixo do diretório root está um número de diretórios para os administradores de sistema e usuários. Os nomes dos diretórios que estão abaixo do diretório root podem variar dependendo do tipo de UNIX que está instalado no seu sistema. Alguns diretórios padrão do UNIX, que estão localizados abaixo do diretório root tanto nos sistemas baseados no AT&T e BSD estão mostrados abaixo:

Diretório	Descrição
/bin	Contém a maioria dos programas utilitários mais comuns do UNIX. Quando você executa um comando comum do UNIX no seu sistema, provavelmente você estará executando um arquivo residente no diretório /bin
/dev	Contém os arquivos especiais para os periféricos de E/S que estão conectados ao sistema, tais como terminais (tty), impressoras (lp) e discos (dsk).
/etc	Contém arquivos de configuração dos sistema, de boot e manutenção que são usados primeiramente pelos administradores de sistema.
/lib	Contém comandos objetos para rotinas de matemática, linguagem de programação orientada e rotinas gráficas e estatísticas.
/usr/bin	Contém comandos e utilitários adicionais do UNIX. Comandos e programas locais estão localizados aqui.
/tmp	O diretório /tmp geralmente é grande, pois os arquivos temporários que são lidos e escritos por todos os usuários do sistema são armazenados aqui .



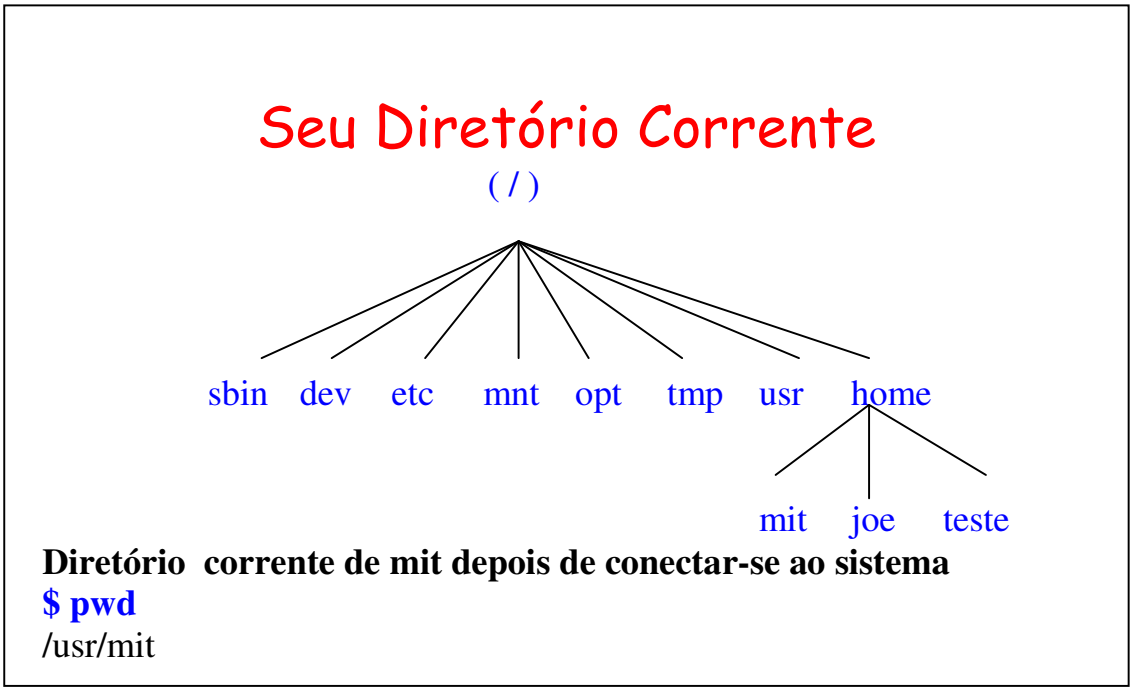
Seu Diretório Home

Quando você se conecta ao sistema UNIX, você automaticamente é colocado em seu *home directory* (algumas vezes chamado seu *diretório de login*).

Por convenção, o nome de seu home directory é o mesmo nome do seu login ID. Isto é montado pelo seu administrador de sistema e é guardado no arquivo */etc/passwd*. Usualmente, home directories estão localizados abaixo dos diretórios /usr ou /home.

Os usuários armazenam arquivos e diretórios pessoais em seus home directory. Este é seu espaço em disco. (Usuários podem também acessar outros arquivos e diretórios no sistema se tiver permissão para tanto. Permissões de arquivos e diretórios serão discutidos na próxima seção) .

No exemplo da próxima página, o home directory de mit esta enfatizado. O caminho completo para o home directory de mit é: */usr/mit*.



Diretório corrente de mit depois de conectar-se ao sistema

$ pwd

/usr/mit

Seu diretório corrente

Enquanto você está conectado no sistema UNIX, você sempre estará em um diretório particular. O diretório que você estiver correntemente é chamado de *diretório corrente*, ou *diretório de trabalho (working directory).* Neste curso, ele sempre será referenciado como o seu diretório corrente.

Quando você se conecta ao sistema, seu home directory é o diretório corrente. O diretório corrente pode ser representado por um ponto (.) ou por seu caminho completo.

A ilustração na página seguinte mostra o diretório corrente de mit quando ela se conecta ao sistema.

Sistema de Arquivos no UNIX

Nesta sessão discutiremos os seguintes comandos:

- pwd
- cd
- file
- find
- mkdir
- rm
- rmdir
- wc

O comando pwd

O comando *pwd* (print working directory) mostra o caminho completo do seu diretório corrente. Não há argumentos ou opções para este comando. Ele diz onde você está localizado na estrutura de arquivos do sistema.

O formato para o comando *pwd* é como segue:

pwd

O exemplo na transparência acima mostra a saída para o comando pwd se mit o executasse ao se conectar.

Criando Diretórios

```
$ pwd
/usr/mit
$ mkdir sec
$ mkdir sec/sec5
$ mkdir bin memos letters
```

Criando diretórios

Em geral, seus arquivos e diretórios residem abaixo do seu home directory. Procure habituar-se a usar diretórios para agrupar e organizar seus arquivos e diretórios em sua área em disco. Fazendo assim, você encontrará facilmente cada um de seus arquivos. Depois que você criou um diretório, você poderá criar arquivos e outros diretórios abaixo dele. Para criar um diretório, use o comando *mkdir*.

Comando *mkdir*

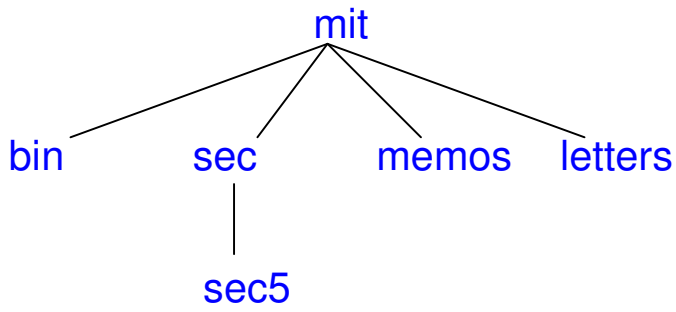
Use o comando *mkdir* (make directory) para criar um ou mais diretórios. O formato do comando mkdir é como segue:

mkdir [dirname...]

dirname Nome(s) do diretório ou diretórios que você quer criar. Não pode existir nenhum arquivo ou diretório com o mesmo nome do diretório novo onde ele está sendo criado.

Criando Diretórios

Depois de executar os comandos acima, a estrutura de arquivos e diretórios de mit será:



Quando um novo diretório é criado, ele está vazio exceto pelas seguintes entradas:

ponto (.)	Representa o diretório corrente
ponto ponto (..)	Representa o diretório pai

As entradas ponto e ponto são criadas automaticamente sempre que você cria um diretório usando o comando *mkdir*. Isto habilita o novo diretório a reconhecer a si mesmo e seu diretório pai.

No exemplo da página anterior, mit está criando um novo diretório, *sec*, para armazenar arquivos criados durante essa aula e outro diretório abaixo dele, *sec5*, para armazenar os exercícios da seção 5. Ele também criará os diretórios *bin*, *memos* e *letters*. mit armazenará os executáveis que ela escreveu abaixo do diretório *bin*. Os outros diretórios permitirá a ela armazenar seu memorandos e cartas separadamente para um acesso mais fácil.

Mudando de Diretórios

O Comando cd

```
$ pwd
/usr/mit
$ cd /usr/bin
$ pwd
/usr/bin
$ cd /
$ pwd
/
$ cd
$ pwd
/usr/mit
```

Mudando de diretório

Use o comando *cd* (change directory) para mudar seu diretório corrente para outro. O comando *cd* permite a você movimentar na estrutura de arquivos do sistema e acessar arquivos e diretórios que estão na sua área em disco ou outras áreas do sistema que você tiver permissão para acessar.

O comando *cd*

O comando cd muda o diretório corrente para o diretório especificado na linha de comando. O formato do comando cd é:

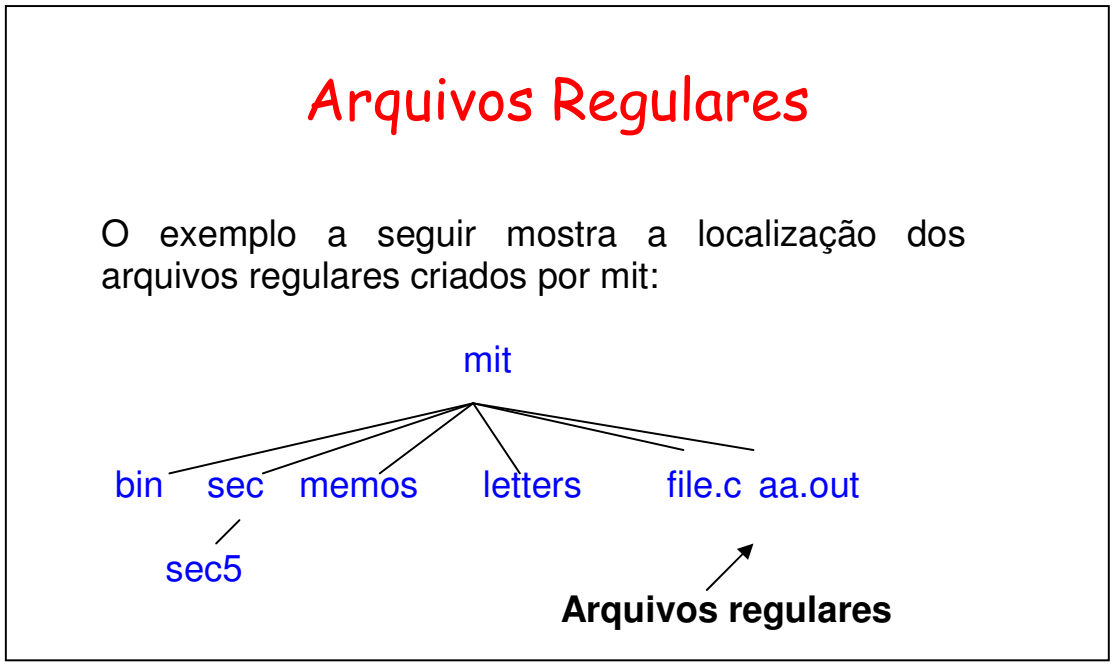
```
cd [directory]
```

directory — diretório que você tem permissão para acessar. Se o nome do diretório não é especificado como argumento para o comando *cd*, o seu home diretory toma o seu diretório corrente. Se você não tem certeza onde você está na estrutura de arquivos do sistema ou qual o seu diretório corrente, entre com o comando cd, que sempre retornará você para seu home directory..

Tipos de Arquivos

Do ponto de vista do usuário há três tipos de arquivos básicos:

- Arquivos regulares
- Arquivos de diretórios
- Arquivos especiais



Arquivos Regulares

Um arquivo regular contém informações ou dados que o usuário coloca nele, tais como texto ou programas binários. Arquivos regulares estão no fim de cada caminho da árvore do sistema de arquivos e nenhum outro arquivo ou diretório podem ramificar deles. (Galhos não podem crescer de folhas).
Usando um editor de texto é uma maneira de criar arquivos textos. (O editor vi será discutido mais tarde). Um arquivo de texto consiste de uma string de caracteres com linhas que terminam com um caracter invisível de "new line" (criada pressionando a tecla RETURN no fim de cada linha). O fim de arquivo é marcado com o EOF, que
também é invisível.

Arquivos binários são criados por compiladores tais como o compilador C. Programas binários são seqüências de instruções como elas aparecem na memória quando o programa começa a ser executado.

Os exemplos na página anterior mostram arquivos que mit criou usando o editor *vi* e o compilador C. Seguindo o exemplo está a estrutura de arquivos de mit que mostra a localização dos seus novos arquivos regulares.

Arquivos de diretórios

Um *arquivo de diretório* contém uma lista dos arquivos e diretórios abaixo dele na estrutura de arquivos e suas localizações físicas no disco. Usualmente, os diretórios são usados para agrupar arquivos relacionados entre si. Organize seus arquivos criando diretórios apropriados para colocá-los.

Inodes

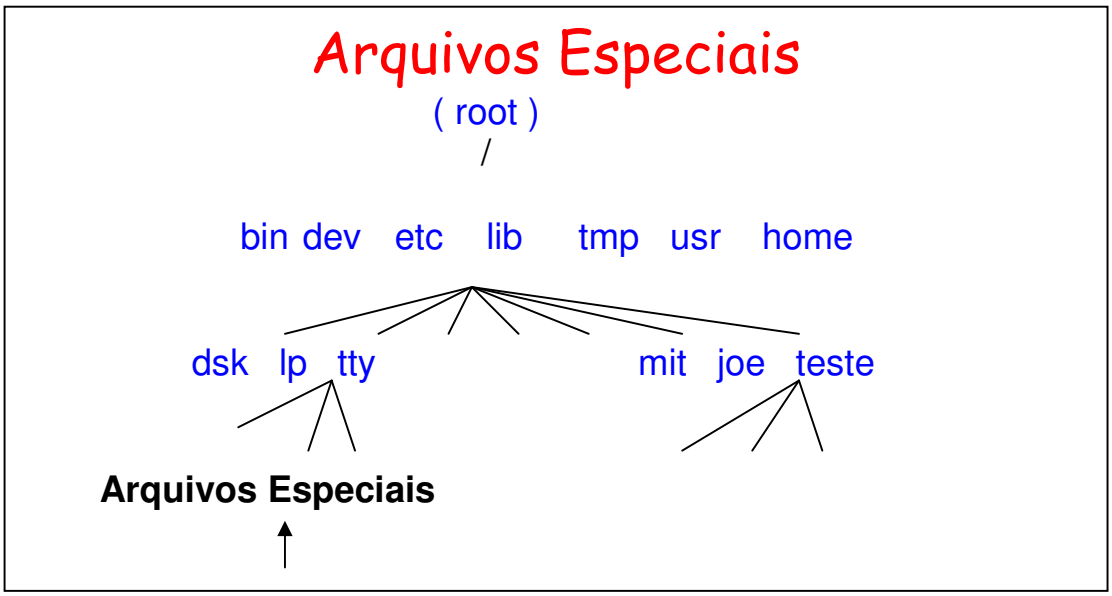
Um *inode* relaciona cada nome de arquivo no sistema a um número de inode, que é um índice para uma tabela de inode. O inode para um arquivo é onde as informações para aquele arquivo está guardado.
Geralmente, usuários UNIX não precisam saber os detalhes internos de como os inodes trabalham, mas eles devem estar atentos da sua existência e suas funções básicas.

Quando você cria um arquivo, você adiciona uma entrada no diretório que descreve o arquivo. Quando você remove um arquivo, o sistema operacional remove a entrada, assim como o espaço do arquivo do disco.

Cada diretório é automaticamente provido com duas entradas quando ele é criado (. e ..). A entrada ponto (.) é um link para o próprio diretório e a entrada ponto ponto (..) é

um link para o diretório pai. Os inodes contém as seguintes informações sobre cada arquivo e diretório:

- Privilégios de acesso (modo ou permissões)
- Número de links para o arquivo
- Dono
- Data de criação, último acesso e última modificação
- Tamanho do arquivo
- Tipo do Arquivo
- Grupo
- Endereço do disco (ponteiros para blocos de dados).



Arquivos Especiais

Arquivos especiais estão associados com os periféricos de E/S. Cada periférico de E/S suportado no sistema é representado por pelo menos um arquivo especial. Esses arquivos contém parâmetros que fornecem ao sistema operacional informações necessárias para acessar tais devices (periféricos). Arquivos especiais são lidos e escritos exatamente como arquivos comuns de disco, mas solicitações para leitura e escrita resultam em disparar o periférico associado. Os arquivos especiais de disco e memória são protegidos contra acesso.

Usualmente, arquivos especiais residem abaixo do diretório */dev* (devices). Cada arquivo neste diretório representa um periférico de E/S específico.
Arquivos especiais estabelecem uma conexão entre um caminho arbitrário (por exemplo, */dev/tty12*) e as rotinas do sistema operacional que lidam com os periféricos que controlam os periféricos físicos. Os arquivos especiais são criados quando o sistema é instalado pelo administrador.

O exemplo de sistemas de arquivos na página seguinte mostra os arquivos especiais */p* (para line printer), *tty* (para terminal device) e *dsk* (para disk device) que residem abaixo do diretório */dev.*

O comando file

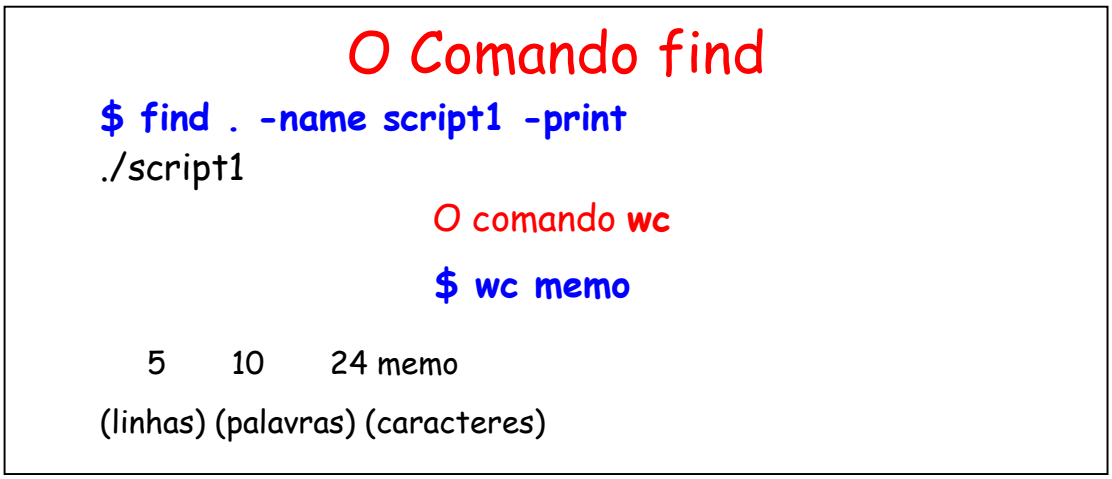
```
$ file memo script1 lixo dir
memo:       ascii text
script1:    commands text
lixo:       empty
dir:        directory
$ file /  /etc /etc/passwd
/:          directory
/etc:       directory
/etc/passwd: ascii text
```

O comando *file*

O comando *file* deixa você determinar que tipo de informação está contido em um determinado arquivo ou arquivos. Entre com o nome do arquivo que você quer obter as informações como argumento para o comando *file*. O sistema examina o começo do arquivo, determina o seu tipo e mostra essa informação na sua tela.

O formato para o comando file é:

```
file [opções] filelist
```



O comando *find*

O comando *find* é útil quando você está procurando por um arquivo ou diretório particular na estrutura de arquivos do sistema. O comando *find* procura o arquivo começando pelo diretório especificado, por arquivos que satisfazem a expressão especificada.

O formato para o comando é como se segue:

find *pathname expression*

Exemplo:

```
$ find / -name xyz - print
```

No exemplo acima, o comando find procura o nome do arquivo, seguindo a designação -name, xyz, partindo do diretório do root, especificado pelo primeiro argumento, /, e imprime o caminho completo do arquivo na tela uma vez que ele seja localizado. Se a opção *-print* for omitida, a saída é descartada e não será mostrada.

O comando *wc*

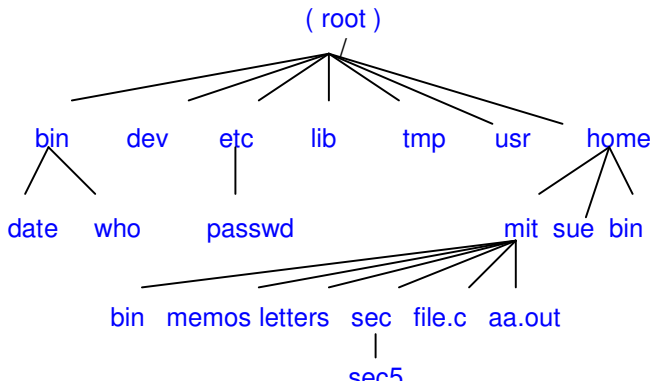
O comando *wc* conta o número de linhas, palavras ou caracteres nos arquivos especificados. O primeiro número mostrado na saída indica o número de linha no arquivo. O próximo número é o número de palavras, depois vem o número de caracteres seguido do nome do arquivo.

O formato para o comando *wc* é:

wc [-lwc] filelist

A opção *-l* retorna o número de linhas; *-w* o número de palavras e *-c* o número de caracteres. O default do comando wc é retornar todas as três opções.

Caminhos Completos



Sair do diretório memo e ir para letters usando o caminho

$ /home/mit/letters

Path Names

Dois arquivos no mesmo diretório não podem ter o mesmo nome. Entretanto, arquivos em diretórios diferentes podem ter o mesmo nome. O sistema será capaz de distinguir entre os dois arquivos, pois cada um deles tem um único *path name* (caminho). Um nome de um arquivo real está em seu path name. Um path name para um arquivo pode ser um *full path name* (*ou absolute path name) (caminho completo)* ou um *relative path name.*

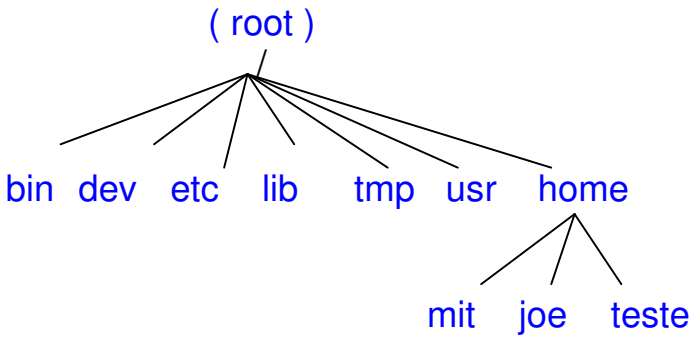
Caminhos Completos

- O caminho completo para o arquivo **who** é /bin/who
- O caminho completo para o diretório de **mit** é /home/mit.
- O caminho completo para o diretório **sec5** é

Full Path Names:

Quando referir-se a um arquivo com um *full path name* você deve especificar cada diretório que faz parte de seu caminho, partindo do root e terminando com o nome do arquivo desejado. Um caminho completo sempre começa com uma barra (/). A barra inicial indica que você está partindo do diretório root; todas as barras subsequentes são usadas para separar diretórios e arquivos. Um arquivo ou diretório à direita da barra é um filho do diretório que está à esquerda da barra (seu diretório pai).

O Diretório "pai"



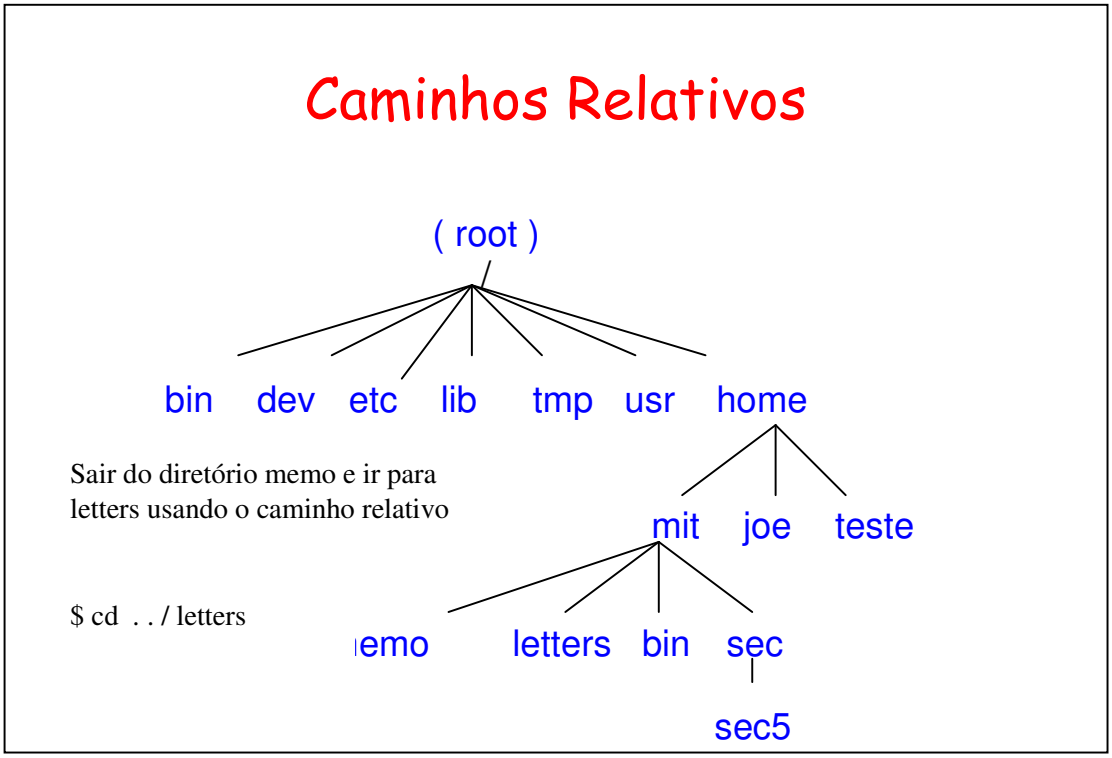
- O caminho completo para o diretório "pai"de mit é /home.

Parent Directory:

O *parente directory (diretório pai)* de um arquivo ou diretório é o diretório imediatamente acima dele na estrutura de arquivos do sistema. Ponto ponto (..) representa o diretório pai do corrente diretório.

Para mudar para o diretório pai, use o comando *cd* e ponto ponto, como no exemplo:

```
$ cd..
```

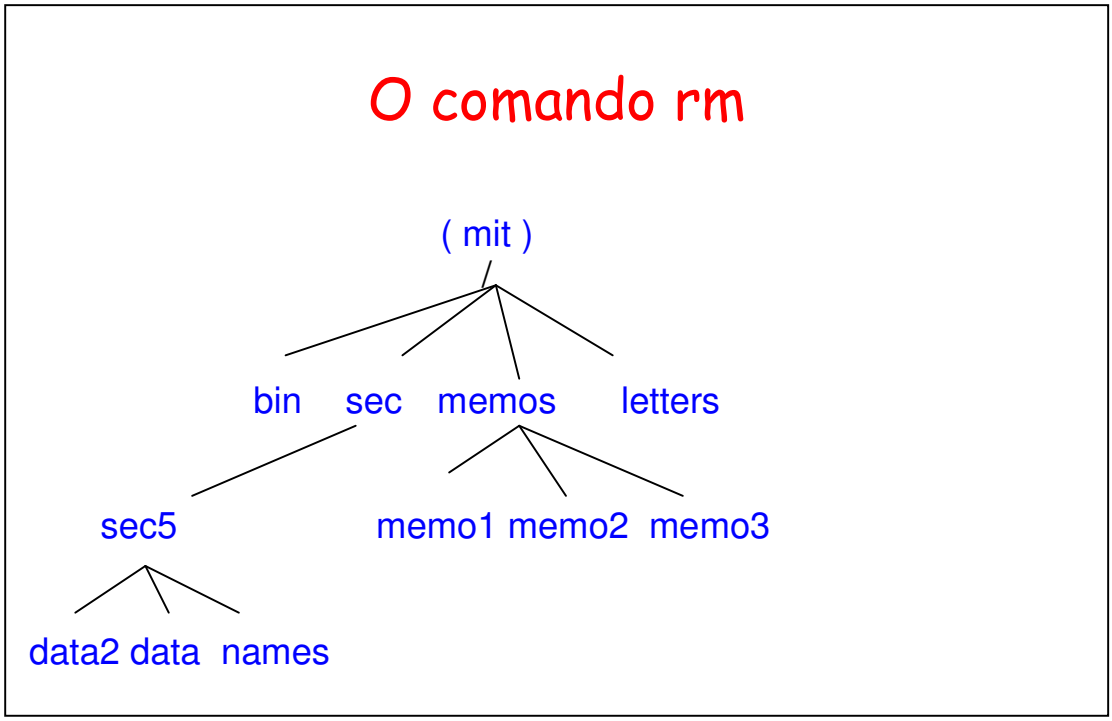


Relative Path Names:

Um *relative path name* é o caminho que é relativo ao seu diretório corrente. Relative path name nunca começam com uma barra (/). Eles podem começar com o nome do arquivo, uma notação de ponto (.) para indicar o diretório corrente ou uma notação de ponto ponto (..) para referir-se ao diretório pai. Barras são usadas como delimitadores.

A única maneira de mover-se para cima na estrutura de diretórios é usando a notação ponto ponto para referir-se ao diretório pai; caso contrário, o movimento será sempre para baixo na estrutura de arquivos do sistema.

Relative path names são usados para evitar a digitação do caminho completo, embora nem sempre o relative path name seja menor do que o full path name.



Removendo Arquivos e Diretórios

As subseções subsequentes descrevem os comandos que você pode usar para remover arquivos e diretórios.

O comando *rm*

O comando *rm* (remove) deixa você remover arquivos regulares. O comando *rm* removendo a entrada na tabela de inodes do arquivo e liberando o espaço em disco que os dados do arquivo ocupavam anteriormente.

O formato do comando *rm* é:

rm [opções] arquivo[arquivo...]

O comando rm

mit remove os arquivos **data2** e **names** do diretório **sec5** e então remove o diretório **memos** e seu conteúdo:

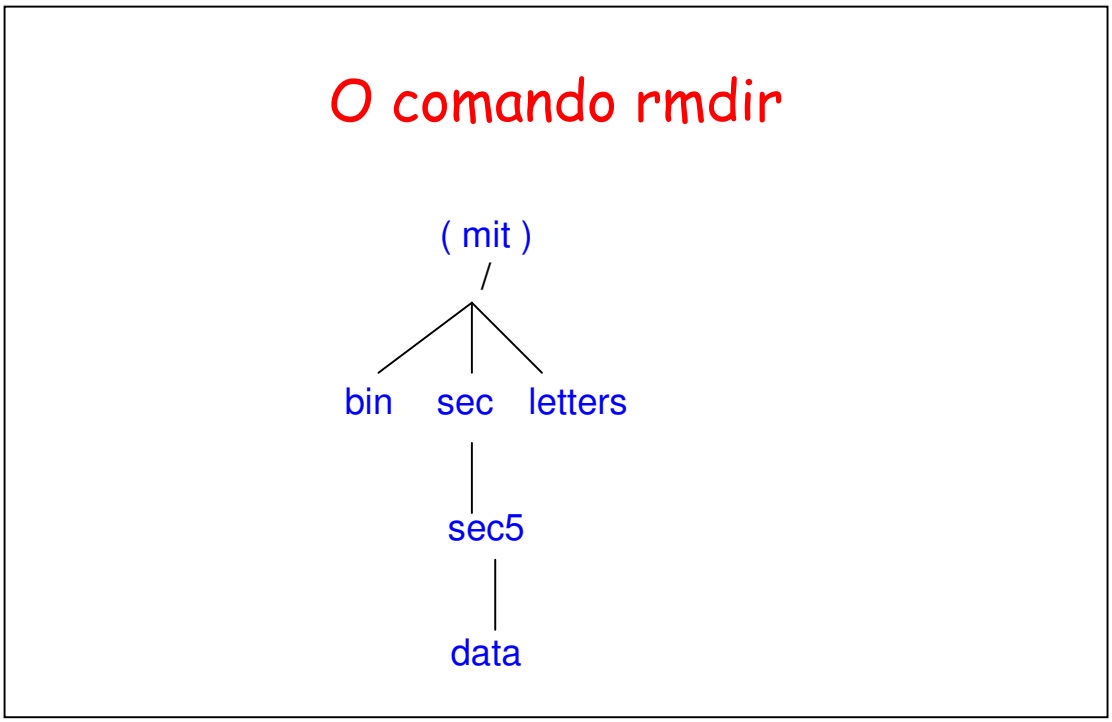
$ pwd
/usr/mit
$ rm sec/sec5/names
$ rm -i sec/sec5/data
uub/sec5/data: ? **Y**
$ rm -ir memos
sec/sec5/memos/memos1: ? **Y**
sec/sec5/memos/memos2: ? **Y**
sec/sec5/memos/memos3: ? **Y**
sec/sec5/memos: ? **Y**

Você pode suprir o comando com múltiplos nomes de arquivos separados por espaço. Depois que um arquivo é removido do sistema, ele não pode ser recuperado a menos que ele tenha sido salvo (backed up) em fita ou algum outro meio anteriormente.

A opção *-i* (interactive) é usada como precaução quando removendo arquivos (principalmente se você estiver usando o usuário root). Quando usando esta opção com o comando *rm*, será pedido para você confirmar a operação com o nome de cada *arquivo* na linha de comando. Para remover cada arquivo, responda digitando Y (yes). Se qualquer outro caracter for digitado, o arquivo não será removido. Se uma lista de arquivos foi dado, o comando rm irá confirmar com você a remoção de cada um deles.

Usando o comando rm com a opção *-r* (recursive), você poderá remover o diretório especificado, assim como todos os arquivos e subdiretórios abaixo dele. Use essa opção com precaução. É recomendado que você sempre use a opção

-i com a opção *-r*.



O comando *rmdir*

O comando *rmdir* remove diretórios vazios da estrutura de arquivos através da remoção das conexões (links) para seu diretório pai.

O formato para o comando *rmdir* é como segue:

rmdir *diretório [diretório...]*

Não há opções para o comando *rmdir*. Os diretórios especificados deve estar vazios para que o comando possa ser executado. Esta é uma precaução para impedir você de remover arquivos ou diretórios inadvertivalmente. Um diretório é considerado vazio se somente as entradas ponto (.) e ponto ponto (..) estiverem nele.

O comando rmdir

mit tenta remover o diretório **sec5** e descobre que ele não está vazio; então remove o arquivo **data** e o diretório vazio **sec5:**

$ pwd

/usr/mit

$ rmdir sec/sec5

rmdir: sec/sec5 not empty

$ rm -i sec/sec5/data

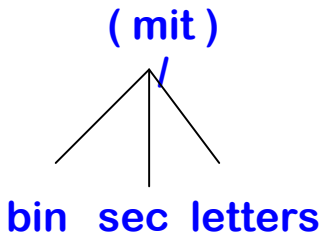
uub/sec5/data: ? **Y**

$ rmdir sec/sec5

Se o diretório não estiver vazio, o comando retornará uma mensagem de erro e não removerá o diretório.

O comando rmdir

A estrutura de diretório de mit agora consiste de:



Exercícios

1. Conecte-se ao seu sistema. Verifique o seu diretório corrente.
2. Mude seu diretório corrente para o diretório */bin*. Que comando você usou?
3. Verifique o que mudou no seu diretório corrente.Mude seu diretório corrente para o diretório "pai" do diretório */bin*. Execute o comando **pwd** para verificar onde você está na estrutura de arquivos do sistema.
4. Volte para o seu diretório "home". Agora mude para o diretório "pai" de seu diretório "home". Observe que agora o seu diretório corrente é o diretório "pai"de seu diretório "home".
5. Volte novamente para o seu diretório home. Crie um diretório chamado **uub**. Mude para o diretório **uub**.
6. Crie os diretórios **sec4**, **sec5** e **sec6** abaixo do diretório **uub**.
7. Sem sair do seu diretório corrente , crie o diretório **exer** abaixo do diretório **sec4**.
8. Novamente, sem sair do seu diretório corrente, remova o diretório **sec6**.
9. Tente remover o diretório **sec4** usando o comando **rmdir**. O que aconteceu?
10. Agora remova o diretório **sec4** usando o comando **rm** com uma opção especial que o permite fazê-lo. Use, também, a opção que confirma a remoção de cada arquivo.
11. Use **/dev/console** como argumento para o comando **file**. Mude seu diretório corrente para **/dev/console**. O que aconteceu?
12. Tente remover o arquivo **/etc/motd**. O que aconteceu? Use **etc/motd** como argumento para o comando **file**.
13. Verifique se você está no seu diretório "home". Crie um diretório chamado **temp**. Crie um subdiretório abaixo de **temp** chamado **abc**. Agora crie um subdiretório chamado **123** abaixo do diretório **abc**. Mude o diretório corrente para o diretório **temp**. Tenha certeza de que você está no diretório **temp**. Entre com o comando **rm** usando asterisco (*) como argumento. **Não** entre com este comando em seu diretório home. O que acontece?

Solução

1. **$ pwd**
2. **$ cd/bin**

   $ pwd
3. **$ cd ..**

   $ pwd
4. **$ cd**

   $ cd ..

   $ pwd
5. **$ cd ; mkdir uub**

   $ file uub

   $ cd uub
6. **$ mkdir sec4 sec5 sec6**
7. **$ mkdir sec4/exer**
8. **$ rmdir sec6**
9. **$ rmdir sec4**

   Você deve receber a seguinte mensagem: rmdir:sec4 not empty.
10. **$ rm -i sec4**
11. **$ file /dev/console**

    $ cd /dev/console

    Você deve receber a seguinte mensagem: /dev/console: Not a directory.
12. **$ rm /etc/motd**

    Você deve receber a seguinte mensagem: rm: /etc/motd removed: Permission denied

    $ file /etc/motd
13. **$ pwd**

    $ mkdir temp

    $ mkdir temp/abc

$ mkdir temp/abc/123

$ cd temp

$ pwd

$ rm *

Você deve receber a mensagem: rm: abc is as directory.

Extensão no nome dos arquivos

- Extensões mais comuns são:

.c Fontes de programas em C (file.c)

.o Objetos - Códigos binários (file.o)

- Extensões ajudam a lembrar o tipo de informação contida no arquivo:

letter.bob sort.c mail.aug1 man.txt

memo.0816 cmd_txtmins.1101 mod_ex

Os arquivos de Usuários

Escolhendo os nomes de seus arquivos

Quando estiver dando nome aos seus arquivos, escolha um nome que descreva o conteúdo do arquivo. Qualquer caracter ASCII é válido em um nome de arquivo. UNIX é caso sensitivo, ou seja faz diferença entre letras maiúsculas e minúsculas.
Por exemplo, *file* e *FILE* são considerados dois arquivos diferentes.

O tamanho permitido para o nome de arquivos varia de entre os vários sistemas UNIX. Algumas versões antigas do UNIX pode permitir nomes de até 14 caracteres ou menores. Versões mais novas do UNIX geralmente permitem mais do que 14 caracteres.

Extensão do nome de arquivos

Os usuários geralmente colocam *extensões* aos nomes de seus arquivos. Uma extensão ajuda a identificar qual o tipo de informação que um arquivo contém. Extensões são adicionadas ao nome do arquivo freqüentemente seguindo um caracter ponto (.) ou undescore (_) e são opcionais na maioria dos casos.
As páginas seguintes mostram alguns exemplos de nomes de arquivos com suas extensões.
Lembre-se, UNIX é caso sensitivo, fazendo diferença entre caracteres maiúsculos e minúsculos. Para o sistema UNIX, o arquivo *abc* é um arquivo completamente diferente de *ABC* ou *Abc*.

Usando caracteres especiais

- **Não** use os seguintes caracteres especiais no nome dos arquivos:

; ( ) & | > < * ? \ ‘ “ ‘ [ ] $ # ! ^ { } :

- Exemplos infelizes de nomes de arquivos:

letter*mom txt\cmd1 mail(feb) man<date
mit&ed sort/c mod_ex? File>out

Usando caracteres especiais do shell em nomes de arquivos

Evite usar qualquer um dos seguintes caracteres especiais da shell em seus arquivos. Esses caracteres têm significado especial para a shell podem ser mal interpretados se usadas nos nomes dos arquivos.

; ( ) & | > < * ? \ ' " ' [ ] $ # ! ^ { }

Os caracteres especiais da shell (algumas vezes chamados de metacaracteres) e suas funções serão discutidos no Capítulo "A Shell". Você deve estar atento a existência de caracteres especiais da shell, de forma que você possa evitar usá-los no nome de seus arquivos.

Embora os caracteres especiais da shell ofereçam uma grande potência e flexibilidade, ele também podem causar problemas inesperados, principalmente para usuários novos. Use-os com precaução.

Arquivos do Usuário

Os seguintes comandos serão descritos nesta seção:

- Cat
- grep
- ls
- more
- tail
- cp
- mv
- chmod
- umask

O comando cat

Os arquivos mang1 e mang2 existem no diretório corrente:

$ cat mang1

Mangueira
Estou aqui na plataforma
Da estação primeira
O morro veio me chamar
De terno branco e chapeu de palha
Vou me apresentar
A minha nova parceira
Já mandei subir o piano pra Mangueira

Mostrando seus arquivos

O UNIX oferece vários comandos para mostrar o conteúdo de seus arquivos textos. Os arquivos podem ser formatados de várias maneiras diferentes, dependendo de qual comando você usa. Esses comandos não são para serem usados em diretórios, arquivos binários ou especiais. Se você tentar mostrar um arquivo binário executável, caracteres estranhos (lixo) serão enviados à sua tela e podem travar o seu terminal.

Os comandos UNIX usados para mostrar o conteúdo de arquivos regulares que são discutidos nas subseções seguintes são *cat, more* e *tail*. Você pode usar outros comandos UNIX, tais como *head, less* (público), *pg* e *pr* para mostrar o conteúdo de arquivos regulares. Veja a man page para esses comandos para maiores informações.

O comando cat

$ cat mang2

A minha musica não e de levantar poeira
Mas pode entrar no barracao
Onde a cabrocha pendura a saia
No amanhecer da quarta-feira
Mangueira
Estacao primeira de Mangueira
Pela vida inteira Mangueira

O comando *cat*

O comando *cat* lê cada arquivo especificado na linha de comando e mostra o conteúdo daquele arquivo na sua tela. O nome do comando *cat* deriva da palavra *concatenate*. Os arquivos especificados como argumento para o comando *cat* são mostrados como se eles tivessem sido concatenados juntos.

O formato para o comando *cat* é:

```
cat [opções] [arquivo...]
```

O comando cat

Os arquivos mang1 e mang2 existem no diretório corrente:

$ cat mang1 mang2
Mangueira
Estou aqui na plataforma
Da estacao primeira
O morro veio me chamar
De terno branco e chapeu de palha
Vou me apresentar
A minha nova parceira
Já mandei subir o piano pra Mangueira
A minha musica não e de levantar poeira
Mas pode entrar no barracao
Onde a cabrocha pendura a saia
No amanhecer da quarta-feira
Mangueira
Estação primeira de Mangueira
Pela vida inteira de Mangueira

Mesmo que o comando *cat* tenha várias opções disponíveis, elas não são usadas freqüentemente. Você pode especificar qualquer número de arquivos na linha de comando. É opcional se você especifica ou não um nome de arquivo. Se você não especificar um nome, o comando *cat* lê da entrada padrão (standard input) (seu teclado) até encontrar um caracter de CONTROL-D no começo de uma linha em branco. Veja no capítulo "A Shell", exemplos de como usar essa função.

Uma desvantagem de usar o comando *cat* para mostrar um arquivo grande é que o conteúdo do arquivo passa direto pela tela, concluindo com a última tela de informação do arquivo e impede você de ver a primeira parte do arquivo.

Criando arquivos com o comando cat

$ cat > arquivo

O comando cat sem qualquer argumento usa a entrada padrão para obter os dados para o comando. Estou digitando os dados agora.
Quando eu entrar com ^d em uma nova linha, a saida sera redirecionada do buffer para o novo arquivo.
^D

Criando Arquivos com o comando *cat*

Para criar um arquivo texto no seu sistema UNIX, você deve aprender um editor de texto tal como editor *vi*, na próxima seção. Até você aprender o *vi* (ou qualquer outro editor disponível no seu sistema), você pode usar o comando *cat* para criar arquivos textos. Criando arquivos usando o comando *cat* é também útil para usuários experientes que querem criar um arquivo pequeno rapidamente.

Use o seguinte formato para o comando *cat* criar um arquivo texto:

cat > arquivo

O sinal de maior (>) seguindo o comando *cat* é usado para indicar a direção da saída, significando que a saída do comando *cat* está direcionada para o arquivo *arquivo*.

Criando arquivos com o comando cat

$ cat arquivo

O comando cat sem qualquer argumento usa a entrada padrão para obter os dados para o comando.
Estou digitando os dados agora.
Quando eu entrar com ^d em uma nova linha, a saida sera redirecionada do buffer para o novo arquivo.

$

Quando você entra com a seqüência acima e pressiona RETURN, tudo que for digitado no teclado será colocado no arquivo que segue o símbolo de direcionamento de saída. Você deve pressionar RETURN no fim de cada linha.

Para indicar o fim de um arquivo, você deve pressionar CONTROL-D como primeiro caracter na linha seguinte do texto.

Esse método é usado para criar arquivos pequenos, porque não é possível voltar para modificar ou corrigir linhas anteriores depois que você pressionou a tecla RETURN.

O exemplo da transparência anterior mostra o uso do comando *cat* para criar arquivos.

Comando grep

$ grep [-opções] regexpr [nome (s) arquivo]

O comando grep é usado para procurar palavras dentro de arquivos. A procura pode ser feita no diretório em que o usuário está ou nos subdiretórios, de forma recursiva

$

O grep e suas variantes são provavelmente os comandos mais úteis do Unix para comparação e localização de cadeias de texto. Grep é a abreviação de "Get Regular ExPression" (obter expressão regular) e, na maioria das versões do Unix, vem em três versões um pouco diferentes. A primeira é o grep regular, que permite buscar um texto que corresponda a uma cadeia de texto que pode incluir caracteres curinga, conjuntos de caracteres e metacaracteres. A segunda é o fgrep (Fast Grep) e a terceira é o egrep (extended grep).

As opções do grep são:

-b Retorna o número do bloco e também a linha correspondente

-c Retorna apenas uma contagem do número de linhas correspondentes

-h Retorna apenas as linhas correspondentes, mas não os nomes de arquivos onde elas estavam

-i ignora maiúsculas/minúsculas durante a comparação

-n Retorna os números de linha e também a linha correspondente

-s Suprime as mensagens de erro

-v Retorna apenas as linhas que não correspondem à expressão regular

-w Compara apenas palavras inteiras

-R Recursiva – avança para os subdiretórios

Comando more

$ more /etc/motd

```
Digital UNIX V4.0D  (Rev. 878); Wed Aug  5  16:03:43  EST  1998

The installation software has successfully Installed your system.

There are logfiles that contain a record of your installation.
These are:
        /var/adm/smlogs/install.cdf      - configuration description file
        /var/adm/smlogs/install.log      - general log file
        /var/adm/smlogs/install.FS.log   - file system creation logs
        /var/adm/smlogs/setld.log        - log for the setld(8) utility
        /var/adm/smlogs/fverify.log      - verification log file
motd: END
```

O comando *more*

O comando *more* mostra o conteúdo de um ou mais arquivos uma tela por vez. É similar ao comando *cat*, mas pára depois da primeira tela cheia de informação em vez de mostrar o arquivo inteiro. Pressione a tecla RETURN para mostrar uma linha a mais e a tecla SPACE para mostrar a próxima página.

O formato para o comando *more é:*

```
more [opção] [arquivo...]
```

Quando estiver usando o comando *more* para mostrar um arquivo, uma mensagem similar a seguinte será mostrada no fim de cada tela cheia de informação:

--More--(43%)

A porcentagem (43%) descreve qual a porcentagem do arquivo já foi mostrada. Algumas opções úteis que podem ser digitadas quando o comando *more* pára no fim de uma página são:

Opção	Descrição
<SPACE>	Mostra a próxima página
q ou Q	Sai de *more*
/partten	Procura pela próxima ocorrência de *pattem*
h	Fornece uma descrição das opções do comando *more*
:p	Pula para o próximo arquivo ou para o começo do arquivo se *more* estiver no meio do arquivo
v	Chama o editor *vi* na linha corrente e volta para o *more* depois de sair do *vi*.

Comando tail

• Usando o comando tail para mostrar as duas últimas linhas do arquivo mang1:

$ tail -2 mang1

A minha nova parceira
Já mandei subir o piano pra Mangueira

• Usando tail para mostrar da 4ª linha até o fim do arquivo mang1:

$ tail +4 mang1

O morro veio me chamar
De terno branco e chapeu de palha
Vou me apresentar
A minha nova parceira
Já mandei subir o piano pra Mangueira

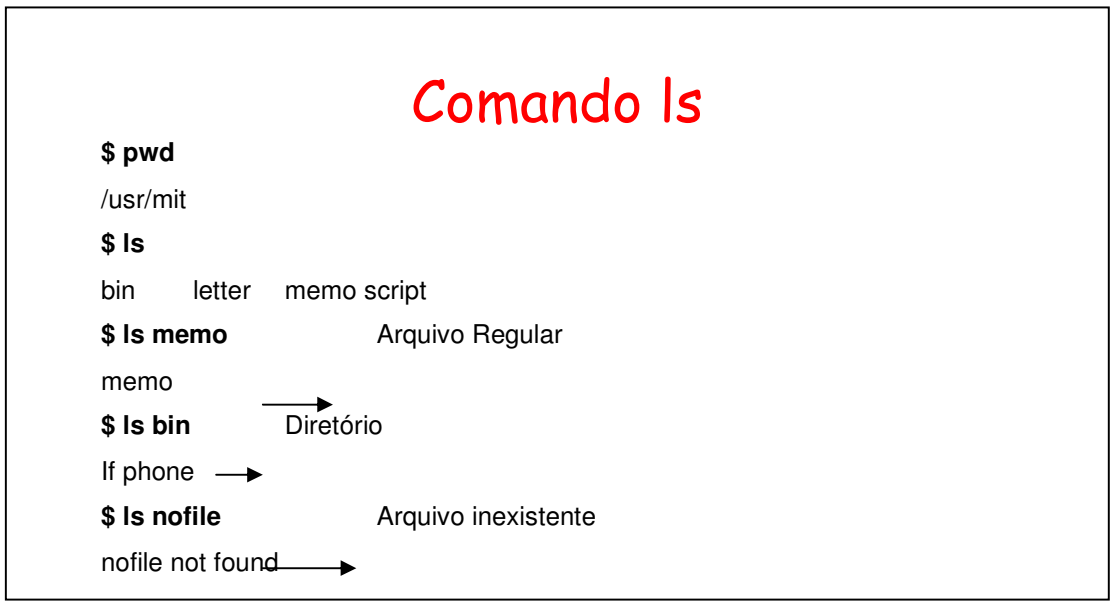
O comando tail

O comando *tail* mostra a última parte de um arquivo na sua tela. Por default, o comando *tail* mostra as últimas 10 linhas de um arquivo. O formato do comando é: tail [+- número] [opções] [arquivo...]

Você pode solicitar a última parte do arquivo para ser mostrada como se uma "marca" do começo ou do fim do arquivo especificando o sinal de mais (+) ou menos (-), seguindo do *número*, onde *número* indica o número de linhas a ser mostrado.

Um argumento da forma *+números* de linhas do começo do arquivo. Por exemplo, o comando *tail +10 arquivo* mostra 11 linhas para o fim do arquivo. Um argumento na forma *-número* mostra *-números* de linhas do fim do arquivo. Por exemplo, o comando *tail -20 arquivo* mostra as últimas 20 linhas do arquivo.

A opção *-f* (follow) no comando *tail* é usado para atualizar continuamente o display quando um arquivo está sendo dinamicamente atualizado. Essa opção é comumente usada para seguir a saída de uma compilação ou monitorar um arquivo de log do sistema. Entre CONTROL-C para terminar o comando *tail* quando usar a opção *-f*.



Listando seus arquivos - O comando *ls*

O comando *ls* (list) mostra informações sobre um ou mais arquivos e diretórios. O formato para o comando é o seguinte:

ls [opções] [arquivo...]

arquivo	Pode ser tanto um arquivo regular, como um arquivo especial ou diretório. Se o arquivo é um diretório, o conteúdo do diretório será mostrado. Se o arquivo é um arquivo regular ou especial, as informações mostrada é para o próprio arquivo. O tipo de informação que será mostrada é determinada pelas opções na linha de comando. Se as opções não forem especificadas, o comando *ls* mostrará uma lista curta e em ordem alfabética que contém somente os nomes dos arquivos.

Os nomes de arquivos como argumentos podem ser compostos de qualquer combinação de arquivos regulares, especiais ou diretórios usando o caminho completo ou relativo (full or relative paths names). Todos os arquivos são listados, exceto arquivos que comecem com ponto (.). Arquivos cujos nomes começam com um ponto são *arquivos escondidos*, porque por default, eles não são listados quando se usa o comando *ls* para listar as informações. Uma opção

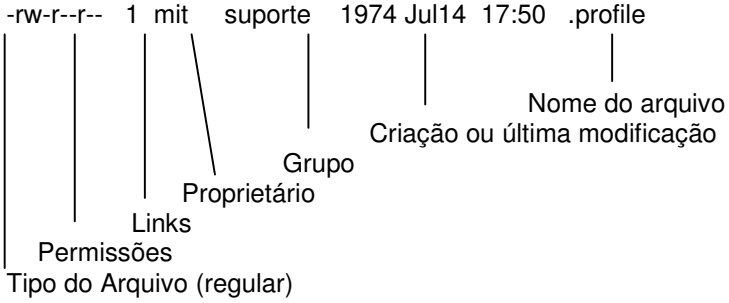
especial para o comando *ls* (-a) é necessária para mostrar os arquivos escondidos. Você pode criar arquivos escondidos, mas geralmente esses arquivos são usados para usos específicos do UNIX. Os arquivos escondidos que são usados para personalizar (customize) seu ambiente de trabalho serão discutidos no capítulo sobre Ambientes do Usuário.

Você examinará as opções de uso mais freqüente do comando *ls* (-l, -a, -d, -F e -C) nas subseções seguintes. Veja a man page para o comando *ls* para obter mais informações sobre outras opções disponíveis.

Comando ls com a opção -l

$ ls -l profile

-rw-r--r-- 1 mit suporte 1974 Jul14 17:50 .profile



O comando *ls* com a opção -l

A opção -l (long format) para o comando *ls* mostra os nomes dos arquivos e suas características para cada arquivo especificado na linha de comando, como segue:

Característica	Descrição
Tipo de arquivo	O primeiro caracter determina o tipo de arquivo: d diretório b arquivo especial do tipo block (disco, fita, etc) c arquivo especial do tipo character (impressora, terminais, modens) l arquivos link - arquivos regulares
Permissões	Essa seqüência de 9 caracteres definem o tipo de permissão que o arquivo tem.
Links	Número de links para o arquivo são indicados por esse número. Cada vez que um arquivo é criado, um link é criado. Usualmente, somente um link existe para cada arquivo regular. Esse link ou ponteiro associa o nome do arquivo com um inode. Cada diretório tem pelo menos dois links, o link para o diretório corrente e outro para o diretório pai. O número de links de um diretório que um diretório contém cresce a medida que forem aumentando o número de seus subdiretórios. Esses links permitem ao sistema localizar a posição de cada diretório no sistema de arquivos.
Owner	Login Id do dono do arquivo
Group	Nome do grupo ao qual o usuário pertence, como definido no arquivo */etc/group* pelo administrador do sistema
Size	Tamanho do arquivo em bytes
Last modification	Data e hora da última modificação
File Name	Nome do arquivo

Comando ls com a opção -a

```
$ pwd
/usr/mit
$ ls -a
.
..
.profile       bin     letter memo       script
$
```

O comando *ls* com a opção *-a*

O comando *ls* com a opção -a lista todas as entradas, incluindo os arquivos escondidos.

Comando ls com opção -d

```
$ ls bin
if
phone
$ ls -d bin
bin
$ ls -ld bin
drwxr-xr-x  2  mit  suporte  98  Aug 15 08:54  bin
```

O comando *ls* com a opção -d

Quando a opção -d é usada, o nome dos diretórios é listado em vez do seu conteúdo. Essa opção é usada mais freqüentemente com a opção -l para obter as características do diretório.

Comando ls com a opção -F

```
$ ls -F
bin/    letter      memo      script*
```

Comando ls com a opção -C

```
$ ls -C
bin         letter      memo      script
```

O comando *ls* com a opção -F

A opção -F coloca uma barra depois de cada diretório e um asterisco (*) depois de cada arquivo executável. Todos os outros arquivos são ou regulares ou especiais.

O comando *ls* com a opção -C

A opção -C lista os arquivos sorteados em colunas verticais. Os arquivos são sorteados em ordem alfabética.

Os exemplos acima mostram como usar o comando *ls* com as opções -a, -d, -F e -C.

Comando cp

mit copia o arquivo script para o arquivo list:

```
$ ls
bin     letter   letter2 letter3 memo script
$ cp scrip list
$ ls
bin     letter           letter2 letter3
list    memo script
```

mit cria o diretório mail e copia o arquivo letter para o diretório mail com o nome de letter1:

```
$ mkdir mail
$ cp letter mail/letter1
$ ls mail
letter1
```

Copiando seus arquivos

As subseções seguintes descrevem os comandos que você pode usar para copiar seus arquivos.

O comando cp

O comando *cp* (copy) copia um ou mais arquivos regulares. Ele tem dois modos de operação. O primeiro copia um arquivo para outro e o segundo copia um ou mais arquivos para um diretório.
O formato para o comando *cp* é:

```
cp file1 [file2...] target
```

Para copiar um arquivo regular em outro arquivo regular, *file1* é o arquivo fonte e *target* é o arquivo de destino. Eles não podem ter o mesmo nome. Se *target* não existe, ele será criado. Se ele existe, todo seu conteúdo será substituído pelo conteúdo *file1*. Se você precisa copiar vários arquivos regulares para outros arquivos regulares, é necessário usar um comando *cp* para cada arquivo.
Para copiar arquivos regulares para um diretório, o último argumento na linha de comando deve ser o nome do diretório. Os arquivos receberão os mesmos nomes dentro do diretório. Quando três ou mais nomes de arquivos são incluídos na linha de comando, o último nome sempre deve ser o nome de um diretório e os outros nomes devem ser nomes de arquivos regulares. Você não pode usar o comando *cp* para copiar um diretório para outro diretório. (Há opções especificas para isso).

Os exemplos mostram como usar o comando *cp* para copiar arquivos.

Comando mv

mit move o arquivo script para o diretório bin, dando-lhe o nome de lister:

```
$ ls
bin letter2      letter3  list      mail      memo   script
$ mv script bin/lister
$ ls
bin      letter2  letter3  list      memo
$ ls bin
ls       lister    phone
```

Movendo seus Arquivos

Nas próximas subseções descrevemos os comandos que você pode usar para mover ou renomear seus arquivos.

O comando *mv*

O comando *mv (move)* é similar ao comando *cp* exceto pelo fato que ele não faz uma cópia dos arquivos; ele move-os (ou renomea-os). Somente uma cópia do arquivo original existirá depois do move. Como o comando *cp*, você pode mover um arquivo regular para outro, ou mover um ou mais arquivos regulares para outro diretório. Você pode também mover ou renomear um diretório se os dois diretórios tem o mesmo diretório pai.
O formato para o comando *mv* é:

```
mv file1 [file2...] target
```

Comando mv

mit move os arquivos letter2 e letter3 para o diretório mail:

$ mv letter2 letter3 mail

$ ls mail

lettr1 letter2 letter3

$ mv list memo bin

$ ls

bin mail

Para mover um arquivo regular para outro, *file1* é o arquivo fonte e *target* é o arquivo de destino. Eles não podem ter o mesmo nome. Como o comando *cp*, se *target* não existe, ele será criado. Se *target* existir, seu conteúdo será substituído pelo conteúdo de *file1*. Para mover um ou mais arquivos para um diretório, *file1, file2 ...* são os nomes dos arquivos fontes e *target* é o nome do diretório de destino. Para renomear um diretório, *file1* é o nome do diretório que existe e *target* é o nome do novo diretório. Quando três ou mais arquivos estão na linha de comando, o último nome deve ser um diretório e todos os outros devem ser arquivos regulares.

Os exemplos mostram como usar o comando *mv* para mover e renomear arquivos.

Categorias de Acesso

- **User** Dono do Arquivo
- **Grupo** Nome do grupo
- **Outros** Todos os outros usuários do sistema

Modos de Acesso

- **Read** Permite a leitura do conteúdo do arquivo
- **Write** Permite criar, modificar ou apagar o conteúdo do arquivo
- **Execute** Permite executar o arquivo, dependendo

Proteção de seus Arquivos

A seguir identificaremos os três tipos de permissão padrão do UNIX e examinaremos como os arquivos podem ser protegidos. Depois disto, você terá condições de ver uma lista de arquivos e interpretar as permissões de acesso de cada arquivo e diretório e poderá modificar as permissões de acesso de seus arquivos.

Proteger seus arquivos significa que você pode restringir seu acesso a outros usuários. Você pode restringir o acesso a seus arquivos de forma que só você pode acessá-los. Todavia, quando você restringe o acesso aos seus arquivos, você está limitando o fluxo de informação. O UNIX foi desenvolvido para fornecer um ambiente multiusuário no qual as informações e recursos fossem facilmente compartilhados. Você deve pesar os efeitos contrários que pode provocar ao restringir o acesso aos seus arquivos.

Categorias de Acesso

Existem três categorias de usuários no sistema UNIX:

Categoria	**Descrição**
User	Login ID do proprietário do arquivo
Group	Nome do grupo ao qual o proprietário do arquivo pertence. O administrador do sistema atribui seu grupo no arquivo */etc/group*
Outros	Consiste de todos os outros usuários do sistema

Modos de Acesso

Três modos de permissão de acesso estão disponíveis:

Categoria	**Descrição**
Read	Deixa você ler o conteúdo dos arquivos
Write	Deixa você escrever, modificar ou remover o conteúdo do arquivo
Execute	Deixa você acessar ou executar o arquivo, dependendo do tipo do arquivo.

Mostrando Permissões de Acesso

```
$ ls -l
-rw-r--r--   1   mit   suporte   258   Aug   5   10:20  job
drwxr-xr-x 2   mit   suporte     24   Jul    20   08:54  bin
```

Examinado as permissões no arquivo job:

```
rw-r--r- -
|    | | |
|    |outros: permissão de leitura somente
|    |
| grupo: permissão de leitura somente

usuário: permissão de leitura e escrita
```

Mostrando a permissão de seus arquivos

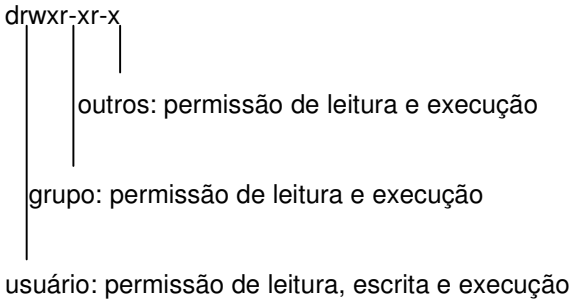
Para mostrar a permissão de seus arquivos, use o comando *ls*. Você deve usar a opção *-l (long)* de forma que as permissões sejam mostradas. Use a opção *-d* com a opção *-l* quando você quiser ver as permissões de um determinado diretório.

Olhe os exemplos de saída do comando *ls -l* na próxima página enquanto lê as seguintes informações:

- A primeira seqüência de 10 caracteres fixa o tipo e permissões do arquivo.
- O primeiro caracter indica o tipo de arquivo. Olhe na linha para o arquivo *job*. O primeiro caracter é um hífen (-), que indica que *job* é um arquivo regular. O primeiro caracter na linha para o arquivo *bin* é uma letra *d,* que significa que *bin* é um diretório.

Mostrando Permissões de Acesso

Examinando as permissões do diretório bin



- Os próximos 9 caracteres são as permissões dadas ao arquivo. Esses caracteres são divididos em três conjuntos de 3 caracteres cada. Cada conjunto determina as permissões para usuário, grupo e outros, respectivamente. Dentro de cada conjunto, os 3 caracteres significam permissões de read, write e execute. O caracter *r* representa permissão de leitura, o caracter *w*, permissão para escrever e o caracter *x*, execução. Os caracteres aparecem dentro de cada conjunto nessa ordem. Para cada permissão que não é encontrada ou proibida, um hífen (-) aparece em seu lugar.
- Cada um desses tipos de permissão tem um efeito diferente sobre um arquivo, dependendo do seu tipo.
- As permissões podem ser definidas com qualquer combinação de read, write e execute. O comando usado para alterar as permissões, *chmod,* será discutido mais adiante.

Permissões de Arquivos e Diretórios Regulares

- Os comandos abaixo do diretório bin devem ter permissões de leitura e execução:

$ ls -l /bin/date

-rwxr-xr-x 1 bin sys 22528 Jan 10 1990 date

- O diretório /bin deve ter permissão de leitura e execução para que os usuários possam ter acesso ao diretório:

$ ls -ld /bin

drwxr-xr-x 2 bin sys 2048 Jul 18 17:18 bin

Permissão de Arquivos Regulares

Para um arquivo regular, as permissões de read permitem que você leia o conteúdo do arquivo como entrada para um comando. Por exemplo, se você não tivesse permissão de read em um arquivo, você não poderia usar o comando *cat* para mostrar esse arquivo. A seguinte mensagem de erro deveria ser mostrada:

filename: permission denied

Quase todos os comandos que usa um arquivo existente como argumento precisa de permissão de read.

As permissões de write permitem que você modifique o conteúdo de um arquivo. Quando você está editando um arquivo existente, você deve ter permissão de write para salvar qualquer alteração feita no arquivo.

As permissões de execute permitem que você execute o conteúdo do arquivo como um comando. Por exemplo, se você compilou um programa em C e gostaria de executar o arquivo binário *a.out* que foi criado, você deve ter permissões de execute naquele arquivo. *Shells scripts* são arquivos que contém comandos UNIX que podem ser executados quando você entra com o nome do script na

linha de comando. Arquivos de shell script também precisam ter permissão de execute para que possam ser executados.

Permissão de Diretórios

Para diretórios, a permissão de read deixa você listar o conteúdo do diretório com o comando *ls.A* permissão de write permite que você modifique o conteúdo do diretório criando novos arquivos, modificando ou removendo qualquer arquivo naquele diretório. Para remover um arquivo usando o comando *rm* você deve ter permissão de write no diretório corrente, mas não precisa ter permissão de write no arquivo.

A permissão de execute permite que você mude do seu diretório corrente para outro usando o comando *cd*. Por exemplo, para mudar seu diretório corrente para outro diretório, o outro diretório deve ter permissão de execute. Quando um caminho relativo ou completo é usado com um comando, você deve ter permissão de execute em cada diretório no caminho para que possa acessar o arquivo.

Permissões de Arquivos Especiais

- No exemplo a seguir, mit usa o comando mesg proibir, e depois permitir, a permissão de escrita no device de seu terminal:

```
$ who am i
mit       tty03    Sep 16 11:17
$ ls -l /dev/tty03
crw--w--w-   1   mit                5,3 Sep 16 13:04 tty03
$ mesg n
$ ls -l /dev/tty03
crw-------     1   mit                5,3 Sep 16 13:05 tty03
$ mesg y
$ ls -l /dev/tty03
crw--w--w-   1   mit          5,2 Sep 16 13:05 tty03
```

Permissão para Arquivos especiais

Os arquivos especiais seguem as regras das permissões de arquivos regulares. Ou seja, read permite que você leia de um periférico. Write permite que você escreva em um periférico e execute permite que você execute o arquivo do periférico.

O exemplo na página acima mostra a colocação de permissões em arquivos especiais. Quando você se conecta a um sistema UNIX, é designado um arquivo de periférico para o seu terminal (tty). Você torna-se o proprietário daquele periférico e pode mudar as suas permissões. Quando você entra com o comando *mesg n*, as permissões do periférico são modificadas. A permissão de write para o periférico *tty* é removido para grupos e outros. A permissão de write pode ser restabelecida com o comando *mesg y*.

Comando chmod-Formato Simbólico

Os exemplos a seguir mostram como mudar as permissões de arquivos, usando o formato simbólico:

$ chmod u+x *arquivo*	Adiciona permissão de execução para o usuário (owner)
$ chmod go-w *arquivo*	Tira a permissão de escrita para grupo e outros
$ chmod -x *arquivo*	Tira todas as permissões de execução
$ chmod go=rx *arquivo*	Dá permissão só de leitura e execução (sem escrita) para grupo e outros
$ chmod o= *arquivo*	Tira todas as permissões para outros
$ chmod u+x,-w *arquivo*	Adiciona permissão de execução para o usuário e tira a permissão de escrita para o usuário, grupo e outros

O comando *chmod*

O comando *chmod* muda o modo de permissão dos arquivos e diretórios. Somente o proprietário de um arquivo ou o superusuário (root) pode alterar as permissões de um arquivo.
As permissões dos arquivos são alteradas de acordo com o modo especificado. O modo pode ser expresso ou em um formato simbólico ou em um formato absoluto. A maioria dos usuários acham o formato simbólico mais fácil de usar.

Formato Simbólico

O formato *simbólico* tem a seguinte sintaxe:

chmod [who] op *permission* file [file...]

Who - Qualquer combinação das letras *u, g, o* e *a*, que especificam o conjunto de modos a ser alterado:

- *u* User (proprietário)
- *g* Grupo
- Outros
- *a* Todos conjuntos (all = user, groups and others)

Se *who* for omitido, o default é *ugo* ou *a* (all)

Op - Os caracteres +, - ou = são usados para especificar adição, eliminação ou colocação exata de uma lista de permissões.

permission Uma combinação das letras *r* (read), *w* (write) ou *x* (execute) representando as permissões a serem alteradas.

Comando chmod-Formato Absoluto

Os exemplos a seguir mostram como mudar as permissões de arquivos, usando o formato absoluto:

```
$ ls -l temp
-rw-r--r--  1  mit  suporte  270 Sep 16 12:34 temp
$ chmod 444 temp
$ ls -l temp
-r--r--r--  1  mit  suporte  270  Sep 16 12:34 temp
$ chmod 640 temp
$ ls -l temp
-rw-r----  1  mit  suporte  270  Sep 16 12:34 temp
$ chmod 777 temp
$ ls -l temp
-rwxrwxrwx  1  mit  suporte  270  Sep 16 12:34 temp
$ chmod 644 temp
$ ls -l temp
-rw-r--r--    1  mit  suporte  270  Sep 16 12:34 temp
```

Formato Absoluto

O *formato absoluto* é um número octal de 3 dígitos. Um dígito octal representa cada conjunto de permissões (usuário, grupo e outros). Se você sabe como representar os números octal de 0-7 em forma binária, a seqüência octal pode ser uma forma mais fácil para você usar. Cada ocorrência das letras *r, w* e *x* representa um (1) binário, um *hífen (-)* representa um zero binário (0).

chmod *mode file [file...]*

Exemplo:

r w x r w - r - -

1 1 1 1 1 0 1 0 0 ----------> Representação binária

7 6 4 ----------> Representação Octal

$chmod 764 file

O comando *chmod* coloca as permissões de *file* iguais as permissões do exemplo anterior.

rwxrw-r--

Uma forma simples de olhar o formato absoluto e associar os valores numéricos com cada modo de acesso:

read	4
write	2
execute	1

Quando você usa o formato absoluto, você está colocando as permissões exatamente como indicado pela representação octal ou binária. A correspondência entre as permissões é a seguinte:

000	= 0	---	Sem permissão
001	= 1	--x	Permissão de execução
010	= 2	-w-	Permissão de escrita
011	= 3	-wx	Permissão de escrita e execução
100	= 4	r--	Permissão de leitura
101	= 5	r-x	Permissão de leitura e escrita
110	= 6	rw-	Permissão de leitura e escrita
111	= 7	rwx	Permissão de leitura, escrita e execução

Comando umask

$ umask 022

Quando criamos uma máscara usando o comando umask, como no exemplo, acontecerá o seguinte quando criamos um arquivo ou diretório:

1. A forma binária é negada:

   000 010 010 torna 111 101 101

2. O valor negado é adicionado aos valores default do sistema:

Arquivo regular default	110 110 110
umask negado	111 101 101
Resultado	rw- r-- r--

O comando *umask*

Quando você cria uma arquivo no sistema, um conjunto de permissões default é aplicado ao arquivo, que estão então "mascarado" com a máscara de criação de arquivos do usuário *(umask)*. (Quando você copia ou move um arquivo, as permissões anteriores do arquivo fonte são preservadas no arquivo de destino). O propósito primário do comando *umask* é ter permissões específicas colocadas automaticamente sempre que você cria um novo arquivo ou diretório. Você pode ainda mudar as permissões manualmente através do comando *chmod*.

As permissões default do sistema para um arquivo regular são *rw-rw-rw-* ou *666*. As permissões default para um diretório são *rwxr-xr-x* ou *755.*

Seu administrador de sistema pode ter colocado um comando *umask* no seu arquivo de iniciação; portanto, as permissões colocadas quando você cria um arquivo ou um diretório podem variar dos valores default. Arquivos de iniciação serão discutidos em mais detalhes no capítulo. "O ambiente do usuário".

Cada usuário pode criar um arquivo de criação de mascara que modifica as permissões default para todos os seus arquivos e diretórios criados subseqüentemente usando o comando *umask.*

Comando umask

Diretório default	111 111 111
umask negado	111 101 101
Resultado	rwx r-x r-x

O formato para o comando *umask* é:

```
umask [000]
```

Cada dígito octal representa a máscara para proprietário, grupo ou outros usuários respectivamente. O comando *umask* usa os dígitos octais (na forma binária) e nega-os. Isso significa que todos os binários "uns" tornam-se "zeros" e "zeros" tornam-se "uns". O comando *umask* faz então um "logical and" com o default do sistema.

Sem qualquer argumento, o comando *umask* mostra o valor corrente da máscara do usuário.

Exercícios:

1. Vá para o seu diretório home e faça uma lista longa de seus arquivos. Você vê a entradas para *ponto (.)* e *ponto ponto (..)*?
2. Faça uma longa lista de seus arquivos incluindo qualquer arquivo escondido.
3. Crie um arquivo chamado **small** usando o comando **cat** e coloque uma linha de texto no arquivo. Crie outro arquivo usando o comando **cat** para listar o arquivo **/etc/hosts**, direcione a saída para o arquivo **large**.
4. Imprima em sua tela o conteúdo dos arquivos **small** e **large** um por vez. Imprima o conteúdo do arquivo **large** usando o outro comando de display estudado nesta sessão.
5. Faça uma longa lista somente do arquivo **small** . Observe as permissões do arquivo **small**. Quais são as permissões para usuário, grupo e outros?
6. Copie o arquivo **small** para um arquivo chamado **test**. Liste seus arquivos para ver o resultado.
7. Crie um diretório chamado **sec5**. Mova os arquivos **small** e **large** para o diretório **sec5**. Liste seus arquivos para certificar-se que eles não estão mais em seu diretório corrente. Faça uma longa lista do diretório **sec5** e observe as permissões de acesso do diretório. Agora liste o conteúdo do diretório **sec5**.
8. Mude o nome do arquivo **test** para **small2**.
9. Olhe as permissões de acesso do arquivo **small2**. Mude as permissões de acesso do arquivo **small2** para leitura, escrita e execução para usuário; leitura e execução para grupo e somente leitura para outros (rwxr-xr--). Faça uma longa lista para verificar as mudanças.
10. Imprima o valor corrente de seu **umask**. Mude o seu **umask** para 077, então imprima novamente o valor de seu **umask**. Crie um arquivo chamado **file1** com uma linha de texto e então crie um diretório chamado **dir1**. Verifique as permissões para o novo arquivo e diretório e observe se elas estão diferentes daquelas do arquivo **small2** e do diretório **sec5**.
11. Use o comando **more** para listar o conteúdo do arquivo **/etc/passwd**. Procure pela linha que contém seu login ID.
12. Veja o manual online para o comando **pr**. Use o comando **pr** para ver o conteúdo do arquivo **/etc/passwd**. Observe que as informações listadas são diferentes daquelas do comando **more**.

13. Veja o manual online para o comando **ls**. Use algumas das outras opções disponíveis para listar os seguintes arquivos e diretórios e seus conteúdos:

/etc
/etc/motd
/etc/group
/bin
/bin/ls

14. Crie um diretório abaixo do seu diretório home chamado **sec5_test**. Copie o arquivo **/etc/motd** para o novo diretório, dando-lhe o nome de **pratica**. Sem mudar de diretório, faça duas cópias deste arquivo no diretório **sec5_test**, chamando-os de **pratica1** e **pratica2**.

15. Verifique as permissões de acesso do diretório **sec5_test**. Mude seu diretório corrente para o diretório **sec5_test**. Verifique as permissões de acesso dos arquivos **pratica**, **pratica1** e **pratica2**.

Entre com o comando:

$ chmod 777 *

Olhe novamente as permissões de acesso dos arquivos. O que aconteceu? Use o caracter especial asterisco (*) e o comando **rm** para mover todos os arquivos no diretório **sec5_test**.

Agora remova o diretório **sec5_test**.

Solução:

1. $ cd ; ls -l
2. $ ls -la
3. $ cat > small <RETURN>
Este eh o arquivo chamado small. <RETURN>
^d
$ cat /etc/hosts > large
4. $ cat small
$ cat large
$ more large
$ tail large
5. $ ls -l small
6. $ cp small test
$ ls
7. $ mkdir sec5
$ mv small large sec5
$ ls
$ ls -ld sec5
$ ls -l sec5
8. $ mv test small2
9. $ ls -l small 2
$ chmod 754 small2 (ou $ chmod u=rwx, g=rx, o=r small2)
$ ls -l small2
10.$ umask
$ umask 077
$ umask
$ cat > file1 <RETURN>

Este eh o arquivo chamado file1. <RETURN>

^d

$ mkdir dir1

$ ls -l file1

$ ls -ld dir1

11.$ more /etc/passwd

/seu_userid

/h

q

12.$ man pr

$ pr /etc/passwd

13.$ man ls

14.$ mkdir sec5_test

$ cp /etc/motd sec5_test/pratica

$ cp sec5_test/pratica sec5_test/pratica1

$ cp sec5_test/pratica sec5_test/pratica2

15.$ ls -ld sec5_test

$ cd sec5_test

$ ls -l

$ chmod 777*

$ ls -l

(Todos os arquivos no diretório sec5_test ficaram com permissão de read, write e execute).

$ rm*

$ cd ..

$ rmdir sec5_test

========== F I M ==========