INSTITUTO FEDERAL
Paraíba

Sistemas Operacionais

Atividade Prática de SA

OBS: Cada tela/resposta requisitada tem valor de 1,0.

Estrutura de Diretórios

Conforme discutido em sala, alguns dos principais sistemas de arquivos usados em ambientes Linux são o ext2, ext3, ext4, reiser, entre outros. Apesar destes diferentes sistemas, em geral, os diretórios de um sistema de arquivos no UNIX têm uma estrutura pré-definida comum, com poucas variações:

- **bin** - *contains system binary executables*
- **boot** - *contains files necessary for the system to boot up*
- **dev** - *contains device files which function as an interface to the various hardware drivers. These will vary greatly depending on the version of Unix*
- **etc** - *contains system configuration settings*
- **home** - *contains the user's home directories (often but not at LSC/ATM Linux network)*
- **mnt –** *mount point for a temporary mounted filesystem*
- **lib** - *contains system libraries for 32- bit applications*
- **lib64** - *contains system libraries for 64- bit applications*
- **opt** - *contains optional applications*
- **proc** - *contains a virtual file system which holds information about running processes and the state of the system*
- **root** - *the root (administrator) user's home directory*
- **sbin** - *contains static binary executables needed for the system*
- **tmp** - *temporary directory used by many applications*
- **usr** - *contains binaries, data and settings for various applications. The structure of /usr mimics the root file system organization*
- **var** - *stores logs, data for services and other transient data*

Uma vez apresentada esta estrutura de diretórios, e considerando alguns dos comandos discutidos em sala, realize as atividades propostas na sequência.

Processos

1) No terminal, vá até o diretório **`/proc`** e liste seu conteúdo (**`ls -l`**). Observe que os subiretórios corespondem aos PIDs dos processos correntes (execute **`ps -lax`** e verifique isso). Apresente um *printscreen* desta tela

> *O /proc é, por vezes, chamado de "pseudo sistema de arquivos de informações de processos" (process information pseudo-file system). O diretório não contém "arquivos de verdade", mas as informações referentes ao seu sistema em tempo de execução (runtime).*
>
> *Entre as informações disponíveis no /proc, você pode encontrar a quantidade de memória presente no sistema, os dispositivos de armazenamento que estão montados, a configuração*

atual do hardware, o tempo que o seu dispositivo está ligado etc.

2) Lembrando que o *bash* é o seu interpretador de comandos, você pode verificar o PID do *bash* executando o comando **`ps`**. De posse do PID do seu *bash*, entre no subdiretório cujo nome seja o PID do seu *bash*. Ali você encontrará várias informações sobre este processo. Consulte algumas dessas informações para o seu *bash*:

`more (ou cat) /proc/PID/cmdline` // Argumentos da linha de comando.
`more (ou cat) /proc/PID/maps` // Mapas de memória para os executáveis e arquivos da // biblioteca.
`more (ou cat) /proc/PID/stat` // Informações gerais de estado do processo:

```
(1) pid  %d
          The process ID.

(2) comm  %s
          The filename of the executable, in parentheses.
          This is visible whether or not the executable is
          swapped out.

(3) state  %c
          One of the following characters, indicating process
          state:

          R  Running

          S  Sleeping in an interruptible wait

          D  Waiting in uninterruptible disk sleep

          Z  Zombie

          T  Stopped (on a signal) or (before Linux 2.6.33)
             trace stopped
(14) utime  %lu
          Amount of time that this process has been scheduled
          in user mode, measured in clock ticks (divide by
          sysconf(_SC_CLK_TCK)).  This includes guest time,
          guest_time (time spent running a virtual CPU, see
          below), so that applications that are not aware of
          the guest time field do not lose that time from
          their calculations.

(15) stime  %lu
          Amount of time that this process has been scheduled
          in kernel mode, measured in clock ticks (divide by
          sysconf(_SC_CLK_TCK)).
```

Apresente um *printscreen* com o resultado da visualização do *status* do processo.

3) Você consegue encontrar o executável do seu SO? Execute **`ls -l`** no diretório raiz. Observe que aparece algo assim:

```
lrwxrwxrwx   1 root root    30 jun 29 06:56 vmlinuz -> boot/vmlinuz-4.15.0-54-generic
```

Esse 1º. caractere na linha indica o tipo de arquivo. Neste caso temos l, para *link*. Apresente um *printscreen* demonstrando o executável do seu SO.

Tipos de Arquivos

Os sistemas Unix e Linux trabalham com diferentes tipos de arquivos. Os tipos suportados pelo sistema são:

- **Arquivos normais/regulares:** sequências de bytes (texto, binário, executável, etc.)
- **Diretórios:** lista de outros arquivos (nome do arquivo e inode)
- **Arquivos especiais (dispositivos):** interface entre o sistema e dispositivos de entrada e saída. Podem ser dispositivos orientados a **c**aractere ou a **b**loco
- ***Links*:** podem ser simbólicos (*soft link*: ponteiro para outro arquivo) ou concretos (*hard link*: atribue mais um nome ao mesmo arquivo que esteja na mesma partição)
- ***Sockets* e *Pipes*:** usados para comunicação entre processos (mecanismo para programação)

Arquivos de Dispositivos

No UNIX, tudo é apresentado na forma de arquivos. Ao plugar um pendrive no computador, por exemplo, um arquivo será criado dentro do diretório **/dev** e ele servirá como interface para acessar ou gerenciar o *drive* USB. Nesse diretório, você encontra caminhos semelhantes para acessar terminais e qualquer dispositivo conectado ao computador, como o mouse e até modems.

4) No terminal, vá até o diretório **`/dev`** e liste seu conteúdo (**`ls -l`**). Observe que o inicio de cada linha impressa indica o tipo de arquivo (**`c`**, **`b`** ou **`d`**… eventualmente algum **`l`**).

Exemplos:

- *disco IDE* */dev/hda, /dev/hdb, /dev/hdc, /dev/hdd, ...*
- *disco SCSI/SATA* */dev/sda, /dev/sdb, /dev/sdc, /dev/sdd, ...*
- *partições disco IDE 1 /dev/hda1, /dev/hda2, /dev/hda3, ...*
- *partições disco SCSI1/dev/sda1, /dev/sda2, /dev/sda3, ...*
- *terminal de controle /dev/tty*
- *terminal serial* */dev/tty1, /dev/tty2, /dev/tty3, ….*
- *subdiretório em que são montados os dispositivos USB /dev/usb*

Um fato curioso sobre os dispositivos está relacionado a existência de quatro arquivos na pasta **/dev**: *full*, *zero*, *random* e o *null*. Estes arquivos não correspondem a dispositivos de fato. Você saberia dizer a função de cada um deles?

5) No terminal, digite:

```
$ echo "Hello World"
```

e depois

```
$ echo "Hello World" > /dev/null
```

… o que aconteceu com a saída do comando? Apresente um *printscreen* com o resultado do comando.

6) No terminal, digite o comando abaixo e observe o resultado.

```
$ echo "Hello world" > /dev/full
```

De forma análoga, você consegue dizer o que está acontecendo? Apresente um *printscreen* com o resultado do comando.

Inodes e Atributos de Arquivos

Cada arquivo ou diretório possuiu um inode associado.

7) No terminal, vá até o diretório HOME (`cd ~`) e digite `$ ls -lai .`

Na coluna mais à esquerda, você encontra os números do inode de cada arquivo.

Agora faça a mesma coisa de dentro do diretório raiz. Alguém com o inode 1?

Nessa distribuição, provavelmente você deve ver o "/" com inode 2. Mas /proc e /sys com inode 1. Isso ocorre na verdade porque esses não são diretórios de fato no sistema de arquivos local. Eles são "montados" (veremos isso daqui a pouco…). Mas se você está curioso, digite no terminal `$ findmnt`

Como vimos em sala, no inode de cada arquivo estão armazenados diferentes atributos (informações de controle sobre o arquivo). Ali encontramos informações como:

- Tipo de arquivo: por exemplo, regular, diretório, PIPE, *links* simbólicos, arquivos especiais representando dispositivos
- Número de *hard links* apontando p/ o arquivo
- Tamanho (bytes)
- ID do dispositivo
- Número do i-node: Dentro de um mesmo dispostivo, um i-node (arquivo) tem um número único
- UIDs e GIDs do proprietário
  - `Quando um arquivo é criado, seu UID é herdado do effective UID do processo criador. Já no caso do GID, depende da versão do UNIX (ex: SVR3: herda o effective GID do processo criador; BSD/Linux: herda o GID do diretório pai).`
- *Timestamps* (último acesso, última modificação e última modificação de atributos)
- Permissões e *mode flags*
  - read, write, execute … Acessos divididos por categorias: owner, group, others

Arquivos executáveis têm um atributo especial, o **suid**: quando um usuário executa um arquivo, o *effective* UID do processo correspondente é setado para o UDI do *owner* deste arquivo.

8) No terminal, digite `$ stat NOME_DO_ARQUIVO`. Faça isso para diferentes tipos de arquivos (um arquivo de texto, um dispositivo no **/dev**, um diretório). Observe os campos "Blocos" e "bloco de E/S" (Obs.: podem aparecer em inglês). Apresente um *printscreen* com os resultados dos comandos.

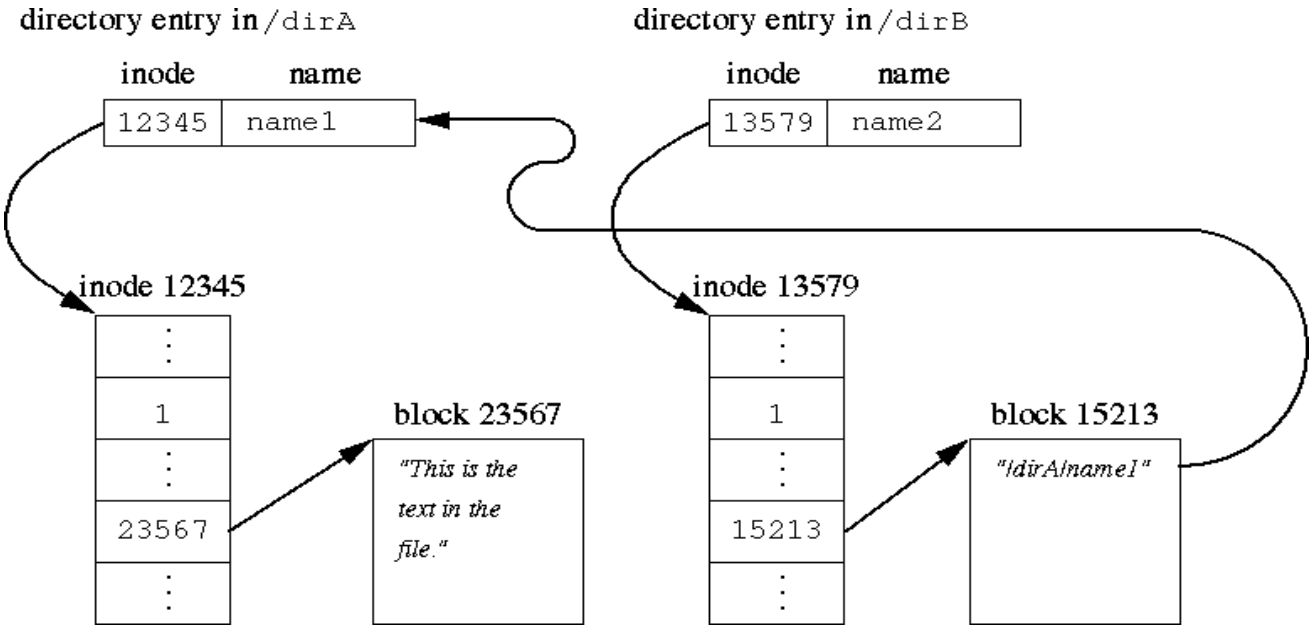
Arquivos do tipo *link*

O *link* é um mecanismo que faz referência a outro arquivo ou diretório em outra localização. Os *links* são arquivos especiais e podem ser identificados com um "l" quando executado o comando **ls -la**.

Symbolic links

No *link* tipo simbólico, o *link* é um arquivo especial de disco do tipo *link*, que tem como conteúdo o caminho para chegar até o arquivo alvo. As principais características são:

- Pode-se fazer *links* simbólicos em arquivos e diretórios;
- O *link* simbólico e o arquivo alvo não precisam estar na mesma partição de disco;
- Se o *link* simbólico for apagado/movido. Somente o *link* será apagado/movido;
- Qualquer usuário pode criar/desfazer um *link* simbólico (respeitando as permissões).



Como criar: **ln –s path1_alvo_do_link path2_nome_do_arquivo_link**

Como visualizar o *link* criado: **ls -l**

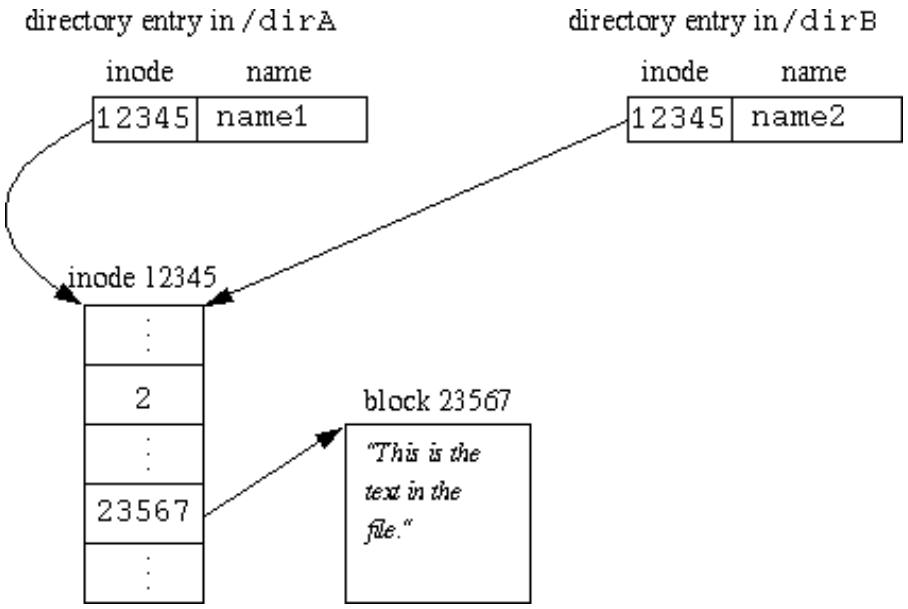
9) No terminal, crie um *link* simbólico usando **ln -s** e depois verifique o resultado usando **ls -l**. Apresente um *printscreen* com o resultado do comando.

Hard links

No *link* tipo *hardlink*, o *link* é apontado para o mesmo inode do arquivo alvo, sendo assim, os dois arquivos serão o mesmo. As principais características são:

- Não é possível fazer um *hardlink* para um diretório;
- Somente é possível fazer *hardlink* em arquivos que estejam em uma mesma partição de disco;
- Se o *hardlink* for apagado/movido, você estará apagando/movendo o arquivo alvo;
- O usuário deve ter permissão de RW no arquivo destino

Como criar: **ln path1_alvo_do_Link path2_nome_do_arquivo_link**



10) No terminal, crie um *hard link* usando **ln** e depois verifique o resultado usando **ls -l**. Tente acessar o arquivo (**more** se for um arquivo ASCII) via o *link* criado. Apresente um *printscreen* com o resultado do comando.

11) Você consegue criar um *hardlink* para um diretório? Apresente um *printscreen* com o resultado do comando.