Cesar Kállas – cesarkallas@gmx.net

Curso GNU/Linux

Realização

CAECOMP

Puc Campinas – 2004

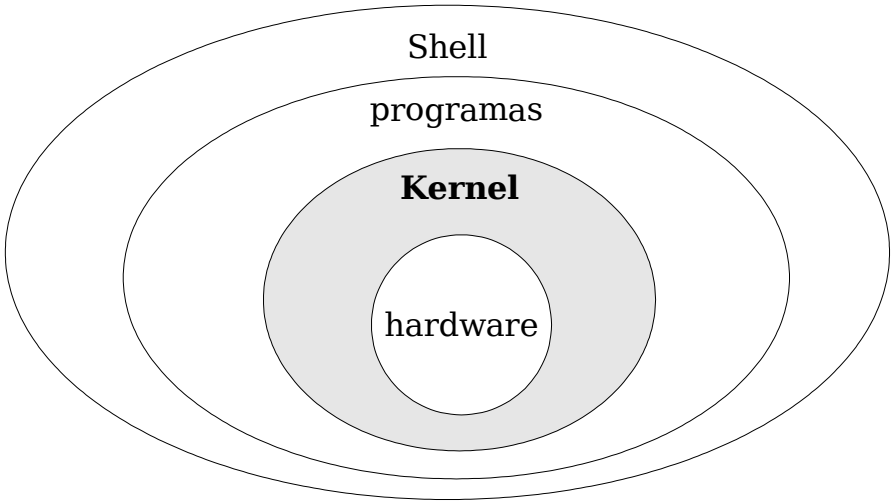
Capítulo 2

2

O Kernel

O kernel (núcleo) é sem dúvida a parte mais importante de um Sistema Operacional, é ele que mantém o contato com a parte física do computador (hardware), gerencia entrada e saída de dados, é o responsável por controlar todas as tarefas do sistema, incluindo controle sobre memória, placas de som, vídeo, discos rígidos, disquetes, sistemas de arquivos, redes e outros recursos disponíveis. Todos os demais serviços e programas fazem chamadas ao kernel durante sua execução.

Veja o exemplo abaixo, como o sistema operacional é dividido em camadas:



Versões r.x.y do kernel, onde x é um número par, são versões estáveis, e enquanto o y é incrementado, apenas reparos de bugs são efetuados. Versões r.x.y do kernel, onde x é um número ímpar, são versões beta destinadas apenas a desenvolvedores, podem ser instáveis, falhar e estarão recebendo novas características o tempo todo. De tempos em tempos, com o atual desenvolvimento do kernel sendo considerado "estável", x é mudado para um número par, e o desenvolvimento continua com uma nova versão (x ímpar).

Módulos

São partes do kernel que são carregadas somente quando são solicitadas por algum aplicativo ou dispositivo e descarregadas da memória quando não são mais usadas. Este recurso é útil por 2 motivos: Evita a construção de um kernel grande (estático) que ocupe grande parte da memória com todos os drivers compilados e permite que partes do kernel ocupem a memória somente quando forem necessários.

Por exemplo, se você tem uma placa de som no seu computador, você

necessita carregar um módulo para que o kernel possa gerenciar ela. O mesmo pode acontecer com sua placa de rede, etc.

Shell

Shell é um interpretador de comandos, ou seja, ele é a conexão entre comandos digitados pelo usuário e o kernel. Além disso, o shell é uma linguagem interpretada, podendo executar loops, comparações (if) e armazenar valores em variáveis.

Entenda o shell como um ambiente, aonde você está em contato com seu Sistema, podendo controlá-lo, programar rotinas entre várias outras coisas. Uma vez que você digita um comando no shell, o interpretador de comandos repassa a função de uma forma simplificada para o kernel, e o mesmo a executa.

É interessante ressaltar que o Linux tem como padrão de operação apenas o shell, e não um ambiente gráfico como outros sistemas operacionais no mercado. Assim você consegue usá-lo sem necessitar de uma máquina muito potente, claro que isso é relacionado a função que você quer executar.

Partindo desse ponto, podemos dizer que o shell é um modo texto (podendo ter suporte ao mouse e imagens), e dele você pode iniciar um ambiente gráfico. Existem vantagens de você ter apenas uma tela e ficar digitando comandos e mais comandos ? Com muita certeza sim, nem tudo roda em torno de ícones e imagens. Servidores de grande porte, optam apenas por essa interface de comunicação.

Scripts

São comandos shell guardados em um arquivo. Quando você aciona esse arquivo, os comandos são executados linha a linha, assim você pode usar uma programação para o sistema, com if, while, for, etc.

Existem diversos tipos de shell, cada um tem uma particularidade em si, mas no fundo tendem ao mesmo sentido, interpretar. O usuário cadastrado no sistema pode optar por qual vai usar. Veja exemplos mais comuns:

Bourne Shell – sh

Shell Padrão do UNIX, feito por Stephen Bourne na Bell Labs. Um dos shell mais utilizados.

Bourne-Again Shell – bash

Shell padrão do Linux. Traz em si implementações relativas ao korn

shell e ao Bourne Shell, sendo totalmente compatível com o Bourne Shell.

Korn Shell - ksh

Nada mais que um Bourne Shell incrementado, sendo compatível com o mesmo.

Exemplo de um terminal rodando shell (bash):

```
[chk@linux /]$
```

Estrutura de diretórios

Todos arquivos do linux são guardados em diretórios, que formam uma árvore seguindo um subpadrão chamado Filesystem Hierarchy System (FHS) do LSB (Linux Standard Base).

É importante ressaltar que não temos letras específicas para designar partições e discos, apenas diretórios que se passam como pontos de montagem de CDROM, Disquete, um segundo HD, etc.

Veja exemplo mais comum:

Diretório	Descrição
/	Diretório raiz, aonde tudo começa. Por ser um ponto de montagem da sua partição (ou hd), todos os outros são subdiretórios dele.
/bin	Aonde fica os comandos essenciais do sistema.
/boot	Arquivos de inicialização (boot) do kernel(s).
/dev	Dispositivos de entrada e saída.
/etc	Arquivos com configurações do sistema local.
/home	Diretório dos usuários.
/home/chk	Arquivos pessoais do usuário chk.
/lib	Bibliotecas compartilhadas e módulos do kernel.
/mnt	Ponto de montagem temporário para outros dispositivos.
/mnt/cdrom	Ponto de montagem do cdrom.
/opt	Softwares opcionais (exemplo kde).

Diretório	Descrição
/proc	Sistema de arquivo virtual com informações do kernel.
/root	Diretório do super usuário, root.
/sbin	Comandos essenciais para administração do sistema.
/tmp	Diretórios com arquivos temporários.
/usr	Todos outros softwares, programas e comandos.
/usr/bin	Comandos e programas não essenciais ao sistema.
/var	Dados variáveis.

Note que, quando você é um usuário cadastrado no sistema, ao logar (entrar no sistema), você será redirecionado para seu diretório e não para a raiz (/).

Por exemplo, se o usuário chk entrar no sistema, o diretório no qual ele inicialmente estará presente é no seu diretório home, no caso: /home/chk .

Hierarquia – usuários e grupos

O Linux é multi-usuários, assim vários usuários (cadastrados) podem acessar no mesmo instante o sistema. Um problema que poderia ocorrer é o não controle sobre arquivos pessoais de cada usuário, assim qualquer um poderia ler, apagar, etc. .

Para solucionar isso, no Linux (Unix também) existe um usuário especial chamado root (raiz). Este usuário é tão especial que seu apelido é "super usuário" (super user). Ele é o único que pode alterar todos os arquivos do sistema.

Também no Linux há os logins de usuários normais, os quais não podem alterar arquivos de sistema e de outros usuários que não permitem. Os únicos arquivos que os usuários normais podem alterar são os de seu diretório HOME. O usuário de login Luke, por exemplo, tem seu diretório HOME localizado em /home/luke. Se este usuário tentar apagar um arquivo, por exemplo, do diretorio /etc não conseguirá, pois o sistema retornará : Permissão Negada (Permission Denied). Agora, se ele tentar apagar um arquivo dentro do diretorio /home/luke, conseguirá.

Quando o usuário é cadastrado no sistema, ele recebe uma espécie de ficha, ou seja, uma conta. Essa conta do usuário, possui um nome(login) que não pode ser igual para outro usuário, um número de identificação (uid), dados pessoais do usuário, uma senha para acesso ao sistema e um local padrão para ser seu diretório (home), todos usuários cadastrados, fazem parte de um (ou mais) grupo(s). Existe a opção também, de você escolher já no cadastro, qual shell o usuário vai usar.

Um grupo é um conjunto de usuários. Cada grupo também possui uma

identificação única no sistema, um nome e um número(gid – group identification). Normalmente administradores do sistema controlam os usuários por grupos, o que podem e não fazer.

Sendo assim, por padrão, você somente terá acesso ao sistema se você tiver uma conta nele(login e senha), claro, incluindo se você tiver a senha do usuário root.

O arquivo que possui todos usuários do sistema, por default é o passwd, que se localiza no diretório /etc . Todas as senhas dos usuários são armazenadas em outro arquivo, o shadow, isso porque elas se tornam criptografadas, impedindo que outro usuário veja (é possivel descriptografar).

Exemplo: /etc/passwd

```
root:x:0:0::/root:/bin/bash
joaoferreira:x:1000:100:João Ferreira Silva,,,:/home/joaoferreira:/bin/bash
marilia:x:14:350:/home/marilia:/bin/bash
carlos:x:225:125:/home/rh/carlos:/bin/bash
cardoso:x:3322:100:/home/cardoso:/bin/bash
```

Note que em todos usuários (cada linha, um usuário) possui um “x”, esse “x” se refere a senha dele que está no arquivo shadow .

Vamos esquematizar a linha do **João Ferreira Silva :**

login: joaoferreira
senha: x
informações adcionais: João Ferreira Silva
uid: 1000
Diretório: /home/joaoferreira
Shell: /bin/bash
gid: users(100)

Exemplo: /etc/shadow

```
joaoferreira:$1$6p.02IdH$xs77Hd8QgEDOIWR3qLM8m.:12290:0:99999:7:::
```

Veja acima, o exemplo do arquivo que contém a senha do João Ferreira, primeiro vem o login, e logo após o ':' vem a senha dele, até o próximo ':'

login: joaoferreira
senha: $1$6p.02IdH$xs77Hd8QgEDOIWR3qLM8m. (senha criptografada)

Note que existe outros campos, referente ao dia da mudança da senha, se ela pode expirar, mas não vamos entrar em detalhes.

O root é o usuário que possui a identificação (uid) valendo 0 (zero).

Exemplo: /etc/group

root::0:root
bin::1:root,bin,daemon,carlos
sys::3:root,bin,adm
users::100:carlos,joaoferreira

O arquivo group contém todos grupos cadastrados no sistema, um usuário pode fazer parte de mais de um grupo, como no exemplo, o carlos faz parte dos grupos bin e users. Claro, você pode adcionar e remover grupos sem problema, bem como trocar usuários de grupos.

Permissões

Depois de você ler sobre hierarquia (usuários, grupos) acima, você deve ter se perguntado: “Bem, muito legal, mas como eu vou usar identificação de usuário pra proteger minhas coisas?”

No Linux, como em outros sistemas Unix, cada arquivo tem uma permissão. As permissões são atributos dos arquivos que especificarão se ele pode ser lido, executado ou escrito. Estas permissões que vão definir o que um usuário pode fazer.

Note que:

Leitura significa: read, mas no sistema é apenas: **r**
Gravação significa: write, mas no sistema é apenas: **w**
Execução significa: execute, mas no sistema é apenas: **x**

Formando assim o famoso trio: **rwx**

Essas permissões podem ser setadas também através de números, veja:

Permissão	Significado
0	Nenhuma permissão
1	Executar (x)
2	Gravar (w)
3	Gravar e executar (wx)
4	Ler (r)
5	Ler e executar (rx)

Permissão	Significado
6	Ler e gravar (rw)
7	Ler, gravar e executar (rwx)

Quando você for setar permissão para algum arquivo ou diretório, você pode escolher em setar por letras (rwx) ou por números, lógico que por letras é mais fácil, porém para você nunca esquecer os números existe uma dica:

Se notar temos 8 opções (números), a primeira opção (zero) executa o que a opção 7 não faz, ou seja, é o contrário. A opção 1 executa, enquanto a 6 apenas lê e grava. A opção 2 grava, enquanto a 5 apenas lê e executa. A opção 3 grava e executa, enquanto a 4 apenas lê.

Tudo isso, tem uma aplicação direta em arquivos, diretórios, e até em dispositivos de io.

Veja o exemplo:

```
[chk@linux /tmp]$ ls -l foto.jpg
-rwxr-xrw-    1 chk      users       57932 Jul  6 17:04 foto.jpg
```

Nesse exemplo acima, estamos listando as propriedades do arquivo foto.jpg, a primeira string a ser listada possui atributos de permissões, que são os 10 primeiros caracteres da linha. Logo após vem quantos arquivos possui vinculados a ele (no caso somente ele mesmo, 1), e ao lado o dono do arquivo (chk) seguido do seu grupo (users). O resto se refere a data de modificação, tamanho, etc , etc ....

Analisando a primeira cadeia de string. Dividindo ela em quatro partes distintas :

-rwxr-xrw-
0111222333

1) **-** (0) O primeiro caracter designa se é um diretório ou apenas um arquivo, como tem o – (traço), subentende que é um arquivo. (ex. Diretório: **d**rwxr-xr-x).

2) **rwx** (111) Aqui temos as definições para o dono do arquivo, assim sendo, podemos notar que o dono do arquivo pode ler (r), gravar (w) e executar (x).

3) **r-x** (222) Essa segunda cadeia de caracteres, possui informações do que o grupo do dono do arquivo (users), pode fazer com ele (arquivo). Veja que o grupo pode ler (r), **não** pode gravar nele (w) e pode executar (x).

4) **rw-** (333) Última permissão, essa se refere a todos usuário que não são do grupo do dono do arquivo e lógico, não é o dono do arquivo. Quem não for nada do dono do arquivo, vai poder ler (r) e gravar (w), não podendo executar.

Processos

Processo é um programa em execução. Mais especificamente, é a estrutura que possui todas informações necessárias à execução de um programa.

Os processos são independentes, e cada processo em si, tem um uma identificação (PID – Process Identification). O kernel é responsável por gerenciar esses processos, quanto a sua utilização de memória, processamento.

No diretório /proc é criado um subdiretório para cada processo em execução. Os nomes desses subdiretórios é o PID de cada processo, em cada um contém informações detalhadas sobre sua execução.

Como normalmente utilizamos apenas um processador e vários processos, o kernel tem que gerenciar todos de modo que nenhum pare muito tempo (multi-processamento), porém é possível alterar a prioridade de execução de cada processo, através do comando nice (somente root).

Foreground e Background

Os processos podem ser executados de duas formas: em foreground (primeiro plano) ou background (segundo plano).

Os processos executado em foreground são aqueles que necessitam de interação direta com o usuário, incluindo troca de informações. Os processo em background não necessitam desta interação com o usuário.

Muitas vezes é preciso passar um processo que está sendo executado em foreground para background e vice-versa. Numa sessão de transferência de arquivos entre máquinas remotas, a velocidade da linha de transmissão pode aumentar demasiadamente o tempo de transferência (horas, às vezes!). Neste caso, seria interessante passar o processo para segundo plano, liberando a shell para outras atividades do usuário.

A passagem de um processo de foreground para background é feita primeiro suspendendo o processo, utilizando o conjunto de teclas CTRL+z, seguido do comando bg, que envia o processo para segundo plano. Deve ficar claro que suspender a execução de um processo não significa finalizá-lo, apenas torná-lo temporariamente inativo.

A lista dos processos executados em background pode ser visualizada com o comando jobs, que mostra cada processo associado com um número de job. Caso o usuário necessite interagir novamente com o processo, deve utilizar o comando fg seguido do número de job.

Para colocar um programa em execução background, pasta acrescentar o & (e comercial) na frente do nome. Ex. netscape &

Tipos de processos

Processo Interativo: são iniciados a partir de, e controlados por uma sessão terminal. Estes processos podem rodar tanto em foreground como em background.

Processos batch ou em lote: são processos não associados em nenhum terminal. Ao invés disso, são submetidos a uma fila, da qual jobs são executados sequencialmente.

Daemons: são processos servidores, inicializados durante o boot, que rodam continuamente enquanto o sistema estiver ativo, esperando, em background, até que um processo requisite seus serviços.

O que é PIPE ?

Pipe é um caracteristica do UNIX/LINUX que permite que vários programas sejam concatenados. Deste modo, o output de um programa é input do próximo. Esse recurso é muito utilizado para aplicar filtros. Por exemplo, para listar os processos de um determinado usuário, mandando listar os processos de saída deste comando serve todas as ocorrências do usuário procura. EX: ps | grep joao