Apostila de Linux
Seatel 2014

PET Computação UFPR

25 de fevereiro de 2015

Sumário

Capítulo 1

Uso básico

1.1 Distribuições

Debian A segunda mais antiga distribuição Linux ainda mantida e a que mais gerou descendentes. É altamente confiável, mas por conta de suas políticas de segurança e estabilidade os pacotes do repositório demoram a ser atualizados. O ciclo de lançamentos costuma ser de 2 anos.

Ubuntu Derivada do Debian, é a mais popular distribuição Linux e tem milhares de colaboradores pelo mundo. Por isso, ela é considerada quase que um sinônimo de Linux entre os leigos. Surgiu com a premissa de ser uma distribuição voltada aos usuários sem conhecimento técnico e foi bem sucedido. Problemas com licenciamento de softwares proprietários eventualmente impedem que esses softwares estejam facilmente disponíveis para ele.

Foi criada para vir com tudo que um usuário comum precisa em um sistema. Em outras palavras, o sistema está pronto para o uso logo após a instalação.

Lança uma nova versão a cada 6 meses e uma versão LTS a cada 18 meses.

Linux Mint Um derivado do Ubuntu que segue ainda mais fielmente a premissa "plug-and-play" e é menos atrelado as questões ideológicas do software livre. Costumeiramente já tem por padrão muitos softwares proprietários presentes em outros sistemas, como o flashplayer e extensões do Java. Recomendada para iniciantes. Lança novas versões junto com o Ubuntu.

Arch Linux Foi feita para usuários avançados. Oficialmente contém apenas o kernel Linux e um gerenciador de pacotes, sendo o resto deixado a cargo do usuário, que então deve "montar" as demais partes do sistema.

1.2 Gerenciadores Gráficos

Apesar de se poder fazer várias coisas pelo terminal, ele nem sempre é intuitivo. As vezes precisamos visualizar vários arquivos de uma vez e não queremos nos preocupar tanto com sintaxe e telas pretas de terminais. Pra isso existem os gerenciadores gráficos, que são os dispositivos que gerenciam os arquivos e os apresentam da maneira que estamos acostumados a ver.

1.2.1 Unity

É a interface padrão do Ubuntu desde a versão 11.04, substituindo o Gnome. Foi bastante criticada pelos usuários por ser muito pesada e, por isso, não é muito recomendada para computadores antigos. "Bugs" que deixam o sistema lento vêm sendo constantemente corrigidos.

1.2.2 Gnome

O Gnome é provavelmente o ambiente de trabalho mais popular no segmento Linux. As versões Gnome 2.x foram usadas por padrão no Ubuntu, Fedora e Debian; e entre outras grandes e populares distribuições. Este ambiente gráfico de trabalho é simples de usar e oferece uma interface muito limpa. A versão 3 não agradou a muitos usuários que ainda usam a versão 2.x ou a versão 3.x classical (que mantém muitas das características anteriores).

1.2.3 KDE

Juntamente com o Gnome, o KDE é outro dos mais populares ambientes gráficos de trabalho. Oferece uma interface bonita e atraente, e, em muitas situações, este ambiente gráfico é muito mais funcional que o Gnome. Para quem gosta do Ubuntu mas não gosta muito do Unity ou Gnome, o KDE é sem dúvida uma ótima alternativa. É bastante conhecido por ser uma das mais bonitas e rebuscadas interfaces, embora seja também a mais pesada depois do Unity.

1.2.4 XFCE

Uma boa opção semelhante ao antigo Gnome, mas é considerada mais leve.

1.2.5 LXDE

Semelhante ao XFCE, porém ainda mais leve. Apesar das vantagens, muitos dos ambientes que o usam sacrificam usabilidade ou possuem muitos bugs, criando uma certa má fama. Muito recomendado para computadores antigos, principalmente notebooks.

1.2.6 Cinnamon

Um fork do Gnome que foi desenvolvido especificamente para o Linux Mint.

Capítulo 2

Terminal

Enquanto em outros sistemas o uso dessa ferramenta parou no tempo e é extremamente precário, nos dias de hoje elas são um verdadeiro canivete suíço que evoluiu junto com o resto dos sistemas Unix.

2.1 Estrutura de um comando

A partir deste ponto indicaremos um comando que pode ser executado em um terminal por:

```
$ comando
```

Comandos são na realidade scripts ou programas que estão instalados no sistema e cada um deles pode ter sido feito de uma maneira diferente. Existem padrões que devem ser usados quando se está criando um comando, mas na realidade os desenvolvedores não são obrigados a usar o bom senso.

Um comando pode receber outros parâmetros para execução, genericamente: [1]

opções curtas se possuir um único traço (-) na frente `ls -l`

opções longas se possuir dois traços (–) na frente `ls --color`

parâmetros se não possuir nenhum traço na frente `ls /bin`

parâmetros para opções algumas opções querem mais detalhes sobre o que devem fazer, então existe a necessidade de passar parâmetros para elas `ls --color=never`

[1] a nomenclatura utilizada não é oficial.

Abaixo existem alguns exemplos de uso de um comando:

```
$ ls -l -a

$ ls -la

$ ls --color

$ ls /bin

$ ls -1 -la --color=never /bin
```

Conceitualmente, a diferença entre opções e parâmetros é que um parâmetro define coisas que devem ser processadas pelo comando, já as opções modificam a maneira que o comando processará aqueles parâmetros.

2.2 man

O comando `man` exibe a função de determinado comando, programa ou função. Ele é muito útil quando não se sabe o que um comando faz, ou quando se pretende aprender mais sobre a sua utilização. Aconselha-se a leitura do manual sempre que houver dúvidas. Este comando é multifuncão e serve como material de referência offline tanto para comandos e recursos do linux quanto para função e bibliotecas da linguagem C. Isso causa ambiguidades pois ao usar o comando `man write` para obter ajuda sobre a função write da linguagem C o terminal exibirá o manual do *comando* `write` que serve para usar o terminal como chat entre os usuários de uma máquina.

```
$ man ls
```

Neste caso, será mostrado no terminal o manual à respeito do comando *ls*.

Existem algumas outras opções de `man` importantes, visto que podem existir mais de um comando com o mesmo nome.

Quando não temos certeza se aquele nome é usado somente uma vez, é importante executar o comando `whatis` ou `man -f`, para depois abrir o manual correto do que queremos.

Para saber todas as aplicações do mesmo nome os dois comandos abaixo são equivalentes:

```
$ whatis nome
$ man -f nome
```

Por exemplo, ao digitar `whatis time`, como a palavra *time* está relacionada a mais de um comando ou função, aparecerá mais de um resultado. Veja o exemplo a seguir.

```
$ whatis time
time (7)             - overview of time and timers
time (2)             - get time in seconds
time (1)             - run programs and summarize system resource usage
```

No caso de mais de um resultado para o comando `whatis nome`, para abrir o manual de uma das opções resultantes:

```
$ man 2 time
```

Assim, o terminal mostrará o manual da função *time* da linguagem C.

2.3 Comandos básicos

Nesta sessão trataremos dos comandos básicos de manipulação de arquivos e diretórios (pastas) no linux. Alguns comandos serão apenas brevemente citados pois sua utilização básica é bastante simples e não existem novos conceitos envolvidos. Mais detalhes e outras opções podem ser encontrados no manual do comando (`man`).

`cd` Ao abrir um terminal estaremos por padrão na nossa pasta pessoal, mas geralmente é mais fácil lançarmos os comandos necessários para alguma tarefa se estivermos em alguma outra pasta conveniente. O comando para alterar nosso diretório atual é o `cd`:

```
$ cd Downloads
$ cd ../Documentos/2014/
$ cd /home/pet/
```

`ls` Lista o conteudo de um diretório.

```
$ ls
$ ls -l
$ ls -a
$ ls -la /
```

`touch` Cria um arquivo.

```
$ touch atualiza.sh
```

`rm` Exclui um arquivo. Arquivos deletados pelo terminal não vão para a lixeira e a princípio não há como recupera-los.

```
$ rm teste.txt
$ rm *.o *~
$ rm Downloads/*.torrent
$ rm -rf /
```

`cp` Faz uma cópia de um arquivo (equivalente ao ctrl+c ctrl+v).

```
$ cp filmes/* ~/Vídeos
$ cp teste.txt link_pendrive
```

`mv` Move o arquivo (equivalente ao ctrl+x ctrl+v). Também usado para renomear arquivos.

```
$ mv Downloads/*.avi Vídeos
$ mv relatorio.pdf Documentos
$ mv teste.txt teste2.txt
```

`mkdir` Criar um novo diretório.

```
$ mkdir fotos
$ mkdir curso_linux curso_linux/exemplos
$ mkdir -p curso_linux/exemplos
```

`rmdir` Deleta diretórios vazios. Se o diretório não estiver vazio use `rm -rf dir`.

```
$ rmdir fotos
```

`cat` Mostra na tela o conteudo de um arquivo de texto.

```
$ cat arquivo.txt
```

`more` e `less` São dois comandos para mostrar o conteúdo de um arquivo na tela porém de uma forma mais esperta do que o `cat`. O `more` mostra inicialmente as primeiras linhas do arquivo enquanto o `less` mostra as últimas.

```
$ more arquivo.txt
$ less arquivo.txt
```

2.4 Entrada e saída de comandos

Todo comando dispõe de alguns meios padrões para comunicação com o meio exterior. Primeiro precisamos conhecê-los para depois aprender a manipulá-los.

Como meios de entrada temos a *entrada padrão* e os *parâmetros*; e como meios de saída, a *saída padrão*, a *saída de erros* e o *código de retorno*.

Vamos explicar os mais importantes deles, mas antes precisamos definir um termo:

buffer é um arquivo ou alguma região da memória do dispositivo que serve para guardar dados temporariamente.

Entrada padrão A entrada padrão é um buffer de onde o comando lê dados, o padrão para comandos é que eles tirem esses dados do teclado.

Saída padrão É o buffer onde o comando coloca os resultados do processamento (quando aplicável). Por padrão é a tela do terminal.

Saída de erros É o buffer onde o comando coloca as mensagens de erro pertinentes. A tela do terminal também é padrão para esta saída, o que causa uma certa confusão na leitura dos resultados.

2.4.1 Redirecionamento de entrada e saída

É possível e geralmente muito útil modificar os buffers que alguns comandos usam para entrada e saída. Exemplos de casos de uso:

- quando um comando produz uma saída muito grande e gostaríamos de analisá-la com calma
- quando a saída de um programa gera uma informação que gostaríamos de guardar

- quando queremos que um comando processe a saída de outro comando
- quando não queremos que os resultados das saídas padrão e de erros sejam mostrados no mesmo local
- quando um programa que lê da entrada padrão deve processar muitos dados (é ineficiente digitar pelo teclado todos os dados)
- quando deixamos um comando demorado executando em background e no dia seguinte queremos ver os resultados dele

Conectando saída padrão em arquivo

Abaixo está o `ls` padrão que lista o conteúdo de um diretório e o mostra na saída padrão:

```
$ ls
```

Mas, e se queremos guardar essas informações em um arquivo? Ctrl+c Ctrv+v da tela? Tem jeito melhor. O comando abaixo mostra como "conectar" a saída padrão do `ls` em um arquivo de texto. Assim, o resultado do comando é escrito diretamente no arquivo.

```
$ ls > arquivo.txt
```

O equivalente para a saída de erros seria:

```
$ ls 2> arquivo.txt
```

Os dois podem ser usados em conjunto:

```
$ ls > arquivo.txt 2> erros.txt
```

Caso você queira ignorar uma das saídas redirecione a saída equivalente para o arquivo "/dev/null"[2]:

```
$ ls 2> /dev/null
```

Conectando um arquivo na entrada padrão

Se para a saída usamos >, para a entrada usamos <:

```
$ grep teste < exemplo.txt
```

[2] o arquivo /dev/null é como um "buraco negro" que faz desaparecer tudo que é mandado para ele

Conectando a saída padrão de um comando na entrada padrão de outro comando

Essa operação é uma das mais interessantes (e complexas) dos interpretadores de comandos e seu uso será melhor estudado a seguir.

```
$ ls | grep teste
```

Capítulo 3

Filtros

Existem comandos de terminal para quase tudo. Afinal, um comando pode ser um programa instalado no sistema e qualquer pessoa pode criar um novo programa e distribuí-lo. Em especial, é útil conhecer alguns comandos padrões do sistema que fazem processamento de texto:

`grep` O grep elimina as linhas de um arquivo que não possuam um determinado texto/padrão e coloca as linhas restantes na saida padrão. Se nenhum arquivo for informado para ser processado a entrada padrão é usada.

`tr` Lê um texto da entrada padrão e substitui uma sequência de caracteres nela. Coloca o resultado na saida padrão.

`wc` Faz contagens em arquivos de texto. Mostra informações como quantidade de palavras, linhas e caracteres dependendo das opções passadas.

`cut` Quebra um texto em colunas a partir de um delimitador dado e mostra uma coluna na saída padrão.

Outros comandos úteis que se encaixam nessa categoria são `paste`, `sort`, `shuf`, `head`, `tail` e `uniq`.

Capítulo 4

Gerenciamento de processos

Nos sistemas operacionais, um processo é a forma de representar um programa em execução. É o processo que utiliza os recursos do computador - processador, memória, etc - para a realização das tarefas para as quais a máquina é destinada. O Linux gerencia os usuários e os grupos através de números conhecidos como UID (User Identifier) e GID (Group Identifier). A identificação de cada processo é feita através do número PID.

Comandos principais Para visualizar os processos em execução:

```
$ ps
```

Para exibir também os processos de outros usuários:

```
$ ps -a
```

Para exibir os processos com detalhes de memória:

```
$ ps -u
```

Para exibir os processos que não estão vinculados a terminais:

```
$ ps -x
```

Caso seja necessário executar mais de uma das opções basta fazer:

```
$ ps -aux
```

Caso seja necessário saber quais processos se relacionam é possivel visualizá-los em forma de árvore:

```
$ pstree
```

O comando "ps"lista os processos em execução no sistema. Porém ele não traz informações sobre o quanto de processamento ou de memória cada processo está consumindo. Apesar disso, o "ps"é uma maneira bem mais ágil de consultar o PID de um processo, principalmente ao ser usado em conjunto com o grep (O grep permite filtrar os processos mostrados no terminal). A utilização do grep é feita da seguinte forma:

```
$ ps -aux | grep -i nomedoprograma
```

A barra vertical, ou pipe (—), faz com que o resultado do "ps"seja direcionado para o comando grep que, por sua vez filtrará apenas as linhas que tenham a palavra "nomedoprograma".

Para exibir os processos que utilizam CPU em tempo real:

```
$ top
```

Para parar a visualização trazida por "top"basta digitar Q ou Ctrl+C.

Comandos úteis para finalizar processos

Para terminar um processo através do número do PID:

```
$ kill PID
```

Caso deseje terminar mais de um processo de uma vez só:

```
$ kill PID1 PID2 PID3
```

Para mostrar as outras opções do comando kill:

```
$ kill -l
```

Caso o processo não seja encerrado com kill, o seguinte comando forçará o encerramento, basta escrever:

```
$ kill -9 PID
```

Existem duas maneiras para encerrar todos os processos do mesmo programa:

```
$ killall nomedoprograma
$ pkill nomedoprograma
```

Para dar o "poder"de matar um processo ao cursor (assim, ao clicar em algum programa na tela o mesmo será encerrado) basta digitar:

```
$ xkill
```

Capítulo 5

apt-get

O APT ou `apt-get`, é um recurso desenvolvido originalmente para o Debian, porém incorporado por várias distribuições Linux, como o Ubuntu, que possibilita o gerenciamento de pacotes .deb de forma rápida e prática. Com ele podemos instalar um programa sem se preocupar em encontrar e instalar as suas dependências, atualizar, remover pacotes e até mesmo deixar o seu Linux na versão mais atual possível.

5.1 Atualizando lista de pacotes

Para encontrar os programas o APT utiliza uma lista de repositórios, que geralmete está em */etc/apt/sources.list*. Porém para que o APT saiba quais pacotes estão no repositório e consiga comparar com a versão instalado no seu computador, temos que pedir para ele atualizar as referências. Fazemos isso usando o comando `apt-get update`.

5.2 Instalando pacotes

Para instalar um pacote, basta utilizar o comando `apt-get install nomepacote`. Por exemplo, caso se queira instalar o pacote *chromium*, o comando ficaria `apt-get install chromium`. Após a execução do comando, o programa deverá ser instalado sem problemas, sendo que caso seja necessário instalar uma dependência o APT pedirá a sua permissão.

5.3 Removendo pacotes

Para desinstalar um programa com o APT, basta utilizar o comando `apt-get remove nomepacote`. Por exemplo, caso queira deletar o pacote *kile*, o comando seria `apt-get remove kile`.

5.4 Atualizando pacotes

Para obter a mais nova versão de um software disponível em seus repositórios, você deve utilizar o comando `apt-get -u upgrade`. A opção "u"mostra a lista de pacotes sendo atualizados e, portando, é opcional.

5.5 Encontrando um novo pacote

O APT também tem um comando para busca de pacotes para caso se tenha alguma duvída quanto ao nome do pacote ou se queira encontrar algo novo. Para tal, basta utilizar o comando `apt-cache search termo`. Por exemplo, para encontrar pacotes com o termo *tux*, o comando seria `apt-cache search tux`.

Obs: Geralmente o comando `apt-get` necessita de permissão de administrador