UNIVERSIDADE DE SÃO PAULO

CIRP

CENTRO DE INFORMÁTICA DE RIBEIRÃO PRETO

SISTEMA OPERACIONAL LINUX

MSC. ENG. ALI FAIEZ TAHA

Sumário

Introdução

O Linux é um clone UNIX de distribuição livre para PCs baseados em processadores 386/486/Pentium, etc.

É uma implementação independente da especificação POSIX, com a qual todas as versões do UNIX padrão estão convencionadas.

O Linux foi primeiramente desenvolvido para PCs baseados em 387/486/Pentium, mas atualmente também roda em computadores Alpha da DEC, Sparcs da SUN, máquinas M68000 (semelhantes a Atari e Amiga), MIPS e PowerPCs.

O Linux foi inteiramente escrito do nada, não há código proprietário em seu interior.

O Linux está disponível na forma de código aberto, bem como em código fonte.

Pode ser livremente distribuído nos termos da GNU (Generic Public License).

Possui todas as características de um Sistema Operacional UNIX.

A maioria dos programas rodando em Linux são freeware genéricos para UNIX, muitos provenientes do projeto GNU.

O Linux está disponível através da Internet por meio de centenas de sites FTP.

O Linux está sendo usado para desenvolvimento de Softwares, networking e Desktop. É uma alternativa efetiva de custo em relação aos caros sistemas UNIX existentes.

História do Linux

O Kernel do Linux foi, originalmente, escrito por Linus Torvalds, do Departamento de Ciência da Computação da Universidade de Helsinki, Finlândia, com a ajuda de vários programadores voluntários através da Internet.

Linux Torvalds iniciou cortando (Hacking) o Kernel como um projeto particular, inspirado em seu interesse no Minix, um pequeno Sistema UNIX desenvolvido por Andy Tannenbaum. Ele se limitou a criar, em suas próprias palavras, "Um Minix melhor que o Minix" ("a better Minix than Minix"). E depois de algum tempo de trabalho em seu projeto, sozinho, ele enviou a seguinte mensagem para a lista comp.os.minix :

Você suspira por melhores dias do Minix-1.1, quando homens são homens e escreverão seus próprios "device drivers" ?

Você está sem um bom projeto e está morrendo por colocar as mãos em um S.O. no qual você possa modificar de acordo com suas necessidades ? Você está achando frustrante quando tudo trabalha em Minix ? Chega de atravessar noites para obter programas que trabalham corretamente ? Então esta mensagem pode ser exatamente para você.

Como eu mencionei a um mês atrás, estou trabalhando em uma versão independente de S.O. similar ao Minix para computadores AT-386. Ele está, finalmente, próximo do estágio em que poderá ser utilizado (embora possa não ser o que você esteja esperando), e eu estou disposto a colocar os fontes para ampla distribuição. Ele está na versão 0.02... contudo eu tive sucesso rodando bash, gcc, gnu-make, gnu- sed, compressão, etc. nele.

1 Sistema Operacional UNIX

O UNIX foi desenvolvido por Bell Laboratories.

Como todos os sistemas operacionais, o UNIX controla as atividades e recursos de um computador, interpretando os comandos que voce digita no teclado e traduzindo-os em ações dirigidas a alguma parte do computador, como a memória, CPU, disco, impressora. O UNIX coordena várias atividades do computador ao mesmo tempo, permitindo que uma pessoa imprima um documento enquanto outra executa um programa de verificação de ortografia, outra digita dados e assim por diante. O sistema operacional canaliza todos os comandos digitados em diferentes teclados por diferentes usuários e todo fluxo de informações gerado pela execução de vários programas. O sistema operacional mantém o computador organizado, evitando que os comandos de um usuário ou programa interfiram nos comandos de um outro usuário ou programa.

Projetado em 1969, o sistema UNIX tinha originalmente a intenção de propiciar um ambiente no qual os programadores pudessem criar programas. Logo ficou evidente que o UNIX também propiciava um ambiente no qual usuários da área comercial, científica e industrial pudessem executar programas para ajudá-los em seu trabalho. O UNIX foi originalmente desenvolvido para microcomputadores de tamanho médio (especificamente a série PDP da companhia DEC), e mais tarde passou a ser usado também em grandes e potentes computadores de grande porte e microcomputadores. A Bell Laboratories têm freqüentemente apresentado novas versões de UNIX ao longo desses anos. Toda vez que uma nova versão é publicada, inclui novas características, tanto para programadores como para usuários. Consequentemente, existem várias versões do UNIX.

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

1.1 Diferenças entre o UNIX e outros Sistemas Operacionais

O objetivo de todos os sistemas operacionais é mais ou menos o mesmo : controlar as atividades de um computador. Os sistemas operacionais diferem na maioria como eles fazem o seu trabalho e nas características adicionais que oferecem. O UNIX é único em sistema modular, que permite aos usuários acrescentar ou remover partes para adaptá-las às suas necessidades específicas. Os programas em UNIX são como peças de um quebra-cabeça; os módulos se encaixam com conexões padrão. Você pode tirar um módulo e substituí-lo por um outro ou expandir o sistema acrescentando vários módulos. Essa característica é especialmente útil nas implementações de microcomputadores, onde as unidades de disco têm capacidade limitada; a remoção de programas desnecessários abre espaço para mais arquivos de dados.

1.2 Características do UNIX

- Capacidade multitarefas
- Capacidade multiusuário
- Transportabilidade
- Ampla seleção de potentes programas
- Comunicações e correio eletrônico
- Biblioteca de softwares aplicativos

1.3 Capacidade Multitarefas

Multitarefa significa executar mais que uma tarefa ao mesmo tempo, por exemplo, falar ao telefone ao mesmo tempo que fazer anotações. Você pode estar usando uma calculadora, o que significa estar fazendo três coisas de uma só vez. Fazer mais que uma coisa ao mesmo tempo em um computador também é multitarefa. Quando você executa dois programas ao mesmo tempo, como a classificação de uma relação de endereços em ordem alfabética e a edição de uma carta que será enviada aos nomes constantes da lista de endereços, o computador está realizando uma operação multitarefa.

A multitarefa em um computador permite que você execute simultaneamente tarefas que anteriormente teriam de ser executadas seqüencialmente. O conjunto de tarefas não só é executado mais rapidamente como também você e o computador ficam livres para fazer outras coisas com o tempo economizado.

1.4 Capacidade Multiusuário

Um sistema multiusuário permite que vários usuários usem o computador simultaneamente. Mais de um terminal (teclado e tela) pode ser conectado a um computador, e os usuários de todos os terminais podem executar programas, acessar arquivos e imprimir documentos de uma só vez. O sistema opercional gerencia os pedidos que os vários usuários fazem ao computador, evita que um interfira no outro e atribui prioridades quando duas ou mais pessoas querem usar o mesmo arquivo ou a mesma impressora simultaneamente.

A capacidade multiusuário economiza tempo na medida em que permite que mais de uma pessoa trabalhe com um conjunto de informações ao mesmo tempo.

1.5 Portabilidade do Sistema UNIX

O sistema UNIX em si é muito portátil. Isto é, é mais fácil modificar o código do sistema UNIX para implementá-lo em um novo computador do que reesecrever um novo sistema operacional para um novo computador. A facilidade de transportar o sistema UNIX de uma marca de computador para outra tem sido uma das razões principais de sua aceitação. O sistema UNIX executa em mais marcas e tipos de computadores, grandes ou pequenos, do que em qualquer outro sistema operacional.

1.6 Ferramentas oferecidas pelo Sistema UNIX

O Sistema UNIX vem com centenas de programas que podem ser divididos em duas classes :

- Utilitários Integrados
- Ferramentas

Os utilitários integrados são partes do Sistema UNIX que fornecem tal assistência ao sistema opercional que são absolutamente necessários para a operação prática do computador. Um exemplo é o interpretador de comando do sistema UNIX, ou programa shell. Sem esse programa você não poderia pedir que seu sistema UNIX realizasse qualquer trabalho. As ferramentas, por outro lado, são programas que não são necessários à operação básicoa do computador mas que fornecem recursos adicionais significativos so UNIX. Essas ferramentas incluem vários programas aplicativos, alguns dos quais podem ser adquiridos separadamente, como planilhas eletrônicas, softwares gráficos e outros.

Os utilitários integrados são utilitários que deixam você preparar sofisticados procedimentos automáticos para realizar uma série de tarefas, que de outra forma teriam de ser efetuadas como váriações separadas.

1.7 Comunicações em UNIX

O Software de comunicação é parte do sistema UNIX. As comunicações em UNIX incluem as seguintes opções :

- Comunicação entre diferentes terminais ligados ao mesmo computador.
- Comunicação entre usuários de um computador e usuários de um outro computador no mesmo escritório.
- Comunicação entre computadores de diferentes tipos e tamanhos em lugares diferentes.

A forma mais simples de comunicação eletrônica dá -se entre dois usuários usando terminais ligados a um mesmo computador. Usando o correio eletrônico ou um programa de transmissão direta ao terminal, eles podem enviar mensagens lembrando um ao outro reuniões ou pedindo informações. Se dois computadores UNIX, mesmo de marcas diferentes, estiverem conectados com os utilitários UNIX, os usuários destes podem enviar correspondência e trocar arquivos como se estivessem no mesmo computador. O sistema UNIX pode também usar linhas telefônicas para efetuar a comunicação entre computadores em localidades diferentes.

1.8 Programas aplicativos de terceiros

Além dos programas aplicativos fornecidos por Bell Laboratories, uma biblioteca de muitos programas aplicativos em UNIX foi desenvolvida e é vendida por "terceiros" - fabricantes de computadores e empresas de software. Esses programas não são parte do sistema UNIX básico, mas podem ser adquiridos separadamente e depois executados de acordo com suas necessidades.

2 Estrutura do UNIX

O UNIX é um Sistema Operacional amplo e complexo que roda em uma grande variedade de plataformas de hardware, dos Pcs aos computadores de grande porte. Ele contém literalmente centenas de comandos com milhares de opções associadas. Além disso, o UNIX tem uma estrutura consistente que funciona em qualquer plataforma de hardware. Para o usuário final ou para o administrador de sistema é muito importante entender sua estrutura básica.

2.1 Sistema de Hardware para o UNIX

Devido à extraordinária variedade de plataformas de hardware em que o UNIX roda, não existe um sistema de hardware característico para ele. Por outro lado, há certos componentes de hardware que são encontrados em todos os sistemas UNIX. Com alguns desses componentes, como a unidade de sistema, não são possíveis de se lidar diretamente. Os principais componentes são descritos a seguir :

- A Unidade de sistema contém a unidade de processamento central do sistema, ou CPU (Unidade Central de Processamento do computador) e os driver's, ou unidades acionadoras de disco, de Hard Disk, de CD ROM's, etc.
- Dispositivos de Armazenamento utilizados para Backup, como um drive de disco flexível ou fita magnética, para se copiar o software e os arquivos de sistema de e para o computador, geralmente para se fazer cópias backup dos arquivos ou sistema de arquivos para o caso de eles serem acidentalmente removidos ou destruídos.

- O console é um terminal, com um monitor e um teclado conectados diretamente à unidade do sistema. As mensagens de erro do sistema são exibidas por este terminal. A administração e operação do sistema geralmente são controladas a partir do console, embora também funcione como um terminal de usuário comum. O console pode ser uma parte da unidade de sistema no caso das estações de trabalho ou microcomputadores pessoais.

- As Linhas de comunicação conectam o sistema a computadores de grande porte ou a outros sistemas com base UNIX, conectam-se às redes de computadores. O UNIX pode estar ou não conectado a outros computadores.

- Os terminais de usuário consistem em um teclado e um monitor de vídeo e são conectados à unidade de sistema diretamente por redes de computadores, por meio de modems e de linhas telefônicas. A comunicação é feita através do teclado do terminal. Pode-se utilizar diversos tipos de terminais com os sistemas UNIX, desde os terminais ASCII, microcomputadores pessoais, até as estações de trabalho mais poderosas.

- As impressoras produzem cópias impressas de textos, gráficos, figuras, etc. É possível se usar vários tipos de impressoras, desde as mais simples, de impacto (como as matriciais), até impressoras a laser e fotocompositoras.

2.2 Componentes de Software

A execução de um sistema operacional envolve muitas tarefas, cada uma é controlada por um componente do software UNIX. Por exemplo: um sistema UNIX pode dar suporte a muitos usuários ao mesmo tempo, cada um rodando um programa diferente - multitarefa . O software que gerencia esse controle não é chamado pelo usuário, mas executado automaticamente sempre que o UNIX é ativado. Para realizar tarefas específicas, os usuários precisam utilizar outros softwares, como processadores e formatadores de textos, calculadoras matemáticas e programas de comunicação entre usuários.

O kernel é o núcleo do sistema UNIX. O kernel se comunica com o sistema de hardware e, portanto, deve ser adaptado à arquitetura específica de cada plataforma de hardware. É o kernel que isola os usuários das diferenças dos vários tipos de hardware.

Além de se comunicar com o hardware, o kernel coordena as muitas funções internas do sistema operacional. Uma dessas funções é a alocação de memória e de outros recursos do sistema para os processos que estão sendo executados em determinado momento. Como o UNIX é um sistema operacional multiusuário e multitarefa, o kernel deve lidar com todo o escallonamento das operações e gerenciamento de memória envolvidos em um ambiente tão complexo. O kernel assegura que muitos programas ou aplicações possam rodar ao mesmo tempo sem que interfiram uns nos outros.

O kernel também mantém o controle do uso do sistema pelos usuários e do conteúdo e localização de todos os arquivos dentro do sistema de arquivos. Uma vez que o shell tenha traduzido os comandos do usuário para instruções que o computador possa realizar, é o kernel que realiza essas instruções. O kernel também mantém registros da atividade do sistema e do uso que cada usuário faz do sistema para fins de contabilidade.

O shell é um programa que conecta e interpreta os comandos digitados por um usuário, chama programas em memória e executa-os individualmente ou em uma seqüência chamada pipe, é uma parte do sistema operacional que funciona como uma ligação entre os comandos que você digita e as atividades que o kernel pode realizar. Quando um comando é digitado, o shell traduz o nome daquele comando no conjunto de chamadas de sistema, em linguagem de máquina, que são as instruções necessárias para que o kernel realize aquela tarefa. Cada comando no UNIX e´, na verdade, apenas um nome fácil de memorizar para cada conjunto específico de chamadas de sistema. O propósito do shell é tornar o sistema

operacional mais amigável. É muito mais fácil para os usuários lembrar dos nomes dos comandos do que o conjunto de chamadas de sistema que estão por trás dele.

A maioria dos sistemas UNIX dispõe de mais de um shell - por exemplo, o Bourne Shell, o C shell e o Korn shell. Cada um tem características ligeiramente diferentes e funções especiais. O sistema permite que se escolha qual shell a utilizar.

A **figura 1** mostra as partes do sistema UNIX: o kernel (núcleo) , o shell (casca) e as ferramentas e aplicativos.

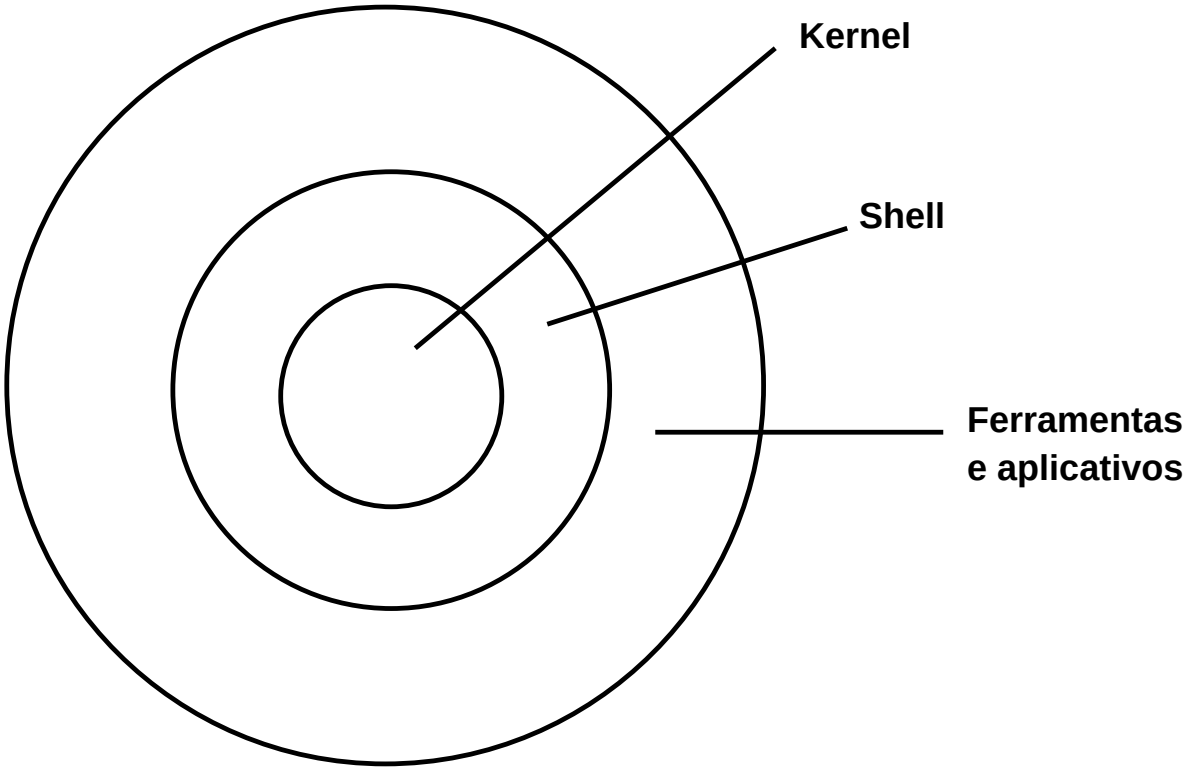

Figura 1

O shell também contém o recurso de encadeamento de comandos, ou piping (canalização). Um arquivo de dados pode ser enviado através de um pipeline (encanamento) com paradas ao longo do caminho, onde programas diferentes agem no caminho. Um único comando de encadeamento pode fazer com que um arquivo de dados seja processado por vários programas seqüencialmente. A saída de um programa flui pelo pipe e se torna a entrada do programa seguinte. Por exemplo, um arquivo de endereços poderia ser enviado no começo do pipeline, ser classificado na primeira parada pela coluna do CEP, ser combinado com um modelo de carta na parada seguinte e copiado, essa cópia sendo então enviada a um local de armazenamento na outra parada ferramentas e impressa na parada final.

2.3 Utilitários do UNIX

Os utilitários do UNIX (frequentemente chamados comandos) são programas que realizam tarefas específicas. Como já foi mencionado, existem centenas de utilitários diferentes do UNIX, embora nem todos estejam disponíveis em determinados sistemas.

Os mais utilizados são:

- Edição de textos.
- Formatação de textos.

- Verificação Ortográfica.
- Cálculos Matemáticos.
- Gerenciamento de arquivos e diretórios.
- Administração de sistemas.
- Manutenção de arquivos e da segurança do sistema.
- Envio de saída para a impressora.
- Desenvolvimento de programas.
- Filtragem de dados.

Como os utilitários realizam uma enorme variedade de funções, muitos usuários se restringem a eles, mas o usuário pode criar seus próprios programas se quiser realizar tarefas mais complexas ou especializadas.

3 Sistemas de Arquivos

Os dados são reunidos em grupos chamados arquivos. Há vários tipos de arquivos :

- **Arquivos de texto comuns**, em que os usuários armazenam os dados.
- **Arquivos de programas**, que são os arquivos em código de máquina.
- **Arquivos especiais**, que controlam os vários dispositivos do sistema, como terminais e impressoras.

Nos sistemas operacionais UNIX, cada arquivo é gravado dentro de um diretório, sendo que os diretórios podem conter vários arquivos e também outros diretórios. Os sistemas de diretórios e arquivos são chamados de sistemas de arquivos. Cada sistema UNIX contém pelo menos um sistema de arquivos. Com os arquivos organizados em um sistema, fica muito mais fácil encontrar um determinado arquivo.

O diretório principal que contém todos os outros diretórios e arquivos é chamado diretório-raiz. No diretório raiz estão os diretórios que contêm as áreas de trabalho do usuário e os arquivos em código de máquina necessários para rodar os utilitários UNIX.

O sistema Unix de arquivos hierárquicos permite que se prepare índices para o grande número de arquivos de dados que geralmente são acumulados nos computadores. Funciona também como a estrutura básica da qual se desloca de uma área de trabalho para outra. É a chamada árvore de diretórios. Esta árvore apresenta os galhos de uma árvore de cabeça para baixo, figura 2.Na parte superior está a raiz (root), ou o primeiro diretório do qual um segundo nível de diretórios derive. O nível inicial de diretórios é padrão na maioria dos sistemas UNIX. Cada u desses diretórios contém um segmento funcional importante do sistema operacional UNIX.

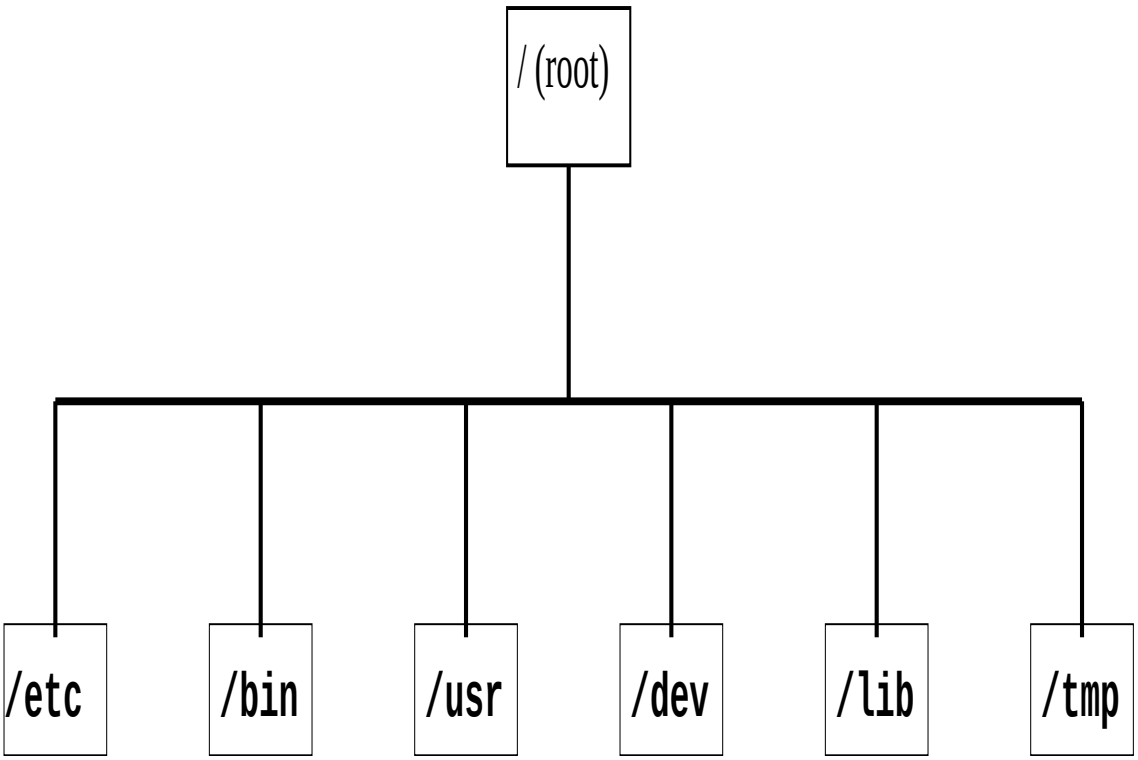

figura 2

/etc - Comandos de administração do sistema

/bin - Comandos UNIX mais comumente usados.

/usr - todas as contas de usuários e alguns comandos.

/dev - arquivos de dispositivos.

/lib - arquivos biblioteca para programação em C.

/tmp - armazenamento temporário.

4 Processos

Sempre que se executa um utilitário ou um programa no UNIX, um processo se inicia.

Como um sistema operacional multiusuário e multitarefa, o UNIX é capaz de rodar vários processos simultaneamente. O kernel do UNIX controla o tempo e as prioridades de execução dos processo. Ele permite que os processos sejam criados (por exemplo, carregando um utilitário para ser executado) e terminados.

Existem dois tipos, ou níveis, de processo no UNIX : os processos do usuário e os de sistema (ou kernel). Sempre que se executa um utilitário UNIX (por exemplo, para se exibir a quantidade de espaço disponível no disco), deve-se iniciar um processo de usuário.

Os processos de sistema, por outro lado, são iniciados pelo kernel para manter o controle preciso do sistema operacional. Por exemplo, o kernel inicia um processo de sistema (ou chamada sistema) sempre que um programa precisa de memória para ser executado. Os processos de sistema são, em sua maior parte, transparentes para o usuário.

Cada vez que um processo de usuário precisa acessar um recurso do sistema, o kernel processa a solicitação. Por exemplo, na execução de um utilitário UNIX que precisa ler

um arquivo de disco, ele faz uma chamada sistema para o kernel fazer a leitura; o kernel é o acionador do sistema UNIX. Ele serve como interface entre o usuário e o hardware para fornecer um sistema de arquivos e coordenar a execução dos processos.

4.1 Escalonamento da CPU

O UNIX é um sistema operacional de tempo compartilhado. Como tal, cada usuário deve ter a ilusão de ter o computador completamente para si. Entretanto, como existe apenas um processador na maioria dos sistemas UNIX, somente um processo pode ser executado por vez. Isso significa que os processos devem ser ativados e desativados a alta velocidade, de maneira que pareçam rodar ininterruptamente. Em milésimos de segundo, o sistema UNIX interrompe o processo corrente e passa a executar outro processo, um mecanismo conhecido como escalonamento apropriativo. Devido à velocidade da CPU, o tempo gasto na passagem de um processo para outo é muito pequeno. Todas as informações sobre o contexto de cada processo são gravadas, desta forma, quando o UNIX retoma a execução de um processo, pode continuar rodando-o exatamente como antes.

Para que os processos pareçam rodar de maneira ininterrupta, o kernel deve controlar o tempo de processamento de maneira "justa". Para fazer isso, ele usa uma abordagem circular, isto é, ele salta de um processo para o seguinte de acordo com um sistema de prioridade. Em geral, os processos de sistema têm prioridade sobre os de usuários; isso assegura que o sistema seja executado de forma uniforme. Com a abordagem circular, embora os processos isolados não sejam completados com o máximo de rapidez, o poder de processamento do sistema fica distribuído de maneira mais equilibrada entre todos os usuários.

4.2 As Ferramentas do Sistema UNIX

A terceira camada, a camada externa, do sistema UNIX contém as ferramentas, que variam de implementação para implementação. Algumas implementações incluem todas as mais de quatrocentas ferramentas UNIX. Algumas, porém, incluem apenas um subconjunto adequado a um tipo de usuário - processamento de palavras, aplicações comerciais, programadores e assim por diante.

4.3 Fundamentos do uso do sistema UNIX

A melhor maneira de se aprender o sistema UNIX é usá-lo. Aprenderemos os ítens a seguir mais detalhadamente :

1. Como ter acesso ao UNIX.
2. Relacionamento entre as principais partes do software com o hardware do computador e o usuário do sistema.
3. Comunicação com usuários.
4. armazenamento de dados e ferramentas para a manipulação dos dados.

4.4 Algumas definições básicas

Durante a maior parte do tempo você estará interagindo com um programa conhecido como shell. O shell é um programa que interpreta os comandos que você digita. Tal como nos Sistemas Operacionais em Disco (DOS), temos um prompt e em seguida devemos digitar o comando que vamos utilizar. Por exemplo :

$ date

$ é o prompt do shell e **date**, o nome do comando.

Para um comando mais complexo, você digita o nome do comando e depois um ou mais espaços seguidos de um ou mais elementos que especifiquem informações adicionais ao comando. Esses elementos adicionais são chamados argumentos ou parâmetros. Por exemplo :

$ cp temp /usr/usuario

$ é o prompt do shell, cp é o nome do comando, e temp e **/usr/usuario** são argumentos.

Existe uma categoria especial de argumentos da linha de comando chamada opções. No sistema UNIX, as opções se parecem com argumentos. Elas aparecem imediatamente após o nome de comando e geralmente consistem em um sinal de menos (-) ou, às vezes, em um sinal de mais (+), seguidos de um ou mais caracteres (geralmente letras e números). A finalidade de uma opção é especificar uma modificação na operação normal do comando. Alguns comandos não permitem opções enquanto outros podem ter várias opções possíveis. Por exemplo :

$ du -a

onde **du** é o nome do comando e o argumento de opção, **-a**.

O argumento de opção pode consistir em mais de uma letra ou números, ou ambos. Cada caracter pode representar uma opção de modificação diferente. Por exemplo :

$ ls -tr

Aqui t e r provocam uma ação de modificação diferente, cada um, no comando ls . Algumas vezes as letras de opção devem ser separadas umas das outras, como em -t, -r; outras, as letras de opção devem ser colocadas juntas, como em -tr, para que o comando as reconheça corretamente.

Ocasionalmente, vários caracteres são combinados para formar uma única opção de modificação.

Por exemplo:

$ pr -w80

os caracteres **-w80** formam uma única opção.

Argumentos adicionais do comando podem também estar presentes. Caracteres brancos separam os argumentos do nome de comando, das letras de opções, ou o nome de comando da letra de opções. Esses argumentos são geralmente os ítens de dados onde o comando opera. Por exemplo:

$ cat -n arquivoA arquivoB

cat é o nome do comando, **-n** é uma opção, e **arquivoA** e **arquivoB** são argumentos adicionais que se referem a dois arquivos específicos, neste caso chamados de **arquivoA** e **arquivoB**.

4.5 Formato da linha de comando

O formato geral das linhas de comando é :

$ comando opção argumento1 argumento2

Como no sistema UNIX existem várias opções e argumentos possíveis para a maioria dos comandos, a declaração de formato de linha de comando permite que se descreva todas as possíveis linhas de comando em termos gerais. O uso de colchetes [ ] em um argumento indica que aquele argumento é opcional. Os três pontos (...) que seguem um argumento indicam que aquele argumento pode ser repetido. Assim, a declaração de formato de linha de comando precedente poderia ser generalizada da seguinte maneira:

$ comando [opção ...] [argumento ...]

Aqui **opção ...** e **argumento ...** indicam que pode haver mais de uma opção ou argumento; e, como cada uma dessas expressões está entre colchetes, ambas são opcionais - isto é, o comando pode ser usado sem nenhuma opção e sem nenhum argumento.

A maioria dos nomes dos comandos é digitada em letras minúsculas. O sistema UNIX distingue as letras maiúsculas das minúsculas; um caracter maiúsculo é interpretado de forma diferente de sua versão minúscula. Assim, você digitaria date para chamar o comando date, ao passo que, se digitasse **Date** ou **DATE**, você obteria uma mensagem de erro: nenhuma das formas é um substituto para o date. A menos que especifique o contrário, você sempre deve digitar o nome do comando em letras minúsculas, para assegurar que o sistema interprete o comando corretamente. Também é importante usar o tipo adequado para as letras de opções e para os argumentos da linha de comando.

O espaço digitado, barra de espaço, é frequentemente exigido quando se digitam comandos. Os espaços são usados como delimitadores, ou separadores dos elementos da linha de comando. O espaço e o caractere tab são conhecidos coletivamente como caracteres brancos. É importante enfatizar que, apesar de parecer inexistente, um caracter branco é tão importante quanto qualquer outro caracter. Na verdade o computador e o sistema operacional UNIX são bastante minuciosos quanto ao uso do caracter branco.

Existe uma classe de caracteres conhecidos como caracteres de controle que estão sujeitos a interpretação especial pelo computador. Assim como você pressiona a tecla SHIFT para datilografar uma letra maiúscula em uma máquina de escrever, você pressiona e segura a tecla CONTROL para digitar um caracter de controle. Ctrl D se faz pressionando e segurando a tecla CONTROL e depois pressionando a tecla D e, soltando a tecla CONTROL. A tecla CONTROL geralmente é abreviada como CTRL ou CTL.

Os caracteres usados pelo UNIX ou são caracteres de impressão visíveis, como letras, números e pontuações, ou são caracteres de controle, invisíveis. Os caracteres de controle em geral não são exibidos na tela do terminal ou do papel, a não ser que tenha havido preparos especiais para representá-los como caracteres impressos. A representação mais comum é usar um acento circunflexo (^) seguido da letra de controle, como em (^D) para Ctrl-D.

5 Iniciando a sessão

O processo de se entrar no sistema UNIX e iniciar o trabalho é chamado logging in (registrar-se, identificar-se). O processo complementar, sair do sistema, pode ser chamado logging out ou logging off.

O prompt de identificação aparece como **"login:", "user:"** ou alguma outra variação.

(IDENTIFICAÇÃO DE SEU SISTEMA UNIX)

login:

Digite seu nome de usuário e pressione RETURN ou ENTER.

Depois que o sistema UNIX lê seu nome de usuário, ele geralmente pede uma senha.

(IDENTIFICAÇÃO DE SEU SISTEMA UNIX)

login: usuario

password:

Alguns sistemas podem não exigir que você especifique uma senha. Se ela for necessária digite-a e pressione RETURN. Ela não aparecerá na tela por motivos de segurança. Depois que o sistema aceitar sua senha, sua sessão com o sistema estará oficialmente iniciada.

Se você cometer um erro de digitação ou digitar o nome do usuário ou a senha de forma incorreta, o UNIX exibirá uma mensagem, como "Login incorrect", pedindo para que você digite o nome do usuário novamente.

Uma vez iniciada oficialmente sua sessão, o UNIX pode exibir uma variedade de mensagens abaixo do prompt da senha.

(IDENTIFICAÇÃO DE SEU SISTEMA UNIX)
login: usuario
password:
(MENSAGENS DE SEU SISTEMA UNIX)

Muitos sistemas informam a última vez que sua conta foi acessada. É aconselhável que você sempre verifique a mensagem que indica seu último acesso para ter certeza de que ela está de acordo com o que você está lembrando. Uma discrepância pode indicar que uma outra pessoa usou sua conta.

Além da última identificação, o sistema pode exibir mensagens do dia, mensagens mais recentes para os usuários do sistema. Tipicamento, o horário de manutenção do sistema, as mensagens de aviso (espaço disponível em disco, dentre outros). Uma outra mensagem pode ser "You have mail" indicando que alguém lhe enviou uma mensagem por correio eletrônico.

Depois de todas as mensagens tiverem sido exibidas, o sistema indicará que ele está pronto para seus comando exibindo o prompt do shell ou % , ou então um prompt diferente.

Para encerrar uma sessão, apenas digite exit ou logout. Assim, tudo retorna ao iníco, ou seja, serão exigidos novamente um login e um password para o novo usuário.

6 O sistema de arquivamento e os tipos de arquivos do UNIX

O sistema UNIX de arquivamento é uma estrutura para a organização de informações e dados que fornece uma armazenagem de longo prazo flexível e eficiente tanto para os dados relacionados ao usuário quanto para os dados relacionados ao sistema.

Os dados são agrupados em entidades chamadas arquivos. Todas as versões do sistema UNIX reconhecem pelo menos três tipos de arquivos :

6.1 Arquivos comuns

Usados para armazenar dados. Os usuários podem acrescentar dados diretamente em arquivos comuns, como, por exemplo, um editor.

Os programas executáveis são também guardados como arquivos comuns.

6.2 Arquivos de diretório

Um arquivo de diretório contém uma lista de arquivos.

Cada inserção na lista consiste em duas partes : o nome do arquivo e um ponteiro para o arquivo real em disco. Por outro lado, os diretórios se comportam exatamente como arquivos comuns, exceto pelo fato de que algumas operações que você usaria para a manipulação de arquivos comuns não funcionam com os arquivos de diretório e vice-versa.

6.3 Arquivos especiais

Esses arquivos são usados para se fazer referências aos dispositivos físicos, como os terminais, as impressoras, os discos e as unidades de fitas. Eles são lidos e gravados como arquivos comuns, mas tais associações causam a ativação do dispositivo físico associado a eles.

6.4 Comandos para a manipulação de arquivos

6.4.1 ls : lista o conteúdo de um diretório

Após iniciar uma sessão no seu sistema, você entra no seu diretório base e tem vários arquivos.

O comando ls (juntamente com alguns parâmetros) lista os arquivos do diretório local.

Exemplos de alguns parâmetros para o comando ls :

ls [opção ...] [arquivo ...]

-l - lista em formato longo. Os arquivos são ordenados pelo nome. Os modos de permissão, o número de ligações, o proprietário do arquivo, a data e a hora da última modificação e o nome do arquivo são listados.

-t - lista pela ordem cronológica em função da hora da última modificação (as mais recentes primeiro).

-a - lista todas as inserções, incluindo os arquivos cujos nomes comecem com um ponto (.)

-s - Informa o número de blocos de disco ocupados por cada arquivo.

-d - lista o nome do diretório em lugar de seu conteúdo.

-r - Inverte a ordem de classificação na listagem.

-u - lista pela ordem cronológica em funcção da hora do último acesso.

-c - Lista pela ordem cronológica em função da hora da última modificação do inode e não do arquivo.

-i - Exibe o número do inode na primeira coluna.

-f - Força cada argumento a ser interpretado como um diretório. Essa opção desativa as opções -l, -t, -s e -r, mas ativa -a. O conteúdo dos arquivos comuns são exibidos de maneira incorreta.

-g - Lista ID do grupo em lugar da ID do usuário na listagem longa.

-o - Lista no formato longo suprimindo o nome do grupo. Normalmente tanto o nome do usuário quanto o nome do grupo são listados.

-p - Acrescenta uma barra (/) ao nome de cada arquivo de diretório.

-x - Classifica o resultado em várias colunas no sentido horizontal em lugar de no sentido vertical.

-F - Exibe os nomes de diretórios com uma barra (/) no final e os nomes de arquivos executáveis com um asterisco (*) no fim.

-R - lista recursivamente os subdiretórios

Estes parâmetros podem ser usados juntamente no mesmo comando ls, por exemplo :

ls -al - irá listar todas as inserções, incluindo os arquivos cujos nomes comecem com um ponto (.) e no formato longo, Os arquivos serão ordenados pelo nome. Os modos de permissão, o número de ligações, o proprietário do arquivo, a data e a hora da última modificação e o nome do arquivo são listados.

ls -alR /etc - irá listar todos os arquivos contidos no diretório /etc recursivamente, todas as inserções, incluindo os arquivos cujos nomes comecem com um ponto (.) e no formato longo, Os arquivos serão ordenados pelo nome. Os modos de permissão, o número de ligações, o proprietário do arquivo, a data e a hora da última modificação e o nome do arquivo são listados.

6.4.2 cat: concatena e imprime um arquivo

cat é a abreviação de concatenar, que significa colocar junto. Este utilitário é usado para exibir o conteúdo de um único arquivo, ou o conteúdo de vários arquivos em sucessão.

cat teste - irá exibir o conteúdo do arquivo chamado teste.

Para exibir o conteúdo de mais de um arquivo :

cat teste teste1 teste2 - irá exibir o conteúdo dos arquivos teste, teste1 e teste2 .

6.4.3 cp: copia um arquivo comum

O comando **cp (copy)** permite que se crie uma duplicata de um arquivo comum. O formato da linha de comando é:

cp arqfont arqdest

arqfont é o arquivo inicial ou fonte, e **arqdest** é o nome da cópia a ser criada. O nome de arqdest deve ser diferente de arqfont.

6.4.4 rm: remove arquivos

O comando **rm (remove)** remove um ou mais arquivos comuns de um diretório, apagando efetivamente o arquivo. O formato da linha de comando para se usar rm para remover um ou mais arquivos é :

rm arquivo1 arquivo2 arquivo3

onde cada arquivo é separado do próximo por espaços brancos.

6.4.5 mv: mover arquivos

O comando **mv (move)** altera o nome associado a um arquivo associando um nome novo ao ponteiro do arquivo físico na inserção do diretório e, depois, removendo o elo do nome antigo. O formato da linha de comando para o uso do mv é :

mv nomeantigo nomenovo

Essa linha de comando muda o nome do arquivo nomeantigo para nomenovo.

O comando mv tambem move arquivos, ou seja, retira um arquivo de um local e o coloca em outro local. Suponha que se queira mover um arquivo de nome **teste.txt** localizado no diretório /usr/manoel para o diretório /usr/manoel/textos . A linha de comando fica assim :

/usr/manoel $ mv teste.txt /usr/manoel/textos

onde **/usr/manoel $** é o prompt que indica o diretório atual e **/usr/manoel/textos** é o diretório destino.

Agora suponha que se queira renomear o arquivo teste.txt para **exemplo.doc**. A linha de comando fica assim :

/usr/manoel $ cd textos

/usr/manoel/textos $ mv teste.txt exemplo.doc

Uma maneira mais direta seria :

/usr/manoel $ mv teste.txt /usr/manoel/textos/exemplo.doc

Essa linha de comando significa que o arquivo **teste.txt** foi movido para o diretório **/usr/manoel/textos** e ficou com o nome definitivo de **exemplo.doc** .

6.5 Permissões para acesso a Arquivos

O sistema UNIX fornece um meio fácil de controlar o acesso que os usuários do sistema possam ter a todos os três tipos de arquivos UNIX (ou seja, arquivos comuns, de diretórios e arquivos especiais). Isso é feito para permitir ou restringir o acesso a certos arquivos importantes, para ajudar a prevenir uma perda acidental do conteúdo desses arquivos. Pode-se manter a possibilidade de ler certos arquivos e restringir a possibilidade de escrever neles. Para um sistema multiusuário a restrição do acesso aos arquivos é muito importante,

pois você pode querer que outros usuários do sistema nem mesmo leiam seus arquivos, alterar ou até apagar seus arquivos. O sistema de permissão de acesso a arquivos permite que você controle o destino de seus arquivos.

Existem três classes de usuários de sistema. Primeiro, todo arquivo tem um proprietário, e este geralmente é o usuário que criou o arquivo. O superusuário pode alterar a posse individual de um arquivo, se necessário. O proprietário tem controle total sobre a restrição ou permissão de acesso ao arquivo a qualquer hora. Além da posse individual do arquivo, é possível que um ou mais usuários do sistema possuam o arquivo coletivamente, em um tipo de propriedade de grupo. Um usuário que não for o proprietário do arquivo pode ter acesso a ele se pertencer ao grupo de usuários que têm permissão para isto. Porém, esse usuário não pode restringir ou permitir acesso ao arquivo; apenas o proprietário do arquivo é quem pode fazer isto. Os usuários que não são nem proprietários individuais, nem proprietários em grupos do arquivo formam a última categoria, conhecida simplesmente como outros. Em resumo :

Proprietário (designado por **u**, de **user, usuário**). O usuário que criou o arquivo.

Grupo (designado por **g**, de **group, grupo**). O grupo é formado por um ou mais usuários que podem ter acesso a um arquivo.

Outros (designado por **o**, de **others, outros**). Refere-se a qualquer outro usuário do sistema.

O sistema de arquivamento UNIX permite que cada classe de usuário tenha acesso ao arquivo independentemente das outras classes. Isto é, os direitos de acesso ao proprietário do arquivo, ao grupo e ao usuário da categoria "outros" podem ser iguais ou diferentes.

Além das classes de usuários dos arquivos, existem três maneiras diferentes de ter acesso a um arquivo. O significado desses modos de acesso é um pouco diferente para os arquivos comuns e para os arquivos de diretórios. Esses modos e seus significados estão na tabela 2.

O significado dos três modos de acesso para os arquivos comuns é relativamente lógico. Os usuários com permissão de leitura podem ler o conteúdo de um arquivo comum (usando, por exemplo, cat). Os usuários com permissão de escrita podem gravar em um arquivo e alterar o seu conteúdo (usando, por exemplo, um editor). A permissão de escrita também é exigida para se eliminar um arquivo usando o comando rm. Mesmo que a permissão de escrita não seja possível de fazer, o proprietário do arquivo sempre pode eliminar o arquivo se ele primeiro possibilitar essa permissão e, depois, remover o arquivo.

Modo de acesso	Arquivo comum	Arquivo de Diretório
Leitura (read)	Examinar o conteúdo do arquivo	Listar os arquivos
Escrita (write)	Alterar o conteúdo do arquivo	Criar e remover arquivos
Execução	Executar um arquivo como comando	Pesquisar o diretório

Finalmente, os usuários com permissão de execução nos arquivos comuns podem executar um arquivo como um comando. Geralmente, um arquivo comum não recebe permissão para ser executado quando é criado. A execução de um arquivo comum somente faz sentido se o arquivo for realmente um programa (comando) ou um shell script . Um shell script é um arquivo que contém uma lista de um ou mais comandos que podem ser executados pelo shell.

O significado desses mesmos modos de acesso é diferente para um arquivo de diretório. O usuário com permissão de leitura pode ler o conteúdo do diretório (por exemplo, usando o comando ls). O usuário com permissão de escrita, por outro lado, pode usar alguns programas privilegiados para gravar em um diretório. A permissão de gravação é necessária para se criar ou para remover arquivos.

Um usuário deve ter permissão de execução em um diretório para ter acesso aos arquivos ali mencionados. Se um usuário tem permissão para leitura e escrita em um arquivo comum que está listado em um diretório mas não tem a permissão de execução para aquele diretório, o sistema UNIX não deixa você ler nem gravar o conteúdo daquele arquivo comum.

As três classes de usuários (proprietário, grupo e outros) podem ser combinadas com os três tipos de acesso (leitura, escrita e execução) :

r w x r w x r w x

Proprietário Grupo Outros

A presença de uma permissão é indicada pela letra apropriada **(r, w, x)**, a ausência é indicada por um traço **(-)** .

Exemplos :

r–r–r–

Este arquivo pode ser lido por todas as três classes. Um arquivo com essa característica é conhecido como read-only (apenas leitura).

–x–x–x

Este arquivo pode ser executado por todas as três classes, é protegido contra leitura e gravação.

rw———-

Este arquivo pode ser lido e gravado apenas por seu proprietário. É um arquivo que o proprietário mantém afastado de todos os outros usuários.

Alguns dos padrões de permissão mais comuns para os arquivos de diretórios e seus significados são :

rwxrwxrwx

Esse diretório é completamente acessível por todos os usuários do sistema.

rwx——

Esse diretório pode ser acessado apenas por seu proprietário; outros usuários não podem pesquisá-lo, lê-lo ou fazer gravações nos arquivos nele contidos.

Para determinar o tipo de um arquivo ou diretório, utiliza-se o comando ls com a opção -l (listagem longa).

Por exemplo :

$ ls -l
$ -rw-rw-rw- 1 maria docum 40 Nov 1 13:23 carta

O arquivo carta pode ser lido e gravado por todos os usuários e o proprietário dele é maria.

Significado dos vários segmentos (ou campos) da listagem longa :

$ -rw-rw-rw- 1 maria docum 40 Nov 1 13:23 carta

- (-) no campo do tipo de arquivo indica que ele é um arquivo comum
- O campo das permissões de acesso informa que todas as permissões são concedidas
- O campo de ligações indica que existe apenas 1 (uma) ligação entre esse arquivo e o diretório, o que significa que esse arquivo tem apenas um nome associado a ele.
- A palavra "maria" indica que o proprietário do arquivo é o usuário chamado "maria"
- O grupo que tem acesso a esse arquivo é designado por "docum"
- O arquivo tem o tamanho de 40 Bytes
- A data e a hora " Nov 1 13:23", mostram a última vez em que o arquivo foi modificado
- O nome do arquivo é "carta"

Obs : No primeiro campo, (-) indica um arquivo comum, e d indica um arquivo de diretório.

Alterando as permissões de acesso a arquivos

O comando chmod permite que se altere os modos de permissão de um ou mais arquivos ou diretórios. O formato da linha de comando e´:

$ chmod [quem] operação permissão... arquivo...

O argumento **quem** informa a **chmod** a classe do usuário e pode ser qualquer um dos seguintes :

u Usuário (proprietário individual do arquivo).

g Proprietário grupal do arquivo.

o Usuários classificados como "Outros".

a Todos (all) os usuários do sistema (proprietário do arquivo, grupo e outros).

O argumento operação representa a operação a ser executada por chmod:

+ Acrescenta as permissões especificadas às permissões existentes.

- Retira as permissões indicadas das permissões existentes.

= Atribui as permissões indicadas.

O argumento permissão usa as mesmas abreviações que dos tipos de acesso a arquivos:

r Permissão de leitura (read).

w Permissão de escrita (write).

x Permissão de execução

Exemplos de utilização :

Para restringir permissões para o arquivo carta de forma que nem os usuários do grupo nem os "outros" possam ter acesso a ele.

Examinando a listagem :

$ ls -l carta

-rw-rw-rw- 1 maria docum 40 Nov 1 13:23 carta

Fazendo **chmod go -rw carta** para retirar as permissões de leitura e escrita das categorias de usuários do grupo e outros.

O resultado será :

$ ls -l carta
-rw——- 1 maria docum 40 Nov 1 13:23 carta

Para colocar permissão de leitura para outros, teremos que fazer :

$ chmod o+r carta
$ ls -l
-rw—-r– 1 maria docum 40 Nov 1 13:23 carta

Para programas executáveis recém instalados, geralmente é necessário que se permita que ele possa ser executado pelo proprietário, pelo grupo de usuários e por outros, e a maneira mais usual de se fazer isso é :

$ chmod ugo+x [programa]

Uma outra maneira de se ter uma descrição dos comandos a serem utilizados é usar o **man** do **UNIX**, ou seja, para o se saber os parâmetros exigidos de um certo comando, basta digitar **man comando**. Por exemplo : **man lpr** mostra o manual de instruções do comando **lpr**. Este manual está disposto em forma de arquivo e em forma de manual de instruções para a versão do Sistema Operacional utilizado.

6.6 Manipulação de Diretórios

6.6.1 pwd: mostra diretório atual

Para se saber em que ponto estamos da árvore de diretórios do sistema de arquivos, utilizamos o comando **pwd**, que faz o mesmo trabalho quando digitamos simplesmente **cd** no **DOS**.

$ pwd
/usr/marcelo
$

6.6.2 mkdir: criar diretórios

Para criar um diretório no **UNIX** usa-se o comando **mkdir** que tem como equivalente no **DOS** o comando **MD** ou mesmo **MKDIR**. A maneira de utilizá-lo é semelhante ao **DOS**. Exemplo:

$ mkdir Lixo

Este exemplo cria um sub-diretório abaixo do diretório corrente, de nome **Lixo**.

6.6.3 cd: troca de diretório corrente

Podemos mudar o diretório de trabalho (diretório default) com o comando cd, que funciona de forma semelhante ao **DOS**. Atente aos detalhes do uso da barra **(/)** e não da contrabarra **(\)**, como no **DOS**, para indicação dos níveis. Exemplos:

$ cd Lixo
$ cd /
$ cd /usr
$ cd hermano
$ cd /usr/marcelo
$ cd
$ pwd

6.6.4 rmdir: remover diretório

O comando **rmdir** elimina um diretório que deve necessariamente estar vazio. Exemplo:

$ rmdir Lixo

6.6.5 Atribuição de pseudônimos para comandos (alias)

O comando **alias** permite criar nomes alternativos para comandos. Sem argumentos fornece a lista dos **"aliases"** definidos. Exemplo:

$ alias
false=let 0
functions=typeset -f
h=fc -l
hash=alias -t
history=fc -l
integer=typeset -I
lo=exit
more=pg -n
nohup=nohup
r=fc -e -
true=:
type=whence -v
$

Para atribuir **ls -F** para **ls**:

$ alias ls 'ls -F'

Os apóstrofes são necessários porque a parte à direita da atribuição contém branco. Se preferirmos os nomes dos comandos do **DOS** para similares **UNIX**, podemos fazer :

$ alias dir ls -l'
$ alias type cat
$ alias del rm -I'
$ alias ren mv

Se quisermos eliminar uma definição de **alias** usamos o comando **unalias**.

$ unalias dir type del ren

7 Segurança

Por ser um sistema multiusuário, muitos usuários trabalham em um mesmo sistema de arquivos.

O acesso de um usuário a informaões gravidas na área de trabalho de outros não é possível a menos que o dono da informação permita.

Serão descritos neste capítulo quais são e como manipular os sistemas de seguranç a que o UNIX oferece para proteger os arquivos e diretórios dos usuários e do Sistema.

7.1 Valores de permissão default

Quando um arquivo é criado, ele recebe um conjunto default de permissões. Esse conjunto, que normalmente é rwxrwxr–, é controlado pelo valor corrente de umask. Assim como chmod, umask recebe três dígitos como parâmetro. Sem parâmetro ele informa seu valor corrente.

$ umask

02

O valor 02 é mostrado geralmente

Interpretação do valor de umask

Criando um arquivo:

$ cat > arquivo

e observando seus atributos,

$ ls -l arquivo

-rw-rw-r– 1 manoel users 0 jun 15 08:58 arquivo

vemos que é **-rw-rw-r–** que denota um arquivo comum no qual estão habilitados as operações de **escrita** e **leitura** para o **owner** e **group** e apenas **leitura** para **others**.

Logicamente podemos trocar essas permissões com **chmod**. Entretanto, se quisermos que um arquivo seja criado com permissões diferentes, devemos trocar o valor de **umask**.

O valor corrente 02 de **umask** deve ser inetrpretado da seguinte forma:

Se acrescentarmos um **0** à esquerda desse valor temos **002**. Cada dígito representa o conjunto de permissões que queremos tirar do grupo em questão. Isto é, tirar nenhuma permissão do **owner (0)**, nenhuma permissão de **group (0)**, tirar permissão de gravação de **others (2)**.

Se desejarmos que arquivos recém-criados tenham o conjunto **rwx r-x —** no momento de sua criação, devemos fazer:

$ umask 037

$ cat > arquivo2

$ ls -l arquivo2

-rw-r—— 1 manoel users 0 jun 15 08:58 arquivo

Embora tenhamos ajustado o valor de **umask** para que o **owner** e o **group** tenham direito de execução **(x)**, arquivos de dados não são executáveis.

7.2 Criação de arquivos de comandos (batch ou shell script)

Os arquivos executáveis são aqueles criados por compiladores (**C**, por exemplo) ou aqueles com comandos para serem executados em **batch**, comandos **shell script** ou arquivos **batch**. Para criar um arquivo **batch** basta criar um arquivo texto com os comandos a serem executados e permitir que ele seja executável **(x)** com o comando **chmod**.

Exemplo: Usar o editor vi para criar um arquivo de nome **"lote"** com os comandos **who**, **date**, **ls -l**, um por linha, e trocar seu atributo com **chmod** para executável.

$ chmod +x lote

Executar o arquivo lote

$ ./lote

manoel ttyp0 Feb 11 10:35

hernamo ttyp1 Feb 10 17:41

Tue Feb 11 11:41:30 CST 1998

```
-rw-r–r–      1 manoel      root      127349     jun   15   08:58     fly-1.6.5
-rwxr-xr-x    1 manoel      users     905705     set    8   1998      latx2ht.gz
drwxr-xr-x    2 manoel      users      1024      ago   28   1998      mail
-rwxr-xr-x    1 manoel      users      5046      ago   25   1998      neped.c
-rwxr-xr-x    1 manoel      root        394      mar   24   18:04     access1.c
-rw-r–r–      1 manoel      root       15360     fev   18   23:17     relat.doc
-rwxrwxrwx    1 manoel      root       21791     fev   19   00:04     relatorio
-rwxr-xr-x    1 manoel      root        6905     mar   24   18:12     rltest.tgz
-rwxr-x–x     1 manoel      users         15     feb   11   11:40     lote
```

A forma de chmod utilizada (chmod +x lote) é uma alternativa para ligar o atributo x (+x) para todos os grupos de usuários. No último exemplo, os três comandos contidos no arquivo "lote" foram executados e suas saídas apareceram uma após a outra.

8 Redirecionamento

8.1 Arquivos padrão de entrada e saída

O **UNIX** trata todos os periféricos conectados ao sistema como arquivos. O teclado, por exemplo, é um "arquivo" de entrada; o vídeo é um arquivo de saída, assim como a impressora.

O arquivo padrão de entrada é aquele que os programas usam para obter seus dados caso nenhum outro arquivo seja especificado. Esse arquivo está geralmente associado ao teclado.

Vejamos, como exemplo desse mecanismo, o funcionamento do mail. Quando executamos o mail na forma "mail usuário", o programa espera que a mensagem seja digitada pelo teclado até o pressionamento de **^D (Control+D)**. Tecnicamente dizemos que o **mail** pegou sua entrada do arquivo **stdin (standard input)** pois não foi dado outro arquivo.

$ mail joao
Olah, Joao ! Como vai ?
^D

O arquivo padrão de saída **(stdout)** é o dispositivo no qual o **UNIX** "despeja" os resultados por **default**. Esse dispositivo geralmente é o vídeo. Praticamente todos os programas componentes do sistema que apresentam dados, o fazem em **stdout** como, por exemplo, o comando **ls**.

8.2 Redirecionando a saída padrão para um arquivo

Utilizando o símbolo > podemos redirecionar a saída que seria para o vídeo para, por exemplo, um arquivo em disco.

Abaixo, a saída do comando date é gravada num arquivo de nome "agora".

$ date > agora
$ cat agora
ter jul 20 11:20:06 EST 1999
$

8.3 Acrescentando em um arquivo

O operador > cria um novo arquivo com o nome dado. Se o arquivo já existia, ele é sobreposto. Usando > >, a saída é acrescentada ao final do arquivo se o mesmo já existe ou um novo arquivo é criado.

$ date > agora
$ cat agora

ter jul 20 11:20:06 EST 1999

$ date > > agora
$ cat agora
ter jul 20 11:20:06 EST 1999
ter jul 20 11:21:06 EST 1999
$

8.4 Redirecionando de um arquivo

Para redirecionar a saída da tela para um arquivo usamos o >. Podemos fazer com que o **UNIX** leia um arquivo texto como se estivesse lendo de stdin (teclado) com o operador < . Abaixo mostramos um arquivo contendo um convite para uma reunião que será mandado para o usuário **jose** através de um **e-mail**.

$ cat > convite
Reuniao sexta-feira às 10:00
$ mail jose < convite

Sabemos que o **mail** pega sua entrada de **stdin**, portanto, é possível fazê- lo ler uma entrada redirecionada de um arquivo. O efeito é como se digitássemos novamente o conteúdo de convite para o mail.

8.5 Redirecionando entrada e saída

O **"sort"** é um programa que classifica alfabeticamente as linhas de um arquivo texto. Ele obtém sua entrada de **stdin** e apresenta a saída em **stdout**.

Se quisermos ordenar os nomes contidos no arquivo nomes utilizando **sort**, podemos fazer:

$ cat > nomes
joaquim
manoel
jose
joao

$ sort < nomes
joao
joaquim
jose
manoel
$

O comando **"sort < nomes"** leu o conteúdo do arquivo **nomes**, classificando-o e imprimindo o resultado em **stdout**. A saída poderia ser redirecionada, ao mesmo tempo, para um arquivo, como mostrado abaixo:

$ sort < nomes > nomes.ord
$ cat nomes.ord
joao
joaquim
jose
manoel
$

O **sort** leu o conteúdo do arquivo **"nomes"**, classificou-o e a saída, que teria como destino a tela, foi **"jogada"** para um novo arquivo chamado **"nomes.ord."** . Um detalhe importante: os arquivos de entrada e saída devem ser diferentes, pois a primeira parte do comando a ser executada é a criação do arquivo de saída, o que implica que se ele for o de entrada, ele é zerado.

8.6 Programas "filtro"

Os chamados programas **"filtro"** pegam sua entrada de stdin, fazem alguma modificação nesses dados (processam-os) e colocam a saída em um **stdout (tela do terminal)**.

O mais simples dos programas filtro é o **cat**, que faz uma cópia fiel da entrada na saída. Outros programas desse tipo são: **sort**, **wc**, que conta as linhas, caracteres e palavras, **grep**, que mostra as linhas de um arquivo contendo uma sequência de caracteres, e muitos outros comandos **UNIX**.

Não são programas filtro: **ls, who, date, cal, write,** pois embora a saída seja para **stdout** não é de **stdin**.

8.7 Pipes (dutos)

Pipes são uma forma pela qual podemos contruir uma conexão de dados entre a saída de um programa e a entrada de outro. A saída do programa anterior serve de entrada para o posterior.

Forma de um pipe: comando1 | comando2

Comandos conectados por pipes formam uma **pipeline**.

Nesse caso a saída padrão do **comando1** é automaticamente entendida como a entrada padrão do **comando2**, ou seja, o **comando2** processa a saída do **comando1**. Por exemplo, salvemos a saída do programa who no arquivo **"whois"**.

```
$ who > whois
$ cat whois
zeferino  console   Feb      27   20:10
guest             tty00         Feb  27   20:00
zeferino  ttx00              Feb  27   20:05
marcelo   ttx01              Feb  27   22:10
```

A saída de **who** é ordenada por terminal e não por nome, como poderia ser desejado.

De posse do arquivo **whois**, podemos usar o **sort** para fornecer uma listagem por nome.

```
$ sort < whois
guest             tty00             Feb  27   20:00
marcelo           ttx01             Feb  27   22:10
zeferino          console           Feb  27   20:10
zeferino          ttx00             Feb  27   20:05
```

A idéia do exemplo foi criar um arquivo temporário com a saída de who para servir de entrada de sort. Essa operação pode ser feita automaticamente com o uso de pipes, não sendo necessária a criação do arquivo temporário ("whois").

```
$ who | sort
guest             tty00             Feb  27   20:00
marcelo           ttx01             Feb  27   22:10
zeferino          console           Feb  27   20:10
zeferino          ttx00             Feb  27   20:05
```

O exemplo seguinte usa o programa wc com a opção -l, que informa o número de linhas de um arquivo. O resultado do comando abaixo pode ser interpretado como o número de usuários conectados ao sistema naquele momento.

```
$ who | wc -l
   2
$
```

Um uso bastante comum de **pipes** é em conjunto com o programa **more** ou **page**, que pagina um arquivo texto quando este tem mais linhas do que o vídeo pode mostrar. Experimente os comandos abaixo:

```
$ more /etc/inittab
$ ls -l /dev
$ ls -l /dev | more
```

8.8 Outros comandos

8.8.1 Comando find

Procurar arquivos por nomes
$ find path [opção]
Opões (algumas):

name: especifica que será informado o nome de um arquivo

print: exibe na tela a rota dos arquivos que satisfazem os critérios

type: especifica que será informado o tipe de arquivo

Exemplos:
$ find . -name arquivo -print - procura por **arquivo** no diretório local
$ find . -name "arq*.dat" -print - procura por **arq*.dat** no diretório local
$ find / -name core -print - procura por **core** a partir da raiz /
Obs: Podem ser utilizados metacaracters: [ ] , ? , * .

8.8.2 Comando grep

Procurar uma expressão em um arquivo
$ grep [opção] "expressão" <arquivo>
O utilitário **grep** é útil para localizar as linhas de texto que contenham uma ocorrência específica. Ele lê as linhas do texto do arquivo de entrada padrão e grava no arquivo de saída padrão apenas as linhas que contenham a ocorrência ou quando não for especificado um arquivo de saída o resultado é mostrado na tela.
Opões (algumas):

i: ignora a diferena entre maiúscula e minúscula

c: mostra número de vezes que a expressão foi encontrada

l: lista somente o nome dos arquivos que contém a expressão procurada

n: numera cada linha que contém a expressão procurada

Exemplos:
$ grep -i alias *
$ grep "inetd" /etc/*.conf

8.8.3 Comando diff

Mostra a diferença entre dois arquivos, linha a linha
$ diff <arquivo1> <arquivo2>

8.8.4 Comando df

Mostra o espaço livre no disco (file system)
$ df -k
$ df -ki

8.8.5 Comando du

Mostra o número de blocos usados por diretórios

$ du [opção] <diretório>

Opões (algumas):

s: relata apenas o número de blocos

a: informa o tamanho de cada arquivo

k: Lista os tamanhos em Kbytes

8.9 Pipelines com três ou mais comandos

Podemos conectar quantos programas desejarmos numa **pipeline**. Vemos abaixo dois comandos conectados por **pipe**, que mostra as linhas que contém a sequência **"ttyp"** na saída do comando **ls -l /dev**.

$ ls -l /dev | grep ttyp

```
crw–w—-    1 root    tty       3,   0 jul 20 12:05 ttyp0
crw–w—-    1 root    tty       3,   1 jul 20 11:07 ttyp1
crw–w—-    1 root    tty       3,   2 jul 19 18:00 ttyp2
crw———-    1 root    root      3,   3 jul 20 04:52 ttyp3
crw-rw-rw-  1 root    root      3,   4 jul  2 17:58 ttyp4
crw-rw-rw-  1 root    root      3,   5 jul  1 14:35 ttyp5
crw-rw-rw-  1 root    root      3,   6 jul  2 17:43 ttyp6
```

$

Note que a sequência especificada aparece na última coluna da listagem.

Se conectarmos o comando **wc -l** ao final da **pipeline**, a saída apresentada será o número de linhas que contém a sequência.

$ ls -l /dev | grep ttyp | wc -l

7 —> é o número de linhas que contém a sequência

Perceba que apenas as informações geradas pelo fim do **pipeline** são mostradas. Agora, supondo que estejamos interessados em informações que circulam no meio do **pipe**, digamos, a saída de **grep**. Podemos obtê-las com o comando **tee**.

8.9.1 Comando tee

Formato : tee arquivo

$ ls -l /dev | grep ttyp | tee /dev/ttyp2 | wc -l

```
crw–w—-    1 root    tty       3,   0 jul 20 12:05 ttyp0
crw–w—-    1 root    tty       3,   1 jul 20 11:07 ttyp1
crw–w—-    1 root    tty       3,   2 jul 19 18:00 ttyp2
crw———-    1 root    root      3,   3 jul 20 04:52 ttyp3
crw-rw-rw-  1 root    root      3,   4 jul  2 17:58 ttyp4
crw-rw-rw-  1 root    root      3,   5 jul  1 14:35 ttyp5
crw-rw-rw-  1 root    root      3,   6 jul  2 17:43 ttyp6
```

7 —> é o número de linhas que contém a sequência

$

São apresentadas as saídas de dois pontos do **pipeline**: logo após o **grep**, devido ao **tee**, e após o **wc** por ser o final.

O **tee** sempre precisa ser informado sobre o arquivo no qual apresentar suas informações. Para que essas informações sejam mostradas no vídeo, deve ser indicado o seu arquivo-terminal. No exemplo era **/dev/ttyp2**. O nome desse arquivo pode ser obtido

através dos comandos **who** ou **tty**. Ao invés de apontar a saída para um terminal **(/dev/ttyp2)**, ela poderia ser apontada para um arquivo texto (p. ex: **arq.saida.txt**).

9 Controlando a execução de Processos

Serão apresentados tópicos que permitirão ao usuário iniciante fazer tarefas interessantes como:

- executar mais de um processo ao mesmo tempo
- deixar um processo rodando mesmo depois de terminar a sessão com o sistema
- rodar um processo em "baixa prioridade"
- fazer um processo iniciar sua execução em uma determinada hora

9.1 O operador &

O **UNIX**, por ser um sistema multitarefa, permite que mais de um programa **(processo)** seja executado ao mesmo tempo. Para que um processo seja colocado em execução e o terminal liberado imediatamente para outras tarefas, usamos o **operador &** no final da linha de comando. Esse operador faz com que o processo rode em **"segundo plano" (background)**.

Forma: linha-de-comando &

Para exemplificar usaremos o utilitário **"du" (disk usage)** que informa o espaço ocupado por cada diretório abaixo do especificado e o total, em **Bytes**. O relatório gerado pelo **du** pode ser extenso dependendo do diretório pesquisado. O comando abaixo pode levar algum tempo para ser concluído:

$ du /etc
3 /etc/profile.d
6 /etc/X11/AnotherLevel/decors
11 /etc/X11/AnotherLevel/scripts
102 /etc/X11/AnotherLevel
1 /etc/X11/TheNextLevel
119 /etc/X11/wmconfig
35 /etc/X11/xinit
2 /etc/X11/fs
4 /etc/X11/twm
1 /etc/X11/xdm/authdir/authfiles
2 /etc/X11/xdm/authdir
28 /etc/X11/xdm
etc...

Para se criar um arquivo com saída do **du**, podemos usar redirecionamento como mostrado a seguir:

$ du /etc > du.saida

Esse é um tipo de tarefa que seria interessante deixar o computador fazendo **"sozinho"**, enquanto fazemos outras coisas. Para tanto, basta fazer com que o processo **du** rode em **"background"**, liberando o terminal.

$ du /etc > du.saida &
[1] 3221
$

Quando iniciamos um processo em **"background"**, aparece um número de ordem entre colchetes, seguido de um número chamado **PID (process identification)** e o nome do processo iniciado.

Só podemos colocar em segundo plano processos **"independentes"** de entrada e saída, pois processos em **background** também **"por default"** podem imprimir no terminal e ler dados do teclado, o que vem se confundir com os processos em primeiro plano. Por exemplo, rode e observe o resultado do seguinte comando:

$ du /usr &

Processos em segundo plano não respondem aos caracteres de controle como parada de tela **(^S)** e interrupção **(^C)**. Para interromper processos em segundo plano devemos usar o comando **kill**.

9.2 Protegendo processos em "background" contra o sinal hang-up

Precedendo uma linha de comando que coloca um processo em **"background"** com o comando **"nohup"**, asseguramos que o processo será completamente executado **mesmo que a sessão seja terminada**.

Forma: nohup linha-de-comando &

Como exemplo, dê o comando abaixo, seguido imediatamente de **exit** e depois verifique que o arquivo foi criado (com o uso de **nohup** não é necessário usar redirecionamento pois ele já redireciona automaticamente a saída para um arquivo chamado **nohup.out**)

$ nohup du /usr &
$ exit

9.3 Executando um processo em baixa prioridade: nice

Forma: nice comando

Quando o comando **nice** é colocado antes de uma linha de comando, o tempo dedicado pela **CPU** para a execução daquela tarefa é reduzido, fazendo com que ele não atrase tanto outros processos mais importantes. Exemplo:

$ nice du /usr saida &

Os comando **nice** ou **nohup** podem ser usados sem que o processo seja colocado em segundo plano. Mas nunca isso é feito. Por quê ?

nice só é usado com processos em segundo plano, pois não faz muito sentido atrasar processos pelo qual estamos esperando uma resposta. E usar **nohup** em processos de primeiro plano também não faz sentido pois estando o processo sendo executado não podemos encerrar a sessão.

nohup e **nice** podem ser utilizados numa mesma linha para colocar um processo em execução, deixá-lo rodando sem estarmos conectados ao sistema e não atrasar os outros usuários.

Exemplo:

$ nohup nice du /usr &

10 Processos

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

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

Status dos processos :

- executando
- bloqueado
- parado esperando uma requisição de entrada/saída
- suspenso

10.1 Processamento em background

Terminar a linha de comando com o símbolo **"&"**

Teclar (Control+Z)

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

$ find / -name "*.c" -print &

$ du / > du.all &

Outros comandos relacionados a processos:

jobs - lista os **jobs** em **background**

fg [job] - leva o **job** para **foreground**

bg [job] - leva o **job** para **background**

10.2 Informaões sobre o status dos processos

Para se obter as informaçoes sobre o estado de cada processo

$ ps [opção]

Opções:

-a: exibe os processos associados a um terminal

-l: informaões completas sobre os processos

aux: exibe todos os processos que estão sendo executados na máquina

O número de opções pode ser vistop no manual do comando **ps.** As opções variam para cada tipo de UNIX.

O comando **ps** mostra os processos que estão rodando

Exemplo:

$ ps

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

PID - número de identificação do processo

TT - Terminal

STAT - Status corrente (Running, Sleeping, Stopped (T), etc)

TIME - Tempo de CPU

COMMAND - nome do comando

Encerrar um processo :

$ kill -9 <pid> , onde pid é o número do processo (primeira coluna à esquerda da listagem)

10.3 Executando um processo em determinada data: at

Podemos dizer ao UNIX que um dado processo deve ser executado em determinada data com o comando **at**. No UNIX System V esse comando é de uso apenas do supervisor por questões de segurana.

Formato:
$ at hora [mes dia] [dia_semana]
comando
$ at 1505
at> wall mensagem
at> ^D

Este comando faz com que o comando **wall** seja executado às **15:05**. **wall** envia o conteúdo do arquivo de mensagem para todos os usuários conectados **(uso exclusivo do supervisor)**.

$ at 0800 dec 24
at> mail maria < feliznatal
at> ^D
$ at 1700
at> echo Hora de encerrar | wall
at> ^D

10.4 Utilizando comandos at com arquivos de entrada

$ at 8am <arquivo>
$ at 2030 sun <arquivo>
$ at 5am jan 3 <arquivo>
Para mostrar os **jobs** que estão em fila para serem executados :
$ at -l

11 Editor de Textos VI

Editor de texto padrão do UNIX - **virtual editor**

Todos os sistemas UNIX possuem um editor **VI** instalado.
$ vi [opção] <arquivo>
Opções (algumas):

- -r: recupera o arquivo se houver queda do sistema
- -b: modo binário
- -l: modo Lisp
- -R: modo de somente leitura

Os três modos fundamentais são:

- Modo de digitação ("a" ou "i")
- Modo comando interno (<esc>)
- Comando na última linha (<esc>":")

Os **Sistemas Operacionais Unix**, como por exemplo o **FreeBSD** e **Linux**, e outros, possuem um **help on-line** do editor de textos **vi**.

Para obter ajuda sobre o **"modo de comando interno"** digite **<ESC>:** e escreva **"help"** .

11.1 Comandos internos - vi

*** Teclar <ESC> e a letra correspondente**:

h move o cursor para a esquerda

l move o cursor para a direita

j move o cursor para baixo

k move o cursor para cima

^f move uma tela para frente

^b move uma tela para trás

a insere caracter à direita do cursor

A insere caracter no final da linha

i insere caracter à esquerda do cursor

I insere caracter no início da linha

O insere linha acima do cursor

o insere linha abaixo do cursor

x apaga caracter(es)

dw apaga palavra(s)

dd apaga linha(s)

s substitui caracter

cw substitui palavra

/string procura string

n procura nov ocorrência

ZZ sai do editor e salva o arquivo

11.2 Comandos da última linha - vi

*** Teclar <ESC>: e o comando correspondente**

set num enumera o texto

set nonu retira numeração do texto

5,10 d apaga da linha 5 até a linha 10

1,2 co 4 copia linhas 1 e 2 para depois da linha 4

4,5 m 6 move linhas 4 e 5 para depois da linha 6

g/sting_procurada/s//string_substituta/gc ——-> substitui palavras

gc no final da linha vai solicitar a confirmação da substituição.

1 posiciona o cursor na primeira linha do texto

$ posiciona o cursor no final do texto

r insere arquivo na posição do cursor

u desfaz ultima alteração

w salva o arquivo e continua editando

q! sai do editor de textos, sem salvar o arquivo

x sai do editor de textos e salva o arquivo

12 Editor de textos PICO

Editor de Textos simples e de fácil utilização. Comumente encontrado em Sistemas UNIX. Também apresenta versão para Sistemas Operacionais Windows 9x. Pode ser obtido no seguintes sites:

http://www.washington.edu/pine/
http://www.math.fu-berlin.de/~guckes/pico/
Para acessar o editor :
$ pico <arquivo>
Comandos internos :
<ctrl><g>: help
<ctrl><o>: salva o arquivo
<ctrl><r>: inclui um arquivo na posição do cursor
<ctrl><y>: página anterior
<ctrl><v>: página posterior
<ctrl><k>: apaga linhas
<ctrl><u>: volta as linhas que foram apagadas
<ctrl><w>: procura uma palavra
<ctrl><c>: mostra a posição do cursor
<ctrl><x>: sai do editor

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

13 Cliente de E-Mail PINE

Programa cliente para **Internet News e E-Mail**

O PINE é uma ferramenta orientada a telas para o manuseio de mensagens **(News e E-Mail)**. Tem um número limitado de funções para usuários novos e pode ser ampliado para os usuários mais experientes, além de poder personalizar algumas preferências.

As caracterísisticas básicas são:

- Ler, Salvar, Exportar, Apagar, Imprimir, Responder e Redirecionar mensagens.
- O editor de mensagens utilizado é o PICO.
- Podem ser usados os recursos de FOLDERS, para armazenamento das mensagens.

Possui algumas ferramentas bastante práticas:

- Folders para mensagens
- Livro de Endereos (Address Book)

- Checagem de mensagens em intervalos pré-determinados

O Pine suporta os seguintes protocolos:

SMTP Simple Mail Transport Protocol

NNTP Network News Transport Protocol

MIME Multipurpose Internet Mail Extensions

IMAP Internet Message Access Protocol

Com o **MIME** pode-se utilizar os recursos de multimídia, anexar objetos de multimídia às mensagens, tais como: documentos formatados, planilhas, arquivos de imagens, etc.

Pode ser usado como cliente de servidores locais ou remotos.

Com o **Pine** no ambiente **UNIX**, o **SENDMAIL** (servidor de E-Mail dos Sistemas UNIX) é geralmente utilizado para as funções de Correio Eletrônico. Porém, pode-se optar pela utilização direta do **SMTP**.

Há vários níveis de configuração do **Pine**:

- **Default** - São as configurações básicas após a instalação do PINE.
- **Através do arquivo pine.conf** - São as configuraões estabelecidas no arquivo pine.conf.
- **Configuração personalizada através od arquivo .pinerc** - Neste arquivo as configuraões são definidas pelo usuário
- **Opções com comandos de linha** - Envio de mensagens através de comandos diretos pelo terminal, sem executar o programa Pine e usar seus menus.
- **Um grande sistema** - Com um único arquivo de configuração para todos os usuários do sistema.

O **Pine**, para o **UNIX**, utiliza as seguintes variáveis:

- TERM DISPLAY - Determina se o Pine pode mostrar arquivos de imagens atachadas.
- SHELL - se não for definido, utiliza o /bin/sh.
- MAILCAPS - delimitado por ponto e vírgula, lista os caminhos dos arquivos de e-mail.

Os arquivos relacionados ao Pine no sistema UNIX:

/var/spool/mail/xxxx - Localização dos arquivos de e-mail.

~/mail - Diretório dos arquivos de e-mail.

~/.addressbook - Arquivo livro de endereços de e-mail.

~/.addressbook.lu - Arquivo índice do livro de endereços de e-mail.

~/.pine-debug[1-4] - Log de diagnóstico.

~/.pinerc - Arquivo de configuração pessoal.

~/.newsrc - Arquivo de inscrição/estado do News.

~/.signature - Arquivo de assinatura.

~/.mailcap - Arquivo de localização dos programas para interpretar os e-mails.

~/.mime.types - Arquivo pessoal para mapeamento dos recursos de MIME

/etc/mailcap - Arquivo do sistema que especifica a localização dos programas para interpretar os e-mails com recursos de áudio, imagem, textos formatados, etc.

/etc/mime.types - Ídem para os arquivos com recursos de MIME.

/usr/lib/pine.conf - Arquivo de configuração do Pine, genérico.

/usr/lib/pine.conf.fixed - Arquivo de configuração do Pine, genérico.

/tmp/.\var\spool\mail\xx - Arquivos de "lock", por folder.

~/.pine-interrupted-mail - Mensagem que foi interrompida.

~/mail/postponed-msgs - Para mensagens que serão enviadas posteriormente

~/mail/sent-mail - Folder de E-Mails enviados.

~/mail/saved-messages - Folder de E-Mails salvos.

O Pine é um aplicativo amplamente utilizado na Internet. Está disponível pela Internet as versões para Windows 3.x/95/98/NT, UNIX e outras plataformas.

O site oficial do pine é: http://www.washington.edu/pine

14 Cliente de E-Mail

Sistema Cliente para Correio Eletrônico (E-Mail)

Todo sistema UNIX possui este programa.
Para utilizar, no **UNIX**, digite **mail** e **<ENTER>**
Serão listadas todas as correspondências recebidas.
Para ler as mensagens: digite o número da mensagem e **<ENTER>**
As mensagens ficam no servidor de E-Mails, no diretório **/var/spool/mail/usuario**

Principais comandos:

? help

h mostra o cabealho das mensagens

m envia uma mensagem

r responde a mensagem corrente

d marca mensagem para ser apagada

s salva a mensagem em um arquivo

~v edita a mensagem utilizando o editor vi

~f adiciona a mensagem corrente ao responder a uma mensagem

~r inclui um arquivo na mensagem corrente

x sai do utilitário Mail, não envia as mensagens para o arquivo "mbox" e não apaga as mensagens marcadas para deleção.

q sai do utilitário Mail, envia as mensagens para o arquivo "mbox" e apaga as mensagens marcadas para deleção.

Exemplo do comando mail quando temos mensagens no mbox (diretório /var/spool/mail/usuario) :

$ mail
> 1 aluno@curso.com.brbr Teste do Curso
> 2 xyzw@dominio.com.br Curso Novo
& more 2 —> vai exibir a mensagem número 2
Para enviar uma mensagem:
$ mail usuario@dominio.com.br
Subject: <assunto_da_mensagem>

A partir desse momento você deverá começar a escrever a mensagem. Para enviar a mensagem basta digitar <CTRL><D>, ou colocar um ponto (.) numa linha, apenas o ponto, e em seguida teclar <ENTER>.

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

15 Cliente de E-Mail ELM

Sistema interativo de Correio Eletrônico (E-Mail)

É de fácil utilização e está presente em quase todos os Sistemas UNIX. Sua configuração permite que se escolha o editor de textos com que se vai trabalhar para editar as mensagens, além de outros recursos.

Para iniciar o programa:
$ elm
Aparecerá um índice com os cabeçalhos das mensagens e um **mini-menu.**
Para ler uma mensagem :
1 - Posicionar a barra de seleção sobre a mensagem e teclar **<ENTER>**
ou
2 - Informar o número da mensagem na linha de comando e teclar **<ENTER>**

Para enviar uma mensagem: Teclar **"m"** na linha de comando
Status das mensagens:
N)ew: mensagem nova
O)ld: mensagem antiga mas ainda não lida
D)elete: mensagem marcada para deleção
o) Opções de configuração

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

16 Manual dos comandos do UNIX

O Sistema Operacional UNIX, e os que são derivados do UNIX (Linux, FreeBSD, etc), possuem muitos comandos e aplicativos. Nem sempre é possível utilizar todos os comandos sem saber como.

As distribuições de UNIX já vêm com os manuais dos comandos incorporados ao Sistema.

Estes manuais de comandos se localizam no diretório **/usr/man.**

As páginas do manual são tradicionalmente divididas em 8 seções principais, a saber:

1. Comandos de usuário - **(/usr/man/man1)**

2. Chamadas de sistema - **(/usr/man/man2)**

3. Subrotinas de programação - **(/usr/man/man3)**

4. Dispositivos - **(/usr/man/man4)**

5. Formatos de arquivos - **(/usr/man/man5)**

6. Jogos - **(/usr/man/man6)**

7. Miscelânea - **(/usr/man/man7)**

8. Administração do sistema - **(/usr/man/man8)**

Para exibir a descrição de um comando do sistema **UNIX**, use a seguinte linha de comando:

$ man comando

Por exemplo:

$ man ls —> Vai exibir o manual do comando ls (list).

Um modo de procura por palavra-chave pode ser usado para encontrar o nome do comando que execute uma determinada tarefa. O formato da linha de comando para a procura da palavra-chave é:

$ man -k palavra-chave

No LINUX os comandos também podem ser usados de forma a exibir suas opções: $ comando –help

Exemplos:

$ ls –help —-> vai mostrar a lista de opções do comando ls

$ cat –help —-> vai mostrar a lista de opçoes do comand cat

Antes de descrever alguns comandos do UNIX, vale ressaltar :

1. A listagem a seguir não é única, pois há muitos outros comandos no **UNIX**.
2. Para o ambiente **X11** (**Ambiente gráfico**) muitos comandos iniciam com a letra **x** (**xis**), por exemplo: **xterm, xhost, xlock, xclock, xosview**, etc.
3. O comando **xman** descreve os manuais de todas de comandos do **UNIX**, como será descrito no comando **man**.
4. Como alternativa, há aplicativos **CGIs** específicas para a descrição de cada comando do **UNIX**. Uma delas pode ser encontrada no seguinte **URL**: **www.de.FreeBSD.org/de/cgi/man.cgi**

5. Para uma lista completa de todos os comandos **UNIX**, tenha sempre um bom manual de referências e guias de comandos, que podem ser encontrados no site **www.conectiva.com.br** e em muitos outros sites pela **Internet.**

6. A listagem dos comandos está em ordem alfabética.

16.1 Comando apropos

Mostra informações sobre um assunto. Equivalente a man -k.

Sintaxe: apropos [chave ...]

Exemplos:

$ apropos directory

$ apropos fsck

16.2 Comando bg

Põe um processo em execução em background.

Sintaxe: bg [%id]

Utilização: Comando interno do **shell**. Quando um processo está ocupando o terminal e digitamos **^Z (CTRL-Z)** o processo recebe um sinal SIGSTOP e a shell retornará o controle do terminal ao usuário. bg irá colocar um processo que está parado no modo de execução em **background** (em segundo plano) e o processo poderá continuar sua execução.

Pode-se iniciar um processo já no modo de execução em segundo plano se após o comando acrescentarmos o caractere **'&'**. No caso de haver vários processos parados podemos identificar qual processo desejamos colocar em background, especificando seu "nome" após o caractere **'%'**. O "nome" do processo será indicado pela shell assim que executarmos um comando em modo **background (com '&')** ou interrompermos um processo com **^Z**.

Para retornar um processo ao modo de execução em foreground utiliza-se o comando **fg**.

Uma lista de processos ativos pode ser obtida com o comando jobs.

Obs: Um processo em background geralmente pode escrever sua saída no terminal, mas interromperá sua execução se precisar ler dados da entrada padrão (stdin, geralmente o terminal). **Exemplos:**

$ find / -print > result.txt

^Z (Usuário digita CTRL-Z e o processo é interrompido) [1]+

Stopped find / -print > result.txt

$ bg %1 [1]+ find / -print > result.txt ...Processo continua sua execução em background...

$ ...controle retorna ao usuário...

$ du -k / &

16.3 Comando cal

Imprime o calendário para um determinado mês/ano.

Sintaxe: cal [[mês] ano]

Parâmetros:

mês - Mês para o qual se quer o calendário, deve preceder ano.

ano - O ano para o qual se quer o calendário.

Exemplos:

$ cal 7 1974

$ cal 1974

Obs: Se invocado sem parâmetros cal **irá imprimir o calendário do mês atual (data do sistema).**

16.4 Comando cat

Concatena e mostra arquivos.

Sintaxe: cat [arquivo ...]

Exemplos:

$ cat > texto.txt

$ cat texto1 texto2 > capitulo1

$ cat arq1 - arq2 > arq3

16.5 Comando chmod

Altera permissões de acesso de arquivos.

Sintaxe: chmod [-R] <modo> <arquivo> [arquivo ...]

Parâmetros:

-R Se o arquivo especificado for um diretório, muda recursivamente o modo de acesso de todos seus arquivos e subdiretórios.

modo Pode assumir forma absoluta ou simbólica, como descrito a seguir:

Modo simbólico:

O modo simbólico é uma lista de expressões da forma[identificador ...] operando [valor] separada por vírgulas.

Identificadores:

u permissões para o dono do arquivo.

g permissões para o grupo do arquivo..

o permissões para outros grupos.

a todos os anteriores (all). Default se o identificador for omitido.

Operandos:

+ Adiciona permissão às existentes no arquivo.

- Retira permissão das presentes no arquivo.

= Assinala explicitamente uma permissão (zerando as outras).

Valores:

r Permissão para leitura.

w Permissão para escrita.

x Permissão para execução.

X Permissão para execução se o arquivo for um diretório ou já houver permissão de execução.

s Bit setgid se atribuído a g, setuid se atribuído a u.

t Bit sticky.

Modo absoluto:

As permissões neste modo são representadas por um número octal de quatro dígitos, da forma EUGO.

Dígitos:

E Atributos especiais

U Permissões para o dono do arquivo.

G Permissões para o grupo do arquivo.

O Permissões para outros grupos.

Para os dígitos UGO temos a seguinte interpretação:

0 Nenhuma permissão

1 Permissão de execução

2 Permissão de escrita.

3 Permissão de execução e escrita.

4 Permissão de leitura.

5 Permissão de execução e leitura.

6 Permissão de leitura e escrita.

7 Permissão de leitura, escrita e execução

Para o dígito E temos a seguinte interpretação:

0 Nenhum atributo especial ligado.

1 Bit sticky ligado.

2 Bit sgid ligado.

3 Bits sticky e sgid ligados.

4 Bit suid ligado.

5 Bits sticky e suid ligados.

6 Bits suid e sgid ligados.

7 Bits sticky, suid e sgid ligados.

Atributos especiais:

setuid - O arquivo é executado como se fosse invocado pelo proprietário, não faz sentido para diretórios.

setgid - O arquivo é executado sob seu grupo, mesmo se o usuário invocador não participar dele; todo arquivo criado em um diretório setgid é criado com o mesmo grupo do diretório.

sticky bit - Um arquivo criado sob um diretório com o bit sticky ligado pode ser apagado apenas por seu proprietário.

A interpretação do sticky bit pode variar entre sistemas Unix.

Exemplos:

$ chmod u+x meu_script

$ chmod ug+rw meu_script

$ chmod u+wx,g-w,o=r meu_script

$ chmod 750 helloworld

$ chmod 0750 helloworld

$ chmod 1777 ~usuario/PUB/

$ chmod 711 ~

Obs:

No modo absoluto, os zeros à esquerda são ignorados.

No modo simbólico só faz sentido omitir valor utilizando o operador = para zerar os bits de permissão.

Apenas o superusuário (root) pode alterar os atributos de um arquivo de outro usuário.

16.6 Comando clear

Limpa o terminal

Sintaxe: clear

16.7 Comandos de compressão e descompressão de arquivos

Sintaxe:

gzip [arquivo ...]

gunzip [arquivo ...]

compress [arquivo ...]

uncompress [arquivo ...]

compress e **uncompress** - são os comandos do UNIX para comprimir e descomprimir arquivos.

gzip e **gunzip** - são os comandos da GNU para compressão e descompressão.

zip e **unzip** - são portes para UNIX do pkzip e pkunzip respectivamente.

16.8 Comando cp

Copia um ou vários arquivos.

Sintaxe: cp [-ipr] <arquivo> [arquivo ...] <destino>

Parâmetros:

-i Pede confirmação para cada arquivo a ser copiado.

-p Mantém na cópia as datas de modificação e permissões do arquivo original.

-r Copia recursivamente arquivos e diretórios. Neste caso destino deve se referir a um diretório.

Obs: Quando vários arquivos estão sendo copiados destino deve se referir a um diretório. Exemplo: cp -r ~leonardo/html/ /www

16.9 Comando date

Mostra a data e hora atuais do sistema.

Sintaxe: date [-u]

Parâmetros:

-u Universal time. Date irá mostrar o horário de Greenwich.

16.10 Comando df

Mostra dados de ocupação dos sistemas de arquivo especificados ou do sistema de arquivo onde residem os arquivos passados como parâmetro. Sintaxe: df [-k] [arquivo...] [filesystem...]

Parâmetros:

-k Mostra a ocupação em kilobytes ao invés de blocos de disco (algumas versões de df).

Exemplo: df /tmp / /var

Obs: Chamado sem parâmetros df mostra a ocupação de todos os sistemas de arquivos montados (mounted).

16.11 Comandos unix2dos e dos2unix

Converte conjunto de caracteres, e sequências de quebra de linha em arquivos texto para o formato do DOS e do Unix.

Sintaxe: dos2unix <original> <convertido> unix2dos <original> <convertido>

Exemplo:

$ unix2dos install.txt

16.12 Comando du

Mostra a quantidade de disco utilizada

Sintaxe: du [-ask] [arquivo ...]

Parâmetros:

-a Mostra uma entrada para cada arquivo que não é um diretório.

-s Mostra apenas o valor total de ocupação para cada diretório especificado.

-k Mostra a ocupação em kilobytes ao invés de blocos de disco (algumas versões de du).

Obs: Se um argumento é um diretório du irá mostrar a ocupação de seus subdiretórios recursivamente.

Exemplo:

$ du -s ~aluno

16.13 Comando echo

Escreve no terminal.

Sintaxe: echo [string ...]

16.14 Editor de Textos GNU EMACS

Editor de textos "free" da Free Software Foundation (FSF).

Para acessar o tutorial on-line do emacs e começar a aprender usando, chame o editor e digite a sequência de teclas CTRL+h t (A tecla CTRL e 'h'simultaneamente, depois a tecla 't').

Comandos Básicos:

Setas do teclado *GERALMENTE* Movimentam o cursor

CTRL-v - Avana uma página

CTRL-p - Vai para a linha anterior

CTRL-n - Vai para a próxima linha

CTRL-f - Vai para a direita

CTRL-b - Vai para a esquerda

ESC-v - Volta uma página

CTRL-e - Vai para o fim da linha

CTRL-a - Vai para o começo da linha

CTRL-k - Apaga até o final da linha

CTRL-x CTRL-f - Abre um arquivo

CTRL-x CTRL-s - Salva um arquivo

CTRL-x CTRL-c - Sai do editor

CTRL-d - Apaga o caractere embaixo do cursor

CTRL-x - u Undo! Desfaz as últimas aões..

CTRL-y - Paste.

16.15 Comando fg

Põe um processo em execução em foreground.

Sintaxe: fg [%id]

Utilização: fg é um comando interno da shell. Quando um processo está parado ou em modo de execução em background (veja bg) o comando fg irá transferir o controle do terminal para o processo.

Se mais de um processo estiver parado ou em background, pode-se especificar qual processo colocar em foreground indicando seu "nome" após o caractere ’%".

Uma lista de processos ativos e seus respectivos identificadores pode ser obtida pelo comando jobs.

Se nenhum identificador de processo for especificado o último processo executado em background será colocado em foreground.

Exemplo:
$ vi & <— processo iniciado em background.
[1] xxx (<— nome e número do processo em background)
[1]+ Stopped (tty output)
$ fg %1
... vi continua a executar, agora em foreground...

16.16 Comando find

Percorre recursivamente os diretórios especificados mostrando arquivos com as características desejadas.

Sintaxe: find [diretório ...] [expressão...]

Parâmetros:

-expressão - As seguintes primitivas podem ser combinadas para definir expressões

-group x - Verdadeiro se o arquivo pertence ao grupo x.

-ok c - Similar a -exec, porém pedindo confirmação prévia.

-name x - Verdadeiro se x coincide com o nome do arquivo corrente.

-newer x - Verdadeiro se o arquivo foi modificado mais recentemente do que o arquivo x.

-perm [-]modo - Verdadeiro se coincidirem as permissões do arquivo e modo. Se precedido por - apenas os bits ligados de modo serão comparados.

-print - Sempre verdadeiro. Imprime o nome do arquivo corrente.

-prune - Não descende subdiretórios.

-size n[c] - Verdadeiro se o tamanho do arquivo for de n blocos. Se seguido pela letra c o tamanho é interpretado em bytes.

-type t - Verdadeiro se o tipo do arquivo corresponder a t, onde t é um entre:

b Block device.

c Character device.

d Diretório.

l Link.

p Arquivos comuns (plain files).

f FIFO.

a Socket.

-user x - Verdadeiro se o arquivo pertence ao usuário x.

(exp) - Verdadeiro se a expressão entre parênteses for verdadeira.

Parênteses são interpretados pela shell se não forem precedido por \.

Operadores: Pode-se combinar diversas expressões usando os seguintes

- operadores (em ordem decrescente de precedência)
- ! exp Operador de negação (o caractere ! é interpretado pela shell e deve ser precedido por \ ).
- exp -and exp Operador ’E’ lógico. Retorna verdadeiro se ambas as expressões forem verdadeiras.
- exp -a exp Algumas versões de find usam -a no lugar de -and.
- exp -or exp Operador ’OU’ lógico. Retorna verdadeiro se pelo menos uma das duas expressões forem verdadeiras.
- exp -o exp Algumas versões de find usam -o no lugar de -or.

Exemplos:

```
$ find ~aluno \! -name "*.c" -print
$ find ~ -perm -100 -print
```

16.17 Comando finger

Mostra informações sobre usuários locais ou remotos.

Sintaxe: finger [-lmsp] [usuário ...]

Parâmetros:

-l Saída em formato longo.

-m Casa apenas o nome de login do usuário e não seu nome completo.

-s Saída em modo simplificado (short).

-p Não imprime o arquivo .plan do usuário.

Exemplos:

$ finger root

$ finger aluno@dominio.com.br

Obs:

Usuários podem ser especificados como userids (ou login names) para usuários locais, ou usuario@nome.da.maquina para usuários de máquinas remotas.

16.18 Comando ftp

Transfere arquivos entre máquinas da rede.

Sintaxe: ftp [-iv] [maquina]

Parâmetro:

-i - Não pede confirmação antes da transferência de arquivos múltiplos. Veja o comando prompt abaixo.

-v - Habilita saída descritiva. Default se a entrada padrão estiver associada a um terminal.

Comandos:

ascii - Ativa o modo de transferência adequado à transferência de arquivos texto.

bell - Soa um alarme a cada transferência de arquivo completada.

binary - Ativa o modo de transferência adequado à transferência de arquivos binários.

bye - Encerra a sessão com o servidor remoto e sai do programa ftp. Um caractere EOF (geralmente ^D) tem o mesmo efeito.

cd dir_remoto - Muda o diretório de trabalho na máquina remota para dir_remoto.

chmod modo arquivo_remoto - Muda o modo de acesso de arquivo_remoto para aquele especificado por modo.

close - Termina a sessão com o servidor remoto e retorna ao interpretador de comandos.

delete arq_remoto Remove o arquivo arq_remoto na máquina remota.

dir [dir_remoto] [arq_local] - Mostra o conteúdo do diretório dir_remoto, opcionalmente colocando a saída no arquivo arq_local.

get arq_remoto [arq_local] - Transfere o arquivo da máquina remota arq_remoto para a máquina local, renomeando-o para arq_local caso este parâmetro seja fornecido.

hash - Mostra um caractere # (hash) para cada bloco de dados recebido (1Kbyte).

help [cmd] - Mostra uma mensagem informativa sobre o comando cmd. Se nenhum comando for especificado lista todos os comandos disponíveis.

lcd [dir_local] - Muda o diretório de trabalho na máquina local. Se dir_local não for especificado o diretório HOME do usuário será usado.

ls [dir_remoto] [arq_local] - Similar a dir, porém incluindo informações dependentes de sistema fornecidas pelo servidor.

mdelete [arqs_remotos] - Similar a delete para múltiplos arquivos.

mget arqs_remotos - Similar a get, para múltiplos arquivos.

mkdir dir_remoto - Cria diretório na máquina remota.

mput arqs_locais - Copia os arquivos dados por arqs_locais para o diretório de trabalho corrente na máquina remota.

newer arquivo - Copia um arquivo somente se a versão remota for mais recente do que a versão local, cujo nome é dado por arquivo.

open host [porta] - Abre uma conexão com o host , na porta especificada.

prompt - Liga ou desliga o modo interativo, onde ftp pede confirmação antes de transferir ou deletar múltiplos arquivos.

put arq_local [arq_remoto] - Armazena um arquivo na máquina remota, opcionalmente com o nome dado por arq_remoto.

pwd - Imprime o diretório corrente da máquina remota.

rename nome_velho novo_nome - Muda o nome do arquivo remoto chamado nome_velho, chamando-o de novo_nome.

rmdir diretório - Remove o diretório na máquina remota.

user user-name - Identifica o usuário no servidor ftp, se uma senha de acesso for necessária o servidor irá pedi-la.

verbose - Aciona o modo detalhado de apresentação. Default se os comandos estiverem sendo entrados de um terminal.

? [comando] O mesmo que help.

Exemplos:

$ ftp sunsite.unc.edu

$ ftp -niv ftp.linux.org < comandos.txt &

16.19 Comando grep

Encontra ocorrências de um padrão dentro de arquivos.

Sintaxe: grep [-cilnsv] &ltpadrão> [arquivo ...]

Parâmetros:

-c - Mostra apenas o número de linhas que continham o padrão.

-i - Não diferencia letras maiúsculas de minúsculas.

-l - Mostra apenas os nomes dos arquivos que continham padrão.

-n - Mostra cada linha que continha padrão precedida por seu número dentro do arquivo.

-s - Não mostra mensagens de erro produzidas por tentativas de acesso a arquivos.

-v - Mostra apenas as linhas que não continham o padrão especificado.

Exemplos:

$ grep -ci ’Jose’ /etc/passwd

$ grep -l ’aluno’ html/*

Obs:

Recomenda-se que padrão apareça entre aspas simples (’) pois alguns caracteres (notadamente $, * , [, ^ , | , (,) e \ ) têm significado especial para a shell e podem ser interpretados erroneamente.

16.20 Comando jobs

Lista processos em execução pela shell.

Sintaxe: jobs [-l]

Parâmetro:

-l - Lista também o número de cada processo.

16.21 Comando kill

Envia um sinal a um processo.

Sintaxe: kill [-sinal] [processo ...]
kill -s sinal [processo ...]
kill -l [sinal]

Parâmetro:

sinal - Pode ser tanto o número do sinal como seu nome.

-s - Em alguns sistemas é necessária a presença da opção -s para se especificar o sinal que se quer enviar.

-l - Quando existente lista todos os nomes e números de sinais.

Alguns sinais comuns:

$ kill -l

QUIT

ABRT

KILL

ALRM

TERM

Exemplos:

$ kill -HUP 1

$ kill -9 %2

Obs: Sua sintaxe varia um pouco de sistema para sistema e entre shells diferentes.

- Em alguns sistemas pode ser encontrado como um comando separado da shell.

- A lista de processos pode ser uma lista de números ou de nomes de processos.

- O sinal SIGTERM (15) é enviado por default.

- Somente o superusuário pode enviar sinais a processos de outros usuários.

16.22 Comando logout

Termina a seção do usuário (função interna da shell).

Sintaxe: logout

16.23 Comando ln

Cria links a arquivos ou diretórios.

Sintaxe: ln [-fs]
ln [-fs] [arquivo ...]

Parâmetros:

-f - Cria o link mesmo se o arquivo destino não exista ou não estiver acesível.

-s - Cria um link simbólico (soft link).

Obs:

ln pode criar tanto links simbólicos (soft links) como diretos (hard links)

ln cria links diretos por default.

16.24 Comando ls

Lista conteúdo de diretórios.

Sintaxe: ls [-aAcCdFLqrRtu1] [arquivo ...]

Parâmetros:

-a - Inclui entradas do diretório cujos nomes começam com ".", normalmente omitidas.

-A - Lista todas as entradas, exceto "." e "..".

-c - Usa a data de modificação do arquivo para a ordenação ou impressão.

-C - Formata a saída em múltiplas colunas, default quando a saída é o terminal.

-d - Trata diretórios como arquivos comuns e não segue links simbólicos.

-F - Diferencia os tipos de arquivos concatenando caracteres a seus nomes:

/ Diretórios

***** Arquivos executáveis

@ Links simbólicos

= Sockets

| Pipes

-l - Lista usando o formato longo.

-L - Se o argumento for um link simbólico lista o arquivo ou diretório referenciado.

-q - Impressão de caracteres não gráficos nos nomes dos arquivos como pontos de interrogação (?).

-r - Reverte a ordenação para obter ordem alfabética reversa ou arquivos mais antigos primeiro.

-R - Lista recursivamente o conteúdo dos diretórios encontrados.

-t - Ordena os resultados cronologicamente.

-u - Usa a data do último acesso ao arquivo para a ordenação ou impressão.

-1 - Mostra um elemento por linha de saída.

Exemplos:

$ ls -la /etc/passwd

$ ls /etc

$ ls -lad /etc

$ ls -la /etc

$ ls -lR /dev | more

Obs:

O formato longo mostra informações sobre o tipo de arquivo e suas permissões de acesso como uma string de dez caracteres, onde o primeiro identifica o tipo dos arquivos da seguinte forma:

b Block device

c Character device.

d Diretórios.

l Links.

p FIFO (named pipe)

s Socket da família AF_UNIX

- Arquivo comum (plain file)

Os outros nove caracteres referem-se cada um a um bit de permissão de acesso, os três primeiros às permissões do proprietário, os três seguintes às permissões do grupo do arquivo e os três últimos às permissões de acesso para outros grupos.

Podem assumir os seguintes valores:

r permissão de leitura.

w permisso de escrita.

x permisso de execução.

- bit de permissão desligado.

16.25 Comando man

Consulta os manuais on-line do sistema

Sintaxe: man [-fk] [seção] [chave ...]
man [-fk] [-s seção] [chave ...]

Parâmetros:

-f - Mostra descrições de uma linha sempre que chave coincidir com uma entrada do manual.

-k - Pesquisa informações sobre palavras chave nas descrições das páginas dos manuais on-line.

-s - seção. Algumas versões de man necessitam do parâmetro -s se se quiser especificar uma seção dos manuais onde buscar a informação.

Exemplos:

$ man man

$ man ls

$ man -k tape

16.26 Comando mkdir

Cria diretórios.

Sintaxe: mkdir [-p] <diretório> [diretório ...]

Parâmetros:

-p - Cria os diretórios intemediários do path se necessário.

Exemplos:

$ mkdir -p pub/docs/livro

$ mkdir dir1 ../dir2 /usr/local/src/dir3

16.27 Comando more

Mostra arquivos texto página a página.

Sintaxe: more [-csu] [arquivo ...]

Parâmetros:

-c - Escreve a partir do topo da tela, apagando o restante de cada linha até o final.

-s - Substitui múltiplas linhas em branco consecutivas por apenas uma.

-u - Trata caracteres de retrocesso (backspace) e sequências CR-LF de uma maneira especial.

Obs:

A barra de espaço faz more avançar uma página.

ENTER - faz more avançar uma linha.

/ string - Avança até a primeira ocorrência da string dentro do texto.

Para mais comandos interativos consulte o manual on-line (comando man).

Exemplos:

$ ls -la | more

$ more /etc/passwd

$ more arquivo.txt

16.28 Comando mv

Move ou renomeia arquivos e diretórios.

Sintaxe: mv [-fi] <arquivo> [arquivo ...] <destino>

Parâmetros:

-f - Não pede confirmação antes de sobrescrever um arquivo já existente.

-i - Pede confirmação antes de mover um arquivo que irá sobrescrever outro.

Exemplos:

$ mv livro livro.old

$ mv linux/docs/*HOWTO ../linux/

Obs: Se múltiplos arquivos forem especificados, destino precisa necessariamente ser um diretório.

16.29 Comando rm

Remove arquivos.

Sintaxe: rm [-firR] <arquivo> [arquivo ...]

Parâmetros:

-f - Deleta arquivo sem pedido de confirmação.

-i - Pede a confirmação antes de remover cada arquivo.

-r ou -R - Causa a remoção recursiva do conteúdo de um diretório (inclusive o próprio).

Exemplos:

$ rm -rf ~aluno/src/compilador

$ rm -i *.gif *.jpg

16.30 Comando rmdir

Remove diretórios.

Sintaxe: rmdir <diretório> [diretório ...]

Obs: Os diretórios devem estar vazios.

16.31 Comando passwd

Altera a senha do usuário.

Sintaxe: passwd [usuário]

Obs: Apenas o superusuário (root) pode mudar a senha de outros usuários.

16.32 Comando ping

Envia pacotes a máquinas da rede e aguarda respostas.

Sintaxe: ping [-sn] <host>

Parâmetros:

-s - Envia um pacote a cada segundo.

-n - Mostra o número dos hosts ao invés de seus nomes.

16.33 Comando rsh

Executa um comando no host especificado.

Sintaxe: rsh [-l user] <host> [comando]

Parâmetros:

-l user Por default o nome do usuário remoto é o mesmo do local, a opção -l permite executar a shell como outro usuário no host remoto.

Exemplos:

$ rsh -l aluno outro.host.exemplo

$ rsh dracula

Obs: O host remoto deve permitir esta operação.

Se o comando não for especificado, o default é executar uma shell interativa.

16.34 Comando sort

Ordena linhas de arquivos.

Sintaxe: sort [-bcdfimMnru] [-o arq] [-tc] [-Tdir] [arquivo ...]

Parâmetros:

-b - Ignora brancos (espaços e newlines) antes das chaves de ordenação.

-c - Verifica se arquivos já estão ordenados, não ordena.

-d - Considera apenas letras e dígitos como parte das chaves de ordenação

-f - Converte letras minúsculas em maiúsculas antes de efetuar comparações.

-i - Considera parte das chaves apenas os caracteres visíveis num terminal.

-m - Mescla (merge) arquivos já ordenados, não efetua ordenação.

-M - Ordena usando string como nome de mês, implica -b.

-n - Ordena usando valor numérico da string, implica -b.

-o - arq Escreve sua saída no arquivo arq ao invés de stdout.

-t - Reverte a ordenação.

-tc - Usa c como separador de campos. Default=espaos e newlines.

-u - Elimina linhas com campos repetidos. Em conjunto com -m

Exemplos:

$ sort -k 2,2 arquivo

$ sort arq -ro arq.ord

$ cat /etc/passwd|sort -nt: -k 3,3

16.35 Comando su

Troca o ID efetivo do usuário.

Sintaxe: su [-] [user]

Parâmetros:

-Executa uma shell de login, carregando todo o ambiente do usuário.

user O nome do usuário para o qual se quer alternar, se omitido su tomará o usuário root como padrão.

Exemplo:

$ su -l aluno

Obs: su pede confirmação através de senha para efetuar a execução da shell, exceto quando executado pelo superusuário (root).

16.36 Comando talk

Conversa com outro usuário.

Sintaxe: talk usuário [term]

Parâmetro:

term - Indica o terminal ao qual se destina a conversa.

Exemplo:

$ talk noel@polo.norte.org

Obs: Se ambos usuários estiverem na mesma máquina, então usuario é o login da pessoa com quem se quer falar, se alguém estiver em outra máquina o formato a ser utilizado será usuario@nome.da.maquina.

Para interromper a conversa basta pressionar o caractere de interrupção (geralmente CTRL-C). Pode-se impedir o recebimento de mensagenss com o comando mesg.

16.37 Comando tar

Armazena vários arquivos e diretórios dentro de um único arquivo ou dispositivo.

Sintaxe: tar [txcrumphvw] [f arq] [-b blk] [-C dir] [arquivo ...]

Parâmetros:

-c - Cria um novo arquivo tar.

-f arq - Usa o arquivo ou dispositivo arq para armazenamento ou recuperação.

-h - Armazena os arquivos apontados por links, ao invés dos links.

-r - Inclui arquivos ao final (append) de um arquivo tar.

-t - Lista o conteúdo de um arquivo tar.

-u - Somente inclui (append) arquivos que não estão presentes, ou são mais novos do que o arquivo tar.

-v - Lista arquivos processados.

-w - Pede confirmação para cada ação a ser tomada.

-x - Extrai arquivos armazenados em um arquivo tar.

Obs:

Os parâmetros podem ser agrupados logo após o comando, (não necessariamente precedidos por -). Neste caso quaisquer valores associados devem ser passados na mesma ordem dos parâmetros correspondentes. Compare os quatro últimos exemplos.

Exemplos:

$ tar cvf files.tar -C /etc arq1 -C /usr arq2

$ tar cvfb images.tar 512 *.gif

$ tar cvf images.tar -b 512 *.gif

$ tar -cv -f images.tar -b 512 *.gif

$ tar -c -v -f images.tar -b 512 *.gif

16.38 Comando telnet

Abre um canal de comunicação entre duas máquinas através do protocolo TELNET.

Sintaxe: telnet [host [port]]

Parâmetros:

host - Identifica a máquina remota com a qual se quer estabelecer a conexão

port - Identifica a porta através da qual se estabelecer a conexão entre as duas máquinas.

Obs:

Se não forem fornecidos parâmetros, telnet entrará no modo de comando interativo, para alternar para o modo interativo durante uma sessão usa-se o caractere de escape CTRL-] (^]) .

Comandos interativos:

$ open host [port]

$ quit

Exemplos:

$ telnet maquina.dominio.com.br

$ tenet 200.201.202.203

16.39 Comando touch

Atualiza a data de acesso do arquivo.

Sintaxe: touch [-c] <arquivo> [arquivo ...]

Parâmetro:

-c - Não cria o arquivo caso ele não exista.

Obs: Caso o arquivo não exista, touch irá criá-lo vazio por default.

Exemplo:

$ touch helloworld

16.40 Comando uname

Mostra informações sobre o sistema operacional e o hardware.

Sintaxe: uname [-amnrsv]

Parâmetros:

-a - Equivale a se especificar todas as opções.

-m - Mostra o nome da plataforma sobre a qual o sistema está rodando.

-n - Mostra o nome da máquina (hostname).

-r - Mostra o "release level" do sistema operacional.

-s - Mostra o nome do sistema operacional.

-v - Mostra a versão (version level) do sistema operacional.

Obs: Se não forem fornecidos parâmetros, a ação default é imprimir o nome do sistema operacional.

16.41 Comando uniq

Notifica ou filtra a ocorrência de linhas repetidas adjacentes em um arquivo texto.

Sintaxe: uniq [-cdu] [-f n] [-s n] [arqent [arqsai]]

Parâmetros:

-c - Precede cada linha de saida com o número de vezes que a mesma ocorreu na entrada.

-d - Não mostra linhas que não aparecem repetidas na entrada.

-f n - Ignora os n primeiros campos (palavras separadas por espaços, tabs ou newlines) na comparação entre linhas.

-s n Ignora os n primeiros caracteres em cada linha de entrada. Se usado com -f os n primeiros caracteres após os campos ignorados serão ignorados.

-u - Não mostra linhas que se repetem na entrada.

Obs: Argumentos adicionais serão usados como o nome de um arquivo de entrada e um de saida, respectivamente.

16.42 Comando users

Mostra quem está usando o sistema.

Sintaxe: users

16.43 Editor de Textos vi

Editor de textos

Sintaxe: vi [arquivo] Utilização:

Há dois modos de trabalho no vi, o modo de comando e o modo de edição ou entrada.

No modo de comando podemos nos movimentar pelo texto, copiar e deletar linhas e caracteres; o modo de edição onde entramos o texto propriamente dito.

Vários comandos nos levam ao modo de edição, mas apenas a tecla de escape (<esc>) nos leva ao modo de comando.

Pode-se fazer com que um comando se repita automaticamente um determinado número de vezes, basta digitar o número de repetições antes do nome do comando, os comandos que permitem esta característica estarão precedidos por [n], omitir n equivale a digitar 1 antes do comando.

Obs:

Note que alguns caracteres especiais podem utilizar mais de uma posição na tela, e algumas linhas de texto podem utilizar mais do que o número de caracteres mostrados em uma linha do terminal.

Modificações se aplicam a toda a linha de texto sobre a qual o cursor se encontra,e não na linha do terminal.

Comandos:

[n] h - Move o cursor n caracteres para a esquerda.

[n] l - Move o cursor n caracteres para a direita.

[n] k - Move o cursor n caracteres para cima.

[n] j - Move o cursor n caracteres para baixo.

/texto - Procura a cadeia texto no arquivo e move o cursor para sua primeira letra.

[n] a - Insere texto, iniciando após o cursor.

[n] i - Insere texto, iniciando antes do cursor.

[n] o - Insere uma nova linha abaixo do cursor e inicia a inserção de texto.

[n] yy - Copia a linha em que se encontra o cursor.

[n] p - Insere a linha copiada após a linha onde está o cursor (paste).

[n] dd - Elimina a linha onde está o cursor (a linha apagada pode ser inserida novamente com p).

[n] x - Elimina o caracter onde o cursor se encontra.

x - Atualiza o arquivo editado e sai do editor.

:w - nome Grava o texto no arquivo especificado por nome.

:q[!] - Sai do editor de textos. Se o texto foi modificado e não foi salvo o caractere opcional ! fará com que qualquer modificação seja ignorada.

<esc> - Entra no modo de comando ou cancela comandos não terminados.

[n] - Repete n vezes o último comando que modificou o texto.

número - Move o cursor para a linha numeroésima linha.

0 (zero) - Move o cursor para o primeiro caractere da linha.

[n] A - Entra no modo de edição, inserindo o texto ao final da linha.

[n] J - Junta n linhas consecutivas do texto, a partir do cursor.

[n] ~ - Inverte a caixa (case) dos n caracteres seguintes ao cursor.

16.44 Comando whatis

Mostra um sumário de uma linha sobre uma palavra chave.

Sintaxe: whatis [chave ...]

Obs: Equivale a man -f

16.45 Comando who

Mostra quem está logado e o que estão fazendo.

Sintaxe: who [am i]

Parâmetro:

am i - Mostra o nome real do usuário (real user name)

Obs: Se invocado sem argumentos, who irá listar o login de todos usuários com processos associados a terminais, o nome do terminal e dados sobre atividade e tempo de login.

16.46 Comando whoami

Mostra o ID efetivo do usuário.

Sintaxe: whoami

16.47 Comando xhost

Controla acesso externo ao terminal gráfico.

Sintaxe: xhost [[+|-][nome] ...]

Parâmetros:

+host - Adiciona host à lista de acesso.

-host - Retira host de lista de acesso.

+ - Permite acesso a qualquer host (desliga lista de acesso).

- - Permite acesso apenas aos hosts da lista (liga lista de acesso).

Obs:

Se invocado sem parâmetros, xhost mostra o estado atual da lista de acesso (on ou off).

Pode-se especificar um usuário em um host no formato user@host.

Exemplos:

$ xhost +maquina.dominio.br

$ xhost +machine@domain.com

$ xhost -xingu

$ xhost

16.48 Comando xman

Exibe os manuais dos comandos do UNIX

Sintaxe: xman [-font <font>...]

Veja também xman –help

17 Referências Bibliográficas

- 1 - UNIX TOTAL - Dr. Rebecca Thomas, Jean Yates
- 2 - UNIX 5 - Guia do Usuário - The LeBlond Group
- Geoffrey T. LeBlonf, Sheila Blust, Wes Modes
- 3 - Open Sources: Voices from the Open Source Revolution
- O'Reilly - 1st Edition January 1999 - http://www.oreilly.com/catalog/opensources/book/toc.html
- 4 - Sistemas Operacionais - William A. Shay - Makron Books
- 5 - Linux Systems Training - GBdirect Ltd. - 1999 - http://www.linuxtraining.co.uk
- 6 - LINUSP - USP - São Paulo - http://www.linusp.usp.br
- 7 - GUL - Guia de usuários Linux - USP - São Paulo - http://gul.linux.ime.usp.br
- 8 - Linux Brasil - UNICAMP - Campinas - SP - http://linux.unicamp.br
- 9 - Conectiva - Curitiba - PR - http://www.conectiva.com.br
- 10 RedHat Linux - http://www.redhat.org