1

Conceitos Básicos

"Não basta ensinar ao homem uma especialidade, porque se tornará assim uma máquina utilizável, mas não uma personalidade. É necessário que adquira sentimento, um senso prático daquilo que vale a pena ser empreendido, daquilo que é belo, do que é moralmente correto. A não ser assim, ele se assemelhará, com seus conhecimentos profissionais, mais a um cão ensinado do que a uma criatura harmoniosamente desenvolvida. Deve aprender a compreender as motivações dos homens, suas quimeras e suas angústias, para determinar com exatidão seu lugar preciso em relação a seus próximos e à comunidade."
Albert Einstein

Visão Geral

Um sistema operacional, por mais complexo que possa parecer, é apenas um conjunto de rotinas executado pelo processador, de forma semelhante aos programas dos usuários. Sua principal função é controlar o funcionamento de um computador, gerenciando a utilização e o compartilhamento dos seus diversos recursos, como processadores, memórias e dispositivos de entrada e saída.

Sem o sistema operacional, um usuário para interagir com o computador deveria conhecer profundamente diversos detalhes sobre o hardware do equipamento, o que tornaria o seu trabalho lento e com grandes possibilidades de erros. O sistema operacional tem por objetivo funcionar como uma interface entre o usuário e o computador, tornando sua utilização mais simples, rápida e segura.

Funções básicas

Um sistema operacional possui inúmeras funções, mas antes de começar o estudo dos conceitos e dos seus principais componentes é importante saber primeiramente quais são suas funções básicas. Nesta introdução, as funções de um sistema operacional são resumidas em duas, descritas a seguir:

- Facilidade de acesso aos recursos

Para a maioria dos usuários, uma operação como a leitura de um arquivo em disco pode parecer simples. Na realidade, existe um conjunto de rotinas específicas, controladas pelo sistema operacional, responsável pelo acionamento do mecanismo de leitura e gravação da unidade de disco, posicionamento na trilha e setor corretos, transferência dos dados para a memória e, finalmente, informação ao programa da conclusão da operação. Cabe, então, ao sistema operacional servir de interface entre os usuários e os recursos disponíveis no sistema computacional, tornando esta comunicação transparente, além de permitir um trabalho mais eficiente e com menores possibilidades de erros. Este conceito de ambiente simulado, criado pelo sistema operacional, é denominado *máquina virtual* e está presente na maioria dos sistemas modernos.

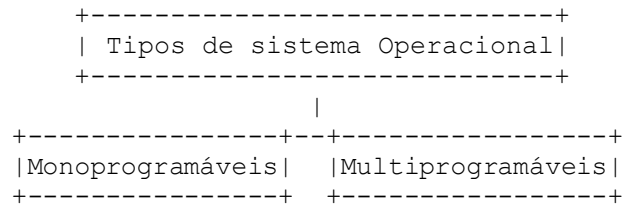
- Compartilhamento de recursos de forma organizada e protegida

Em sistemas onde diversos usuários compartilham recursos do sistema computacional, é necessário controlar o uso concorrente desses recursos. Se imaginarmos uma impressora sendo compartilhada, deverá existir algum tipo de controle para que a impressão de um usuário não interfira nas dos demais. Novamente é o sistema operacional que tem a responsabilidade de permitir o acesso concorrente a esse e a outros recursos de forma organizada e protegida.

Não é apenas em sistemas multiusuário que o sistema operacional é importante. Se pensarmos que um computador pessoal nos permite executar diversas tarefas ao mesmo tempo, como imprimir um documento, copiar um arquivo pela Internet ou processar uma planilha, o sistema operacional deve ser capaz de controlar a execução concorrente de todas essas atividades.

Tipos de sistemas operacionais

Os tipos de sistemas operacionais e sua evolução estão relacionados diretamente com a evolução do hardware e das aplicações por ele suportadas. Muitos termos inicialmente introduzidos para definir conceitos e técnicas foram substituídos por outros, na tentativa de refletir uma nova maneira de interação ou processamento. Isto fica muito claro quando tratamos da unidade de execução do processador. Inicialmente, os termos programa ou job eram mais utilizados, depois surgiu o conceito de processo e subprocesso e, posteriormente, o conceito de thread.



Sistemas monoprogramáveis ou monousuário

Os primeiros sistemas operacionais eram tipicamente voltados para a execução de um único programa. Qualquer outra aplicação, para ser executada, deveria aguardar o término do programa corrente. Os sistemas monoprogramáveis, como vieram a ser conhecidos, se caracterizam por permitir que o processador, a memória e os periféricos permaneçam exclusivamente dedicados à execução de um único programa.

Sistemas multiprogramáveis ou multitarefa

Os sistemas multiprogramáveis ou multitarefa são a evolução dos sistemas monoprogramáveis. Neste tipo de sistema, os recursos computacionais são compartilhados entre diversos usuários e aplicações. Enquanto em sistemas monoprogramáveis existe apenas um programa utilizando os recursos disponíveis, nos multiprogramáveis várias aplicações compartilham esses mesmos recursos.

2

Conceitos de Hardware

Visão Geral

Um sistema computacional é um conjunto de circuitos eletrônicos interligados, formado por processadores, memórias, registradores, barramentos, monitores de vídeo, impressoras, mouse, discos magnéticos, além de outros dispositivos físicos (hardware). Todos esses dispositivos manipulam dados na forma digital, o que proporciona uma maneira confiável de representação e transmissão de dados.

Todos os componentes de um sistema computacional são agrupados em três subsistemas básicos, chamados *unidades funcionais*: CPU (processador), memória principal e dispositivos de entrada e saída. Estes subsistemas estão presentes em qualquer tipo de computador digital, independente da arquitetura ou fabricante.

Processador

O *processador*, também denominado Unidade Central de Processamento (CPU), gerencia todo o sistema computacional controlando as operações realizadas por cada unidade funcional. **A principal função do processador é controlar e executar instruções presentes na memória principal**, através de operações básicas como somar, subtrair, comparar e movimentar dados.

Cada processador é composto por unidade de controle, unidade lógica e aritmética e registradores. A unidade de controle é responsável por gerenciar as atividades de todos os componentes do computador, como a gravação de dados em discos ou a busca de instruções na memória. A unidade lógica e aritmética, como o nome indica, é a responsável pela realização de operações lógicas (testes e comparações) e aritméticas (somas e subtrações).

Os registradores são dispositivos com a função principal de armazenar dados temporariamente. O conjunto de registradores funciona como uma memória interna do processador de alta velocidade, pórem com uma capacidade de armazenamento pequena em comparação a memória principal e custo mais elevado. O número de registradores e sua capacidade de armazenamento variam em função da arquitetura de cada processador.

Memória Principal

A *memória principal* ou *real*, é o local onde são armazenados instruções e dados. A memória é composta por unidades de acesso chamadas células, sendo cada célula composta por um determinado número de bits. O bit é a unidade básica de memória, podendo assumir o valor lógico 0 ou 1.

O acesso ao conteúdo de uma célula é realizado através da especificação de um número chamado *endereço*. O endereço é uma referência única, que podemos fazer a uma célula de memória. Quando um programa deseja ler ou escrever um dado em uma célula, deve primeiro especificar qual o endereço de memória desejado, para depois realizar a operação.

Memória Cache

A *memória cache* é uma memória volátil de alta velocidade, porém com pequena capacidade de armazenamento. O tempo de acesso a um dado nela contido é muito menor do que se este dado estivesse na memória principal. O propósito do uso da memória cache é minimizar a disparidade existente entre a velocidade com que o processador executa instruções e a velocidade com que os dados são acessados na memória principal.

Memória Secundária

A *memória secundária* é um meio permanente, isto é, não-volátil de armazenamento de programas e dados. Enquanto a memória principal precisa estar sempre energizada para manter suas informações, a memória secundária não precisa de alimentação.

O acesso a memória secundária é lento, se comparado com o acesso à memória principal, porém seu custo é baixo e sua capacidade de armazenamento é bem superior. Enquanto a unidade de acesso à memória principal é de milissegundos, o acesso a memória principal é de nanossegundos. Podemos citar, como exemplos de memórias secundárias, a fita magnética, o disco magnético e o disco óptico.

Dispositivos de entrada e saída

Os *dispositivos de entrada e saída* são utilizados para permitir a comunicação entre o sistema computacional e o mundo externo e podem ser divididos em duas categorias: os que são utilizados como memória secundária e os que server para a interface usuário-máquina.

Os dispositivos utilizados como memória secundária (discos e fitas) caracterizam-se por ter capacidade de armazenamento bastante superior ao da memória principal.

Outros dispositivos têm como finalidade a comunicação usuário-máquina, como teclados, monitores de vídeo, mouses e impressoras.

Barramento

O *barramento* ou *bus* é um meio físico de comunicação entre as unidades funcionais de um sistema computacional. Através de condutores, informações como dados, endereços e sinais de controle trafegam entre processadores, memórias e dispositivos de entrada e saída.

Os barramentos são classificados em três tipos: barramentos processador-memória, barramentos de Entrada e Saída e barramentos de backplane. Os barramentos processador-memória são de curta extensão e alta velocidade para que seja otimizada a transferência de informação entre processadores e memórias. Diferentemente, os barramentos de Entrada e Saída possuem maior extensão, são mais lentos e permitem a conexão de diferentes dispositivos.

Arquiteturas RISC e CISC

Cada processador possui um conjunto definido de instruções de máquina, definido pelo seu fabricante. As instruções de máquina fazem referências a detalhes como registradores, modos de endereçamento e tipos de dados, que caracterizam um processador e suas funcionalidades.

Um programa em linguagem de máquina pode ser diretamente executado pelo processador, não requerendo qualquer tipo de tradução ou relocação. Quando escrito em linguagem de máquina de um determinado processador, um programa não pode ser executado em outra máquina de arquitetura diferente, visto que o conjunto de instruções de um processador é característica específica de cada arquitetura.

Um processador com arquitetura RISC (Reduced Instrution Set Computer) se caracteriza por possuir poucas instruções de máquina, em geral bastante simples, executadas diretamente pelo hardware. Na sua maioria, estas instruções não acessam a memória principal, trabalhando principalmente com registradores, que, neste tipo de processador, se apresentam em grande número. Estas características, além de permitirem que as instruções sejam executadas rapidamente, facilitam a implementação da técnica de pipelining (pipelining é a técnica que permite que o processador execute múltiplas instruções em estágios diferentes).

Os processadore com arquitetura CISC (Complex Instrution Set Computer) já possuem instruções complexas que são interpretadas por microprogramas. O número de registradores é pequeno e qualquer instrução pode referenciar a memória principal. Com estas características as instruções demoram mais para serem executadas e a implementação da técnica de pipelining é mais díficil.

3

Introdução ao Unix

Visão Geral

Neste documento vamos abordar os comandos mais utilizados no dia-a-dia de um utilizador de um sistema operacional derivado de Unix. Serão apresentados:

- comandos para manipulação de arquivos;
- comandos de monitoração de sistemas;
- comandos de monitoração de rede;

Antes de falar sobre comandos...

Antes de apresentarmos os comandos para a manipulação de arquivos faz-se necessário a apresentação de algums conceitos:

- O que é um arquivo?
- Permissões de arquivo
- O que é um bit?
- Interpretador de comandos

O que é um arquivo?

Um *arquivo* é constituído por informações logicamente relacionadas. Estas informações podem representar instruções ou dados. Um arquivo executável, por exemplo, contém instruções compreendidas pelo processador, enquanto um arquivo de dados pode ser estruturado livremente como um arquivo texto ou de forma mias rígida como em um banco de dados relacional.

Nos sistemas operacionais derivados do Unix, os arquivos são classificados em tipos. Existem 7 tipos de arquivo, porém como o propósito deste documento é apenas introdutório iremos abordar apenas os seguintes tipos:

- Regular
- Diretório

Arquivos Regulares

São os arquivos que não tem um propósito especial para o sistema operacional. O que vêm a ser um propósito especial? Um sistema operacional é a representação lógica dos dispositivos físicos que existem no sistema computacional. Logo, existem determinados tipos de arquivos que tem como função representar estes dispositivos, por exemplo, para representar o disco rígido (HD), o sistema operacional utiliza um arquivo especial do tipo, dispositivo de bloco.

Os arquivos regulares são geralmente, arquivos de texto ou arquivos binários.

Diretório

O *diretório* é um tipo de arquivo especial, pois tem a capacidade de armazenar dentro de si outros arquivos e/ou diretórios.

Alguns nomes de diretório são padronizados, como o diretório de programas executáveis do sistema (/bin), o diretório de arquivos especiais ligados aos dispositivos de entrada e saída (/dev), o diretório de bibliotecas (/lib) e o diretório que agrupa os diretórios dos usuários (/usr). Geralmente, cada usuário possui seu diretório default de login, denominado diretório /home.

Permissões de arquivo

O acesso a um arquivo é controlado pelo sistema operacional. Para que você execute alguma operação com um arquivo, é necessário que você tenha permissão de acesso para executar tal operação.

As permissões que temos no Unix são três:

- r = read (leitura);
- w = write (escrita);
- x = execute (execução);

O arquivo tem três grupos de permissões:

- user (proprietário);
- group (grupo);
- others (outros);

Como diria Arnaldo César Coelho, a regra é clara, cada grupo pode ter desde nenhuma permissão a todas as 3 permissões. Com a permissão de *read* o usuário pode ler o arquivo, com a permissão de *write* o usuário pode alterar e apagar o arquivo e com a permissão de *execute*, caso o arquivo seja um arquivo executável o usuário pode executar o arquivo.

O que é um bit?

Bit é a abreviação de "binary digit", é a menor unidade possível de representação lógica de informação para o computador. O bit é logicamente representado pelos números 1 e 0. A letra **A** é representada pelo agrupamento de **8 bits** que são **01000001**. Este agrupamento de 8 bits é chamado de **Byte**. Vamos as unidades de medida da informática:

1 Byte	=	8 bits
1 KByte	=	1024 Bytes
1 MByte=	1024 KBytes	
1 GByte=	1024 MBytes	
1 Tbyte	=	1024 GBytes

Interpretador de Comando

O *interpretador de comando* ou *shell*, é um programa que é inicializado logo após o login do usuário. Este programa tem como responsabilidade, captar os comandos enviados pelo usuário, interpretá-los e assim permitir que a ação desejada pelo usuário seja executada.

Finalmente vamos aos comandos...

Todo comando é composto de uma estrutura básica:

```
comando [ parâmetros ] [outros]
```

Sempre que você consultar uma documentação de sistema operacional, o que estiver entre os sinais de [ ] não é obrigatório.

Inicialmente, vamos conhecer os comandos para navegação entre os diretórios do sistema operacional. Em seguida, vamos conhecer alguns comandos para a manipulação de arquivos.

cd – current directory

Sintaxe:

```
cd [ diretório ]
```

O comando cd move você do diretório atual para um outro diretório. Se você não especificar um parâmetro, no caso um diretório, o comando cd move você para o seu diretório home.

pwd – print working directory

Sintaxe:

```
pwd
```

O comando pwd imprimi na saída padrão, o caminho completo (absoluto) do diretório onde você encontra-se atualmente. Todos os diretórios são separados por uma / (barra).

cat – conCATenate files

Sintaxe:

```
cat [ parâmetros ] [ arquivos ]
```

O comando cat lê cada arquivo em seqüência e imprimi-os na saída padrão. Se você não especificar um nome de arquivo, o comando irá ler tudo que for passado para a entrada padrão.

Aproveitando o texto acima, vamos esclarecer o que é entrada e saída padrão. O sistema operacional, assume como entrada padrão o teclado, e assume como saída padrão o monitor. Sendo assim, faríamos da seguinte maneira para criar um arquivo utilizando o comando cat:

```
$ cat > file001.txt [ ENTER ]
Este e' o conteudo do arquivo file001.txt [ ENTER ]
[ CTRL + D ]
```

Vamos analisar a linha de comando acima:

- Como vimos anteriormente o comando cat sem parâmetro nenhum, captura a entrada padrão e imprimi na saída padrão;
- Como não queremos que o conteúdo capturado na entrada padrão seja impresso na saída padrão para com isso criar um arquivo, utilizamos o sinal de maior (>) para redirecionar, mudar o destino da impressão da captura do comando cat, para um arquivo.

ls – list directory contents

Sintaxe:

```
ls [ parâmetros ] [ arquivos ]
```

O comando ls, lista informações sobre os arquivos (do diretório corrente por padrão).

Alguns parâmetros do comando ls:

-l Lista as permissões, número de links, proprietário, grupo, tamanho do arquivo (em bytes), e data e hora da última atualização de cada arquivo.

-a Lista todos os arquivos no diretório incluíndo os arquivos ocultos.

-t Lista os arquivos ordenados de forma decrescente pela data de modificação.

-r Inverte a ordem da listagem apresentada.

-s Mostra o tamanho do arquivo em Kbytes.

Sendo assim, para listarmos o conteúdo de um diretório, basta executarmos o comando ls:

```
$ ls
Documents  public_html
```

Se quisermos uma lista mais detalhada do conteúdo do diretório:

```
 $ ls -l
 total 8
 drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
 drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
```

mkdir – make directories

Sintaxe:

```
mkdir [ parâmetros ] diretório
```

O comando mkdir, cria um ou mais diretórios, se eles não existirem ainda.

Um parâmetro muito utilizado com o comando mkdir:

-p Cria os diretórios em níveis.

Para criar um diretório chamado edinfor001:

```
$ mkdir edinfor001
```

Para verificar se o diretório foi criado:

```
$ ls -l
total 12
drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
drwxr-xr-x    2 sed0148  users        4096 2004-05-03 12:38 edinfor001
drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:14 teste001.ed
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:17 teste002.ed
```

cp – copy

Sintaxe:

```
cp [ parâmetros ] ORIGEM DESTINO
```

O comando cp, copia um arquivo ou diretório de uma determinada origem para um determinado destino. Vale ressaltar que a ORIGEM e o DESTINO são obrigatórios neste comando, como vocês podem ver na sintaxe do comando, as palavras ORIGEM e DESTINO **não** aparecem entre os sinais de **[ ]**.

Alguns parâmetros do comando cp:

-i permite o modo interativo do comando, ou seja, no caso de o comando cp ter que sobrescrever algum arquivo ele perguntará antes se realmente é para sobrescrevê-lo.

-p preserva as permissões, proprietário, grupo e atributos dos arquivos copiados.
-r copia arquivos recursivamente.

Para fazer uma cópia do arquivo teste001.ed para teste002.ed, execute o seguinte comando:

```
$ cp teste001.ed teste002.ed
```

Para verificar o resultado do comando, use o comando ls:

```
$ ls -l
total 8
drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
drwxr-xr-x    2 sed0148  users        4096 2004-05-03 12:38 edinfor001
drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:14 teste001.ed
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:17 teste002.ed
```

mv – move

Sintaxe:

```
mv [ parâmetros ] ORIGEM DESTINO
```

O comando mv, movimenta arquivos de uma determinada ORIGEM para um determinado DESTINO.

Para movimentar o arquivo teste001.ed para o diretório edinfor001:

```
$ mv teste001.ed edinfor001/
```

Para verificar se o arquivo foi movimentado:

```
$ ls -l edinfor001/
total 0
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:14 teste001.ed
```

O comando mv também é utilizado para renomear arquivos. Por exemplo, para renomear o arquivo teste002.ed para teste003.ed, basta emitir o seguinte comando:

```
$ mv teste002.ed teste003.ed
```

Para verificar o resultado do comando:

```
$ ls -l
total 12
drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
drwxr-xr-x    2 sed0148  users        4096 2004-05-03 12:44 edinfor001
drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
-rw-r--r--    1 sed0148  users           0 2004-05-03 12:17 teste003.ed
```

rm – remove

Sintaxe:

```
rm [ parâmetros ] arquivos
```

O comando rm remove arquivos e/ou diretórios. Este comando não criado para remover diretórios seu objetivo inicial é remover entradas em um diretório.

Alguns parâmetros do comando rm:

-i modo interativo, com está opção o comando rm pedirá confirmação para a exclusão

do arquivo.

-r com está opção, o comando rm funcionará recursivamente.

Para removermos o arquivo teste003.ed, emitimos o seguinte comando:

```
$ rm teste003.ed
```

Para verificar o resultado:

```
$ ls -l
total 12
drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
drwxr-xr-x    2 sed0148  users        4096 2004-05-03 12:44 edinfor001
drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
```

rmdir – Remove a directory

Sintaxe:

```
rmdir diretório
```

O comando rmdir, remove um diretório vázio.

Para remover o diretório edinfor001, emitimos o seguinte comando:

```
$ rmdir edinfor001/
```

Para conferir o resultado:

```
$ ls -l
total 8
drwxr-xr-x    2 sed0148  users        4096 2004-04-30 19:00 Documents
drwxr-xr-x    2 sed0148  users        4096 2003-12-17 14:53 public_html
```

Alguns exercícios…

1. Utilizando os comandos que você aprendeu até agora, dentro do seu diretório home, crie um arquivo chamado exercicio001.ed, o conteúdo fica a seu critério.
2. Crie um diretório chamado edinfor001.
3. Crie um diretório chamado edinfor002.
4. Faça uma cópia do arquivo exercicio001.ed dentro do diretório edinfor002.
5. Remova o arquivo exercicio001 do seu diretório home.
6. Movimente o arquivo do diretório edinfor002 para o diretório edinfor001.
7. Remova o diretório edinfor002.

grep – generalized regular expression processor

Sintaxe:

```
grep [ parâmetros ] <item de pesquisa> [ arquivos ]
```

O comando grep, pesquisa um determinado ou um conjunto de caracteres em um ou mais arquivos e imprimi o resultado na saída padrão.

Alguns parâmetros do comando grep:

-i Ignore maiúsculas e minúsculas.
-v Excluí do resultado pesquisado o item da pesquisa.

Para procurar a palavra sed0148 nos arquivos /etc/passwd e /etc/group, emitimos o seguinte comando.

```
$ grep sed0148 /etc/passwd /etc/group
/etc/passwd:sed0148:x:708:100:Rodrigo Nascimento:/home/sed0148:/bin/bash
/etc/group:uucp:x:14:sed0148
/etc/group:dialout:x:16:sed0148
/etc/group:video:x:33:sed0148
```

find – search for files in a directory hierarchy

Sintaxe:

```
$ find [ caminho ] [ expressão ]
```

Com o comando find você pode procurar por qualquer arquivo utilizando uma grande variedade de critérios de busca. Para proporcionar a grande variedade de critérios de busca, o comando find utiliza uma grande quantidade de parâmetros, apresentaremos aqui somente os parâmetros de utilização mais comuns no dia-a-dia.

Alguns parâmetros do comando find:

-name É utilizado para procurar um determinado conjunto de caracteres no nome do arquivo. Por exemplo, caso você esteja procurando um arquivo que inicie com as letras rel, -name rel*.
-user Procura somente os arquivos de um determinado usuário.
-group Procura somente os arquivos de um determinado group.
-inum Procura o arquivo pelo número de i-node.
-type Procura o arquivo pelo tipo.
-exec Executa comandos do sistema operacional com o resultado da procura realizada pelo comando find. Este parâmetro deve ser finalizado com os caracteres {} \;

df – report filesystem disk space usage

Sintaxe:

$ df [parâmetros] [mount-point]

O comando df lista as seguintes informações sobre os filesystems: Nome do filesystem, tamanho do filesystem, quantidade de utilização, espaço disponível, porcentagem utilizada e nome do mount-point.

Um parâmetro muito utilizado com o comando df:

-k Imprime as informações relacionadas a espaço em KBytes.

du – disk usage

Sintaxe:

$ du [parâmetros] [diretórios]

O comando du imprime na saída padrão um relarório sobre a utilização de disco. Por exemplo, com este comando é possível identificar qual a quantidade de espaço em disco utilizado por um diretório, para isso emitimos o seguinte comando:

```
$ du –sk /home/sed0148
104     /home/sed0148
```

Com está saída conseguimos concluir que o diretório /home/sed0148 ocupa em disco 104 KBytes.

Alguns parâmetros do comando du:

-k Imprime as informações relacionadas a espaço utilizando a unidade KBytes.
-s Imprime somente o total de espaço consumido por diretório.

date – print or set the system date and time

Sintaxe:

$ date [parâmetros] [MMDDhhmmYY]

O comando date pode ser utilizado para imprimir na saída padrão informações de data e hora configuradas no sistema, ou ainda, para configurar informações de data e hora no sistema.

Parâmetro com mais utilizado no comando date:

-u Mostra a informação de horário absoluto do sistema.

Por exemplo, para imprimir a data e hora do sistema emitimos o comando:

```
$ date
Tue May 18 13:24:43 BRT 2004
```

Para visualizar o horário absoluto do sistema emitimos o seguinte comando:

```
$ date -u
Tue May 18 16:28:33 UTC 2004
```

chown – change owner and group

Sintaxe:

chown [parâmetros] owner[:group] arquivo

O comando chown altera o proprietário de um ou mais arquivos para um novo proprietário. Este comando permite também a alteração do grupo detentor de permissões relacionadas ao arquivo.

Alguns parâmetros do comando chown:

-f Não imprime mensagens de erro sobre arquivos que não puderam ter o seu proprietário alterado.

-R Executa o comando de modo recursivo, aplicando assim o comando para os sub-níveis do diretório.

chmod – change file access permissions

Sintaxe:

$ chmod [parâmetros] {arquivo | diretório}

O comando chmod altera as permissões de acesso aos arquivos em um sistema operacional Unix. Neste comando o proprietário, o grupo e os outros usuários são representados pelos caracteres **u**, **g**, **o**, respectivamente. Também utilizamos os sinais + e – para dar e tirar permissões.

Alguns parâmetros do comando chmod:

-R Executa o comando de modo recursivo, aplicando assim o comando para os sub-níveis do diretório.

-f Não imprime mensagens de erro sobre arquivos que não puderam ter o seu proprietário alterado.

ps – report process status

Sintaxe:

$ ps [parâmetros]

O comando ps fornece informações sobre os processos que estão sendo executados no sistema operacional.

Alguns parâmetros do comando ps:

-a Imprimi na saída padrão informações sobre todos os processos do sistema, exceto os que não estão associados com o terminal.

-e Imprimi na saída padrão informações sobre todos os processos do sistema, exceto os processos de kernel.

-f Gera uma listagem completa.

-k Lista os processos de kernel.

-u Mostra uma saída orientada ao usuário, isto incluí informações como usuário, process ID, % de utilização de CPU, % de utilização de memória e outras informações.

-x Lista processos que não estão associados com o terminal.

4

Bibliografia

"O que sabemos é uma gota. O que ignoramos é um oceano."
Isaac Newton

Arquitetura de Sistemas Operacionais, 3ª Edição.
Francis Bereger Machado e Luiz Paulo Maia
LTC

Essential System Administrator
Aeleen Frisch
O'Reilly

Linux in a nutshell – A Desktop Quick Reference
Jessica Perry Hekman
O'Reilly

Man Pages
Sistema Operacional SuSE Linux

Man Pages
Sistema Operacional IBM AIX