Básico: A interface mais simples existente no linux é o *shell*. O *shell* é um interpretador de comandos que analisa o texto digitado na linha de comandos e os executa produzindo algum resultado.

Além de ser um interpretador de comandos, o *shell* também é um poderoso ambiente de programação capaz de automatizar praticamente tudo em um sistema Linux. Veja mais algumas considerações referentes ao *shell* e outros aspectos básicos.

1. Durante a execução do *bash* são mantidas algumas variáveis especiais que contêm alguma informação importante para a execução do *shell*. Estas variáveis são carregadas no início de sua execução e também podem ser configuradas manualmente em qualquer momento.
    1. ***PS1(Prompt String):*** Esta variável guarda o conteúdo do *prompt* de comando do *bash* quando ele está pronto para receber instruções.
    2. ***PS2:*** Guarda o conteúdo do prompt de comandos quando são necessárias múltiplas linhas para completar um comando.
    3. ***PATH:*** O path guarda uma lista dos diretórios que contêm os programas que poderão ser executados sem passar na linha de comandos o caminho completo de sua localização.
2. Para imprimir o conteúdo de uma variável de ambiente utilize o comando *echo*.
3. É importante saber que o interpretador de comandos do bash segue a seguinte ordem para achar e executar os comandos digitados:
    1. *O comando digitado é um comando interno do interpretador de comandos?*
    2. *Se não for, o comando é um programa executável localizado em um diretório encontrado no PATH?*
    3. *A localização do comando foi explicitamente declarada?*
4. Depois de criar uma variável do shell, é preciso exportá-la para o ambiente com o comando *export*. Quando uma variável é exportada para o ambiente, ela fica disponível para todos os processos filhos do shell.
5. Alguns comandos aceitam opções com "-" ou "--". Valendo observar que aqueles que utilizam "--" não podem ser combinados.
6. Para alguns comandos os argumentos são opcionais. Para outros são necessários. Os argumentos são os parâmetros que os comandos aceitam ou de que necessitam.
7. É importante ter em mente que o linux somente vai executar os comandos que sejam internos do interpretador, ou comandos cuja localização esteja na variável PATH ou comandos chamados com o seu caminho explícito. O bash também permite uma sequencia de comandos em uma mesma linha. Para isso deve-se separar os comandos com o símbolo ";".
8. O bash escreve no arquivo chamado ***.bash_history,*** localizado no diretório *home* de cada usuário, o histórico de todos os comandos digitados pelos usuários.
9. O símbolo "~" faz referência ao diretório *home* do usuário atual.

Redirecionamento e Condutores (PIPE): Linux foi criado por programadores para programadores, fez-se necessário que os comandos e processos tivessem a habilidade de tratar as entradas e saídas de dados com grande facilidade.

O Linux possui três tipos de entradas padrão, são elas:

1. **stdin**: é a entrada de fluxo de dados. Como exemplo temos o teclado, o mouse, o Driver de CD/DVD, etc. Todos eles alimentam o computador com informações. Pode ser representado pelo número 0.
2. **stdout**: é a saída de um fluxo de dados em condições normais. Como exemplo temos o monitor, a impressora, o disquete, um arquivo, etc. Todos eles recebem informações do computador. Pode ser representado pelo número 1.
3. **stderr**: é a saída de um fluxo de dados em condições de erro ou insucesso em um determinado processamento. A saída de erro poderá ser direcionada para o monitor ou para um arquivo de LOG. Pode ser representado pelo número 2.

Para redirecionar um resultado de uma saída para uma outra saída utilizamos o sinal de maior (>) e para direcionar uma entrada para outra entrada usamos o sinal de menor (<). Ainda podemos redirecionar uma saída para uma entrada usando uma barra vertical especial, o sinal em inglês chamado "pipe" (|) ou condutor.

Gerenciando arquivos

Vários sistemas operacionais (SO) oferecem diferentes sistemas de arquivos, mas todos tem a mesma finalidade: ***Oferecer ao SO a estrutura necessária para ler/gravar os arquivos/diretórios***. Nos sistemas de arquivos, de modo geral, os objetos são organizados de forma hierárquica e cada um possui identificação única dentro da tabela.

A identificação dos objetos de um sistema de arquivo no Linux é conhecida como ***inode***. Ele carrega as informações de onde o objeto está localizado no disco, informações de segurança, data e hora de criação e última modificação, dentre outras. Quando criamos um sistema de arquivos no Linux, cada dispositivo tem um número finito de ***inodes*** que será diretamente proporcional ao número de arquivos que este dispositivo poderá acomodar.

A estrutura hierárquica do Linux segue o padrão chamado de *Filesystem Hierarchy Standard* ou *FHS.* Segue agora uma breve descrição de cada um dos diretórios principais:

- /bin: Contêm os comandos que podem ser utilizados pelos usuários e pelo administrador do sistema.
- /boot: Contém tudo que é necessário para carregar o sistema, exceto os arquivos de configuração e o gerenciador de boot. O /boot inclui o setor master de carga do sistema (master boot sectors) e arquivos de mapa de setor.
- /dev: Contém um arquivo para cada dispositivo que o Kernel pode suportar.
- /etc: Contém arquivos necessários à configuração do sistema. No diretório /etc/skel deste diretório é replicado para o diretório padrão (home) dos novos usuários criados no sistema.
- /home: Contém os diretórios locais do usuário.
- /lib: Arquivos de bibliotecas essenciais ao sistema, utilizadas pelos programas em /bin e módulos do kernel.
- /mnt: Este diretório foi previsto para que o administrador possa montar temporariamente sistemas de arquivos quando for necessário.
- /proc: é utilizado para manipular informações de processos do sistema. Mantém informações sobre os sistemas de arquivos suportados, partições de disco disponíveis, portas de entrada e saída (I/O), interrupções (IRQ) e diversas outras informações.
- /root: utilizado como pasta padrão para o superusuário.
- /sbin: tipicamente contém arquivos essenciais para carga e manutenção do sistema.
- /tmp: Arquivos temporários gerados por alguns aplicativos. A permanência da informação que é armazenada em /tmp é limpada em cada carga do sistema.
- /usr: Os arquivos deste diretório pertencentes aos usuários.
- /var: Contém arquivos com informação variável.

Sistema de Arquivos e Dispositivos

Linux tem suporte a inúmeros dispositivos como discos IDE, Sata, SCSI, CD/DVD, dentre inúmeros outros. Cada um destes dispositivos podem ser formatados para o sistema de arquivos Ext2, padrão no Linux.

Cada disco pode ter de uma a dezesseis partições. As partições funcionam como um contêiner para o sistema de arquivos. No Linux cada partição é representada por um número inteiro. Ex.: /dev/hda1. Existem quatro tipos de partições possíveis:

- ***Partições primárias:*** Cada disco pode conter no máximo quatro partições primárias. As partições primárias podem ser nomeadas como: /dev/hda1, /dev/hda2, … /dev/hda4. Uma destas partições primárias deve ser marcada como ativa para que a carga do SO (boot) seja possível.
- ***Partições estendidas:*** São uma variação das partições primárias, contudo **não** podem conter um sistema de arquivos. Elas funcionam como um contêiner para as partições lógicas. Um disco somente pode ter uma partição estendida e que toma lugar de uma partição primária.
- ***Partições lógicas:*** Existem um conjunto com uma partição estendida e pode-se ter de uma a doze partições deste tipo.
- ***Partição de Swap (Arquivo de troca)***: Esta possibilita que o Linux tenha uma memória virtual em disco. Este tipo de memória é usada como arquivo de troca de dados entre a memória física e o disco.

Montando e desmontando Sistema de Arquivos

O sistema de arquivos do Linux é hierárquico e admite que diversos dispositivos sejam mapeados e utilizados a partir da raiz do sistema (root). Estes diretórios que servirão como hospedeiros para os dispositivos são chamados de ponto de montagem.

Qualquer diretórios pode servir como ponto de montagem para outros dispositivos. Se os diretórios não estiverem vazios, seu conteúdo não ficará disponível enquanto o dispositivo não for desmontado.

/etc/fstab: armazena a configuração de quais dispositivos devem ser montados e qual o ponto de montagem de cada uma na carga do sistema operacional (veja o cardápio *de estudos sobre arquivos de configuração*).

Trabalhando com permissões

Como o Linux é um SO multiusuário, as permissões de acesso a arquivos, diretórios e outros dispositivos são necessárias para garantir que os usuários tenham acesso somente aos recursos que eles podem utilizar. Segue a estrutura das permissões:

	Usuário			**Grupo**			**Outros**		
d	**r**	**w**	**x**	**r**	**w**	**x**	**r** *(Leitura)*	**w** *(Gravação)*	**x** *(Execução)*

- *Permissões de usuário:* Definem a permissão para o usuário que é o "dono" do arquivo, quem o criou e o mantém.
- *Permissões de grupo:* Definem a permissão para o grupo de usuários ao qual ele pertence.
- *Permissões para outros usuários:* Definem a permissão para todos os outros usuários (não dono e de outros grupos).
- O primeiro campo pode conter uma das seguintes definições:
    - d: diretório
    - -: arquivo
    - l: link
    - b: dispositivo de bloco
    - c: dispositivo de caractere

As definições de leitura, escrita e execução têm nuances diferentes se estamos trabalhando com arquivos ou diretórios. Estas restrições podem ser descritas na forma binária, onde nota-se outras opções para o primeiro campo, veja:

SUID	SGID	Sticky	leitura	gravação	execução	leitura	gravação	execução	leitura	gravação	execução
0	0	0	0	0	0	0	0	0	0	0	0

Os três primeiros bits da esquerda para a direita são bits de atributos especiais, a seguir:

- *SUID (Set User ID):* O bit SUID afeta somente os arquivos executáveis. Normalmente os programas são executados com a permissão do usuário que os executou. O SUID muda esta condição, fazendo com que o programa seja executado sob as permissões do usuário dono do arquivo, não importando quem o chamou. O SUID geralmente é utilizado para dar ao programa permissões de *root*.
- *SGID (Set Group ID)*: Executam sob a permissão de grupo do dono do arquivo.
- *Stick (Colando na memória)*: O bit especial chamado de sticky ou bit de colado na memória faz com que os programas permaneçam na memória mesmo depois de terminados. Este bit quando aplicado a diretórios faz com que somente o dono do diretório, o dono do arquivo ou o *root* possam renomear ou apagar arquivos neste diretório.

Como os bits de permissão especiais são utilizados com pouca frequência, e sob condições especiais, eles são representados pela letra (s) no lugar do (x) de execução para os bits SUID e SGID nas classes de dono do arquivo e grupo, e (t) no lugar de (x) para o bit de Sticky na classe de outros.