Introdução ao Sistema Operacional

LINUX

Guia Básico para Iniciantes

Oliver Tompson Lessa

ÍNDICE

Introdução

O software livre vem ganhando espaço nos últimos anos dentro dos mais diversos setores. Seja por razões econômicas, técnicas ou até mesmo políticas, adotar medidas que levam a soluções relacionadas com o software livre tem se tornado uma opção para muitos. Vemos, inclusive no Brasil, iniciativas de organizações e algumas partes do setor público fazendo esta opção e alcançando seus objetivos com bastante sucesso. Diante disto, não corremos nenhum risco em afirmar: Software livre tem qualidade!

Neste cenário é impossível não destacar o S.O. Linux. Alguns acreditam que o surgimento do Linux e sua trajetória até aqui tenham sido o acontecimento mais importante dentro da estória do software livre. Sem dúvida seu crescimento e sua recente popularidade são impressionantes. Sem dúvida o Linux é uma opção sólida. Acredito que estas e outras razões, inclusive de natureza técnica, sejam motivações suficientes para estudarmos um pouco deste S.O., mesmo que de uma forma introdutória.

Embora este documento esteja voltado para pessoas familiarizadas com algum outro tipo de sistema operacional, ele não tem a pretensão de ser um guia altamente técnico ou que aborde com profundidade conceitos relacionados a hardware ou até mesmo o tema S.O. Não é, também, nenhuma tentativa de advogar em defesa do Linux com relação a outros sistemas operacionais ou fazer qualquer tipo de confrontação ou classificação.

O objetivo de deste documento é apresentar uma introdução ao S.O. Linux abordando algumas de suas principais características, sua operação básica e outras informações que possibilitem um entendimento razoável sobre seu funcionamento de forma a servir de base para um estudo mais detalhado no futuro.

Um pouco de história

O criador do Linux, hoje já bastante conhecido, chama-se Linus Torvalds. Linus iniciou seu trabalho com o Linux enquanto ainda era estudante (graduando) de Ciência da Computação na Universidade de Helsinki, Finlândia. Naquela época, o departamento de Ciência da Computação da universidade usava uma versão do Sistema Operacional UNIX chamada Minix - Criada por Andrew Tanenbaum.

O UNIX é um dos sistemas operacionais mais populares em todo o mundo e que conta com uma imensa base de suporte além de ter sido largamente distribuído ao longo dos anos. O UNIX foi criado pela AT&T entre o final da década de 60 e o início dos anos 70 para ser um sistema multitarefa de microcomputadores e mainframes, tendo já sido classificado por alguns de: “O único sistema operacional de verdade”.

Insatisfeitos com o S.O., Linus e outros usuários resolveram enviar pedidos de modificações e melhoramentos ao autor do Minix, que considerou estas alterações como não sendo necessárias. A partir destas respostas, Linus resolveu desenvolver seu próprio sistema operacional. Sua idéia inicial partiu do princípio de criar um "Minix" melhorado ou, em suas próprias palavras: “... a better Minix than Minix”.(um Minix melhor que o Minix).

Depois de algum tempo trabalhando no seu projeto, em 5 Outrubro de 1991, Linus anuncia o que seria a primeira versão do seu *kernel*: o Linux 0.02. Neste ponto ele já era capaz de rodar uma versão do *bash* (the GNU Bourne Again Shell), do *gcc* (the GNU C compiler) e algumas outras coisas, mas não muitas.

Linus, então, anunciou seu trabalho deixando-o aberto para todos os que desejassem contribuir com o desenvolvimento do *kernel*. A resposta veio rápido e muitas pessoas se juntaram a ele estabelecendo um processo de evolução surpreendente. Deste momento até Dezembro de 1993, o Linux chegou em sua versão de *kernel* 0.99.pl14, alcançando praticamente o momento de sua primeira versão “oficial” pronta para ser distribuída (versão 1.0). Mensagem enviada por Linux Torvalds a **comp.os.minix:**

> *"Você suspira por melhores dias do Minix-1.1, quando homens serão homens e escreverão seus próprios "device drivers" ? Você está sem um bom projeto e esta morrendo por colocar as mãos em um S.O. no qual você possa modificar de acordo com suas necessidades? Você está achando frustrante quando tudo trabalha em Minix? Chega de atravessar noites para obter programas que trabalhem correto? Então esta mensagem pode ser exatamente para você.*
>
> *Como eu mencionei a um mês atrás, estou trabalhando em uma versão independente de um S.O. similar ao Minix para computadores AT-386. Ele está, finalmente, próximo do estágio em que poderá ser utilizado (embora possa não ser o que você esteja esperando), e eu estou disposto a colocar os fontes para ampla distribuição. Ele está na versão 0.02... Contudo eu tive sucesso rodando bash, gcc, gnu-make, gnu-sed, compressão, etc. nele.*"

Durante muito tempo, Linus Torvalds foi o mantenedor direto do *kernel* do Linux. Hoje esta responsabiliade está sobre os ombros de outros e o desenvolvimento do sistema como um todo countinua sendo feito com a colaboração de muitas pessoas ao redor do mundo. Entretando, Linus ainda é quem dá a palavra final sobre o que vai ou não entrar no *kernel* do Linux e quando uma versão está pronta para ser anunciada como estável e distribuida.

> *Após sua saída da universidade, Linus mudou-se para os EUA e passou a trabalhar em uma empresa chamada Transmeta. Seu trabalho foi o de ajudar no desenvolvimento de uma versão de kernel do Linux que pudesse ser alojada em uma fhash ROM. Isto foi usado no desenvolvimento do "Crusoé", um chip voltado para equipamentos portáteis e usando em Note Books.*

O Linux hoje

Com o passar dos tempos muitas aplicações foram portadas ou desenvolvidas para rodar no Linux, tanto livres quanto comerciais. Seguindo este crescimento, existe hoje também um enorme suporte a hardware e periféricos, inclusive comparando com os outros sistemas operacionais.

Hoje o Linux entrou no mercado do desktop. Apesar da inegável dominação da Microsoft nesta área, o Linux tem se mostrado e se estabelecido como uma opção aceitável e competitiva. Isto é notado pelo desenvolvimento de interfaces gráficas cada vez mais amigáveis e outras aplicações que incluem as compatíveis com os produtos Microsoft como: processadores de texto, planilhas, programas de apresentação ou até mesmo pelo fato de ser possível rodar estas próprias aplicações dentro do Linux.

No campo dos servidores o Linux inegavelmente se estabeleceu como uma referência no que diz respeito a ser uma plataforma estável e de bom desempenho. Prova disto é o fato de organizações como Amazon, Exército Alemão e Correios dos EUA utilizarem o Linux para rodar servidores de banco de dados. Provedores de acesso e conteúdo da Internet também adotam largamente o Linux seja como Web servers, firewalls ou em serviços como o Google. Existem ainda outras áreas interessantes aonde o Linux vem ganhando espaço: Clusters de máquinas rodando Linux foram usados para a criação de filmes como “Titanic” e “Shrek”, por exemplo.

Há espaço também no mundo dos portáteis e dos appliances. Alguns PDA´s e muitos equipamentos como pequenos servidores, roteadores e outros dispositivos desta natureza

já têm o Linux portado e rodando como seu sistema operacional principal.

Mas, o que dizer a respeito do ponto de vista do usuário? Trabalhar com o Linux é difícil?

No passado, ser um expert ou um usuário experiente era uma espécie de pre-requisito para usar o Linux. Instalar e configurar o sistema realmente exigia um bom nível de conhecimento tanto do sistema quanto da máquina na qual ele seria instalado. Outro fator que poderia “assustar” os iniciantes interessados era a postura de alguns dos tais experts no assunto que frequentemente respondiam questionamentos com alguma coisa como: “RTFM” (leia os manuais). Sem dúvida um grande número de manuais e documentação em geral sempre estiveram presentes em todos os sistemas. Entretando, a forma muito técnica como eles eram escritos poderia desanimar alguns inicantes.

Esta não é a realidade de hoje. A evolução da interface com o usuário e as ferramentas adicionadas tornaram esta convivência muito mais amigável e atraente para um usuário qualquer. Empresas como a RedHat, SuSE, Mandrake e outras, vem se esforçando em tornar suas respectivas distribuições produtos que possam ser distribuídos facilmente em massa, a exemplo de outros sistemas operacionais.

O Linux e a Iniciativa GNU/Open Source

A idéia de buscar a colaboração dos desenvolvedores e usuários de software, por meio de comentários, sugestões e, principalmente, pelo trabalho direto que foi aplicada por Linus Torvalds no Linux, não é recente e nem uma caraterística exclusiva da trajetória do Linux. Richard Stallman, que trabalhou por algum tempo em um laboraboratório de Inteligência Artifical no Massachusetts Institute of Technology(MIT), é um defensor desta filosofia desde os anos 70. Ele é um dos pioneiros no conceito de "free software" (software livre).

Ao sair do MIT, Richard Stallman, fundou o projeto GNU, que tem como objetivo o desenvolvimento e manutençao de "software livre". A palavra "livre", nesta expressão, não quer dizer necessariamente "grátis". Alguns poderiam acreditar que "domínio público" ou "liberado" seriam expressões que classificariam bem um software. Entretando, ainda poderia haver certa confusão com "grátis" e com o conceito de propriedade. Um programa é um "software livre", para você, um usuário particular, se:

- você tem a liberdade de executar este programa, para qualquer propósito;
- você tem a liberdade de modifica-lo para atender as suas necessidades (para que isto seja uma realidade, você precisa ter acesso ao código fonte do programa);
- você tem a liberdade de redistribuir cópias, gratuitamente ou não;
- você tem a liberdade de distribuir versões modificadas do programa. Desta forma, outras pessoas teriam acesso aos melhoramentos feitos no programa por você.

Além das citadas nos ítens acima, que resumem os termos de uma licença GNU, uma outra idéia interessante relacionada como o projeto GNU é a de “Copyleft”. Esta idéia tem como objetivo prevenir que algum "software livre" venha a se tornar, por qualquer razão, propriedade de alguém ou alguma organização.

Uma vez que "livre" não se refere a preço, não existe contradição entre vender cópias de programas "open source" e o conceito de "software livre". Nestes casos o que ocorre **não é a venda do software**, mas sim do trabalho de "empacotar" este software em algum tipo de mídia e de ferramentas de auxílio à instalação e configuaração desenvolvidas pelo prórpio distribuidor com seus respectivos manuais. Na realidade, esta forma de distribuição tem sido de fundamental importância para a proliferação do Linux pelo mundo.

Para entender melhor a relação do Linux com o Porjeto GNU, vamos voltar um pouco no tempo: Por volta de 1991, já existiam condições para a criação do Linux(SO). Entretanto, Linus Torvalds havia apenas desenvolvido o *kernel* e testado com alguns programas. Ele não tinha todo o sistema operacional criado(e não tinha condições para tal), como ele mesmo disse: "...sozinho, um kernel não leva você a lugar algum. Para ter um Sistema Operacional funcional você precisa de um *shell*, compiladores, bibliotecas e etc".

Do outro lado o projeto GNU tinha a intensão de crirar um sistema operacional e já havia desenvolvido várias aplicações e ferramentas para o S.O. Mas, não tinha um *kernel* - o coração do sistema operacional. Nas palavras de Richard Stallman: "... o *GNU Hurd(*seu projeto de SO), não está pronto para utilização. Felizmente, outro kernel está disponível. Em 1991, Linus Torvalds desenvolveu um kernel UNIX compatível que chamou de Linux".

Deste ponto veio a combinação dos programas necessários fornecidos pelo projeto GNU e o *kernel* desenvolvido por Linus Torvalds. Isto foi o nascimento do SO que chamamos de Linux. Podemos ver, portanto, que o sistema opercional chamdado Linux, na realidade é a união de dois esforços: os softwares do projeto GNU e o *kernel* cujo nome é Linux. Baseado nisto, vemos frequentemente a referência ao SO como Linux/GNU ou GNU/Linux. Não podemos separar ou dar maior importância a uma das partes. Sua integração e desenvolvimento conjunto é o que forma um sistema operacional robusto e estável.

Alguns Prós e Contras do Linux

Prós

- É “Open Source”: O Linux é distribuido sob a lincença GNU. Isto significa que você não precisa necessariamente pagar para tê-lo; nem mesmo comprar uma distribuição. O Linux, em praticamente todas a suas distribuições, está inteiramente disponível para download na Internet. Não há taxas para registro, por número de cópias nem mesmo para atualizações ou pelo código fonte.

- É portável: Uma vez que qualquer pessoa pode ter acesso ao código fonte do sistema e modifica-lo, isto permite que se façam adaptações para praticamente qualquer plataforma de hardware.

- É escalável: Tornou-se comum vermos o Linux rodando desde um Palmtop até clusters com cententas de nodos ou mesmo em um relógio de pulso (produto desenvolvido pela IBM). Seja qual for o projeto: reciclar velhos micros 386 ou 486 até o uso em sistemas embarcados (embutidos – como alguns preferem), o Linux pode ser usado com bom rendimento e sucesso.

- Oferece versatilidade e um bom grau de estabilidade e segurança: Estas são características herdadas do UNIX. O que se espera naturalmente de sistemas como estes é que eles se mantenham em funcionamento durante todo o tempo sem a necessidade de um "reboot" e que seja possível que várias tarefas possam ser executadas durante as 24 horas de um dia.

Contras

- Existe um número muito grande de distribuições: Apesar de a variedade e a "briga" dos distribuidores por espaço parecerem, a primeira vista, um ponto positivo, isto tem um lado ruim. Cada distribuidor adota um caminho: uma forma de "empacotar" o Linux e escolhe que aplicações vão incluir em sua distribuição. Desta forma não temos uma convergência ou uma inicitiva de padronização em qualquer aspecto. Isto leva o usuário iniciante a uma certa confusão sobre o S.O. como: Existem diferentes tipos de Linux? Qual a melhor distribuição? Que escolha fazer?

- Certos processos para instalção e configuração não são amigáveis para o iniciante: Ter uma informação prévia do S.O. e conhecer a máquina na qual ele será instadado ainda é um pré-requisto. Entretanto, muitos esforços têm sido empregados para fazer com que estas informações estejam cada vez mais facilmente ao alcançe dos usuários iniciantes.

- Ainda existe resitência por parte do mercado: O Linux ainda não é aceito sem reservas em muitos setores e pela grande maioria dos usuários. As grandes empresas que dominam o mercado e ainda formam a opinião da maioria dos uauários, tem buscado defender seu espaço da "ameça" Linux com algum sucesso. Entre os argumentos estão: colocar em questão a qualidade do software Open Source, dizer que não há suporte adequado e suficiente, que se houver suporte este terá maior custo do que comparando com software comercial e assim por diante.

Sobre este assunto, o desenvolvedor do Enlightment (um gerenciador de janelas) – que se entitula: The Rasterman – fez a seguinte declaração: ***"Não nos desktops, não nos PCs. Em nada que lembre o que você chama de desktop. O Windows venceu conforme-se. O mercado não é governado por um Kernel superior ou por um sistema que não trava. Os usuários não se importam, eles simplesmente reiniciam e continuam com ele. Eles querem aplicativos e se os aplicativos que eles querem e gostam não estão aqui, então é perda de tempo."***

O Linux e as Distribuições

Como já foi citado anteriormente o Linux tem certas características de estrutura e construção que permitem a qualquer pessoa personalizar uma instalação e/ou configuração em qualquer plataforma. Isto motivou algumas pessoas, grupos e empresas que passaram a distribuir o Linux pela Internet e em CDs vendendo, não o S.O. mas, o trabalho de gravar em CD, uma vasta documentação e muitas ferramentas de auxilio a instalação e configuração do S.O. e das aplicações que o acompanham. Tudo isto com a intensão de "livrar" o usuário das etapas mais complicadas dos processos de instalação e configuração.

Entres essas distribuições surgiram, naturalmente, diferenças que vão desde a quantidade e tipos de softwares que acompanham o S.O., manuais de instalação, pequenas diferenças de organização, locais de instalação dos softwares e até outros aspectos que caracterizam o distribuidor como logotipos e ícones na inicialização do sistema.

Aqui, vale destacar de um detalhe em que, com muita frequência, ocorre algum tipo de confusão: **Uma distribuição não é o mesmo que uma versão do S.O. E, mais especificamente: uma versão de uma distribuição também não é o mesmo que uma versão do S.O.** Assim, em uma mesma plataforma de hardware, peguntas do tipo: "Existem diferentes tipos de Linux?" ou "Qual é o Linux que você usa?", não estariam corretas ou corretamente contextualizadas.

Uma distribuição nada mais é do que uma forma de "empacotar" o Linux. Ou seja, a distribuição contém o S.O. seja qual for a versão em que ele esteja. Nada mais natural do que, com o passar do tempo, as distribuições sofrerem suas alteraçaões e melhorias. A isto, os distribuidores chamam versão. Entretando, no Linux, determinamos a versão observando o *kernel* e as ferramenas essenciais do sistema; em qual versão está o *kernel*.

Esta separação pode parecer um pouco difícil de perceber mas é importante. Se alguém escolhe uma distribuição, com a qual não se sinta confortável, para alguma aplicação e não atinge seus objetivos, sem esta consciência, fatalmente irá "culpar" o Linux por seu fracasso e não a sua escolha inadequada de distribuição.

Então, qual a melhor distribuição? Na teoria e na prática tudo o que podemos fazer com uma distribuição, poderemos fazer com outras. Tentar classificar uma distribuição como melhor ou pior do que outras não é um bom caminho. Sendo assim, cabeira uma outra pergunta: Qual distribuição escolher? Uma das respostas mais simples e diretas para esta questão é: depende do usuário.

Exitem distribuições que poderíamos incluir no grupo das mais recomendadas para usuários iniciantes que são aquelas que, por padrão, já ativam em seu processo de instalação o suporte aos mais variados tipos de hardware(incluindo aqueles que raramente os usuários dispõem) bem como a instalação e pré configuração de um grande número de aplicações e ferramentas. Exemplos de distribuições deste grupo:

- Fedora Linux
- SuSE Linux
- Mandrake
- Knoppix
- Red Hat
- Conectiva (Brasil - baseada na **Red Hat**)
- Kurumin (Brasil - baseada na **Knoppix**)

Por outro lado, alguns usuários mais experiêntes e/ou administradortes de sistema podem não gostar do fato de não ter total controle sobre todo o processo de instalação do sistema e da posterior configuração de tudo. Para estes, também existem distribuições que se adequam bem, por exemplo:

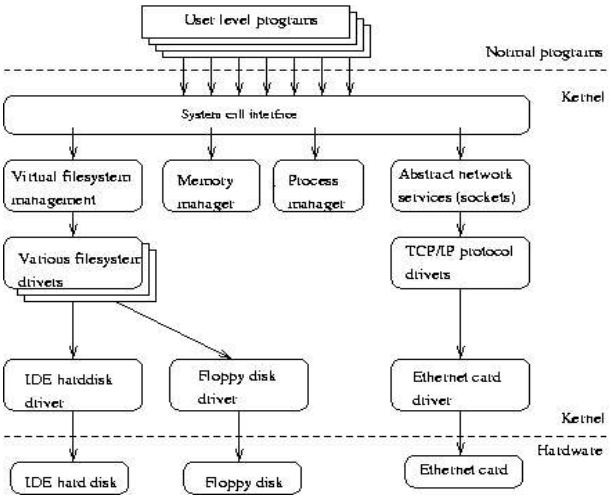
- Slackware (alguns consideram um pesadelo para qualquer iniciante)
- Debian

Kernel

Vamos agora falar, rapidamente e sem aprofundamento teórico, sobre a parte essencial em qualquer sistema operacional: o *kernel*. Ele é o núcleo do S.O.; seu cérebro. A grosso modo, é ele quem "diz" diretamente ao computador o que fazer, quando e como.

O gerenciamento de memória, de processos, de entrada e saída são, em linhas gerais as funçoes básicadas do *kernel*. Em alguns sistemas podemos ter também , dentro do *kernel*, controladores de dispositivos(drivers). No Linux, esta prática é comum. Logo nas primeiras versões, todos os controladores de dispositivos faziam parte do kernel. Com o seu desenvolvimento, no que se refere a suporte de dispositivos e periféricos, o *kernel* cresceu bastante em tamanho. Apesar da possibilidade de compilarmos o *kernel* com apenas os controladores para os dispositivos presentes em nosso equipamento, o tamanho ainda

poderia representar algum problema. A figura a seguir ilustra as funções básicas do *kernel* no Linux:



Vieram os módulos. Os módulos, no Linux, são controladores(drivers) e algumas outras partes do *kernel* como suporte a outros sistemas de arquivos, preparados para serem usados sob demanda e que não estão inseridos no kernel que é carregado logo após o processo de *boot*.

Normalmente, compilamos o *kernel* do Linux da seguinte maneira: todos os componentes vitais, incluindo controladores de E/S e rede com seus protocolos, incluimos no kernel; os demais, como controladores de dispositivos e periféricos pouco utilizados, compilamos como módulos para serem carregados sob demanda ou logo após o *kernel* assumir o controle da máquina.

Usuários e Contas de Usuários

Hoje já é comum encontrarmos em quase todos os sistemas a prática de identificar os usuários: solicitar nome de usuário e senha para autorizar o uso do sistema. No Linux, esta característica sempre existiu por ser um sistema derivado do UNIX. Entretando, esta autenticação tem uma função muito maior do que a mera personalização de configurações visuais.

O Linux é um sistema multi-usuário e multitarefa real. Isto significa que vários usuários podem estar executando muitas aplicações ao mesmo tempo em uma mesma máquina. Isto não ocorre em outros sistemas voltados para desktop como algumas versões do Windows, por exemplo.

No Linux, ao fornecer seu nome de usuário e senha ao sistema você estará efetuando um processo chamado de *log in* em sua *conta de usuário*. Uma *conta* não apenas identifica um usuário mas também representa todos os arquivos, recursos e quaisquer outras

informações pertencentes a um usuário, por exemplo: espaço em disco que pode ser usado, poder ou não imprimir em uma determinada impressora, cotas de impressão, quantidade dos recursos da CPU que podem ser usados, poder de gravar em diretórios, prioridades e etc.

Um exmplo da tela de *log in* em modo texto:

```
Welcome to Linux 2.4.22 (tty1)

darkstar login: _
```

Ao criarmos uma *conta de usuário* no sistema devemos fornecer algumas informações para indentifica-lo. Cada usuário tem um número, único, que o identifica no sistema. As informações fornecidas são:

- *username* (nome de usuário ou login)
- *password* (senha – é armazenada de forma criptografada)
- *user id* ou *uid* (número único de cada usuário)
- *group* id ou *gid* (número do grupo ao qual ele pertençe)
- *fullname* (nome completo e outras informações)
- *home directory* (diretório pessoal do usuário)
- *login shell* (o programa que será excutado ao efetuar-se o *log in*)

Na lista com os ítens acima, aparece um outro conceito interessante: os *grupos de usuários*. No Linux, todos os usuários cadastrados são obrigatoriamente classificados em grupos. Isto, amplia bastante as possibilidades de organização e segurança do sistema. Os grupos, assim como os usuários, tem seus direitos dentro do sistema sobre arquivos, diretórios, recursos e etc. Isto nos permite combinar os direitos(nívies de segurança) de usuarios e grupos, facilitando a administração do sistema e garantindo a integridade desta organização.

Os usuários herdam os direitos do grupo aonde estão incluídos, sobre os recursos do sistema dos os quais possuam o domínio. Por exemplo: um usuário, isoladamente pode ter somente o direito de leitura sobre um determinado arquivo. Entretando, se este mesmo usuário estiver cadastrado em um grupo que possua o direito de gravar neste arquivo, ele automaticamente herdará este direito.

root – um usuário especial

O *root* é um usuário muito especial do sistema, com "poderes" especiais. Na realidade, o *root* tem todos os poderes possíveis dentro do sistema; ele é o administrador, o "super usuário". As configurações do sistema, quase que em sua totalidade, só podem ser feitas por este usuário. Inclusão e exclusão de usuários e de grupos, configuração de dispositivos, gerência de permissões para usar os recusros do sistema, desligar o sistema e etc; tudo isso é tarefa e privilégio do *root*. É o usuário que pode ler e gravar em qualquer parte do sistema de arquivos. O *root* é quem tem todas as "chaves" e vê tudo.

Com todos estes "poderes" é extremamente necessário muita cautela ao usarmos o sistema como *root*. Pressupõe-se que o administrador do sistema tem plena consciência de todos os seus atos. Dizemos isto porque o *root*, normalmente, não é advertido ou impedido pelo sistema(em condições normais) ao executar comandos e configurações. Quando ao apagar um arquivo ou um diretório inteiro, por exemplo, o sistema não faz ao *root* perguntas do tipo: "... tem certeza?".

Como vimos anteriormente, o Linux sendo um sistema realmente multi-usuário, nos permite ter vários usuários cadastrados e habilitados a usa-lo ao mesmo tempo. Todos estes estão abaixo do *root* na hierarquia de poderes dentro do sistema e, por esta razão, são advertidos ou simplesmente impedidos de executar tarefas classificadas como de configuração ou danosas ao sistema. Portando, é saudável e recomendável em nosso trabalho do dia-a-dia, usarmos o sistema como um usuário comum, mesmo tendo privilégios maiores, e reservar o uso do *root* para a administração e configuração do sistema – o que ocorre na menor parte do tempo.

Arquivos e o Sistema de Arquivos (f*ilesystems*)

Ao migrar de outros sistemas como o Windows para o Linux ou algum outro sistema baseado em UNIX, uma das maiores diferenças percebidas pelo usuário será no que diz respeito ao sistema de arquivos e sua organização.

Um sistema de arquivos ou *filesystem* é um conjunto de métodos e estruturas de dados que são usados pelo sistema para manter os arquivos em um disco ou partição. Ou seja, é o que define a estrutura e forma como as informações serão armazenas e os meios de recuperação delas. Basicamente, temos a seguinte situação: armazenamos informações em arquivos, arquivos em diretórios e estes formam uma estrutura organizada.

É o sistema de arquivos o responsável por tornar isto possível. Além disto, saber onde se encontra uma informação e como recupera-la também é seu papel. A expressão *filesystem*

pode ser usada para fazer referência a uma partição ou a um disco que contém arquivos, como para um tipo de partição.

Falando em tipos de *filesystem*, podemos destacar um outra característica encotrada no Linux que é o suporte a vários tipos de *filesystem*. Isto permite que, usando o Linux, possamos ler e gravar em uma partição com outro sistema instalado como o Windows, por exemplo. Alguns tipos de *filesystem* suportados pelo Linux:

- minix
- xia
- ext3
- ext2
- ext
- reiserfs
- msdos
- umsdos
- vfat
- iso9660
- nfs
- smbfs
- hpfs
- sysv

Uma outra diferença marcante entre o Linux – ou sistemas derivados do UNIX, de modo geral – e outros sistemas operacionais é com relação ao tratamento dado a arquivos. Uma descrição bastante usual empregada em sistemas UNIX relacionada a arquivos é:

"Em um sistema UNIX, tudo é um arquivo; se alguma coisa não é arquivo, então é um processo."

No Linux, isto também se aplica pelo fato de se ter: arquivos e arquivos especiais. O Linux não faz diferença entre diretórios e arquivos, uma vez que considera os diretórios como sendo apenas arquivos que contém os nomes de outros arquivos. Dispositivos de entrada e saída e, em geral, todos os outros dispositivos e periféricos também são considerados arquivos.

Uma vez que tudo é arquivo, o Linux tem um sistema de classificação de **tipos de arquivos**. Todos estes tipos de arquivos são organizados em uma estrutura tipo ávore, também chamada de hierarquia do sistema de arquivos, que veremos um pouco mais adiante. Por hora vamos aos tipos de arquivos:

- Regulares: são a maioria dos arquivos. São os que contém dados normais como arquivos de texto, imagens, arquivos executáveis, programas, arquivos que contém entrada ou saída de algum programa e assim por diante.

- Diretórios: aqueles que contém listas de outros arquivos (também conhecidos como "pastas").

- Arquivos especiais: são o mecanismo utilizado para entrada e saída; arquivos ligados aos dispositivos, usandos para acesso direto a eles ou como interface. A

maior parte dos arquivos especiais se encontram no diretório `/dev` que veremos mais tarde.

- Links: é um meio de fazer com que um arquivo esteja visível em várias partes do sistema de arquivos. Também nos referenciamos a eles como atalhos ou simplesmeste ligações.

- (Domain) sockets: é um tipo de arquivo especial similar ao *socket* do TCP/IP. Ele é usado para proporcionar uma comunicação inter-processos pela rede, protegida pelo controle de acesso do sistema de arquivos.

- Named pipes: é também uma forma de comunicação inter-processos sem usar a mesma semântica dos *sockets* na rede.

No prompt de comando, quando solicitamos a listagem dos arquivos do diretório atual, podemos verificar a indentificação dos tipos de arquivos no primeiro caractere da coluna dos atruibutos à esquerda:

```
oliver@darkside:~/sources/palm$ ls -l
total 9749
drwxr-xr-x    2 oliver   users         168 2004-02-15 15:49 acalc-0.21/
-rw-------    1 oliver   users       28257 2004-02-14 17:56 acalc021.zip
-rw-rw-r--    1 oliver   users        7633 2000-11-30 21:27 BinaryBuddy.prc
-rw-------    1 oliver   users        6543 2004-02-14 17:58 BinaryBuddy.zip
```

A tabela a seguir resume a forma de identificação:

Caractere	***Tipo de arquivo***
-	Regular
d	Diretório
l	Link
c	Arquivo especial
s	Socket
p	Named pipe

Direitos de Acesso (permissões)

No Linux, todo arquivo é propriedade de um usuário e de um grupo de usuários. Para cada categoria de usuários podem ser dadas as permissões para ler, escrever e executar arquivos podem ser dadas ou retiradas. Pode existir também uma categoria de usuários que não tem propriedade sobre nenhum arquivo e que não pertence a nenhum grupo. Alguns afirmam que está é a primeira linha de defesa do Linux.

No exemplo anterior, quando falavamos sobre os tipos de arquivos, vimos a utilização do comando `ls -l` para listar o conteúdo do diretório. Vimos também que a identificação do tipo se localiza no mesmo bloco de informações onde encontramos as especificações sobre os direitos de acesso(permissões) sobre cada programa. Após o primeiro caractere, temos a sequência de caracteres que identificam os atributos.

```
oliver@darkside:~/sources/palm$ ls -l
total 9749
drwxr-xr-x    2 oliver   users         168 2004-02-15 15:49 acalc-0.21/
-rw-------    1 oliver   users       28257 2004-02-14 17:56 acalc021.zip
-rw-rw-r--    1 oliver   users        7633 2000-11-30 21:27 BinaryBuddy.prc
-rw-------    1 oliver   users        6543 2004-02-14 17:58 BinaryBuddy.zip
```

Observando novamente a listagem acima, da direita para a esquerda, temos: o nome do arquivo, a data e hora da criação ou modificação, o tamanho do arquivo(em bytes), o grupo dono do arquivo, o usuário dono do arquivo e os atributos(permissões).

Na parte dos atributos(permissões), logo após o primeiro caractere que indentifica o tipo de arquivo, temos(olhando da esquerda para a direita):

- Os três primeiros caracteres indicam as permissões para o usuário dono do arquivo.
- Os três caracteres seguintes indicam as permissões para o grupo dono do arquivo.
- A última sequência de três caracteres indicam as permissões para outros usuários.

Os códigos para cada permissão são:

Código	*Significado*
`-` ou `0`	A permissão desta posição não foi definida
`r` ou `4`	Permissão de leitura
`w` ou `2`	Permissão de gravação
`x` ou `1`	Permissão de execução/busca

Recorrendo mais uma vez ao exemplo anterior, vamos verificar somente os atributos do primeiro arquivo(que é um diretório):

```
oliver@darkside:~/sources/palm$ ls -l
total 9749
drwxr-xr-x    2 oliver   users         168 2004-02-15 15:49 acalc-0.21/
```

De acordo com as definições que vimos anteriormente temos: `drwxr-xr-x`

`d`	O arquivo é do tipo diretório
`rwx`	O dono do arquivo tem permissões de *leitura, escrita* e *execução*(busca)
`r-x`	O grupo do qual o usuário faz parte tem permissões de *leitura* e *execução*(busca)
`r-x`	Outros usuários têm permissões de *leitura* e *execução*(busca)

Sistema de Arquivos – Hierarquia de Diretórios

A estrutura hieráquica de diretórios no Linux também é algo diferente se compararmos com um sistema como o Windows, por exemplo. Assim como no UNIX, o Linux optou por ter uma única e simples estrutura de diretórios que começa a partir do `root`(raiz), representado por `/`, que se expande em sub-diretórios.

No Windows poderíamos colocar arquivos em qualquer lugar: drive C, drive D e assim por diante, uma vez que a estrutura de diretórios é controlada pelos próprios programas ou usuários e não pelo sistema operacional. No caso do Linux, os diretórios são organizados de forma descendente a partir do `root(/)` de acordo com sua importância no processo de *boot*.

Se você achou estranho o uso da barra / (inclidada para adireita) e não da barra \ (inclidada para a esquerda), a razão para isto é simples: o Linux segue a tradição do UNIX. Da mesma forma, faz diferença entre letras maiúsculas e minúsculas na formação dos nomes de arquivos – em qualquer parte do nome. Por exemplo: `teste.tar.gz` é diferente de `Teste.tar.gz`. Ou seja, são arquivos diferentes para o sistema.

Outro detalhe desta hierarquia é que sua ordem – o local aonde os arquivos serão colocados – está relacionada com a função do arquivo e não com o contexto do programa.

Listando o conteúdo do diretório `root` ou `/` (raiz) temos:

```
oliver@darkside:/$ ls -l
total 107
drwxr-xr-x    2 root     root         2416 2004-05-25 21:33 bin/
drwxr-xr-x    2 root     root          304 2004-02-11 13:25 boot/
drwxr-xr-x   16 root     root        62400 2004-10-05 11:19 dev/
drwxr-xr-x   52 root     root         5208 2004-10-05 17:44 etc/
drwxr-xr-x    5 root     root          120 2004-02-14 17:43 home/
drwxr-xr-x    5 root     users        2944 2004-05-29 11:34 lib/
drwxr-xr-x    7 root     root          168 2004-02-25 21:17 mnt/
drwxr-xr-x    9 root     root          264 2004-07-13 14:12 opt/
dr-xr-xr-x   74 root     root            0 2004-10-05 07:19 proc/
drwxr-xr-x   39 root     root         2016 2004-08-27 22:53 root/
drwxr-xr-x    2 root     root         5880 2004-06-10 02:18 sbin/
drwxrwxrwt   25 root     root         7288 2004-10-05 11:20 tmp/
drwxr-xr-x   18 root     root          544 2003-01-06 08:21 usr/
drwxr-xr-x   16 root     root          456 2004-04-30 11:21 var/
```

/bin

Atualmente este é um dos diretórios mais importantes do Linux. Ele contém a maioria dos comandos considerados essenciais e que estão disponíveis para todos os usuários (tanto para o *root* quanto para outros usuários comuns). Se olharmos atentamente a listagem do conteúdo veremos os comandos `ls` e o `bash`, *shell* que estamos utilizando, bem como `cp`, `mv`, `rm`, `cat` e outros. Os demais comandos também tem o mesmo grau de importância no sistema e, por isso, dize-se que o coneúdo deste diretório é essencial. Este diretório também contém programas dos quais os *scripts* de *boot* podem depender.

Listando parcialmente o `/bin`:

```
oliver@darkside:/bin$ ls
arch*           df*             gunzip@     mt@             sleep*
ash*            dialog*         gzip*       mt-GNU*         sln*
awk@            dircolors*      head*       mt-st*          stty*
bash*           dmesg*          hostname*   mv*             su*
bunzip2@        dnsdomainname@  ipmask*     netstat*        sulogin@
bzcat@          domainname@     kill*       nisdomainname@  sync*
bzip2*          du*             killall*    ping*           tar*
bzip2recover*   echo*           ksh*        ping6*          tar-1.13*
cat*            ed*             ln*         ps*             tar-1.13.25@
--More--
```

/boot

Aqui, estão todos os arquivos de configuração úteis e necessários para o precesso de *boot* tais como: cópias do *master boot records*, arquivos do tipo *sector/system map* e, em alguns casos, também o *kernel* que está sendo utilizado. Alterar o conteúdo dos arquivos contidos neste diretório, sem o conhecimento necessário, significará perder o acesso ao sistema.

Listando o `/boot`:

```
oliver@darkside:/boot$ ls
boot.0300  config-ide-2.4.22  System.map@            vmlinuz@
config@    map                System.map-ide-2.4.22  vmlinuz-ide-2.4.22
```

/dev

Listando o conteúdo do diretório `/dev` veremos uma grande quantidade de nomes normalmente mostrados em amarelo. Todos estes, são dispositivos que o sistema usa ou pode usar. Assim, temos um arquivo para cada dispositivo como, por exemplo, para o primeiro disco rígido instalado na máquina: o arquivo `/dev/hda`.

Aqui, voltamos a notar uma característica interessante do Linux, citada anteriormente: tudo no sistema é um arquivo. O primeiro disco rígido(master) instaldado na máquina é o aquivo `/dev/hda`. Logo, se apagarmos este arquivo, perderemos o acesso ao disco rígido. Esta característica nos permite muitas outras coisas interessnates. Se, ao tentarmos visualizar o conteúdo de um arquivo, usarmos um redirecionador disponível no *shell* desviando sua saída para um dispositivo, estaremos realmente "escrevendo" naquele dispositivo.

Exemplo:

```
oliver@darkside:~$ cat musica.au > /dev/audio
```

No exemplo acima, usando o comando `cat`, estamos "listando" o conteúdo do arquivo `musica.au` e, usando o redirecionador > do *shell*, desviamos a saída para o dispositivo `/dev/audio` - a placa de som. Neste caso o resultado será, caso o dispositivo esteja devidamente configurado, ouvirmos o som ou a música. Vale ressaltar que, para realizar esta operação, precisamos das devidas permissões - o que pode não ser dado a usuários comuns(acesso direto a dispositivos).

Se fizermos o mesmo procedimento redirecionando a saída para `/dev/lp0` estaremos imprimindo o arquivo em uma impressora conectada na primeira porta paralela. Desviando para `/dev/ttyS0`, o conteúdo do arquivo será mandado para o dispositivo conectado na primeira porta serial; um modem, por exemplo.

Este tipo de interface, entre os dispositivos e o usuario, dá ao sistema uma flexibilidade enorme, facilitando seu uso de e aumentando as possibilidades de acesso aos dispositivos.

Listagem parcial do `/dev`:

```
oliver@darkside:/dev$ ls -l
total 238
drwxrwxrwx    2 root     root         6192 2002-06-11 14:40 amiraid/
crwxrwxrwx    1 root     sys       14,  11 2003-08-29 03:47 amixer0
crwxrwxrwx    1 root     sys       14,  27 2003-08-29 03:47 amixer1
crwxrwxrwx    1 root     sys       14,  43 2003-08-29 03:47 amixer2
crwxrwxrwx    1 root     sys       14,  59 2003-08-29 03:47 amixer3
crw-------    1 oliver   root      10, 134 1996-06-07 17:25 apm_bios
drwxrwxrwx    2 root     root        1200 2002-05-19 19:41 ataraid/
crwxrwxrwx    1 root     sys       10,   3 1994-07-17 19:49 atibm
lrwxrwxrwx    1 root     root            6 2004-02-09 15:59 audio -> audio0
crw-------    1 oliver   root      14,   4 2003-08-29 03:47 audio0
crw-------    1 oliver   root      14,  20 2003-08-29 03:47 audio1
lrwxrwxrwx    1 root     root          8 2004-02-09 16:24 cdrom -> /dev/hdc
--More--
```

/etc

Aqui, residem todos os arquivos de configuração do sistema. Um arquivo de configuração é definido como um arquivo local que controla a operação de um programa; tem que ser estático e não pode ser um binário executável.

Da mensagem de boas vindas ao iniciarmos o sitema, passando pela configuração de rede, impressoras e outras aplicações úteis até o arquivo de senhas e a configuração do *boot*; todos os arquivos estão aqui. Ou seja, é praticamente o centro nervoso do sistema. Por este motivo, é sempre bom fazer *backups*(cópias) regulares de todo o conteúdo deste diretório. Isto será um fator de grande economia de tempo quando for necessário fazer alguma reconfiguração em casos de reinstalção ou perda de dados.

São arquivos de texto simples. Esta é uma das características do Linux. Suas configurações são feitas por meio de mudanças em arquivos texto que, na maior parte, podem ser lidos por qualquer usuário. Qualquer alteração que fizermos em um destes arquivos, estaremos mudando a configuração do sistema.

Listagem parcial do `/etc`:

```
oliver@darkside:/etc$ ls
a2ps.cfg          gimp/                krb5.conf-sample  netgroup               scrollkeeper.conf
a2ps-site.cfg       gnome-vfs-2.0/       krb5kdc/          networks               securetty
acpi/               gnome-vfs-mime-magic ld.so.cache       nntpserver             security/
adjtime             gnopernicus-1.0/     ld.so.conf        nscd.conf              serial.conf
apache/             gpm-root.conf        lftp.conf         nsswitch.conf          services
asound.state        gpm-syn.conf         libgda/           nsswitch.conf-nis      sgml/
at.deny             gpm-twiddler.conf    libuser.conf      ntp/                   shadow
bonobo-activation/  group                lilo.conf         ntp.conf               shadow-
bootptab            gshadow              localtime         openldap/              shells
--More--
```

/home

Como visto anteriormente o Linux é um sistema multi-usuário. Assim, cada usuário tem associado a ele um diretório que deve estar somente acessível a ele e ao administrador do sistema.

Dentro do diretório `/home` estão todos estes diretórios particulares dos usuários cadastrados no sistema. Listando com `ls` o conteúdo do `home/`, veremos todos os diretórios dos usuarios(seus *home*), cujos nomes são exatamente os mesmos cadastrados no sistema(*login*).

O diretório `home/$USUARIO` é a área de trabalho particular de cada usuário e está sob seu domínio. Ali ele pode criar arquivos, apaga-los, instalar programas e etc. Além disso, este diretório pode armazenar arquivos que guardam configurações do seu ambiente de trabalho ou de aplicações que utiliza. Estes arquivos geralmente tem seus nomes começando com "." (ponto) e por isso são chamados aquivos "." (ponto).

Estes arquivos normalmente ficam escondidos e para vê-los precismos utilizar o parâmetro adequado do comando `ls`: **`ls -a`**. Se houver algum conflito entre as configurações dos arquivos pessoais e as configurações globais do sistema, as configurações pessoais do usuário prevalecerão.

Alguns dos arquivo ponto:

`$HOME/.bash_profile` - executado automaticamente no *login*
`$HOME/.bashrc` - executado automaticamente na inicialização do *shell*
`$HOME/.bash_logout` - executado automaticamente no *logout*
`$HOME/.bash_history` - grava os últimos comandos das últimas secões

/lib

Os programas que nós criamos podem usar bibliotecas de onde buscam suas funções ou métodos. Vários programas podem usar a mesma biblioteca de funções e métodos, tanto na hora de sua compilação quanto em execução. Para este fim, no Linux existe este diretório que abriga todas as bibliotecas compartilhadas.

Desta forma todos os programas "sabem" onde encontrar as bibliotecas de funções. Alguns módulos do kernel podem estar presentes neste diretório para serem "carregados" à medida da necessidade. Ao instalarmos algumas aplicações novas nos sistema, elas podem pôr neste diretório suas bibliotecas. As bibliotecas são facilmente identificas por meio de suas extensões como: `*.so`. Alguma coisa equivalente no Windows seriam os arquivos `*.dll`.

Listagem parcial do `/lib`:

```
oliver@darkside:/lib$ ls
cpp@                        libe2p.so.2.3*          libpamc.so.0.77*
evms/                       libext2fs.so.2@         libpam_misc.so@
ld-2.3.2.so*                libext2fs.so.2.4*       libpam_misc.so.0@
ld-linux.so.2@              libgpm.so.1@            libpam_misc.so.0.77*
libanl-2.3.2.so*            libgpm.so.1.18.0*       libpam.so@
libanl.so.1@                liblvm-10.so@           libpam.so.0@
libblkid.so.1@              liblvm-10.so.1@         libpam.so.0.77*
libblkid.so.1.0*            liblvm-10.so.1.0*       libpcprofile.so*
--More--
```

/mnt, /mnt/cdrom, /mnt/floppy e etc

Se olharmos dentro destes diretórios, normalmente, não veremos conteúdo algum. A função destes diretórios é servirem como ponto de montagem de dispositivos. No Linux não temos nada parecido com "A:" ou "C:" para termos acesso aos dipositivos.

Um drive de disquete ou uma unidade de CD-ROM, tem de ser expressamente montado (ativado) em algum diretório no sistema para que possamos ter acesso a ele. Montar é o processo pelo qual tornamos um sistema de arquivos disponível para o sistema. Assim, normalmente usamos o `cdrom/` para montarmos drives de CD, `mnt/` de forma genérica e o `floppy/` para drives de disquete.

/opt

Este diretório é reservado para todo e qualquer software ou pacote adicional que não faça parte da instalação padrão do sistema naquela distribuição. Por exemplo: StarOffice, OpenOffice, Kylix, KDE e outros são algumas aplicações que, por padrão, podem ser instaladas neste diretório.

/proc

Este é também um diretório bem especial por ser um *filesystem* virtual. Alguns se referem a ele como um pseudo-file system de informações sobre processos. Ele não contém arquivos "reais" mas informações sobre o estado do sistema e processos como: memória do sistema, dispositivos montados, configuração de hardware e etc. Muitos dos utilitários do sistema simplesmente fazem uma chamada aos arquivos deste diretório para cumprirem com suas funções. Por exemplo, usar o `lsmod` é o mesmo que o comando `cat /proc/modules` ou ainda: `lspci` é o mesmo que `cat /proc/pci`. Se alterarmos alguns dos arquivos presentes neste diretório podemos estar lendo/alterando parâmetros do *kernel* (`sysctl`) enquanto o sistema está em funciomamento.

Listagem do `/proc`:

```
oliver@darkside:/lib$ ls
1/     1032/  1092/  4/    902/  apm          fs/         mdstat      stat
10/    1034/  11/    44/   903/  asound/      ide/        meminfo     swaps
1006/  1035/  1125/  5/    916/  bus/         interrupts  misc        sys/
1008/  1036/  1622/  6/    917/  cmdline      iomem       modules     sysvipc/
1010/  1038/  1623/  861/  925/  cpuinfo      ioports     mounts@     tty/
1012/  1040/  1624/  864/  951/  crypto       irq/        mtrr        uptime
1013/  1042/  1625/  867/  953/  devices      kcore       net/        version
1014/  1068/  1627/  870/  957/  dma          kmsg        partitions
--More--
```

/root

Este é o diretório *home* do usuário *root;* seu diretório particular. Como já vimos o diretório *home* de um usuário é sua área de trablaho particular e, assim, outros usuários tem restrições de acesso a ele. No caso do *root*, estas restrições se aplicam a quaisquer outros usuários, o que não é recíproco pois, como falamos anteriormente, o *root* tem acesso total a todas as áreas do sistema.

Por que em um diretório separado e não dentro do `/home` como todos os outros? Uma das razões para isto é que este diretório deve sempre estar acessível ao *root* e podem ocorrer situações onde o diretório `/home` esteja localizado em uma outra partição ou mesmo em outro sistema, o que representa um risco de eventualmente não estar acessível.

/sbin

Este é um outro diretório onde o acesso para usuários comuns é restringido. Como no diretório `/bin`, ele abriga muitos dos comandos/programas mais frequentemente utilizados. Etretando, é um diretório em que podemos ver mas, não devemos mexer. Os comandos do diretório `/sbin` são para o uso do administrador do sistema, o *root*. O comando `shutdown`, por exemplo está aqui. Se qualquer usuário tentar executar este comando, os sistema negará. Outros comandos podem, eventualmente, ser executados por usuários comuns mas podem gerar mensagens de erro.

Listagem parcial do `/sbin`:

```
oliver@darkside:/sbin$ ls
accton*                  jfs_fscklog*          pvdisplay*
adjtimex*                jfs_logdump*          pvmove*
agetty*                  jfs_mkfs*             pvscan*
arp*                     jfs_tune*             pwdb_chkpwd*
arping*                  kallsyms@             quotacheck*
arytst*                  kernelversion*        quotaoff@
badblocks*               killall5*             quotaon*
blkid*                   ksyms@                raid0run@
blockdev*                ldconfig*             raidhotadd@
--More--
```

/tmp

Neste diretório, como o próprio nome sugere, ficam os arquivos temporários. Estes arquivos podem ser gerados tanto pelo próprio sistema quanto por aplicações que usamos no dia-a-dia. Por padrão, a maioria das aplicações desenvolvidas para o Linux, se utilizam deste diretório para suas informações temporárias. Frequentemente encontramos "lixo" neste diretório. É um diretório que pode ser usado por todos os usuários(leitura e escrita).

/usr

Aqui, encontram-se arquivos e programas que devem estar diponíveis a todos os usuários do sistema. Ele é umas maiores áreas do sistema e sua função é guardar informações (programas e arquivos) que podem e dever ser compartilhadas e cujo conteúdo tenha pouca ou nenhuma alteração com o passar do tempo.

Dentro do `/usr` veremos muitos diretórios que abrigam aplicações e seus arquivos de configuração que, uma vez instalados/configurados, não mudarão por um bom tempo ou nunca. Um exemplo disto é o diretório `/usr/X11R6/` ou `X11@`. Nele vamos encontrar o X Windows, o sistema para ambiente gráfico mais utilizado no Linux.

Algumas pessoas podem fazer confusão como o nome do diretório e o seu significado pensando que `/usr` estaria relacionado com "user"(ou diretório dos usuários – equivalente ao `/home`) mas, na realidade está relacionado com "User System Resources".

Listagem do `/usr`:

```
oliver@darkside:/usr$ ls
adm@   doc/    i486-slackware-linux/  lib/      man/    spool@  X11@
bin/   etc/    include/               libexec/  sbin/   src/    X11R6/
dict/  games/  info/                  local/    share/  tmp@
```

/var

Este diretório contém arquivos cujos conteúdos são variáveis. Normalmente, estes arquivos armazenam informações sobre ocorrência ou status de aplicações/programas que estão sendo executados no momento. Estas infomações e arquivos estão organizadas em vários sub-diretórios sob o `/var`. O sistema também mantém aqui informações desta natureza em sub-diretórios como: `/var/pid`, `/var/log`, `/var/spool` e etc. Algumas das informações contidas aqui podem ser compartilhadas enquanto outras não.

Listagem do `/var`:

```
oliver@darkside:/var$ ls
adm@    empty/  lock/  mail@  named/  rwho@   state/  www/    yp/
cache/  lib/    log/   man/   run/    spool/  tmp/    X11R6/
```

Shell

Podemos definir um *shell* como um programa que interpreta comandos os quais podem ser recebidos diretamente do usuário ou lidos de um arquivo que será chamado de *shell script* ou *programa shell*. Scripts de *shell* são interpretados; não compilados. O *shell* lê os comandos do *script*, linha por linha, e procura por estes comandos no sistema e então executa-os.

Quando digitamos comandos para apagar arquivo, criar diretórios e etc, estamos usando um *shell*. Todo sistema operacional usa um *shell* como meio receber os comandos digitados no teclado para a máquina. Ele é uma forma amigável e flexível de comunicação – uma espécie de linguagem - entre o usuário e o S.O.; um tipo de interface.

Além de ser uma forma de enviar comandos para o *kernel*, uma das principais funções de um *shell* é proporcionar ao usuário um ambiente de trabalho que possa ser configurável. Ou seja, que possar ser personalizado.

Tipos de shell

- **sh** ou Bourne Shell: o *shell* original e ainda usado em ambientes UNIX. É um shell básico, um programa pequeno com um número não muito grande de funcionalidades. Embora não seja o shell padrão ainda é suportado no Linux por razões de compatibilidade com ambientes UNIX.

- **csh** or C shell: é um *shell* cuja sintaxe lembra a linguagem de programação C. Alguns o chamam de *shell* para programadores.

- **tcsh** or Turbo C shell: é uma *superset* do C shell, com alguns melhoramentos quanto a velocidade e a se tornar mais amigável para o usário.

- **ksh** or the Korn shell: é uma *superset* do Bourne shell; em sua configuração padrão pode ser um pouco “assustador” para usuários iniciantes.

- **bash** ou Bourne Again shell: o *shell* padrão do GNU. Shell, intuitivo e muito flexível. Além de ser, provavelmente, o mais indicado para usuários iniciantes é também uma ótima ferramenta para usuários avançados e proficionais. É o *shell* padrão para a maioria dos usuários no Linux.

Sobrevivência no Linux – Modo Texto

A partir de agora vamos tratar das tarefas mais comuns e como executa-las. Na medida do possível, veremos algumas outras coisas úteis e interessantes.

Consoles Virtuais

O Linux, assim como sistemas operacionais do tipo UNIX, oferecem uma opção muito interessante: os consoles virtuais. Lembrando do antigo MS-DOS, ainda antes do Windows, tinhamos a situação em que somente era possível executarmos um progrmama por vez. Trabalhar no Linux em modo texto lembra os tempos do MS-DOS. Etretanto, o Linux é um sistema multi-tarefa e multiusuário. Diferente do velho DOS, podemos trabalhar com mais de um programa e até como mais de um usuário, ao mesmo tempo.

Vamos a um exemplo: você esta trabalhando na máquina em um programa, como o usuário *paulo*, e tem necessidade de fazer algum tipo de configuração no sistema, como *root*. Não é necessário encerrar o programa em que está trabalhando. Usando as teclas ALT+F2 o Linux mostrará novamente a tela inicial de login . Então, é só entrar com o nome e senha do novo usuário com que se quer trabalhar e pronto.

A combinaçao de ALT e uma das teclas de função no teclado, irá permitir o trabalho como mais de um usuário em mais de um programa. Estes são os consoles virtuais.

Facilidades do Bash

O Bash dispõe de algumas facilidades que certamente agilizam nosso trabalho no dia-a-dia quando utilizamos os terminais em modo texto. A tabela a seguir mosta algumas das principais:

Combinação de teclas	Função
Ctrl+**A**	Move o cursor para o início da linha de comando
Ctrl+**C**	Interrompe um programa em execução e retorna ao *prompt*

Ctrl+**D**	Efetua o *log out*. Equivalente aos comandos `exit` e `logout`
Ctrl+**E**	Move o cursor para o final da linha de comando
Ctrl+**H**	Equivalente a tecla *backspace*
Ctrl+**L**	Limpa a tela do terminal(console)
Ctrl+**R**	Faz uma busca no histórico dos comando já digitados
Ctrl+**Z**	Suspende um programa (sem interrompe-lo)
Setas ↑ e ↓	Busca sequencial no histórico dos comandos já digitados
Shift+**PageUp/PageDown**	Fazem a rolagem da tela do terminal
TAB	Recurso de completar automaticamente o nome de um arquivo ou comando
TAB TAB	Mostra as possibilidades para o recurso de completar automaticamente.

Metacaracteres (coringas)

*****	Substitui qualquer *string* de zero ou mais caracteres
?	Substitui 1(um) caractere simples na posição
[abc...]	Qualquer dos caracteres delimitados pelos colchetes (um ífen pode ser usado para especificar uma faixa (Exemplo: a-z, A-Z, 0-9)
[!abc...]	Qualquer combinação **diferente** dos caracteres delimitados pelos colchetes (um ífen pode ser usado para especificar uma faixa (Exemplo: a-z, A-Z, 0-9)

Exemplos:

```
oliver@darkside:/sbin$ ls ba*
```

```
oliver@darkside:/sbin$ ls ch?
```

```
oliver@darkside:/sbin$ ls [c-i]*
```

Pedindo Ajuda

Todas as distribuições do Linux vêm com uma farta documentação. Mas mesmo sem esta documentação temos ao nosso dispor os recuros do próprio sistema que são os manuais dos comandos e da maioria das aplicações instaladas no sistema.

Duas formas de lermos esses manuais são os comandos: `man` e `info`

Exemplos:

```
oliver@darkside:~$ man ls
```

```
oliver@darkside:~$ info cd
```

Outros comandos para ajuda:

- **`whatis`**: mostra uma explicação curta sobre o comando

- **`apropos`**: Busca por todas as informações disponíveis a respeito de um comando ou de um uma palavra-chave que não seja necessáriamente o nome de um comando ou programa.

```
oliver@darkside:~$ apropos email
audiosend            (1)  - Send an audio email message
mailpost             (8)  - feed an email message into a news group
news2mail            (8)  - a channel script to gateway news into email
pine                 (1)  - a Program for Internet News and Email
showaudio            (1)  - Play an audio email message
```

- A opção --**`help`**: a maior parte dos comandos aceita este parâmetro para mostrar uma pequena ajuda sobre seu uso.

```
oliver@darkside:~$ cat --help
Usage: cat [OPTION] [FILE]...
Concatenate FILE(s), or standard input, to standard output.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonblank output lines
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      never more than one single blank line
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation,
                           except for LFD and TAB
```

Mais comandos

Vejamos uma lista de outros comandos com uma breve descrição de sua função

- **cd**

  Abreviatura de “change directory”. Utilizamos para mudarmos de um diretório para outro

  ```
  oliver@darkside:~$ cd /bin
  ```

- **ls**

  Lista o countéudo de um diretório. É um dos comandos com o maior número de parâmetros do sistema. Veja sua página de manual.

  ```
  oliver@darkside:/bin$ ls a*
  arch*  ash*  awk@
  ```

- **mkdir**

  Cria um novo diretório no local atual.

  ```
  oliver@darkside:~$ mkdir dir-teste
  ```

- **cp**

  Copia os arquivos listados na linha de comando para o arquivo ou diretório especificado. Aqui podemos usar o sinal "."(ponto) para fazer referência ao diretório atual.

```
oliver@darkside:~$ cp teste/arquivo.txt .
```

- **mv**

  Move um arquivo de um lugar para outro podendo neste mesmo processo alterar o seu nome.

```
oliver@darkside:~$ mv teste/arquivo.txt ./arquivo.doc
```

- **rm**

  Remove/apaga um arquivo ou diretório. Aceita vários parâmetros.

```
oliver@darkside:~$ rm teste/arquivo.txt
```

- **rmdir**

  Remove/apaga um diretório mas somente se este estiver vazio.

```
oliver@darkside:~$ rmdir teste
```

- **chmod**

  Muda os atributos/permissões de um aruivo. Use este comando para dar um pouco de segurança aos seus arquivos.

```
oliver@darkside:~$ chmod go-xrw teste/arquivo.txt
```

- **chown**

  Muda os proprietários(usuário e/ou grupo) de um arquivo

```
oliver@darkside:~$ chown oliver.mail /var/spool/mail/oliver
```

- **more**

  Lista o conteúdo de um arquivo na tela de terminal. Mostra uma parte do arquivo suficiente para preenchar a tela(página) e aguarda pela ação do usuário para continuar; linha por linha ou página a página.

```
oliver@darkside:~$ more teste/arquivo.txt
```

- **cat**

  Concatena o conteúdo de arquivos. Também usamos para exibir todo o conteúdo de um arquivo, na tela de terminal(console), de uma só vez.

```
oliver@darkside:~$ cat teste/arquivo.txt
```

- **file**

  Mostra de que tipo é o arquivo. Ou seja, que tipo de dados então contidos no arquivo.

```
oliver@darkside:/bin$ file sh
sh: symbolic link to bash
```

- **which**

  Procura o arquivo especificado nos diretórios que fazem parte do caimho de busca. Caso encontre o arquivo, mostra sua localização.

```
oliver@darkside:~$ which bash
/usr/bin/bash
```

- **find**

  Procura por um arquivo a apartir do diretório especificado. Vários critérios de busca podem ser especificados como: nome, tamanho, conteúdo e etc.

```
oliver@darkside:~$ find / -name bash
```

- **less**

  É a versão GNU do `more` e acrescenta algumas outras funcionalidades como busca por expressões, popr exemplo.

```
oliver@darkside:~$ less teste/arquivo.txt
```

- **head**

  Mostra as *n* primeiras linhas de um arquivo.

```
oliver@darkside:~$ head -10 .bash_history
```

- **tail**

  Mostra as *n* últimas linhas de um arquivo.

```
oliver@darkside:~$ tail -10 .bash_history
```

Juntando comandos

Na linha de comando ou em *script*. Temos a possibilidade de usar os símbolos "|" (pipe) e ";" para combinarmos as ações de dois comandos diferentes.

`cm1; cmd2`	Executa os somandos na seqência em que aparecem.
`cm1 | cmd2`	Usa a saída do primeiro comando como entrada para o segundo.

Exemplo:

```
oliver@darkside:~$ cat /etc/passwd | grep oliver
oliver:x:1000:100:Oliver T Lessa,,,:/home/oliver:/bin/bash
```

Scripts de shell

Como já vimos, um *shell* como o `bash` pode interpretar comandos colocados em um arquivo e executa-los. Este é o *script*; em certos casos é bem parecido com um programa escrito em uma liguagem de programação qualquer. Falando especificamente do `bash` é notória a grande flexibilidade que ele oferece e que faz com que os *scripts* sejam quase tão versáteis quanto programa em outras linguagens.

Não vamos tratar aqui da construção de scripts e de tudo o que o `bash` nos oferece nesta área. Vamos listar algumas de suas funcionalidade e mostrar um exemplo.

Funcionalidades

- Permite uso de variáveis;
- Arrays
- Trabalha com passagem de parâmetros
- Tem todos os operadores aritiméticos comuns
- Trabalha com expressões e substituições de expressões e comandos
- Permite construções condicionais: **`if`** `expressao;` **`then`** `comandos;` **`fi`**
- Comparações numéricas e de *strings*
- Loops com: **`while`** `expressao;` **`do`** `comandos;` **`done`**
- Suporte a funções(incluindo passagem de parâmetros)
- etc

Um exemplo de um *script* simples:

```
#!/bin/bash
clear
printf "This is information provided by mysystem.sh.  Program
starts now."

printf "Hello, $USER.\n\n"

printf "Today's date is `date`, this is week `date +"%V"`.\n\n"

printf "These users are currently connected:\n"
w | cut -d " " -f 1 - | grep -v USER | sort -u
printf "\n"

printf "This is `uname -s` running on a `uname -m`
processor.\n\n"

printf "This is the uptime information:\n"
uptime
printf "\n"

printf "That's all folks!\n"
```