Laboratório 1.3: Sistema de Arquivos - Explorando

Introdução

Organização dos Diretórios

Um sistema Linux é composto por diversos diretórios, cada um com objetivos bem determinados. Essa forma de separação dos arquivos em diretórios baseados no tipo/objetivo deles resulta em um sistema bem organizado e fácil de se manter. Por exemplo, todos os arquivos de configurações de todos os programas instalados ficam no diretório `/etc`, permitindo encontrá-los rapidamente.

Outro exemplo de praticidade são os arquivos executáveis, que ficam dentro do `/bin`, permitindo que estes sejam rapidamente encontrados ao serem executados. Já as bibliotecas compartilhadas, ficam no `/lib`, permitindo que elas sejam facilmente encontrados por todos os programas que precisam delas.

Pacotes

Uma desvantagem desta forma de organização é que os arquivos de um único programa ficam espalhados pelo sistema. Isso estimulou, desde o início, o uso de "pacotes" (e.g., *deb*, *rpm*), que são uma forma padronizada de se distribuir programas. Um pacote possui instruções de instalação, dependências, configuração e desinstalação do seu programa. Hoje em dia, praticamente todos os arquivos do Linux fazem parte de algum pacote e o sistema de instalação inicial do Linux nada mais é do que a instalação de todos os pacotes necessários. Falaremos mais da instalação de pacotes futuramente.

Discos, Partições e Montagem

No Linux, não existe o conceito de "*drives*" (e.g., `C:`). Todos os discos/partições são "montados" em um diretório. O diretório raiz (`/`) é uma partição, enquanto outros subdiretórios, como o `/home` ou `/boot`, podem ser outras partições/discos. Montar uma partição significa que estamos mapeando um disco/partição para um diretório.

Diretório Raiz

Voltando para o sistema de arquivos, o diretório principal do Linux/Unix é o diretório raiz (`/`), também conhecido como *root*. Execute o comando:

```
ls /
```

Diretórios do Sistema

Note que no diretório raiz (`/`) têm-se diversos outros subdiretórios, descritos a seguir:

Diretório	Descrição
`/etc`	Os arquivos de configurações do sistema e dos diversos programas/pacotes instalados são colocados neste diretório.
`/bin, /sbin`	Contém os arquivos binários (executáveis). O `/bin` contém os executáveis normais, enquanto o `/sbin` contém executáveis essenciais do sistema. Outros executáveis podem ser encontrados nos diretórios `/usr/bin` e `/usr/sbin`. Em alguns sistemas, por exemplo, o `/bin` é uma referência para o `/usr/bin`. Veremos o diretório `usr` mais abaixo.
`/lib`	Contém as bibliotecas compartilhadas do sistema. Outras bibliotecas podem ser encontradas no diretório `/usr/lib`.
`/boot`	Contém os arquivos principais para a inicialização do sistema como o kernel Linux e os arquivos do gerenciador de boot (`grub`).
`/home`	Dentro desse diretório, encontram-se os *homes* dos diferentes usuários. Em geral, tem-se um diretório por usuário normal do sistema. O seu *home* provavelmente se encontra aqui.
`/root`	É o diretório *home* do usuário *root*. O usuário *root* é o administrador principal de qualquer sistema Linux/Unix.
`/dev`	Conforme mencionado, tudo no Linux é arquivo. Neste diretório, encontram-se arquivos para acessar os diversos dispositivos de hardware do sistema e outros módulos/componentes do kernel. O nome `dev` vem do inglês *devices*, que significa "dispositivos".
`/proc, /sys`	Contém outros arquivos especiais para permitir a comunicação com módulos e componentes do kernel. Diferentemente do `/dev`, os arquivos não são necessariamente relacionados a algum dispositivo de hardware. O `/proc` é uma estrutura mais antiga, enquanto o `/sys` é uma estrutura mais atual. Mas ambos possuem finalidades semelhantes.
`/media, /mnt`	Nestes diretórios, são montados outras partições e discos. Em geral, o `/mnt` é usado para discos do próprio hardware,

	enquanto o `/media` é usado para discos removíveis como *pedrives USB*.
`/tmp`	Usado para arquivos temporários. Normalmente os arquivos são removidos na reinicialização do sistema. É um dos poucos lugares que um usuário normal pode escrever, com exceção do seu próprio *home*.
`/var`	Contém arquivos "variáveis", que mudam com relativa frequência. Um exemplo são os arquivos de logs do sistema (`/dev/logs`) e arquivos de cache (`/var/cache`).
`/opt`	Diretório para aplicações opcionais (*optional*). Usado por alguns programas para armazenar todos os seus arquivos em um único lugar. Pouco usado na prática.
`/usr`	Este diretório contém vários subdiretórios com o mesmo nome e propósito dos encontrados no `/`. Alguns exemplos incluem o `/usr/bin`, `/usr/lib`. Entretanto, neste diretório são colocados os arquivos de programas de usuários, menos relacionados ao sistema em si. Neste diretório encontram-se também documentações em geral (`/usr/share`) e cabeçalhos/códigos-fontes (`/usr/include` e `/usr/src`).

Partições

Um disco (rígido, SSD, pendrive) pode ser dividido em partes diferentes, com tipos e tamanhos diferentes, conhecidas como partições. Para o sistema operacional, cada partição se parece com um disco diferente. No Windows, é comum termos apenas uma única partição. Já no Linux/Unix, é comum termos várias partições.

Um caso comum no Linux é termos uma partição para o `/`, uma partição para o `/boot` e outra para o `/home`, por exemplo. A vantagem de ter uma partição para o `/boot` é permitir que o sistema pelo menos inicie no modo de recuperação caso o `/` fique sem espaço livre. Já uma vantagem de ter o `/home` em uma partição diferente é que o sistema não ficará comprometido caso algum usuário use todo o espaço da partição. Por fim, outra vantagem de colocar `/home` em uma partição própria é que podemos formatar as outras partições (reinstalar a máquina) sem perda de dados dos usuários.

Cada partição possui um tipo, também conhecido como tipo do sistema de arquivos. A tabela a seguir mostra alguns tipos comuns:

Tipo da Partição	Descrição
`ext4`, `ext3`	*Fourth extended filesystem* (`ext4`) é o sistema de arquivos padrão do Linux, sucessor do `ext3`.

`ntfs`, `vfat`	*New Technology File System* (NTFS) e *Virtual File Allocation Table* (VFAT) são dois tipos de partições usadas no Windows.

1. Explorando o /etc

Conforme mencionado, o diretório `/etc` contém vários arquivos de configurações do sistema. Você já deve ter visto alguns nos laboratórios anteriores. Por exemplo, o arquivo `/etc/passwd` contém as contas de usuários do sistema. Falaremos mais dele no laboratório de segurança. Um arquivo complementar ao `passwd` é o `shadow`, no mesmo diretório. Ele contém as senhas criptografadas dos usuários e só pode ser lido pelo administrador do sistema (usuário *root*). Já o arquivo `group`, contém os grupos de usuários do sistema.

```
$ ls -flh /etc/passwd /etc/shadow /etc/group
```

```
-rw-r--r-- 1 root root   1.7K Feb 26 17:39 /etc/passwd
-rw-r----- 1 root shadow  975 Feb 26 17:39 /etc/shadow
-rw-r--r-- 1 root root    817 Feb 26 17:39 /etc/group
```

Um outro arquivo interessante é o `/etc/fstab` que contém as partições montadas na inicialização do Linux:

```
$ cat /etc/fstab
```

```
LABEL=cloudimg-rootfs / ext4 defaults 0 1
```

O arquivo `/etc/hostname` contém o nome do seu computador:

```
$ cat /etc/hostname
```

```
Marcos
```

Um arquivo interessante é o `/etc/protocols`, que possui uma lista de protocolos que usam o IP (é um arquivo um pouco histórico, menos usado hoje em dia):

```
$ cat /etc/protocols | grep UDP
```

```
udp     17      UDP             # user datagram protocol
udplite 136     UDPLite         # UDP-Lite [RFC3828]
```

Semelhantemente, tem-se o `/etc/services`, que possui uma lista serviços TCP e UDP e suas respectivas portas.

```
$ cat /etc/services | grep -P "ssh|http\t|https|ircd"
```

```
port-numbers/service-names-port-numbers.xhtml .
ssh             22/tcp                          # SSH Remote
Login Protocol
http            80/tcp          www             # WorldWideWeb
HTTP
```

O arquivo `/etc/shells` lista os *shells* (interpretadores de comandos) instalados e disponíveis no sistema:

```
$ cat /etc/shells
```

```
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

Como você pode ver, o `bash` não é o único interpretador de comandos disponível.

Já o arquivo `/etc/timezone` contém o fuso horário de onde a máquina está localizada:

```
$ cat /etc/timezone
```

```
America/Manaus
```

Por fim, é importante mencionar que o `/etc` contém configurações apenas do sistema e dos programas instalados no sistema. Configurações específicas de cada usuário são armazenados dentro do *home* do próprio usuário, normalmente em arquivos ou diretórios ocultos:

```
$ ls -alh ~
```

2. Explorando o /dev, /proc e /sys

Diretório `/dev`

Os diretórios `/dev`, `/proc` e `/sys` contém vários arquivos especiais que são, na verdade, portas de entrada e saída para componentes e módulos do kernel Linux.

Já vimos o **/dev/null**, que serve como lixeira para bytes que queremos descartar.

Outro arquivo interessante é o **/dev/random**, que gera bytes aleatórios para ser usado pelos programas. Você pode ver o conteúdo dele para ver os bytes sendo gerados:

```
$ cat /dev/random                    # Pressione Ctrl+C para sair
```

A base para geração de dados aleatórios são eventos do sistema como teclado, mouse, escrita no disco, etc. Tudo pode ser usado como uma semente para gerar bytes aleatórios. Uma característica interessante é que ele pode ficar sem informações aleatórias suficientes para gerar dados e simplesmente parar. Para gerar outros bytes aleatórios, mova o mouse, digite algo no teclado, etc. Devido a isso, foi criado o gerador **/dev/urandom**, que não tem essa limitação. Mas não se recomenda dar um `cat` nele, pois a quantidade de dados gerados é muito grande e rápido. O `bash` usa este último para gerar números inteiros aleatórios:

```
$ echo $RANDOM                       # Execute várias vezes
```

```
27708
```

Observe também os arquivos **stdin**, **stdout** e **stderr** dentro do `/dev`. Yep, eles também são arquivos:

```
$ cd ~
$ echo "Timber Hearth" > OuterWilds.txt
$ cat OuterWilds.txt > /dev/stdout
```

```
Timber Hearth
```

Diretório `/proc`

O `/proc` contém arquivos especiais disponibilizando informações diretamente do kernel para os aplicativos. Já vimos o **/proc/cpuinfo**, que mostra informações da CPU. O arquivo **/proc/diskstats** disponibiliza informações atualizadas sobre o uso dos dispositivos de armazenamento:

```
$ cat /proc/diskstats | grep -v loop
```

```
1       0 ram0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1       1 ram1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1       2 ram2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1       3 ram3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1       4 ram4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1       5 ram5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Neste comando, estamos ignorando as linhas que possuem a palavra loop (opção `-v` do `grep`). No Linux, loop devices são pseudo-dispositivos que montam um *arquivo* como se fosse um *disco*.

Como você pode ver, a sintaxe do arquivo `/proc/diskstats` não é exatamente amigável. Ele foi feito mais para ser lido por outros programas. Você pode ver a documentação do arquivo no repositório do kernel do Linux. Além disso, você verá diversos dispositivos de entrada e saída. O que possuir os maiores números, provavelmente é o seu disco principal. Este arquivo é usado por vários programas para mostrar e analisar o uso do disco. Um deles é o comando `iostat`, que mostra informações de entrada e saída (*input*/*output*) do sistema:

```
$ iostat  | grep -v loop
```

```
Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s
kB_read    kB_wrtn    kB_dscd
sda               6.09         0.72      2814.86         0.00
225     877504          0
sdb               2.29       148.09         0.00         0.00
46165          0          0
```

Note como as unidades no arquivo `/proc/diskstats` e no comando `iostat` são diferentes. No `/proc/diskstats`, o sexto campo mostra a quantidade de setores lidos. Já no `iostat`, a coluna `kB_read` é mostrada em bytes. O motivo é que as leituras em disco são feitas em blocos chamados setores. Em geral, um setor possui 512 bytes. Para saber o tamanho do setor de um disco, execute:

```
$ cat /sys/block/<DEVICE>/queue/hw_sector_size  # Substitua <DEVICE>
pelo id do disco
```

```
512
```

Mudando de arquivo, o **`/proc/cmdline`** mostra as opções passadas para o kernel do Linux quando ele foi chamado pelo gerenciador de boot (provavelmente o GRUB):

```
$ cat /proc/cmdline
```

```
initrd=\initrd.img panic=-1 nr_cpus=8 swiotlb=force
```

Em geral, as opções do arquivo de imagem do kernel e do dispositivo/partição que contém o diretório raiz (*root*) são passados.

Outro arquivo de informações muito úteis é o **`/proc/loadavg`**. Ele mostra a carga de utilização do processador e é dividido em 5 campos:

Campo #	Descrição

`1, 2, 3`	Carga média de utilização nos últimos 1, 5 e 15 minutos, respectivamente. Uma média de 0 (zero), significa que o sistema está (esteve) ocioso. Um número maior que a quantidade de núcleos do processador significa que o sistema está sobrecarregado. Para saber o número de núcleos, execute: `nproc --all`.
4	Número de processos (programas) ativamente em execução / Número de processos totais no sistema.
5	Número de identificação de processo (PID) do último programa executado.

```
$ cat /proc/loadavg
```

```
0.00 0.01 0.00 1/148 506
```

O código-fonte do componente que gera esse arquivo no kernel do Linux é bem simples. Ele basicamente imprime um vetor, já existente e atualizado pelo kernel, que contém esses valores.

O arquivo **`/proc/meminfo`** mostra informações detalhadas sobre a memória do sistema. No comando abaixo, filtramos os dados para ver apenas algumas informações da memória principal:

```
$ cat /proc/meminfo | grep Mem
```

```
MemTotal:       12179960 kB
MemFree:        11708860 kB
MemAvailable:   11681416 kB
```

O arquivo **`/proc/uptime`** mostra quanto tempo faz desde a última inicialização do sistema:

```
$ cat /proc/uptime
```

```
508.11 3741.92
```

O primeiro valor diz o tempo desde a última inicialização em segundos. Para um administrador de servidores Linux, um valor alto no `uptime` (alguns anos) é motivo de orgulho :). Já o segundo valor, diz a soma total de tempo que cada núcleo de processador ficou em modo ocioso. Portanto, o segundo valor pode ser maior que o primeiro em sistemas com vários núcleos. Novamente, o código-fonte do kernel que implementa esse arquivo é bem simples.

Por fim, o arquivo `/proc/version` contém a versão do kernel do Linux usado no sistema:

```
$ cat /proc/version
```

```
Linux version 5.10.16.3-microsoft-standard-WSL2 (oe-user@oe-host)
```

Para finalizar, existem diversos outros arquivos não mencionados nesta seção nos diretórios `/dev`, `/proc` e `/sys`. Recomenda-se fortemente que você explore e teste esses arquivos. Fique tranquilo que, se você estiver explorando como usuário normal (sem ser *root*), você não conseguirá danificar o sistema.

3. Explorando o /bin e /sbin

Conforme mencionado, os diretórios `/bin` e `/sbin` contém arquivos binários (executáveis) do sistema. O formato de arquivos executáveis no Linux segue o padrão ELF (*Executable and Linkable Format*). Por não serem arquivos textos, não faz muito sentido dar um `cat` neles. Mas podemos usar o utilitário `hexdump` para ver o conteúdo deles em hexadecimal e, quando possível, em caracteres:

```
$ hexdump -C /bin/ls | head -3
```

```
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00
|.ELF............|
00000010  03 00 3e 00 01 00 00 00  d0 67 00 00 00 00 00 00
```

Os quatro primeiros bytes de todo ELF começa com o que é conhecido como *magic number*, usado para identificar unicamente um tipo de arquivo. No caso do ELF, esse *magic number* é "`7F 45 4C 46`", sendo que os três últimos bytes leem "ELF" em ASCII. O próximo byte `02` identifica um formato de 64bits. Seguido de `01` para identificar o uso de *little-endian* em campos de vários bytes. Em seguida, tem-se outro `01` que é a versão do ELF. Por fim, tem-se 8 zeros. A segunda linha, começa com `03 00`, indicando que esse arquivo é um executável com bibliotecas dinâmicas, seguido de `3e 00`, indicando que o arquivo foi compilado para um processador `AMD x86-64`. Obviamente, esses valores podem mudar dependendo da arquitetura do seu Laptop/PC. Para uma descrição completa desse cabeçalho de arquivo, clique aqui.

Voltando para o diretório `/bin`, execute o `ls` nele para listar os arquivos.

```
$ cd /bin
$ ls
```

Note como vários dos comandos que já aprendemos estão localizados aqui. Aprenderemos vários outros aqui e futuramente. Note também o comando **bash**, que é o interpretador de comandos que você está usando. Você pode até executá-lo novamente para ter um `bash` dentro de outro (execute `exit` para voltar).

```
$ bash                                  # Inicia um novo interpretador
de comandos
$ cd /tmp                               # Muda de diretório
$ exit                                  # Sai do bash iniciado e volta
para o anterior
```

Apesar dos comandos anteriores não terem feito nada de útil, a moral da história é mostrar que o `bash` é um comando normal, como qualquer outro.

Veremos, muito rapidamente, mais alguns comandos no diretório `/bin`, começando com o **`date`**, que mostra a data e hora atuais:

```
$ /bin/date
```

```
Fri May  6 22:58:38 -04 2022
```

Aproveitamos o comando `date` acima para mostrar que podemos executar um arquivo usando o seu *caminho completo*. Entretanto, para os arquivos no `/bin` isso não é necessário (podemos executar apenas `date`) porque este e outros diretórios fazem parte da variável de ambiente `PATH`. Falaremos mais das variáveis de ambiente futuramente, mas para ver os outros diretórios que fazem parte do `PATH`, execute:

```
$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Windows/system32:
```

Estes são os diretórios que o sistema irá procurar quando comandos são executados sem o caminho completo.

O comando **`df`** (*disk free*) mostra informações de utilização dos discos/partições:

```
$ df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc        251G  1.2G  237G   1% /
none            5.9G   28K  5.9G   1% /mnt/wslg
none            5.9G  4.0K  5.9G   1% /mnt/wsl
tools           477G  420G   57G  89% /init
```

O comando **touch** cria um arquivo novo em branco ou muda a data de modificação de um arquivo existente:

```
$ touch ~/ArquivoTouch.txt
$ ls -lh ~/ArquivoTouch.txt
```

```
-rw-r--r-- 1 marcos marcos 0 May  6 23:11
/home/marcos/ArquivoTouch.txt
```

O comando **uname** mostra informações do sistema:

```
$ uname -a
```

```
Linux Marcos 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2
22:22:49 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

O que foi mencionado nesta seção é uma pequena parte dos executáveis. Na verdade, a maioria dos executáveis no Linux ficam no `/usr/bin`. Por exemplo, um comando muito usado por administradores é o comando `who`, que mostra os usuários logados no sistema:

```
$ who
```

```
Marcos
```

O `who` é um exemplo de comando que, normalmente, fica localizado no `/usr/bin`. Portanto, não deixe de explorar estes diretórios. Você pode executar os comandos usando a opção `--help` ou usar o `man` para ter mais detalhes deles. Novamente, fique tranquilo que, se você estiver explorando como usuário normal (sem ser *root*), você não conseguirá danificar o sistema! :)

Be excellent to each other. — Bill

π