

Laboratório de Redes de Computadores e Sistemas Operacionais

O Sistema de Arquivos

Fabricio Breve

O que você encontra no Sistema de Arquivos do Linux...

- Processos
- Portas seriais
- Canais de comunicação entre processos
- E também alguns arquivos...

Sistema de Arquivos UNIX

- É natural e conveniente associar objetos no espaço de nomes do sistema de arquivos
- Vantagens:
 - Interface de programação consistente
 - Fácil acesso no shell
- Desvantagens:
 - Implementações de sistemas de arquivos tipo Frankenstein

Sistema de Arquivos

- Compreende quatro componentes principais:
 - **Um espaço de nomes:** uma maneira de dar nome às coisas e organizá-las de forma hierárquica
 - **Uma API:** um conjunto de chamadas de sistema para navegar e manipular objetos
 - **Um modelo de segurança:** um esquema para proteger, ocultar e compartilhar coisas
 - **Uma implementação:** software que amarra o modelo lógico ao hardware em si

Os diversos Sistemas de Arquivos

- Sistemas de arquivos baseados em disco:
 - ext3fs: padrão do Linux
 - ext2fs: versão anterior, ainda usada
 - ReiserFS: maior velocidade, contorna algumas limitações do ext3, etc...
 - Etc...
 - Estrangeiros:
 - FAT, FAT32, ISO-9660
- **NFS**: manipulados por um driver que encaminha as operações

Nomes e caminhos

- O sistema de arquivos é hierárquico e começa no diretório raiz /
- Nome de caminho:
 - Absoluto: /etc/passwd
 - Relativo: book4/filesystem
 - Acessado a partir do diretório atual
- Restrições:
 - Componente de caminho:
 - 255 caracteres
 - Não pode conter / ou caracteres nulos
 - Espaços podem ser usados, porém pode ser problemáticos
 - Caminho: 4095 caracteres

Montando e desmontando sistemas de arquivos

- A maioria dos sistemas de arquivos são partições de disco
- Os sistemas de arquivos são anexados à árvore com o comando **mount**, que associa um diretório dentro da árvore de arquivos existente, chamado de ponto de montagem, à raiz do sistema de arquivos recém-anexados. O conteúdo anterior de um ponto de montagem se torna inacessível enquanto um outro sistema de arquivos esteja montado lá.
- Exemplo:
 - **mount /dev/hda4 /users**

/etc/fstab

- O arquivo **/etc/fstab** mantém uma lista dos sistemas de arquivos montados costumeiramente
 - Permite que os sistemas de arquivos sejam checados (**fck -A**) e montados (**mount -a**) automaticamente no momento da inicialização

umount

- Para demontar um sistema de arquivos utilize **umount**
 - Não é possível desmontar um sistema de arquivos “ocupado”
 - Não podem haver nenhum arquivo ou processo aberto cujos diretórios estejam lá localizados
 - Para saber que arquivos estão ocupando o sistema use **fuser**
 - Exemplo: **fuser -mv /usr**

Códigos utilizados por **fuser**

f	O processo possui um arquivo aberto para leitura ou gravação
c	O diretório atual do processo se encontra no sistema de arquivos
e	O processo está executando um arquivo no momento
r	O diretório-raiz do processo (configurado através de chroot) se encontra no sistema de arquivos
m	O processo associou um arquivo ou biblioteca compartilhada (normalmente um executável inativo)

Determinando os processos

- Para determinar com exatidão quais os processos transgressores, utilize os com a lista de PIDs retornadas por **fuser**
- Exemplo: **ps -fp "3419 3420"**
- **fuser** aceita o parâmetro **-k** para extinguir os processos transgressores (perigoso e exclusivo para o root)
- Alternativa para o **fuser**: **Isof** (Red Hat)

```
[root@localhost ~]# ps -fp "3420 3419"
UID          PID    PPID    C  STIME TTY          TIME CMD
gdm          3419    3274    0  15:12 ?            00:00:01 /usr/bin/gdmgreeter
root         3420    2816    0  15:12 tty1         00:00:00 -bash
```

Organização da Árvore de Arquivos

- Sistemas UNIX nunca foram bem organizados
 - Convenções de nomes incompatíveis usadas ao mesmo tempo
 - Tipos de arquivos espalhados aleatoriamente pelo espaço de nomes
 - Arquivos divididos por função e não por probabilidade com que mudam, dificultando atualização do SO
 - Exemplo: `/etc` tem arquivos jamais personalizados e outros locais

Organização da Árvore de Arquivos

- Algumas inovações (como o **/var**) ajudam a diminuir a bagunça, mas no geral a maioria dos sistemas ainda é bem bagunçada
- Não se deve bagunçar a estrutura-padrão da árvore
 - Pacotes de software e suas instalações fazem suposições abrangentes com relação a localização de arquivos
 - Outros administradores também

Organização da Árvore de Arquivos

- Diretório raiz (/) tipicamente contém:
 - **/boot** – normalmente chamado (ou associado a) **vmlinuz**
 - **/dev** – dispositivos
 - **/etc** – arquivos críticos
 - **/sbin** e **/bin** – utilitários importantes
 - **/tmp** – arquivos temporários
 - **/usr** – programas-padrão, manuais, bibliotecas
 - **/var** – diretórios spool, arquivos de log, informações contábeis e outros arquivos que se modificam rapidamente
 - **/home** – diretórios de usuários

Diretórios-padrão e seu conteúdo.

Nome do caminho	Conteúdo
/bin	Comandos necessários para a mínima operação do sistema.
/boot	Kernel e arquivos necessários para carregar o kernel.
/dev	Entradas de dispositivos para terminais, discos, modems etc.
/etc	Arquivos de configuração e inicialização críticos.
/lib	Bibliotecas e partes do compilador C.
/opt	Pacotes de software aplicativo opcionais e agregáveis.
/proc	Imagens de todos os processos em andamento.
/root	Diretório de usuário do superusuário (normalmente apenas /).
/sbin	Comandos para inicialização, reparos ou recuperação do sistema.
/tmp	Arquivos temporários que desaparecem entre as reinicializações.
/usr	Hierarquia de arquivos e comandos secundários.
/usr/bin	A maioria dos arquivos executáveis e comandos.
/usr/include	Arquivos de cabeçalho para programas em C.
/usr/lib	Bibliotecas; também, arquivos de programas suporte para programas-padrão.
/usr/local	Software local (coisas que instalamos).
/usr/local/bin	Executáveis locais.
/usr/local/etc	Arquivos de configuração e comandos locais.
/usr/local/lib	Arquivos de suporte locais.
/usr/local/sbin	Comandos locais de manutenção de sistema linkados estaticamente.
/usr/local/src	Código-fonte para /usr/local/* .
/usr/sbin	Comandos menos essenciais para administração e reparo de sistema.
/usr/share	Itens que podem ser comuns a vários sistemas (somente para leitura).
/usr/share/man	Páginas de manuais on-line.
/usr/src	Código-fonte para pacotes de software (não locais).
/usr/src/linux	Área de trabalho de construção de kernel, arquivos de configuração.
/var	Arquivos de configuração e dados específicos ao sistema.
/var/adm	Vários: logs, registros de configuração de sistema, pequenas coisas estranhas para administração do sistema.
/var/log	Vários arquivos de log de sistema.
/var/spool	Diretórios de spool para impressão, e-mails etc.
/var/tmp	Mais espaço temporário (preservado entre reinicializações).

Tipos de Arquivos

- O Linux define sete tipos de arquivos:
 - Arquivos regulares
 - Diretórios
 - Arquivos de dispositivos de caracteres
 - Arquivos de dispositivos de blocos
 - Sockets de domínio local
 - Pipes com nome (FIFOs)
 - Links simbólicos

Codificação de tipos de arquivos usados por ls

- **ls -l**

Codificação de tipos de arquivos usados por ls.

Tipo de arquivo	Símbolo	Criado por	Eliminado por
Arquivo comum	-	editores, cp etc.	rm
Diretório	d	mkdir	rmdir, rm -r
Arquivo de dispositivos de caracteres	c	mknod	rm
Arquivo de dispositivos de blocos	b	mknod	rm
Pipe com nome	p	mknod	rm
Link simbólico	l	ln -s	rm

```
[fbreve@localhost ~]# ls -l
total 24
drwxr-xr-x  2 fbreve fbreve 4096 Ago  3 03:36 Desktop
-rw-rw-r--  1 fbreve fbreve   15 Ago 16 12:21 fabricio.txt
drwxrwxrwx  2 root   root   4096 Ago 28 11:46 win
```

Arquivos Regulares

- Apenas uma coletânea de bytes
 - Arquivos texto
 - Arquivos de dados
 - Programas executáveis
 - Bibliotecas compartilhadas
- Permitem acesso seqüencial e aleatório

Diretórios

- “Pasta” no Windows
- Referências com nomes para outros arquivos
- Criados com **mkdir** e eliminados com **rmdir** se estiverem vazios
 - Não vazios podem ser removidos com **rm -r**
- Entradas especiais:
 - . – o próprio diretório
 - .. – o diretório pai
 - No diretório raiz (/) .. equivale a .

Hard Links

- O nome do arquivo fica armazenado no diretório
- Você pode ter mais de uma entrada apontando para o mesmo arquivo
 - Tais entradas são indistinguíveis do arquivo original
 - O Linux mantém uma contagem do número de links que apontam para cada arquivo e só liberam o espaço em disco quando todos os links tenham sido eliminados
 - Criamos hard links com **ln** e eliminamos **rm**

Arquivos de dispositivos de blocos e caracteres

- Permite que os programas se comuniquem com o hardware e periféricos do sistema
- Os módulos que sabem como se comunicar com cada dispositivo são associados ao kernel
- O módulo para um determinado dispositivo (driver) cuida dos detalhes de gerenciamento do dispositivo
- Os arquivos de dispositivos são pontos de encontros usados para se comunicar com o driver, eles permitem que seus drivers associados façam seu próprio buffering de entrada e saída
- Exemplos: **/dev/lp0** **/dev/ttyS0**

Sockets de domínio local

- Permitem que processos se comuniquem de maneira sadia
- Socket de domínio local são acessíveis somente do host local e referidos através de um objeto do sistema de arquivos em vez de uma porta de rede
- Não podem ser lidos ou gravados por processos não envolvidos na conexão
- Sistemas que usam socket de domínio local:
 - Serviço Impressão, X Window e Syslog

Pipes com nome

- Assim como os sockets de domínio local permitem a comunicação entre dois processos rodando no mesmo host
 - São também conhecidos como FIFO (First In, First Out)
 - Raramente requerem interferência administrativa

Links simbólicos

- Também chamados soft links
- Apontam um arquivo pelo nome, quando o kernel o encontra redireciona para o arquivo real
- Diferente do hard link que é uma referência direta
- Criados com **ln -s** e eliminados com **rm**
- Pode conter um caminho absoluto ou relativo

Atributos de Arquivos

- Todo arquivo possui um conjunto de bits de permissão que controla quem pode ler, gravar e executar seu conteúdo
 - Junto com outros 3 bits que afetam a operação de programas executáveis constituem o “modo” do arquivo
- Os 12 bits são armazenados juntos com os quatro bits de informação do tipo de arquivos (configurados quando o arquivo é criado)
 - Os 12 bits podem ser alterados usando **chmod**

Bits de permissão

- Usados para determinar que operações podem ser executadas em um arquivo e por quem.
- Não é possível fazer a configuração por usuário, há conjuntos de permissões para:
 - Proprietários do arquivo
 - Proprietários do grupo do arquivo
 - Todos os outros
- Cada conjunto possui 3 bits:
 - Leitura: arquivo pode ser lido e aberto
 - Gravação: arquivo pode ser modificado ou truncado (apagar depende da permissão do diretório)
 - Execução: permite que o arquivo seja executado

Arquivos Executáveis

- Existem dois tipos:
 - Binários: executados diretamente pela CPU
 - Scripts: devem ser interpretados por um shell ou algum outro programa
 - Por convenção começam com uma linha na forma:
`#!/bin/csh -f`
 - Quando não são binários e não especificam um interpretador, é assumido **bash/sh**

Bits de permissão

- Em diretórios:
 - Bit de execução: permite que o diretório seja introduzido ou passado enquanto um caminho está sendo avaliado, mas não permite que seu conteúdo seja listado
 - Combinando os bits de leitura e execução arquivos podem ser criados, eliminados e renomeados dentro do diretório

Visualizando atributos de arquivos


- Os mais úteis para o administrador são:
 - Contagem de links
 - Proprietário
 - Grupo
 - Modo
 - Tamanho
 - Horário do último acesso
 - Horário da última modificação
 - Tipo
- O restante normalmente só é útil para o próprio sistema de arquivos.
- Para inspecionar os atributos use **ls -l** (ou **ls -ld**)

Visualizando atributos de arquivos

```
[root@localhost fbreve]# ls -l
total 32
drwxr-xr-x  2 fbreve fbreve 4096 Ago  3 03:36 Desktop
-rw-rw-r--  1 fbreve fbreve  15 Ago 16 12:21 fabricio.txt
-rw-rw-r--  1 root   root   37 Ago 30 04:40 fab.txt
drwxrwxrwx  2 root   root  4096 Ago 28 11:46 win
```

- R – leitura, W – escrita, X – execução
- Grupos: dono, grupo e outros
- Número: contagem de hard links do arquivo
 - Links simbólicos não afetam a contagem
- Proprietário e proprietário do grupo
- Tamanho do arquivo em bytes
- Data de última modificação
- Nome do arquivo

chmod: modificando permissões

- Modifica permissões em um arquivo
 - Pode ser usado somente pelo proprietário e pelo root
 - Notação Octal 
 - Exemplo:

chmod 755 myprog

- Dá ao dono permissão de leitura, gravação e execução e aos outros apenas leitura e execução

Octal	Binário	Permissões
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwX

chown: modificando propriedade e grupo

- Modifica propriedade de arquivo e de grupo de um arquivo
- Sintaxe:
 - **chown usuário.grupo arquivo**
- Exemplo:
 - **chown fbreve.alunos texto.txt**
 - **chown -R fbreve.fbreve ~fbreve**
 - Modifica recursivamente todos os arquivos dentro do diretório ~fbreve

umask: atribuindo permissões-padrão

- Manipula permissões-padrão dos arquivos que criamos
- O arquivo criado atende tudo o que o programa criados solicitar menos o que umask proibir
- O padrão é **umask 022**

Octal	Binário	Permissões
0	000	rwX
1	001	rw-
2	010	r-X
3	011	r--
4	100	-wX
5	101	-w-
6	110	--X
7	111	---

Questões

- Qual a diferença entre um soft link (link simbólico) e um hard link?
- Qual a diferença entre executáveis binários e scripts?
- O que é umask? Crie uma umask que não de nenhuma permissão ao grupo ou demais usuários
- Por que é uma boa idéia criar partições num drive separado para **/var** **/home** e **swap**?
- Como você configuraria um arquivo texto para ser lido e gravado apenas por você, e apenas lido pelos demais usuários (inclusive do grupo)?

Referências Bibliográficas

- NEMETH, Evi.; HEIN, Trent R.; SNYDER, Garth. *Manual Completo do Linux: Guia do Administrador*. Makron Books, 2004.