1 Introdução

Este material é apresentado como parte de uma série de minicursos que serão ministrados sobre tópicos variados com o intuito de *nivelar* o conhecimento dos alunos com relação a conceitos básicos que não estão inclusos nas ementas das disciplinas regulares do curso, bem como colocá-los em contato com ferramentas úteis e boas práticas da Computação.

Os conteúdos são dispostos aqui apenas como um apanhado geral com o objetivo de prover um embasamento mínimo e recomenda-se que outras fontes de pesquisa sejam utilizadas para o aprofundamento nestes assuntos.

1.1 Unix

O sistema **Unix** é um sistema operacional desenvolvido por Ken Thompson, Dennis Ritchie, Douglas McIlroy e Joe Ossanna em 1970, na American Telephone & Telegraph Company (AT&T).

Este sistema foi projetado para ser portável, multi-tarefa e multi-usuário. Sistemas Unix são caracterizados por vários conceitos, como o uso de arquivos de texto para guardar dados, um sistema de arquivos hierárquico, o tratamento de certos dispositivos como arquivos e a idéia de que é melhor ter vários programas pequenos, cada um desempenhando uma única tarefa, de forma que eles possam ser encadeados para realizar uma tarefa maior do que o uso de um só programa que possua todas as funcionalidades. Estes conceitos são conhecidos como a "filosofia Unix".

1.2 Tanenbaum e o MINIX

Em 1987, Andrew S. Tanenbaum, enquanto professor da Vrije Universiteit em Amsterdã, desenvolveu o **MINIX** (mini-Unix), um sistema que seguia os mesmos princípios do Unix mas que seria usado somente propósitos educacionais, como material de apoio ao seu livro "Operating Systems: Design and Implementation" (em português: "Sistemas Operacionais: Projeto e Implementação").

Os princípios aplicados por Tanenbaum no MINIX, influenciaram um de seus alunos, Linus Torvalds quando da criação de seu próprio sistema operacional, que mais tarde se tornaria o *kernel* do **Linux**.

2 Conhecendo o Linux

2.1 GNU/Linux

Embora a maioria das pessoas conheça apenas o termo Linux, a nomenclatura mais acurada para o sistema que se usa nos dias de hoje é "GNU/Linux". Isso se dá devido ao fato de que a maioria das distribuições Linux (vide 2.3) na verdade contêm o *kernel* (vide 2.2) do Linux em conjunto com *softwares* GNU.

O projeto GNU foi iniciado em setembro de 1983 por Richard Stallman, enquanto trabalhava no laboratório de inteligência artificial do Massachusetts Institute of Technology (MIT). O intuito do projeto é criar um ambiente de colaboração em que usuários de computadores possam ter acesso a programas que são distribuídos juntamente com o seu código-fonte (*software* livre). A idéia da disseminação do *software* livre é dar aos seus usuários total controle sobre os programas que são utilizados, ao contrário do *software* proprietário, em que é provido o executável, mas não há como realizar modificações, nem mesmo saber o conteúdo do código que gerou o mesmo.

Juntamente com o projeto GNU, foi criado o movimento "Free software" e a instituição de fins não lucrativos "Free software foundation", que atualmente, entre outras atividades, administra o projeto GNU e as licenças GNU.

2.2 O kernel

O *kernel* é a parte do sistema que faz a ponte entre os recursos de *hardware*, como memória, CPU, dispositivos de entrada e saída; e a camada de *software*.

Numa configuração monolítica (Linux, Windows, etc), os programas rodam em "espaço de usuário", o que significa que um programa usa apenas o espaço de memória reservado a ele e não possui acesso direto a recursos do sistema. Para que um programa efetue uma ação como por exemplo exibir uma mensagem na tela, é necessário que uma "chamada de sistema" seja feita através do *kernel*, já que o programa não tem acesso à placa gráfica do computador.

A figura 1 exemplifica a arquitetura básica de um sistema Linux: Na camada superior, em espaço de usuário, estão as aplicações de usuário e a biblioteca GNU. Estas aplicações interagem com o *kernel* através de uma interface de chamadas de sistema. O *kernel*, por sua vez, faz acesso ao hardware.

2.3 Distribuições

Como visto anteriormente, o que se conhece popularmente como Linux, é na verdade apenas o *kernel*. Então, para que possamos ter um sistema funcionando, é necessário fazer a escolha de uma "distribuição". Distribuições nada mais são que o *kernel* do Linux juntamente com alguns outros *softwares* (em sua maioria GNU, mas não necessariamente) que dão funcionalidade ao sistema.

Algumas distribuições mais comuns são: Ubuntu, Debian, Fedora, entre várias outras.

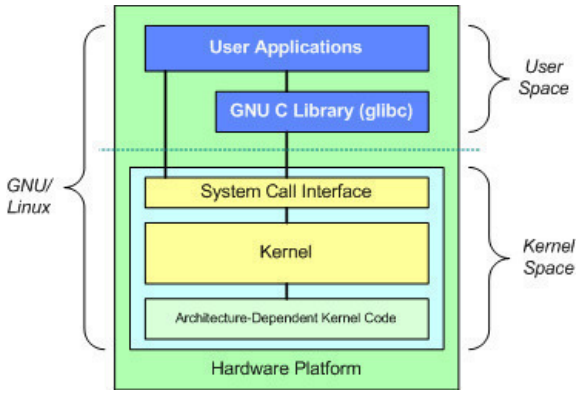

Figura 1: Arquitetura de um sistema GNU/Linux

O conceito de distribuição (também chamadas de "distro") geralmente gera alguma confusão entre os usuários de Linux, já que todos os *softwares* com os quais a pessoa está interagindo no uso cotidiano do sistema são na verdade *softwares* que são providos pela distribuição. Isto faz com que um usuário, ao experimentar várias *distros* em seu computador, não perceba que na verdade, o *kernel* rodando por baixo da camada de *software* de usuário é o mesmo para todas as distribuições.

2.3.1 Desktop Environments e Window Managers

Outra distinção importante que deve ser feita é quanto ao conceito de *Desktop Environments* (ambientes de *desktop*) e *Window Managers* (gerenciadores de janelas). Falando de maneira grosseira, *Desktop Environments* são a parte da distribuição responsável por partes do sistema como o "menu iniciar", gerenciamento de sessão e o próprio *desktop*. Os *Window Managers* são a parte que gerencia a aparência das janelas e o comportamento de funções como a troca de janelas (alt + Tab).

A maioria das distribuições já provê um DE e um WM, porém algumas delas trazem apenas os programas essenciais, ficando a instalação da parte gráfica a cargo do usuário. É importante entender a diferença entre *distro*, *desktop environment* e *window manager* para que no caso de um problema ou caso alguma funcionalidade não lhe agrade, você saiba exatamente em qual parte do sistema está o problema.

3 Como obter?

A maioria das distribuições Linux é disponibilizada grátis na *internet* na forma de imagens *.iso*. Para a instalação, as seguintes alternativas existem:

1. Manter o sistema operacional nativo e rodar Linux em uma máquina virtual. Recomendado para testes iniciais, antes de se decidir por uma distribuição.

2. Manter o sistema operacional nativo e o Linux em dual-boot.
   Recomendado para aqueles que usam com frequência as funcionalidades do sistema nativo, como jogos ou *softwares* proprietários não disponíveis para Linux.

3. Remover o sistema nativo e usar somente Linux.
   Recomendado para aqueles que já tem certa experiência com Linux e não necessitam de outro sistema operacional.

4. Trocar o sistema nativo pelo Linux e rodar o sistema antigo em uma máquina virtual.
   Para aqueles que já optaram por usar Linux, mas que em uma pequena fração do tempo, ainda precisam de funcionalidades do outro sistema.

3.1 Máquina Virtual

Uma máquina virtual (**VM**) é, como o nome indica, um computador virtual baseado somente em *software* que funciona como um emulador, reproduzindo a arquitetura e funcionalidades de um computador de verdade. Através da máquina virtual, pode-se ter vários sistemas operacionais instalados em uma mesma máquina e utilizá-los todos sem afetar o sistema base do computador.

Figura 2: Exemplo do uso de uma máquina virtual

Observe a imagem 2. Nela pode-se ver um sistema Windows (XP) servindo como base para outro sistema Windows (Win 7) rodando dentro da VM. Para efeitos de nomenclatura, chama-se de *host* o sistema que está como base para executar a máquina virtual e *guest* o sistema que está sendo emulado na VM. No exemplo acima, o *host* é definido pelo Windows XP, enquanto o *guest* é o Windows 7.

O sistema que está dentro da VM não afeta de maneira alguma o sistema base. Desta forma, já que a VM nada mais é que um *software* rodando em um sistema qualquer, podemos levantar as seguintes vantagens de seu uso:

- Nada que seja feito no sistema virtual afetará o sistema real.
- Pode-se instalar e desinstalar vários sistemas com facilidade.
- Em situações em que o sistema base possui acesso restrito, como em um ambiente acadêmico, a VM pode ser uma boa alternativa, já que dentro da máquina virtual o usuário possui controle total.

Porém, pelo mesmo fato de ser apenas um programa rodando em seu computador a VM tem as desvantagens de que:

- Como estão sendo utilizados recursos de um sistema para rodar outro sistema, ocorrerá diminuição da eficiência, e o sistema poderá ficar lento ou não ter memória suficiente para executar algum programa.
- Para uma utilização prolongada, será sempre melhor instalar o sistema diretamente em seu computador.

3.2 Dual-boot

Outra alternativa bastante popular na instalação de sistemas operacionais é o *dual-boot*, ou *multi-boot*. Nessa situação, os dois (ou mais) sistemas serão instalados totalmente em uma máquina, cada qual recebendo uma porção do disco rígido. Quando o usuário liga o computador, uma tela é exibida permitindo que se escolha qual dos sistemas deve ser carregado. Para alternar entre sistemas, basta reiniciar o computador e escolher o outro sistema.

A principal vantagem dessa abordagem é que ambos os sistemas possuem a máquina toda para si, exatamente como ocorre quando há apenas um sistema operacional instalado.

Uma desvantagem que pode ser citada é que não é possível executar os dois sistemas ao mesmo tempo e deve-se aguardar o *reboot* para selecionar o outro sistema.

4 Porque usar?

Existem algumas características dos sistemas baseados em Linux que fazem com que eles sejam atrativos para o uso na Ciência da Computação. Pode-se dizer que um sistema Linux possui foco nos desenvolvedores de sistemas enquanto que outros sistemas operacionais possuem foco no usuário.

4.1 Familiaridade geral com termos e conceitos

Pelo fato de se basear em software de código aberto, existem muitos conceitos teóricos da computação que podem ser verificados muito facilmente quando se usa Linux. Um exemplo simples é o próprio *kernel*, que é escrito na linguagem C. Ao estudar programação ou Sistemas Operacionais, o código do *kernel* pode ser explorado para que se entenda como os conceitos estudados são aplicados em um sistema operacional "de verdade".

Outro exemplo são as aplicações na área de Redes de Computadores, que em sua maioria se baseiam em Linux. Ao usar Linux no seu dia a dia, você já estará se familiarizando com alguns conceitos nesta área.

4.2 Aprendizado de boas práticas

O uso contínuo do sistema pode levar ao conhecimento de boas práticas de programação. Um bom exemplo é a "filosofia" de se projetar programas que façam apenas uma tarefa, mas a façam de maneira bem feita.

Outro ponto que se pode ressaltar é o fato de o Linux ser bastante documentado. Isso significa que os programas possuem um manual de instruções de utilização. Acostumar-se a ler manuais e pesquisar sobre funções desconhecidas é um ponto importante dentro da computação, já que constantemente nos deparamos com código escrito por outras pessoas ou situações novas nas quais não possuímos domínio. Saber buscar informação nos lugares certos é algo valioso.

4.3 Gerenciamento de pacotes

Além dos benefícios dentro da computação, as distribuições Linux também trazem algumas funcionalidades que são úteis para qualquer pessoa. O gerenciamento de pacotes é uma delas.

Normalmente, as distribuições Linux aplicam o conceito de "pacote". Um pacote nada mais é que um programa preparado para ser instalado em seu sistema. Pacotes são específicos para cada distribuição e garantem que programas serão instalados e desinstalados de forma correta. Além disso, há o conceito de "repositórios" que são servidores que abrigam vários pacotes. A vantagem dessa abordagem é que quando se quer instalar um programa, ao invés de se procurar em um site de busca pelo programa que se deseja, baixar um executável e instalar o programa; um usuário Linux pode simplesmente digitar o nome do programa em seu gerenciador de pacotes que irá automaticamente percorrer

os repositórios procurando pelo pacote correto, mais atual e com garantia de idoneidade. Isto mantém o sistema muito fácil de gerenciar, já que o gerenciador de pacotes toma conta de tudo. Os pacotes também são, em geral, submetidos ao julgamento da comunidade de usuários do sistema, tornando pacotes "quebrados" ou com código malicioso mais difíceis de serem encontrados.

5 Uso da linha de comando

Quando utilizamos um computador, estamos acostumados a usar a interface gráfica. Isso significa que fazemos bastante uso do *mouse* para acessar as variadas funções do sistema e os resultados das nossas ações são exibidos na tela de maneira muito clara. Ex: ao mover um arquivo, podemos simplesmente clicar sobre ele e arrastá-lo para outra pasta. Esta interação é feita de maneira muito intuitiva.

Embora hoje em dia, quase todas as distribuições Linux possuam uma interface gráfica, o seu ponto forte é a interface de linha de comando. A CLI (*Command Line Interface*) não é baseada nos preceitos de *mouse* + gráficos com os quais estamos acostumados. A sua utilização se faz através de comandos de texto. Para realizar tarefas como a criação de uma pasta, ou fazer a cópia de um arquivo, um comando será digitado em um console. Para executar a mesma ação do exemplo anterior, digitaríamos o comando: `$ mv local_arquivo local_destino` . O comando não retornaria resposta alguma, o que em um sistema Unix significa que tudo ocorreu com sucesso. No começo, esta nova forma de interagir com o computador e estes comandos podem parecer intimidantes, porém com o tempo e costume, a utilização da CLI fará com que a navegabilidade se torne rápida e focada.

Além da navegação básica pelas pastas, os comandos da CLI também podem ser combinados de forma a executar tarefas complexas, como manipulação de arquivos ou gerenciamento de rede, atividades que em outros sistemas operacionais provavelmente teríamos que procurar por um programa na internet para realizá-las.

A seguir, veremos alguns comandos básicos e sua utilização.

Como um guia, todas as linhas de comandos terão o caracter $ no começo, apenas para indicar a linha onde será digitado o comando em questão. O caracter não faz parte do comando e deve ser ignorado.

Ao abrir uma janela de terminal, primeiramente precisamos saber "onde estamos". Para saber em qual pasta você está trabalhando na linha de comando, digite **pwd**, que significa "print working directory".

```
$ pwd
/home/usuario
```

A resposta do comando é a pasta atual. "/home/usuario", seria similar ao "C:\Usuários\usuário\"

de um sistema Windows.

Há alguns "atalhos" que podem ser utilizados na hora de digitar comandos:

- o caracter ~ é uma outra forma de representar a pasta "/home/usuario"
- um **.** significa o diretório atual e **..** é equivalente ao diretório anterior. Portanto, no caso acima, **.** significaria "/home/usuario" e **..** significaria "/home"

A mudança de diretórios, ou seja, pastas, deve ser feita através do comando **cd** ("change directory").

```
$ cd /home/usuario/Área\ de\ Trabalho/
$ cd ..
$ pwd
/home/usuario
$ cd ./Área\ de\ Trabalho/
$ pwd
/home/usuario/Área\ de\ Trabalho
```

É importante lembrar que o / também é um diretório! Esta é a pasta raiz do seu sistema, portanto:

`$ cd /home/usuario/` é **diferente** de: `$ cd home/usuario/`

Você pode criar uma pasta usando o **mkdir** ("make directory"). Lembre-se que sempre que você não souber em que pasta você está, você pode utilizar o pwd.

```
$ cd /tmp/
$ mkdir NovaPasta
$ mkdir NovaPasta/Pasta2
```

Se você desejar saber o conteúdo de uma pasta, basta utilizar o comando **ls** ("list"). Este comando mostrará tanto pastas quanto arquivos presentes.

```
$ cd /tmp/
$ ls
NovaPasta
$ cd NovaPasta
$ ls
Pasta2
```

Para remover uma pasta, um simples comando pode ser usado, o **rmdir** ("remove directory"), porém a pasta só será removida se ela estiver vazia, caso contrário uma mensagem de erro aparecerá na tela. No nosso exemplo, podemos excluir a Pasta2 dessa forma:

```
$ cd /tmp/
$ rmdir ./NovaPasta/Pasta2
```

Você pode criar um arquivo vazio com apenas um comando, o **touch**.

```
$ cd /tmp/NovaPasta
$ touch teste.txt
```

Para podermos copiar um arquivo, utilizaremos o comando **cp**.

```
$ ls /tmp/NovaPasta
teste.txt
$ cp /tmp/NovaPasta/teste.txt /tmp/teste2.txt
```

Utilizando o **cp** com o parâmetro **-r** somos capazes de copiar pastas.

```
$ cp -r /tmp/NovaPasta /tmp/NovaPasta2
```

Remover um arquivo definitivamente é bem simples quando utilizamos o comando **rm**. Apenas é necessário colocar o nome do arquivo que se deseja excluir e estar na pasta correta.

```
$ cd /tmp/
$ rm teste2.txt
```

Novamente, para excluirmos uma pasta, é preciso usar o parâmetro **-r**. É preciso ter muito cuidado ao usar essa funcionalidade, pois esse comando apagará definitivamente qualquer pasta mesmo que haja conteúdo dentro.

```
$ cd /tmp/
$ rm -r NovaPasta2
```

Há outros comandos que são interessantes:

- `cat ArquivoQualquer.txt` - Imprime na tela o conteúdo de ArquivoQualquer.txt
- `find /home/usuario -name ArquivoQualquer` - Procura dentro da "home" um arquivo de nome ArquivoQualquer.
- `grep string caminho_para_pasta/` - Procura dentro dos arquivos da pasta a string colocada após grep.
- `shutdown -h now` - Desliga o computador.
- `shutdown -r now` - Reinicia o computador.
- `sudo comando` - Executar um programa como um super usuário.

- `tar -cf Arquivo.tar.gz caminho_para_pasta/` - Comprimir arquivos na pasta indicada pelo caminho.
- `tar -xf Arquivo.tar.gz` - Extrair arquivos de um .tar.gz.

6 Passos seguintes

Como já mencionado anteriormente, os conceitos aqui apresentados servem apenas como um direcionamento inicial. Espera-se que o leitor continue a buscar fundamentação teórica sobre o uso e funcionamento de sistemas operacionais como um todo. As referências apresentadas a seguir podem servir como um ponto de partida.

Recomenda-se que o usuário experimente com várias instalações e configurações de sistema diferentes e que durante o seu processo de aprendizagem faça uso intensivo de sites de busca e dos manuais e outros tipos de documentação disponíveis na Internet. O uso das máquinas virtuais como um "local seguro" para experimentos também é incentivado.

Outro ponto importante de ser ressaltado é que embora, em alguns momentos, o uso de sistemas Unix e software livre em geral possa parecer difícil ou complicado, o conhecimento e aprofundamento nestes tópicos com certeza trará frutos para sua vida acadêmica e profissional.

Desta forma, os autores deste documento colocam-se à disposição para o esclarecimento de dúvidas e suporte de maneira geral através da lista de emails `psl_pg@googlegroups.com`.

Referências

GNU.ORG. Disponível em: <https://www.gnu.org/gnu/linux-and-gnu.html>.

IBM. CT316, *Anatomy of the Linux kernel*. 2007. Disponível em: <http://www.ibm.com/developerworks/library/l-linux-kernel/>.

IBM. CT316, *Learn Linux, 101: A roadmap for LPIC-1*. 2012. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-lpic1-v3-map/index.html>.

SHOTTS, W. *Linux Command Line*. No Starch Press, 2012. ISBN 9781593274269. Disponível em: <http://books.google.com.br/books?id=ZBKsMYz1Q4kC>.

TANENBAUM, A. S. *Modern Operating Systems*. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007. ISBN 9780136006633.

WIKIPEDIA. *Linux kernel*. 2013. Page Version ID: 576143299. Disponível em: <http://en.wikipedia.org/w/index.php?title=Linux_kernel&oldid=576143299>.