

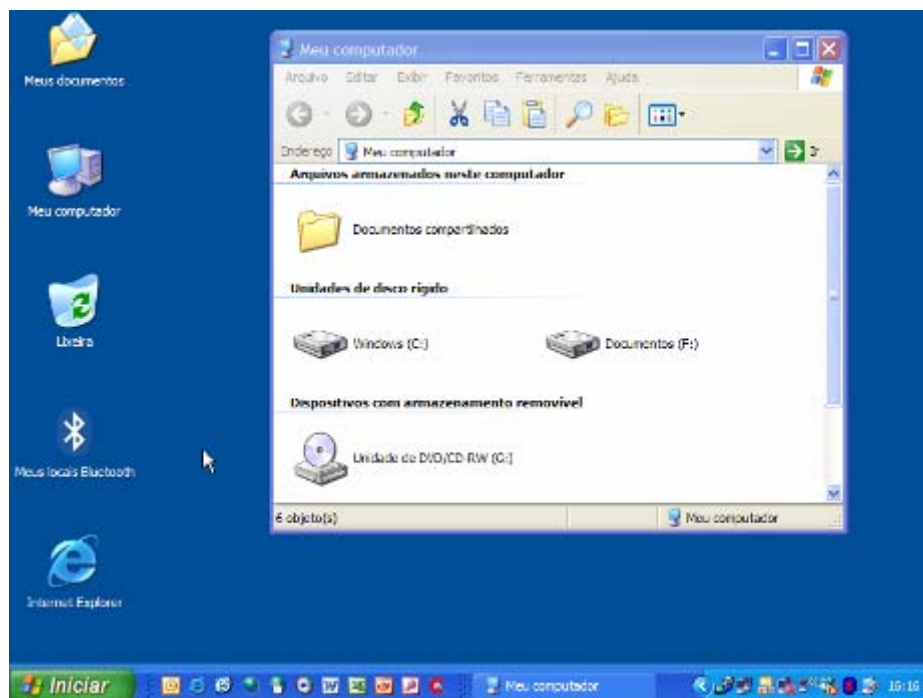
SISTEMA OPERACIONAL LINUX

Introdução

Um Sistema Operacional é um programa que tem por função controlar os recursos do computador e servir de interface entre ele e o usuário. Essa definição é mostrada em qualquer livro que fale a respeito desse tipo de software.

Um sistema operacional, é, portanto, um programa que tem a obrigação de controlar a máquina, permitindo-nos comandá-la através de ordens pré-definidas. Sem um sistema operacional, por exemplo, não seria possível usar uma planilha eletrônica, um editor de textos ou mesmo acessar à Internet.

Quando pensamos em sistema operacional, nos vem à mente, imediatamente, o **Microsoft® Windows®**, utilizado na grande maioria dos computadores pessoais do planeta. Não há como contestar a presença hegemônica da gigante de Bill Gates em nossas vidas computacionais, mas é bom que se saiba que o Windows não é o único sistema operacional que podemos utilizar nos nossos micros.



O Advento de um novo sistema há alguns anos tem tirado o sono dos executivos da Microsoft, não somente por mostrar-se, em muitos aspectos, melhor que seu concorrente, como bem mais barato! Esse novo sistema chama-se **Linux**.

Surgimento do Linux

Em 1991, segundo reza a lenda, o então estudante finlandês **Linus Torvalds** resolveu desenvolver um sistema operacional que se assemelhasse ao **UNIX** (que ele usava na universidade de Helsinque) porque esse sistema não era compatível com o seu PC doméstico. E nesse esforço, surgiu o Linux, que, naquela época, ainda não tinha esse nome.

Explicando: O UNIX é um sistema operacional muito robusto, usado em computadores de grande porte nas empresas e universidades. O UNIX foi desenvolvido, inicialmente, em 1969, na Universidade de Berkeley, na Califórnia. A grande maioria dos cursos de

computação das universidades do mundo utiliza o UNIX em seus servidores, por isso o estudo de Torvalds se baseava nesse ambiente.

Até os dias de hoje, o aprendizado e a utilização do UNIX é privilégio de alguns poucos mortais que se aprofundam no estudo da computação e têm acesso a computadores de grande porte, onde esse sistema pode ser instalado. Mas o Linux trouxe essa utilização para níveis mais cotidianos, permitindo que qualquer um que possua um PC comum possa ter acesso ao universo dos sistemas **UNIX-Like** (termo que significa "semelhantes ao UNIX").

```
-rw----- 1 root root 1589 2005-04-08 20:18 .xsession-errors
drwxr-xr-x 4 root root 112 2005-04-06 23:13 Concursos
drwxr-xr-x 3 root root 104 2005-04-06 21:17 Desktop
drwxr-xr-x 5 root root 120 2005-04-06 23:23 GNUstep
drwx----- 7 root root 520 2005-04-06 22:38 Mail
%
%ps -r
Warning: bad ps syntax, perhaps a bogus '-?' See http://procps.sf.net/q.html
  PID TTY          STAT       TIME COMMAND
  4653 pts/1    R           0:00 -sh
  4668 pts/1    R+          0:00 ps -r
%ls r
ls: r: No such file or directory
%ps r
  PID TTY          STAT       TIME COMMAND
  4653 pts/1    R           0:00 -sh
  4670 pts/1    R+          0:00 ps r
%ps
  PID TTY          TIME CMD
  4648 pts/1    00:00:00 bash
  4650 pts/1    00:00:00 sh
  4653 pts/1    00:00:00 ssh
  4671 pts/1    00:00:00 ps
%[]
```

Linus havia criado um sistema parecido com o UNIX e concorrente do MINIX (um outro sistema UNIX-Like, criado por um conhecido professor americano e autor de diversos livros: Andrew Tanenbaum), mas que só funcionava na máquina dele, em casa (afinal, ele o havia projetado com base no processador e na arquitetura de seu micro caseiro).

Linus escreveu uma mensagem numa lista de discussão na Internet (que, na época, não havia sido descoberta comercialmente em muitos países, incluindo o Brasil) encorajando os outros programadores e usuários do UNIX a ajudá-lo na tarefa de criar um sistema operacional semelhante para micros domésticos que superasse, em muitos aspectos, o MINIX (até então, uma das pouquíssimas opções de sistema UNIX-Like para PCs).

Para pôr em prática seu desejo de ter um sistema semelhante ao UNIX que funcione nos PCs, Linus enviou, aos interessados, o código-fonte do seu sistema, para que os outros programadores pudessem entender e modificar o seu projeto.

Em tempo: Código-Fonte é o nome dado ao conjunto de instruções escritas pelo programador em uma linguagem compreensível para ele (e, normalmente, para mais ninguém!). O projeto de Linus foi escrito na linguagem C, que é bastante poderosa e versátil, sendo a preferida pelos programadores que desenvolvem softwares básicos (aqueles que falam diretamente com a máquina).

O código-fonte não é o programa pronto para ser executado, em vez disso, é a "receita" de como o programa foi criado e do que ele vai fazer. Para que o código-fonte se transforme no arquivo que será executado pelo computador, é necessário um processo de tradução que reescreva o programa na linguagem que o computador entende, também chamada de linguagem de máquina, esse processo de tradução é chamado **compilação**.

Pois é, quando Linus Torvalds distribuiu o código-fonte de seu programa, ele tornou possível para outros programadores fazer alterações em seu sistema, permitindo que muitos se tornassem os co-desenvolvedores do Linux, nome, aliás, que só seria dado ao sistema alguns anos depois, em homenagem ao seu pai original.

O Linux original possuía poucos recursos visuais e de aplicativos, mas era o suficiente para Linus entender que ele poderia melhorar e atingir níveis de usabilidade altos. Ou seja, o negócio ainda não prestava, mas ia melhorar e um dia prestaria para alguma coisa!

Só para você ter uma idéia do que Linus criou e o que ele fazia na época, segue uma foto. O Linux era um sistema basicamente textual, como o DOS, e cujos comandos eram semelhantes aos comandos do UNIX (claro, por ser UNIX-like).

Os Direitos Sobre o Linux

O Linux é um sistema operacional, como já foi dito, e, por isso, tem a função de controlar o computador e permitir que o usuário dê comandos a ele. Isso não é privilégio do Linux, visto que outros programas fazem exatamente o mesmo, como o conhecido Windows, da Microsoft.

Entre outras características diferentes entre eles, podemos citar uma que tornou o Linux algo muito interessante e digno de atenção: **o Linux é um software livre**. Isso significa que aqueles que adquirem o Linux têm certos direitos em relação a ele que os usuários do Windows não possuem em relação ao sistema da Microsoft.

Mas, o que é ser “**software livre**”?

Para explicar esse termo, recorremos ao conceito de um software comercial como o Windows: para usar o Windows em um computador, o usuário tem que ter pago uma taxa a título de “direito autoral” pelo programa, chamada de licença de uso. A licença de uso do Windows é paga por cada computador onde o programa esteja instalado na empresa.

Além disso, não será permitida a cópia do CD original do programa para a instalação em outro computador sem o pagamento de uma nova licença. Então, se um usuário comprar o Windows em uma loja e resolver instalá-lo em mais de um computador, estará cometendo crime de pirataria de software, ou desrespeito às leis de copyright que regem os programas comerciais como o Windows.

No Linux, a coisa muda de figura radicalmente. Os desenvolvedores do Linux, incluindo seu criador, Linus Torvalds, classificaram o Linux numa licença chamada GPL (General Public License – Licença Pública Geral), da FSF (Free Software Foundation – Fundação do Software Livre).

A FSF é uma instituição supostamente sem fins lucrativos que desenvolveu uma “legislação específica” para todos os programadores que quisessem que seus projetos fizessem parte do mundo dos programas sem rigidez de copyright.

Como Assim? Simples: Se você é um programador e criou um software que deseja vender para explorar seu valor financeiro, exigindo pagamento da licença a todos os usuários, vá em frente, não é com a FSF! Mas, se você desenvolveu um programa para ser distribuído por aí, para quem quiser usar, abrindo mão da licença de uso, a FSF criou a GPL para você!

Na GPL, 4 direitos são garantidos aos usuários dos programas regidos por ela (os chamados Softwares Livres):

1. Um Software Livre poderá ser usado para qualquer finalidade;
2. Um Software Livre poderá ser estudado plenamente (para isso, é necessário possuir o código-fonte do programa);
3. Um Software Livre poderá ser alterado em sua totalidade (para isso, é necessário possuir o código-fonte do programa);
4. Um Software Livre poderá ser distribuído (copiado) livremente, sem a exigência de pagamento de licença de uso em nenhum dos casos...

Para os desenvolvedores (programadores) é muito interessante que a GPL determine que o código-fonte do programa seja liberado (ou aberto, como chamamos). O Linux e os demais programas regidos pela GPL são **Open-Source** (código aberto – ou seja, têm seu código-fonte necessariamente acessível a todos).

Para nós, usuários comuns, a característica mais importante dos programas regidos pela GPL é o seu custo, que, devido a não obrigação de pagamento de licença (atrelada à idéia de copyright), torna a implantação do Linux e de outros softwares livres nas empresas algo financeiramente convidativo.

Então, que se saiba: **O Linux é um Software Livre!**

Então, entenda que, devido às diversas vantagens que um software livre tem em relação aos tradicionais programas pagos (como custos, possibilidade de modificação do programa), a mudança de ares na informática de empresas públicas e privadas, saindo do quase onipresente Windows para o onisciente Linux, é inevitável.

Cada vez mais, e isso é sensível, os até então clientes da Microsoft estão se entregando aos prazeres (e desafios) de utilizar o sistema do pingüin (alusão ao Linux porque seu “mascote” ou “logomarca” é um simpático exemplar desta ave, chamado Tux).

A Comunidade Linux - Criadores de Pingüin

Lembra dos programadores com quem Linus Torvalds entrou em contato para ajudá-lo na tarefa de alimentar o Linux de conteúdo e funcionalidade a fim de fazê-lo crescer? Pois é, eles também cresceram em número!

Atualmente, cerca de 10.000 pessoas no mundo todo dão sua contribuição valiosa para a manutenção e evolução do Linux, seja criando novos aplicativos e drivers, seja melhorando o funcionamento do próprio sistema (que é trabalho dos programadores) ou até mesmo traduzindo as interfaces para que o Linux se apresente disponível nos mais variados idiomas (portanto, qualquer poliglota pode fazer parte desse grupo, não precisa conhecer a linguagem C).

Esta, leitor, é a **Comunidade Linux**, que é formada pelos mais diversos profissionais que desejam, simplesmente, em seus tempos livres, colocar mais um tijolo no já muito firme alicerce do ambiente Linux, a fim, talvez, de que um dia o sistema de Linus (e de todos eles) possa desbancar o Windows.

Agora vamos às comparações (é inevitável também): No sistema Windows, qualquer mudança é feita pela detentora do código-fonte, a Microsoft, que disponibiliza a atualização em seu site “Windows Update”. Quanto ao Linux, qualquer usuário conhecedor de C pode mudar alguma coisa que não ache satisfatória no sistema, permitindo melhorias imediatas sem a dependência de uma suposta fabricante. Isso, é claro, porque o usuário é o detentor do código-fonte!

Certas mudanças ficam restritas ao computador do usuário que as fez, mas algumas são enviadas à comunidade, que avalia a relevância da mudança e julga se ela pode ser ou não adicionada na próxima versão do Linux.

O objetivo da Comunidade não é somente criar coisas novas (embora faça isso também), mas, também, modificar constantemente o centro do sistema Linux, o seu **Kernel**.

O Kernel do Linux - A Alma do Sistema

Todo sistema operacional é complexo e formado por diversos programas menores, responsáveis por funções distintas e bem específicas. O **Kernel** é o centro do sistema operacional, que entra em contato direto com a CPU e os demais componentes de hardware do computador, sendo, portanto, a parte mais importante do sistema.

Se o Kernel é mal feito, o sistema operacional nunca funcionará direito, travando constantemente, e executando operações que ele mesmo não permitiria (lembra de algum exemplo?). Mas, se o Kernel é bem construído, existem garantias de que o sistema se comportará sempre da melhor maneira, tornando travamentos e desafetos com o usuário coisas raras (o Linux é muito bom nesse critério, é muito difícil vê-lo travar e comprometer a estabilidade de um computador).

O Kernel é um conjunto de subprogramas, revistos e alterados pela Comunidade Linux o tempo todo, ou seja, existem milhares de pessoas no mundo todo, nesse momento, alterando alguma característica do Kernel do Linux no intuito de melhorá-lo.

Mas o que garante que, sendo o Kernel alterado por tantas mãos, ele não se torne uma "colcha de retalhos" de códigos que gerem incompatibilidades e problemas? Ou seja, o que o faz tão estável e robusto se é "filho de tantos pais"? Ou ainda: o que garante que alguém, dentre esses milhares, não colocaria algo prejudicial no código do Linux para fazê-lo intencionalmente perigoso?

Simples: A comunidade tem direito de alterar o Kernel do Linux, mas todas as alterações são analisadas e julgadas pertinentes ou não por alguns "gurus", os **Mantenedores do Kernel** ou **Guardiães do Kernel**.

Entre os poucos guardiães do Kernel, podemos citar três loucos (no sentido carinhoso da palavra), são eles: Linus Torvalds, não por acaso; Marcelo Tosati (um brasileiro escolhido pelo próprio Linus); e Jon "MadDog" Hall, um dos criadores e principais defensores da idéia de Software Livre. São eles (e mais alguns) que ditam as regras quanto ao que será adicionado ou retirado da próxima versão do Kernel do Linux.

"Sim, Joao, mas como funciona esse negócio de versão do Kernel?"

Novamente, é simples: De tempos em tempos (não há uma exatidão), é lançada uma nova versão do Kernel do Linux, seu "centro nervoso". Esse lançamento é realizado pelos gurus, que analisaram todas as propostas de alteração enviadas pela comunidade e, aceitando algumas e rejeitando outras, decidem que a nova versão está pronta.

Atualmente, encontramos-nos na versão 2.6 do Kernel do Linux (a versão estável mais recente do sistema). Normalmente, as versões do Kernel são batizadas com três ou quatro níveis de números, que identificam sua geração.

Há algum tempo, tínhamos a versão 2.4 e todas as "mini-versões" dentro dela, como 2.4.1, 2.4.15, 2.4.29, etc. Hoje, a versão mais difundida já é a versão 2.6 e toda a sua família (2.6.3, 2.6.11, etc.).

A mudança da versão 2.4 para a 2.6 trouxe muitas novidades, especialmente no tocante às tecnologias que o Kernel novo é capaz de suportar (redes sem fio, bluetooth, novos dispositivos, etc.). Essa "mudança da água para o vinho" também deverá ocorrer quando os gurus lançarem a versão 2.8 e, da 2.8 para a 2.10... Mas, com certeza deverá ser **muito mais significativa** quando sairmos da versão 2 para a 3 (não sei quando isso ocorrerá).

É fácil perceber que a mudança do primeiro nível (o 2) é muito mais demorada (claro!), até mesmo porque deve haver muitas mudanças cruciais no sistema para que se justifique a saída da "geração 2" para a entrada da 3! A mudança do segundo nível demora um certo tempo também, mas as mudanças no terceiro e quarto níveis são bem mais frequentes.

Aí, você pergunta: "João, notei que a mudança do segundo nível da versão acontece apenas com números pares (2.4, 2.6, 2.8, etc.)... Por quê?"

Os mantenedores preferiram criar as versões X.Y, fazendo o Y ímpar quando querem indicar que essa versão não está estável, ou seja, que existe alguma tecnologia nova que está sendo testada nessa versão. É assim: a versão 2.3 trazia novas tecnologias (instavelmente, ainda) que, quando foram devidamente testadas e aprovadas, deram origem à versão 2.4. A 2.5 também é precursora da atual 2.6 e, claro, já se está

trabalhando na versão 2.7 (a comunidade já iniciou seu desenvolvimento para que, quando as novidades estiverem perfeitamente funcionais no Kernel, este possa ser batizado de 2.8 e lançado para o público em geral).

Aí, você pergunta, de novo: “Certo, entendi! Mas, e por que existem os outros níveis de mudanças? Por exemplo, porque existe a versão 2.6.11 se as novas tecnologias só estarão disponíveis na 2.8?”

Ótima pergunta! Às vezes, a versão original do Kernel (2.6, por exemplo) apresenta certos probleminhas com alguns modelos de dispositivos, ou falhas de programação, ou qualquer outra chatice. Quando esses inconvenientes são detectados por alguém da comunidade, este avisa aos mantenedores que lançam uma “nova versão 2.6” com as correções devidas. Ou seja, a versão 2.6.11 é mais recente que a 2.6.10 e, provavelmente, traz correções para os bugs (defeitos de programação) da anterior.

O Kernel é, para que se entenda de forma simples, o Sistema Operacional em si. Quer dizer, o **Linux é seu Kernel** (o restante do Linux são programas extras, desenvolvidos por diversos programadores da comunidade, como aplicativos e jogos).

Pergunta, novamente: “Quer dizer que, basta eu ter o Kernel do Linux e eu posso usar esse sistema em meu computador sem problemas? Basta o Kernel do Linux para o meu micro ser utilizável?”

Não! Nem só de Kernel vive o sistema operacional! O Kernel do Linux em si é muito pequeno e não tem muita coisa, mas claro que tem o mais importante, já que ele é o sistema em si! Porém, para que o Linux seja utilizável, é necessário que existam, também, outros programas que, junto com o Kernel, fazem o sistema completo e amigável para um usuário qualquer.

É aí que entram os Shell (ambientes onde o usuário pode comandar o sistema através de comandos de texto), as interfaces gráficas (ambientes que apresentam ícones e janelas, como o Windows), os aplicativos (para digitar textos, construir planilhas, desenhar e acessar a Internet, por exemplo) e outros mais.

Muitas empresas e programadores obtêm o Kernel do Linux e juntam a ele outros programas que julgam importantes, como aplicativos de escritório e desenho e até mesmo jogos. Cada uma dessas mesmas pessoas ou instituições relança o Linux com seu próprio nome, ou com algum “pseudônimo”. Esses variados “sabores” de Linux são as **Distribuições Linux**.

Distribuições do Linux - Linux Para Todos os Gostos

Como foi dito, o Linux é basicamente seu Kernel. mas aquilo que nós, usuários, utilizamos no Linux é mais que isso, com certeza!

Como vimos ainda, o Kernel e os demais programas que formam o Linux são livres e, na maioria dos casos, open-source (sim, nem todos os softwares livres são open-source) e, por causa disso, podem ser adquiridos e modificados da maneira como os distribuidores querem.

Um **distribuidor** é uma pessoa ou instituição que pega o Kernel do Linux, une esse programa a outros, criados por ele ou por outrem, e “encaixota” o resultado, dando-lhe nome e oferecendo suporte a ele (ou seja, responsabilizando-se pela obra), criando uma nova **Distribuição do Linux**.

Note que diversas distribuições são semelhantes entre si, afinal, têm o mesmo centro, e, muitas vezes, os mesmos programas auxiliares, como aplicativos de escritório e jogos, portanto, a escolha por essa ou aquela distribuição é um processo pessoal e vai mais pelo gosto do usuário (eu mesmo uso duas: o Conectiva Linux 10 e o Slackware 10.1).

Seguem algumas das principais distribuições do Linux (mas lembre-se: são basicamente a mesma coisa, porque têm se baseiam num único centro: o Kernel):

- **Conectiva Linux:** é a distribuição da empresa brasileira Conectiva. Um dos mais amigáveis Linux para o Brasil, apresenta uma interface de instalação muito boa (ou seja, ele é fácil de instalar!). Atualmente (Maio de 2005), está na versão 10, usando o Kernel 2.6. Lembre-se de que versões anteriores do Conectiva usavam versões anteriores do Kernel, claro! o Conectiva pode ser usado tanto em casa como em servidores.
- **Red Hat:** Uma distro (“distribuição”, para os íntimos) americana que recentemente deixou de ser distribuída gratuitamente. A empresa Red Hat simplesmente fornece seu Linux para servidores de rede, não mais para usuários de computadores (e não se pode mais pegar essa distro na Internet de graça!). A última versão gratuita foi a 9, usando o Kernel 2.4.
- **Slackware:** considerada por muitos (os especialistas, normalmente) como a melhor distro de todas, por ser a mais estável (a última versão do “Slack”, a 10.1, por exemplo, utiliza o Kernel 2.4, ainda, que, segundo eles, é mais confiável que o 2.6). O pessoal que mantém o Slack é muito tradicionalista e sempre pregou a criação de uma distro muito enxuta, sem firulas. O Slack é um dos mais difíceis de instalar e de configurar, além disso, traz poucos programas consigo, portanto, é mais recomendado para servidores. Essa é para experts!
- **Suse Linux:** uma distro alemã, também muito famosa e gostosa de usar. Traz diversos programas para usuários finais (como programas de escritório, por exemplo).
- **Mandrake Linux:** também muito fácil de usar, dando preferência aos usuários finais, o pessoal da Mandrake coloca sempre muitos recursos bons para que o Mandrake Linux possa ser usado em casa por qualquer usuário.
- **Fedora Core:** é o projeto de distro gratuita da empresa Red Hat (para não saírem mal na foto com a comunidade Linux, eles – da Red Hat – mantiveram um projeto com ela – a comunidade – de atualização desta distro). É muito completa, cheia de recursos para servidores e usuários finais.
- **Debian:** uma distribuição muito boa de usar (para experts também). O pessoal que usa e mantém o Debian é “o outro extremo da linha” do pessoal do Slack, há uma certa “rivalidade” entre eles.

Acho que já deu para conhecer algumas das principais distribuições do Linux, embora haja muitas outras que podem ser escolhidas! Algumas, inclusive, podem servir de “estágio” na transição Windows/Linux pois podem ser executadas diretamente do CD, sem a necessidade de se instalar o sistema no micro, o que poupa muitas dores de cabeça da maioria dos usuários. Um excelente exemplo de Linux que executa direto no CD é o **Kurumin**, do Carlos Morimoto (www.guiadohardware.net).

Pode ser também, leitor, que algumas dessas distros sejam abandonadas em algum ponto, e outras nasçam com o tempo (a exemplo do **Mandriva**, que é resultado da fusão das distros **Mandrake** e **Conectiva**). O mundo Linux é assim: completamente nômade e mutante (mas fiel às suas origens e seus ideais, pelo menos, até agora!).

Resumindo, caro leitor: Todas as distribuições do Linux são iguais? A resposta é **Não!** Há pequenas diferenças entre elas, mas nada que impossibilite o aprendizado delas, afinal, estamos falando do mesmo produto (o Linux), embalado por várias empresas diferentes (como uma “Torta Floresta Negra” feita por vários restaurantes ou confeitarias diferentes: a torta é a mesma, pois se baseia na mesma receita, mas que dá para sentir diferenças pequenas no sabor de cada uma, dá sim!).

Questões a Considerar Quanto ao Linux

Depois de tudo o que foi apresentado, aparentemente estaremos vivenciando um processo de transição que não tem precedentes no mundo da informática e que não mostra sintomas de que será acalmado ou, muito menos, impedido! Mas mesmo sendo

essa transição tão “inevitável”, há certos aspectos a serem analisados de forma mais crítica.

Vamos dar uma olhada em certas questões relevantes ao movimento de mudança Windows/Linux:

- **O custo da implantação e da manutenção do Linux é muito baixo:** essa, não por acaso, é a mais gritante das características favoráveis ao novato. Por tudo o que vem sendo discutido aqui, é bastante compreensível o porquê do uso do Linux ser tão barato e ainda o porquê do Windows não conseguir, nem de longe, oferecer qualquer resistência a ele nesse aspecto.
Essa questão é muito importante porque permite a empresas (e até mesmo países) em desenvolvimento ou recuperação conseguirem atingir estabilidade e auto-suficiência sem estar sob o jugo da Microsoft e de outras detentoras de copyright de softwares! Ou seja, o Linux permitirá uma verdadeira revolução tecnológica e mesmo econômica para diversas instituições e até nações.
- **O Windows é mais conhecido e mais usado:** Nesse ponto, o fato de a maioria utilizar o Windows em suas casas e nas empresas dá um ponto ao veterano sistema, pois, a esse fato aparentemente banal, está atrelado algo que deve ser pensado pelas empresas: os custos com treinamento de pessoal. **Sim, o Linux pode ser mais caro** durante o início das atividades dele na organização, sendo pública ou privada, porque denotará a necessidade de treinamento dos funcionários! Ainda tem mais: é sabido que, se os funcionários não conseguem utilizar uma determinada tecnologia devidamente, a produtividade vai cair consideravelmente e só será retomada quando a novidade se tornar habitual (ou seja, quando se acostumarem com o Linux).
- **O Plug and “Pray” do Linux:** pois é, para alguns fabricantes de hardware, como impressoras, scanners, web cams e afins, parece que o Linux não existe! Ao comprar um equipamento novo, é muito raro (eu ainda não encontrei um) se deparar com um que traga, no seu CD de drivers, a versão para o sistema do pingüim. Em outras palavras, a maioria dos fabricantes só desenvolve drivers para o Windows, permitindo que seus dispositivos possam ser instalados apenas nesse sistema. O Linux ainda é posto em segundo plano, porque, apenas em alguns poucos casos, o fabricante coloca, no máximo, o driver para Linux em seu site.
- **O Windows é mais “amigável”:** outra coisa que parece incontestável, mas vai de ponto de vista! Como estamos acostumados a lidar com o sistema da Microsoft, então claro que ele nos parecerá mais fácil e intuitivo. O Linux ainda nos apresentará alguns segredos que, quando perfeitamente descobertos, o tornarão tão fácil de usar quanto o Windows.
- **O Linux é mais complicado:** Bom, o outro lado da questão citada acima é esse! O Linux não é complicado, é complexo (e completo). Quero dizer, no Linux, podemos fazer uma operação de várias maneiras, algumas mais fáceis (com mouse e ícones), algumas mais difíceis, através de comandos estranhos e arquivos de configuração assombrados. Isso só dependerá do seu nível de conhecimento e intimidade com o sistema (os experts preferem os arquivos de configuração e os comandos, porque talvez isso os faça parecer mais experts!).
- **Sistema “Cabra Macho”:** o Linux não trava! Pelo menos é isso que se ouve por parte dos entusiastas exacerbados! “O Linux não pega vírus” – dirão eles também! Em ambos os casos, está errado! O Linux trava sim, mas não com a frequência com que o Windows o faz. O Linux também está sujeito às intempéries causadas por vírus de computador, mas numa escala muito inferior à do concorrente! O importante é que o Linux foi feito para não travar, portanto a comunidade Linux vai criar correções para todos os problemas que o sistema apresentar, deixando-o mais seguro e menos propenso a erros, visando manter a integridade do sistema e a confiança que todos depositam nele.
Refazendo o parágrafo anterior: eu já enfrentei travamentos do sistema Linux, mas

nos dois casos, o culpado não foi o Kernel (o sistema em si), mas algum programinha que estivesse em execução causando a instabilidade do computador e impedindo o Linux de resolver a questão (e isso já faz alguns anos! Ultimamente, não tenho sido testemunha de nenhum mal estar do sistema).

Espero que essa apresentação sucinta do sistema operacional Linux tenha aberto sua mente para compreendê-lo e, assim como eu, gostar dele e utilizá-lo! Sei que ainda faltam características importantes para que ele se torne um sistema tão facilmente utilizável e perfeito para ser usado em casa ou no trabalho, mas com o tempo, e a cada versão do Kernel ou nova distribuição, vejo as evoluções aparecerem, o que prova que a Comunidade está trabalhando duro em seu filhote! Vamos lá! Aprenda a criar pingüim você também!

Primeiros Passos com o Linux

Ligar um computador e ver mensagens e figuras diferentes daquelas com que estamos acostumados quando iniciamos o Windows pode ser uma experiência traumática! Esse medo, por parte dos ex-usuários do Windows, acontece todas as vezes que o Linux é iniciado (e, especialmente, na estréia!).

Lembro-me que, quando o fiz pela primeira vez, fiquei assustado e, quando finalmente a inicialização terminou, me senti completamente perdido naquela tela preta que aguardava ansiosamente por meus comandos. Não se sinta mal: é frustrante não saber o que fazer diante de um computador que você julgava perfeitamente subordinado a você.

Os contatos iniciais com o Linux podem parecer muito difíceis (e realmente o serão se o usuário não estiver preparado para entender o sistema e suas principais facetas). Saber o que há no Linux e como se faz para as coisas funcionarem é um pré-requisito para o sucesso do usuário nesse novo ambiente (eu costumo chamar de "novo universo").

Conceitos Gerais

Para utilizar o Linux, não é necessário nenhum conhecimento prévio em Windows ou qualquer outro sistema operacional, mas, é claro que se o usuário que pretende usar o Linux já entende conceitos de outros programas, as comparações serão um excelente modo de estudo (que, por sinal, usarei em demasia no decorrer desse livro).

Alguns dos principais pontos a serem discutidos no Linux são:

- **O Linux é um sistema multiusuário:** O que significa que várias pessoas podem utilizar o Linux em um computador (inclusive ao mesmo tempo, mas eu explico isso depois). Cada usuário é reconhecido pelo sistema quando inicia suas atividades mediante a apresentação de um nome e uma senha (previamente cadastrados). Isso significa que será necessário, todas as vezes que um usuário for utilizar o computador, que ele realize o processo de **Logon**. O Logon consiste na apresentação do **Login** (nome cadastrado no sistema para o usuário) e da **Password** (senha).
- **O Linux pode ser utilizado graficamente:** quer dizer que o sistema Linux pode se apresentar para o usuário do mesmo modo amigável com que o Windows se mostra. O Linux tem ambientes gráficos, e muitos! Claro que o normal, para os usuários experts, é preferirem o Linux com sua interface básica: texto! Tela preta, letras brancas e uma série de comandos diferentes decorados sofridamente! Aqui vai um lembrete para os usuários mais céticos e amedrontados: **O Linux usa mouse e ícones; janelas e menus, como o Windows, e isso facilita o aprendizado.**
- **Algumas coisas no Linux são mais difíceis de fazer:** Isso, é claro, pode até ser relacionado com o fato de usarmos mais o sistema da Microsoft, mas não é bem assim! O Linux complica certas coisas sim! Esse é o preço que se paga pelo direito de ter o controle total sobre o sistema operacional.

Conhecendo o Super Usuário - root

Como já foi citado, o Linux admite a existência de diversos usuários. Os cadastros dos usuários que o sistema possui são feitos em registros exclusivos chamados contas (ou **contas de usuário**). Então, se você pretende usar o Linux, deve possuir uma conta cadastrada no sistema.

Essa conta consiste, entre outras informações, no Login (o seu nome perante o sistema) e a senha (seqüência de caracteres secreta). As contas também definem os privilégios de acesso que o usuário tem no sistema, como por exemplo, se ele vai poder alterar um determinado arquivo, ou se só vai poder lê-lo.

Há várias formas de criar contas de usuário no Linux depois que o sistema está em funcionamento (essas formas serão vistas depois), mas uma conta é criada quando o Linux é instalado no computador, a conta da pessoa que tem direito a fazer qualquer coisa no sistema: o **Administrador** ou **Super Usuário**.

O Super Usuário é o "cara", simplesmente! Ele pode tudo! Se você é o super usuário de sua máquina, você é o dono, o manda-chuva dela. O Login cadastrado para a conta do Administrador é: **root**. Ou seja, para ser reconhecido como super usuário do sistema Linux, é necessário, na inicialização do sistema, que o usuário digite **root** e a senha apropriada.

Se o seu caso é diferente, como por exemplo: você solicitou a alguém (um técnico) que instalasse o Linux em seu computador, depois de realizar a tarefa (e ser pago por isso), o técnico entrega a máquina a você com um papel junto: usuário: **fulano** e senha: **1234**. Isso significa que para poder acessar o sistema, você deverá apresentar essas informações toda vez que o micro for ligado.

"Certo, mas e daí?" Simples: Você não é o root! O técnico, provavelmente, criou a senha para o root, que só ele sabe, e criou uma conta de usuário para você poder utilizar inocentemente o computador. **Quem instala o Linux define a senha do root, porque esta é uma das exigências feitas durante o processo de instalação do sistema.**

Trabalhar com o Linux não é privilégio do usuário que detém a senha do root, mas este usuário poderá fazer qualquer coisa, entrar em qualquer lugar, abrir qualquer arquivo e até mesmo apagar ou criar quaisquer outros usuários. Por exemplo, há certos comandos que só podem ser executados pelo root e quando os usuários comuns tentam executá-los, recebem mensagens de erro informando que o acesso não será permitido.

Recomendação ao Administrador do Sistema: se você é o proprietário da conta de super usuário, aqui vai uma dica interessante: **não use a conta de root constantemente** para fazer qualquer coisa (digitar textos, acesso à Internet, jogos). Ao invés disso, crie uma conta de usuário qualquer (sem privilégios administrativos) para poder realizar as tarefas cotidianas.

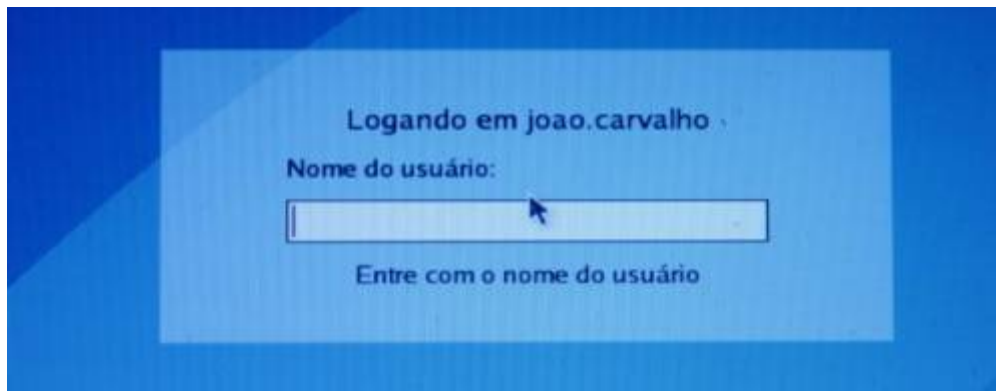
A idéia é que se, durante um acesso à Internet, por exemplo, seu computador for infectado por um vírus o outro programa malicioso, o referido programa será executado em modo root, e terá acesso completo ao sistema (podendo "ferrar" o sistema completamente)! Se, no momento da infecção, você estiver logado como um usuário convencional, os limites de acesso impostos a você pelo próprio Linux serão responsáveis por conter os programas bisbilhoteiros.

Ou seja, não banalize a conta de root, apenas faça uso dela em casos necessários (mudanças de configuração, ajustes do sistema, instalação de programas, etc.).

Entrando no Linux - o Processo de Logon

Quando se inicia o computador com o Linux, depois de alguns procedimentos necessários, uma tela de Logon é apresentada para o usuário. Dependendo de uma série

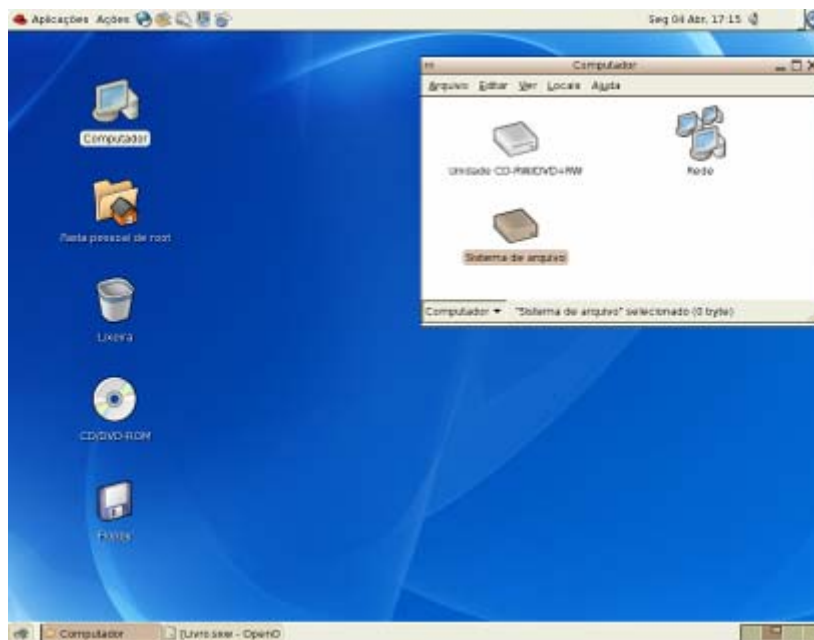
de fatores (incluindo a distribuição utilizada), essa tela pode ser diferente, mas no geral solicita sempre duas informações aos usuários: **Login e Senha**.



Logon é tão-somente o nome dado ao processo de apresentação do Login e da senha, ou seja, logon é "entrar no sistema", "identificar-se". Não confunda, porém, com Login, que é o nome do usuário que está se logando!

Depois de efetuado o logon no sistema, alguns processos são realizados para que Linux apresente sua área de trabalho comum aos usuários (é uma tela muito semelhante à área de trabalho do Windows) e os permita comandar o sistema.

Essa área de trabalho também pode se apresentar de formas diferentes, dependendo da distribuição, mas, mais precisamente, dependendo do ambiente gráfico utilizado pelo usuário (veremos o conceito disso nos próximos tópicos). Veja, abaixo, um exemplo da área de trabalho do Linux (ambiente GNOME) usado no Conectiva Linux 10:



Note algumas semelhanças com o Windows, como a presença de janelas, ícones, menus, ou seja, o Linux não é mais tão assustador assim! É perfeitamente possível migrar da plataforma Windows para a plataforma de software livre! É só querer!

Aproveitando: o termo **plataforma** é muito usado para descrever um "jeito" ou um "ambiente" de trabalho em informática. Então, a plataforma Windows significa o uso ou as características apresentadas pelo Windows e pelos programas que funcionam nele.

Vamos nos prender à utilização destes ambientes gráficos daqui a algumas páginas (não se afobe, preciso apenas mostrar-lhe mais alguns conceitos antes de "soltar" você no Linux!).

Como o Linux Entende as Unidades de Disco

Bom, em primeiro lugar, se você espera ter, no Linux, ícones que ajudem-no a acessar a **Unidade C:, D:, E:** e outras afins, tire isso da cabeça! Aqui, a nomenclatura para as unidades de armazenamento é diferente do Windows. E isso, confie em mim, pode gerar problemas sérios!

Veja, na figura a seguir, uma janela aberta do ícone "Computador", que é comum nos ambientes gráficos atuais que funciona como o manjado "Meu Computador" existente no Windows. Note que não existem as unidades C: ou D: e que há alguns componentes meio estranhos (como o famigerado "Sistema de Arquivo").



Deixe-me tentar explicar em poucas palavras: a forma de nomenclatura dos discos por parte do Linux não se parece, em nada, com a do Windows. Enquanto que no Windows, a estrutura de diretórios (pastas) começa em cada unidade de disco devidamente nomeada (C:, D:, E:, etc.), no Linux, todos os diretórios são subordinados a um grande diretório pai de todos: o **diretório (ou pasta) raiz**, ou **sistema de arquivo** (nessas novas distribuições, essa nomenclatura também tem sido usada).

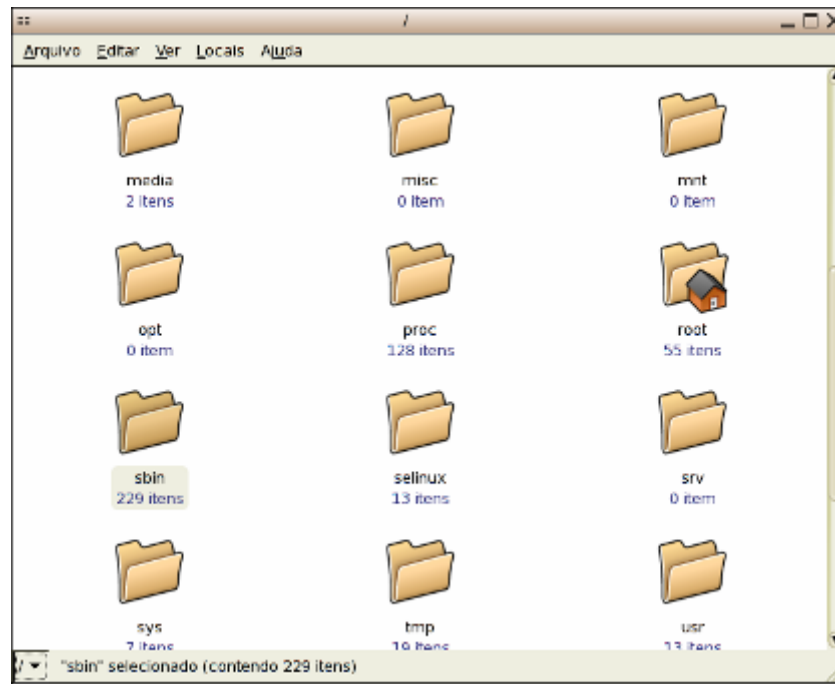
É como se o diretório raiz representasse, simplesmente, o "universo" dentro do sistema Linux. Os demais diretórios estão dentro do sistema de arquivo. Para os mais tradicionalistas e para os comandos usados no sistema Linux, é comum ainda fazer referência a esse diretório principal como **/ (barra)**, simplesmente.

Então fica simples: o Linux não tem unidade C:, nem D:, nem E: ... Mas tem um único e grande repositório de informações que armazena todos os arquivos e diretórios contidos nas unidades de disco (Cds, disquetes, DVDs ainda vão continuar existindo, mas, no Linux, não ganham letras seguidas de dois pontos). Em outras palavras, o diretório raiz, ou sistema de arquivo, ou ainda / (barra) é o "início" de tudo o que está armazenado no computador e a que o Linux tem acesso: tudo, no computador, está dentro do diretório raiz!

Aí, você questiona: "Mas João, estou vendo a unidade de CD-RW/DVD+RW como um ícone separado do Sistema de Arquivo. Como você me diz que ela (a unidade de CD) está dentro do Sistema de Arquivo?"

Acredite em mim! Essa unidade de CD-RW/DVD+RW é apenas um atalho para a verdadeira unidade de CD do computador (que será mostrada como um diretório dentro do diretório raiz).

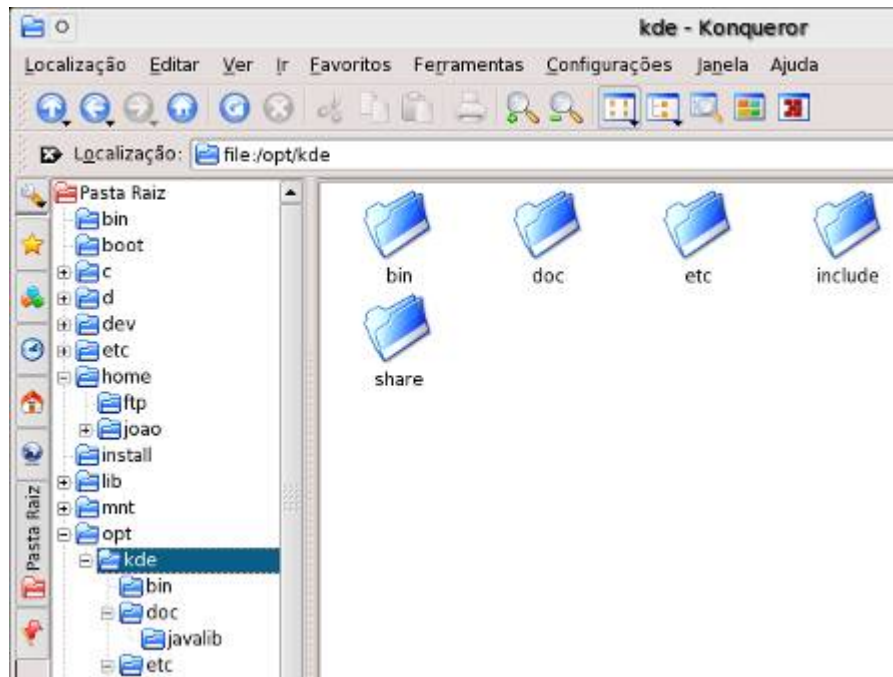
Veja, quando abrimos o ícone **Sistema de Arquivo**, na janela mostrada há pouco, que temos acesso a vários outros diretórios (pastas) e, por sua vez, aos diretórios dentro desses diretórios, como no Windows. Mas é bom lembrar que, quando entramos no ícone **Meu Computador** do Windows, visualizamos as unidades separadas, e aqui elas são subordinadas ao sistema de arquivo.



Note o nome descrito na barra de título da janela (/), que é o nome oficial do diretório raiz. Todas as demais pastas ficam dentro de /, até aquelas que representam discos rígidos diferentes! Sim, os discos rígidos diversos que um micro pode ter são representados por pastas dentro do sistema de arquivo. Veremos essas pastas daqui a pouco.

Só como um exemplo mais fácil de entender, quando se vai salvar algo no Linux, não se define aquele manjado endereço de "C:\pasta\arquivo" (é assim que salvamos no Windows, não é?). OK, no Linux a gente salva um arquivo em **/pasta/arquivo** (porque tudo, no micro, está localizado dentro de "/" - que, no endereço, é a primeira barra, antes do nome da pasta).

Note bem, na figura seguinte, uma foto do **Konqueror** (um programa semelhante ao **Windows Explorer**) navegando pelo conteúdo da pasta **/opt/kde** e note a relação entre essas pastas (**kde** está dentro de **opt**, que, por sua vez, está dentro de **/ - a raiz**). Verifique, também, que, na árvore de pastas (diretórios), o diretório raiz (/) é chamado de **pasta raiz**.



Então, como você pode perceber, copiar e mover arquivos e pastas, organizar o conteúdo do seu computador (seus arquivos de documentos e músicas mp3, por exemplo) não será uma tarefa tão difícil, não acha?

Usar o Linux, do ponto de vista de usuário leigo, se tornou muito mais fácil, porque, hoje em dia tudo está visualmente agradável. A prova disso é que todos os principais recursos e telas que mostramos parecem muito com as janelas no Windows, e isso é graças a programas conhecidos como **Ambientes Gráficos**.

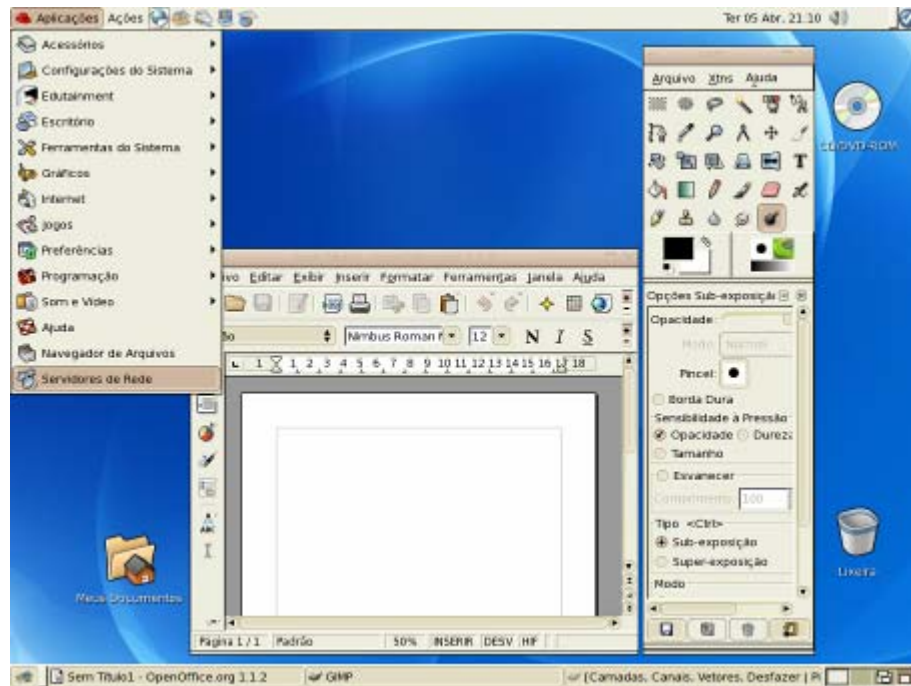
Gerenciadores de Janelas - Os Ambientes Gráficos do Linux

Uma distribuição comum do sistema Linux é formada por uma série de programas, como venho dizendo há algumas páginas. Além do Kernel em si (que é a alma do sistema), temos vários aplicativos de escritório, utilitários de manutenção e até mesmo jogos de diversos estilos. Dentre os programas que acompanham o Linux (saiba que uma distribuição atual pode conter mais de 2000 programas diferentes!), há uma categoria muito especial e bastante necessária para os usuários leigos no sistema: os **Gerenciadores de Janelas** (também conhecidos como **Ambientes Gráficos**).

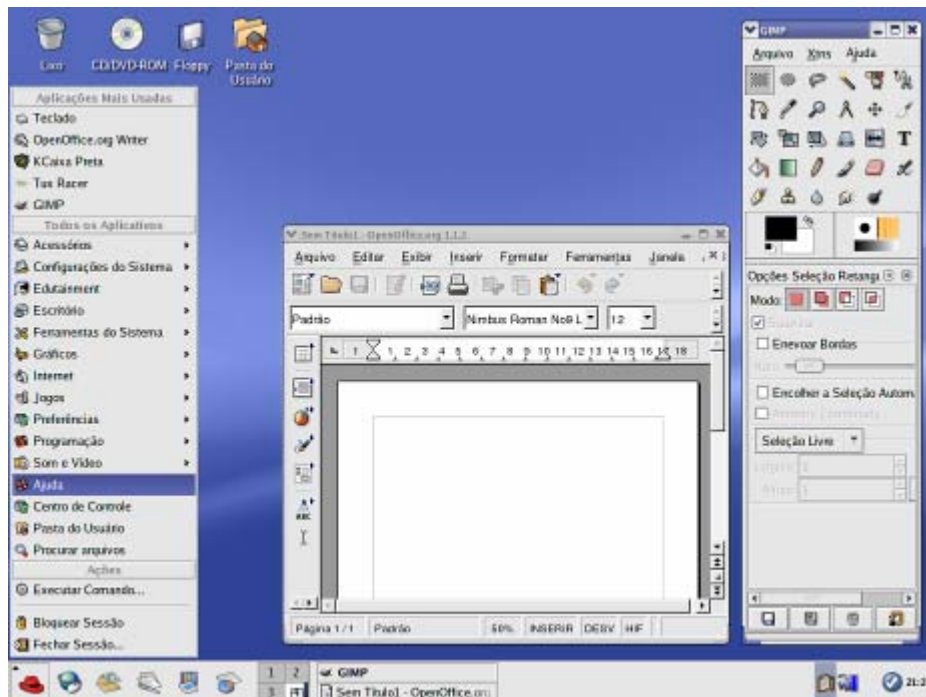
Um Ambiente Gráfico é um programa que permite que o Linux se apresente de forma amigável, como o Windows, através de janelas, ícones, menus, e botões. Um ambiente gráfico é considerado um "programa extra" porque o Linux, naturalmente, não apresenta a "cara bonita" que esses programas criam. Lembre-se de que o Linux é baseado no UNIX, portanto, ele é nativamente **textual** (controlado através de comandos de texto! Tela preta, letras brancas e aquele blá blá blá todo!).

Uma distribuição do Linux pode conter diversos Ambientes Gráficos diferentes, mas os dois mais famosos são, sem dúvida, o **KDE** (K Desktop Environment) e o **Gnome**. A escolha entre um e outro vai simplesmente de decisão pessoal porque ambos são excelentes e a maioria (para não dizer todos) dos programas que funcionam em um também funcionam no outro!

Na figura abaixo é mostrado o ambiente Gnome com dois programas abertos: o **GIMP** (um programa de edição de imagens que chega a se comparar ao Photoshop da Adobe) e o **OpenOffice.org Writer**, a versão open-source do Microsoft Word.



O outro conhecido Gerenciador de Janelas é chamado KDE, é normalmente é mais bonito que o Gnome (novamente, isso é questão de gosto e gosto não se discute!). Note que é possível utilizar os mesmos programas que são utilizados no Gnome, e também é possível acessar os mesmos recursos (como diretórios e arquivos, por exemplo).



A escolha do Ambiente Gráfico que será utilizado acontece no momento no Logon: na parte inferior da tela de logon, normalmente, há um botão chamado **Sessão** que permitirá ao usuário escolher se deseja utilizar o KDE ou o Gnome (ou qualquer outro que esteja instalado no Linux).

Os aplicativos (programas com funções definidas na resolução de problemas dos usuários), em sua maioria, são executados sobre as interfaces gráficas, ou seja: grande parte dos programas que iremos utilizar no Linux é apresentada em formato de janela, portanto, necessita de um ambiente de janelas funcionando.

Por exemplo, o programa OpenOffice.org Writer (ou simplesmente Writer), mostrado nas figuras anteriores, só é executado (passa a funcionar) se um ambiente gráfico (como o KDE ou o Gnome) já estiver em execução. E isso vale para qualquer outro aplicativo dos que se podem acessar pelos menus dos ambientes.

Aviso: Como a liberdade é algo inerente ao uso do Linux, eu não me prenderei a uma interface gráfica apenas, ao invés disso, serão mostradas figuras com fotos ora do KDE, ora do Gnome, mas, devidamente informadas e explicadas.

A inicialização do Sistema Linux

Como todo sistema operacional, o Linux tem a responsabilidade de, entre outras coisas, controlar a inicialização do sistema, realizando todas as tarefas necessárias para que o sistema esteja completamente apto a responder às requisições do usuários quando este assumir o controle da máquina. Muitas operações são realizadas antes de o usuário poder executar qualquer comando ou abrir qualquer janela. Resolvi, então, listar alguns passos importantes desse complexo conjunto de acontecimentos:

- 1) O Computador é ligado, passando a receber alimentação elétrica. Nesse estágio, a memória RAM (principal) está vazia (sem conteúdo) e os programas e arquivos do usuários estão armazenados nos discos (memórias auxiliares).
- 2) O BIOS (Sistema Básico de Entrada e Saída), que é um pequeno programa armazenado numa memória ROM (memória que não necessita de eletricidade) na placa-mãe do computador, vai ser acordado para acordar o restante do computador, especialmente, seu Sistema Operacional. O BIOS é o primeiro programa executado por um computador e possui um papel importante: localizar e executar o Sistema Operacional do computador (seja o Linux ou outro qualquer).
- 3) Depois de encontrar o Sistema Operacional (que estava no Disco Rígido), e iniciar seu processo de carregamento (carregar = jogar na memória principal – RAM), o BIOS entrega a responsabilidade ao Sistema Operacional. É nesse ponto, no caso do Linux, que o Kernel é jogado na RAM e o Linux é efetivamente iniciado.
- 4) Depois de iniciados o Kernel e outros componentes do Linux, o sistema solicita ao usuário as informações de Logon (Nome de usuário e Senha). Nesse ponto, também, é possível escolher como o Linux vai se apresentar, se em modo texto (shell) ou em modo gráfico (através de alguma interface gráfica, como o KDE ou o Gnome). Como nós usaremos, por enquanto, o Linux em modo gráfico, seguiremos os passos seguintes a essa escolha.
- 5) O Gerenciador de Janelas é iniciado, realizando, inicialmente, alguns procedimentos necessários à sua utilização, para, enfim, entregar ao usuário o controle do computador, permitindo que ele execute qualquer programa disponível no sistema (desde, é claro, que seus privilégios de acesso permitam).

Resumindo Até Aqui

- O Linux é um sistema operacional, portanto, tem a obrigação de controlar o computador e mantê-lo funcionando, enquanto recebe nossos comandos e os repassa à máquina. Só podemos utilizar programas de texto e acessar a Internet porque nossos computadores possuem sistemas operacionais.
- Para usarmos o Linux, devemos nos identificar, informando um login (que pode ser **ana, maria, joao, pedro, adm, financeiro, root**) e uma senha. O login **root** permite, ao seu detentor, o controle total do sistema Linux (ele pode fazer o que quiser!).
- O Linux pode ser adquirido de várias formas (inclusive comprando-o), através de vários “fornecedores” e com vários “nomes” diferentes. Esses vários tipos de Linux são

chamados de **Distribuições** e podem apresentar pequenas diferenças entre si, mas todos são Linux, porque se baseiam em um mesmo “alicerce” (o Kernel).

- No Linux, não há unidades de disco separadas, como C:, D: e afins. No Linux, todo o armazenamento de arquivos é feito dentro do diretório raiz: **a barra (/)**. Inclusive, se um computador possuir vários discos rígidos, todos eles serão representados como diretórios dentro do diretório raiz.
- Pode-se trabalhar com o Linux de várias formas: a mais agradável é através de **ambientes gráficos**, presentes em quase todas as distribuições. A forma mais tradicional é utilizar um **Shell** (interface textual – recebe comandos de texto apenas), mas isso exige do usuário o conhecimento em vários comandos diferentes.
- O Linux está sempre em evolução (a cada semana, lança-se uma nova versão menor do Kernel e, em alguns meses, as novas versões maiores são lançadas), o que garante que erros são corrigidos com certa rapidez.
- Se você estiver impaciente quanto a essas mudanças, **faça você mesmo a sua atualização!** Afinal, você também possui o código-fonte e, até mesmo pode baixar um mais novo da Internet.

USANDO O LINUX

Antes de podermos “meter a mão na massa” no Linux, caro leitor, sinto necessidade de apresentar para você alguns conceitos mais aprofundados sobre o sistema. Dentre esses conceitos, vamos falar de como o Linux interpreta arquivos, o que são arquivos executáveis (ou melhor, “como” eles são), qual é a estrutura de diretórios do Linux (como os diretórios do Linux estão organizados), a noção de permissões de acesso e algumas outras informações.

Depois que você ficar conhecendo esses conceitos apropriadamente, partiremos para o uso do sistema, mas saiba que, em alguns casos, iremos voltar a esses conceitos para aprendê-los de forma mais aprofundada pois, nesse capítulo, esses conceitos são apresentados apenas superficialmente (ou quase).

Como o Linux Interpreta Arquivos

Um **arquivo** é qualquer conjunto sólido de informações gravado em uma unidade de armazenamento (memória auxiliar, como um disco rígido ou um CD, por exemplo). Normalmente, um arquivo é criado pela execução do comando **Salvar**, comum em tantos programas aplicativos.

Então, em outras palavras, quando você digita algo em um programa de texto, por exemplo, e salva, está criando um **Arquivo**. Mais precisamente, está criando um **Arquivo de Dados**.

Arquivos, no Linux, são divididos em alguns tipos, como os que seguem:

- **Arquivos Comuns:** podem ser subdivididos em:
 - Arquivos de Dados: contém dados de diversos tipos, os maiores exemplos são os arquivos que manipulamos: textos, documentos, planilhas, figuras, fotos, MP3, etc.
 - Arquivo de texto ASCII: é um tipo específico de **Arquivo de Dados**, escritos por programas editores de texto. São arquivos muito simples e só contém texto (caracteres). Esses arquivos não admitem outro tipo de dado, como figuras ou tabelas. Não são possíveis nem mesmo as formatações normais (negrito, itálico e sublinhado). **Um arquivo do Word, por exemplo, não é um arquivo de texto ASCII.**

- Arquivos de Shell Script: são arquivos escritos como textos ASCII, ou seja, em programas editores de texto. Seu conteúdo é formado por comandos que o Linux consegue interpretar. Esses arquivos são como “roteiros” com várias instruções que o Linux vai executar.
- Arquivos binários (executáveis): são arquivos escritos em linguagem de máquina (zeros e uns) que podem ser executados pela CPU do computador. Esses arquivos são, na verdade, chamados de programas ou arquivos executáveis. Eles não são escritos para serem lidos pelo usuário, eles são criados para serem compreendidos pelo Linux e executados por ele. Para criar tais arquivos, deve-se escrever um programa em alguma linguagem (como C, por exemplo) e compilá-lo a fim de que se transforme no arquivo binário.
- **Diretórios:** Sim, os diretórios (pastas) são considerados arquivos no Linux. O sistema entende que um diretório é um arquivo especial, que tem em seu conteúdo um apontador para todos os arquivos que se mostram “dentro” do diretório. A idéia é a mesma de uma pasta no Windows: ou seja, um diretório é uma “gaveta” onde podemos colocar outros arquivos (inclusive outras pastas).
- **Links (Vínculos):** uma idéia similar à dos atalhos no Windows. Um link é um arquivo que aponta para um outro arquivo qualquer (de qualquer tipo, inclusive diretório). Um link pode apontar, inclusive, para outro link. Exemplo: se há um arquivo chamado **teste.doc** dentro de **/documentos/antigos**, você poderá criar um link para ele na pasta raiz, com o nome de **teste**. Quando você quiser fazer referência ao arquivo, pode-se informar ao programa **/teste** ou **/documentos/antigos/teste.doc** que vai dar no mesmo!

Veja, na figura abaixo, um exemplo de cada tipo de arquivo (os ícones diferem do Windows, claro – eu acho até mais bonitos!). Conheceremos mais sobre esse assunto em tópicos posteriores, não se preocupe!



Só explicando: **Captura de Telas** e **Documentos** são diretórios (pastas) mas, eu acho que você já havia notado; **figura.jpg** é um arquivo de dados (mais precisamente, uma foto); o arquivo **bzip2** é um arquivo executável (binário); **texto** e **velox.sh** são arquivos ASCII (texto puro), a diferença é que o segundo é um Shell Script, ou seja, é composto de vários comandos que serão interpretados pelo shell do Linux; e, finalmente, o arquivo **teste1** é um link (atalho).

Você, então, imagina: “O que vai acontecer quando cada um for aberto (duplo clique)?”.

Depende do tipo: um arquivo de dados (seja ele ASCII ou não) normalmente, quando recebe o duplo clique que solicita sua abertura, faz o Linux chamar o programa que é capaz de abri-lo. É fácil de entender: no Windows, quando nós damos um clique duplo

num arquivo do Word, o que acontece? Isso! O Word é aberto para poder abrir o arquivo que o usuário executou!

Quando o arquivo for um binário, o Linux jogará seu conteúdo na memória principal e começará a executar os comandos existentes nele (afinal, um binário é um programa compilado – um executável em linguagem de máquina).

Se o arquivo for um Shell Script, ou outro script qualquer (existem vários), o Linux se encarregará de “ler” e interpretar seu conteúdo (lembre-se: scripts são roteiros cheios de comandos).

Finalmente, ao se aplicar duplo clique em um link, ele vai apontar para o arquivo original e é o tipo desse arquivo original que definirá o comportamento do Linux após a execução.

Opa! Quase ia me esquecendo delas: se o clique duplo for dado numa pasta, ela será aberta diretamente pelo programa gerenciador de arquivos (no caso da foto, o *konqueror*, do KDE), exatamente como acontece no Windows Explorer.

Nomenclatura dos Arquivos

Uma grande diferença entre o Windows e o Linux é o jeito como cada um trata os seus arquivos no que se refere às regras de nomenclatura deles em cada sistema. Ou seja, nem sempre o que aprendemos e usamos para salvar arquivos no Windows pode ser usado no Linux e vice versa.

Vamos tomar como exemplo um arquivo qualquer do Windows: **carta.doc**. Esse arquivo é, sem dúvida, um arquivo de **Documento do Word**, que será aberto pelo programa **Microsoft Word**. Como eu sei disso? Pelo “sobrenome” do arquivo: a sua extensão (doc). Uma primeira regra simples é: mesmo não sendo necessária hoje, a extensão é utilizada pelo Windows para reconhecer o arquivo, definindo quem irá abri-lo!

Em outras palavras, o arquivo acima citado poderia se chamar somente **carta**, mas isso não deixaria o Windows classificar-lhe como documento do Word. No sistema Windows, os tipos de arquivos são identificados por sua extensão. Nesse sistema, são comuns nomes de arquivos bem definidos, com apenas um ponto entre seu nome e sua extensão, como em: **orçamento.xls**, **carta.doc**, **filmagem.mpg**, **What a Wonderful World.mp3** e assim por diante.

No Linux, realmente não há necessidade de extensão para os arquivos (binários, dados, links, diretórios) existirem e serem identificados como tal. Um arquivo é normalmente identificado pelo seu conteúdo, ou seja, mesmo que um arquivo se chame somente **texto** (como o arquivo mostrado na figura 2.1), ele será identificado como um arquivo de texto puro (ASCII): note o ícone que foi dado a ele!

Não estou dizendo que no Linux não são usadas extensões, porque são sim! Estou dizendo que, para diferenciar os tipos de arquivos entre si, o Linux, na maioria das vezes, não precisa da extensão porque analisa o conteúdo do arquivo para definir seu tipo. Imagine dois arquivos de imagem (são arquivos de dados): uma foto JPEG (JPG) e uma imagem GIF. Mesmo que você não ponha extensões neles, o Linux será capaz de identificá-los por seus conteúdos (porque cada arquivo tem uma espécie de “assinatura” no sistema).

Vale salientar que os ícones não são dados pelo Linux em si, mas pelo ambiente de janelas (KDE, Gnome ou outro qualquer), portanto, se você está usando um ambiente gráfico diferente do meu (estou usando o KDE), os ícones podem ser diferentes dos apresentados nesta janela, e podem até mesmo ser iguais entre si.

Outra coisinha interessante sobre os nomes dos arquivos é que nem sempre há apenas um ponto no nome. É simples entender: como não há essa rigidez toda quanto às extensões, não há obrigatoriedade de identificá-las com um ponto, portanto, o ponto é

um caractere perfeitamente utilizável no nome do arquivo. Claro que a seqüência de caracteres que sucede o **último ponto** é considerada a extensão oficial do arquivo.

É comum encontrar, no Linux, arquivos com esses tipos de nome (vê que loucura!):

ethereal-0.10.10-i486-2jim.tar.gz

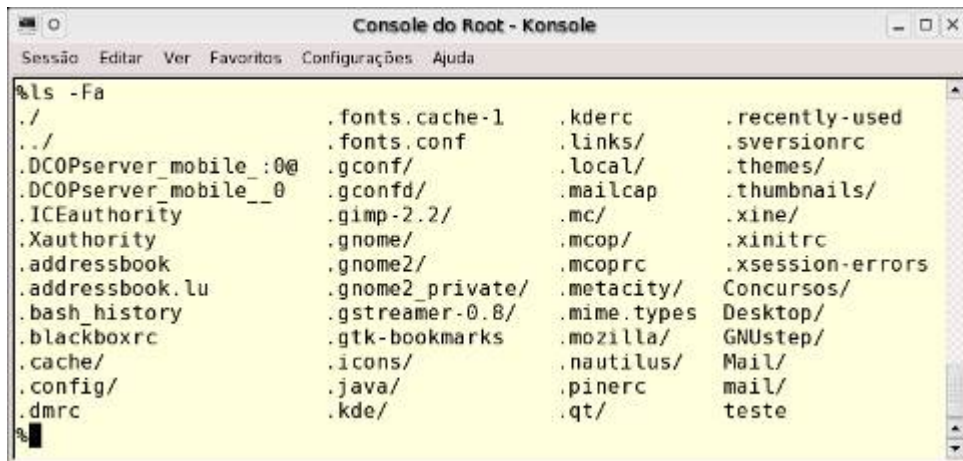
Ooo_1.1.3_LinuxIntel_install.pt-br.rpm

Há ainda necessidade de se comentar algumas regras básicas na nomenclatura de arquivos. Regras que, inclusive, até o Windows tem, embora com algumas diferenças:

- Os nomes de arquivos podem ter até 255 caracteres (igual ao Windows).
- São aceitos espaços no nome dos arquivos (igual ao Windows).
- Praticamente **todos os caracteres** podem ser usados em nomes de arquivos (incluindo alguns dos que o Windows julga proibidos, como *, ?...).
- Não pode haver dois ou mais arquivos com o mesmo nome dentro da mesma pasta (igual ao Windows).
- O Linux possui um sistema de arquivos **Case-Sensitive**, ou seja, ele diferencia maiúsculas de minúsculas. Sendo assim, os arquivos **Casa**, **CASA**, **casa** e **cASA** possuem nomes diferentes (para o Windows, não há essa diferença: todos os nomes listados acima são iguais!). Normalmente, no Linux, prefere-se criar arquivos com **letras minúsculas apenas**.
- Arquivos ocultos, no Linux, têm seus nomes iniciados com um . (ponto). Em outras palavras, todos os arquivos que apresentarem seus nomes começando com um ponto (como em **.profile**, ou **.segredos**), são considerados ocultos (não aparecem nas janelas comuns do gerenciador de arquivos).

Note a presença dos arquivos ocultos nos diretórios do Linux: na imagem a seguir, podemos ver, na pasta **/root**, a existência de apenas 5 pastas e 1 arquivo. Quando executamos o comando que permite visualizar os arquivos ocultos (calma, conheceremos ele), aí conseguimos constatar a presença de mais arquivos (todos precedidos de "." - ponto).





```

%ls -Fa
./                .fontconfig-cache-1  .kderc            .recently-used
./                .fontconfig.conf     .links/           .sversionrc
.DCOPserver_mobile_0@ .gconf/              .local/           .themes/
.DCOPserver_mobile__0 .gconfd/             .mailcap          .thumbnails/
.ICEauthority      .gimp-2.2/           .mc/              .xine/
.Xauthority         .gnome/              .mcp/             .xinitrc
.addressbook        .gnome2/             .mcp/             .xsession-errors
.addressbook.lu     .gnome2_private/    .metacity/        Concursos/
.bash_history       .gstreamer-0.8/     .mime.types       Desktop/
.blackboxrc         .gtk-bookmarks       .mozilla/         GNUstep/
.cache/             .icons/              .nautilus/        Mail/
.config/            .java/               .pinerc           mail/
.dmrc               .kde/                .qt/              teste

```

Grupos de Usuários

Vimos que, para se conectar ao sistema Linux, podendo utilizar seus recursos, um usuário precisa ter uma conta (Login e Senha). Vimos também que uma dessas contas é especial porque fornece, a seu detentor, o direito de fazer qualquer coisa no computador (**root**). Mas ainda faltou falar sobre uma coisinha nas contas de usuários: os grupos.

Um **Grupo de Usuários**, ou somente grupo, é, como o nome já diz, um conjunto de usuários. Um grupo é bom para reunir vários usuários e atribuir a ele (o grupo) certos privilégios de acesso. Quem estiver dentro do grupo vai ter acesso aos recursos que o forem fornecidos ao grupo todo.

Exemplo: o usuário **root** pertence a um grupo chamado, adivinha... **root**. Os demais usuários, como **joao**, **ana**, **pedro**, **paula** podem pertencer a outros diversos grupos, como **financeiro**, **rh**, e assim por diante. Quando se estipulam, para um grupo, privilégios de acesso a um determinado recurso, todos os usuários daquele grupo automaticamente receberão aqueles direitos sobre o recurso.

Quando se cadastra uma nova conta de usuário no Linux, é possível definir em que grupos aquele usuário vai ser inserido (Sim, um usuário pode pertencer a mais de um grupo simultaneamente). Vamos ver posteriormente como fazer para adicionar novos usuários e novos grupos, não se "aveche" (apresse).

Permissões dos Arquivos

Entre os vários recursos que tornam o Linux um sistema seguro está a rigidez dele no tocante às permissões que um usuário tem de utilizar um determinado arquivo. Os sistemas Windows domésticos (95, 98, ME e XP Home) não chegam nem perto do que o Linux pode fazer para proteger os arquivos de um usuário.

Vejamos: imaginemos que os arquivos mostrados na figura abaixo foram criados por um usuário chamado **joao** que pertence ao grupo **contab**. Note, pela figura, que os arquivos em questão estão localizados na pasta **/empresa/documentos/geral**.



Supondo que o usuário clique com o botão secundário do mouse (normalmente o direito) em um dos arquivos mostrados acima (por exemplo, o arquivo **orcamento**), será apresentado um menu de opções onde haverá **propriedades** como uma delas. Ao clicar em propriedades, o usuário obterá a seguinte janela:



Note logo que há 3 níveis de permissões: um para o **Dono**, um para o **Grupo** e um para **Outros**, mas o que é isso?

- **Dono do Arquivo:** é, normalmente, como o nome já diz, o usuário que criou o arquivo. Eu disse “normalmente” porque um arquivo pode ter seu dono atribuído posteriormente (o título de dono do arquivo é transferível a outros usuários). Note que o dono do arquivo é o usuário **joao**, como se pode ver na parte inferior da figura acima.
- **Grupo:** descreve o grupo de usuários ao qual o dono do arquivo pertence. O arquivo mostrado na figura pertence ao usuário joao, que está cadastrado no grupo **contab**.
- **Outros:** descreve os privilégios de acesso dos outros usuários do computador (aqueles que não pertencem ao grupo **contab**).

Note também que há permissões para 3 tipos de operações: **Escrever**, **Ler** e **Executar**.

- **Escrever:** esse privilégio permite, ao seu detentor, modificar o conteúdo de um arquivo (salvá-lo). Normalmente, esse direito está atrelado ao direito de ler o arquivo (porque, na maioria dos casos, modificar o arquivo requer que se abra ele primeiro).

No caso da figura acima, apenas o dono do arquivo (**joao**) poderá alterar seu conteúdo.

- **Ler:** permite que o detentor desse privilégio possa apenas ler o conteúdo de um arquivo, sem poder alterá-lo (salvar). Nem vem! Se você não tiver acesso a um arquivo para modificá-lo, nada feito, o Linux não deixa mesmo! O grupo **contab** foi agraciado com o direito de ler o arquivo orcamento, segundo as informações mostradas na figura acima.
- **Executar:** define que o arquivo em questão poderá ser executado como um programa qualquer pelo usuário. Como já vimos, no Linux, vários arquivos são considerados executáveis, como os binários e os shell scripts. Para que o sistema os possa executar quando o usuário pedir, é necessário que este (o usuário) tenha privilégio para executar o arquivo. O arquivo orcamento, na figura acima, **não é executável!!!**

Note que, aos usuários que não pertencem ao grupo do usuário joao (grupo contab), não foi dada nenhuma permissão ao arquivo mostrado na figura acima, ou seja, com o arquivo **orcamento**, os demais usuários do computador **não podem fazer nada (nem ler)!**

Essas permissões podem ser alteradas nesta mesma janela, bem como através de comandos do sistema (conhecemos os comandos logo logo). Veja a mesma janela com direitos de leitura, escrita e execução dados a todos os usuários do computador (“todo mundo pode tudo”). O dono do arquivo e o grupo também foram alterados.



Ainda há um alerta a se fazer: não adianta atribuir o privilégio de execução para um arquivo qualquer de dados. Um arquivo só vai ser executado se ele possuir um conteúdo que o permita isso (um binário ou um script). Arquivos de dados comuns, como uma foto ou uma música mp3, não se beneficiam do privilégio de execução porque o Linux não vai conseguir executá-los mesmo que o privilégio esteja ativado.

Outro Lembrete oportuno: Mesmo que um usuário qualquer defina limites de acesso aos demais usuários do computador para um determinado arquivo, o **root** pode fazer qualquer coisa com aquele arquivo: ler, escrever e executar. O **root** pode até mesmo destituir o usuário da propriedade do arquivo, fazendo com que o arquivo passe a ter outro dono!

Com isso, caro leitor, acredito que já possamos passar para a parte mais interessante do nosso livro: o trabalho braçal com o Linux. Vamos conhecer seus componentes principais e vamos aprender a utilizá-los!

Pastas Pessoais dos Usuários

Cada usuário cadastrado no sistema Linux tem uma pasta própria, onde recomenda-se que este guarde seus arquivos pessoais (como "Meus Documentos" no Windows). Claro que essa pasta será usada **se o usuário quiser**, pois nada (realmente) o obriga a usá-la! É apenas uma questão de organização e praticidade.

Para todos os usuários do sistema (com exceção do usuário **root**), a pasta pessoal fica localizada em **/home/xxxx**, onde **xxxx** é o login do referido usuário. Exemplo: o usuário **pedro** vai ter, quando cadastrado, sua pasta pessoal criada como **/home/pedro**.

Para o super usuário, a pasta pessoal dele é **/root**, fora da estrutura de **/home**. (é... quem pode, pode!).

E, é claro, a menos que se determinem permissões diferentes, o diretório **/root** é acessível somente pelo usuário **root** e os diretórios pessoais dos outros usuários estarão acessíveis apenas por eles respectivamente (cada um no seu) e pelo **root** (novamente, quem pode, pode!).

ENTENDENDO O SHELL (O AMBIENTE DOS COMANDOS DE TEXTO)

A janela de comandos do Linux possui um prompt (aviso) e um cursor (para inserir caracteres). O prompt é apresentado assim, normalmente:

```
[usuario@computador diretório]$
```

Onde:

usuario: login do usuário que está logado

computador: nome do computador que se está usando

diretório: nome do diretório atual (ou seja, a pasta onde se está trabalhando no momento).

Às vezes, porém, o prompt apresenta informações variadas (não exatamente essas que falei): por exemplo, é possível encontrar o prompt apenas com um sinal, que pode ser:

\$ se o usuário logado é um usuário comum;

se o usuário logado é o root (administrador);

(note que esses sinais acompanham os prompts grandes também!)

Além de saber como o Linux se apresenta em modo texto, é interessante ver como se processam as respostas dadas aos comandos que digitamos: é bem simples... Digite o comando; pressione ENTER; o Linux vai responder a você!

```
[prompt]$ comando <ENTER>
```

Resposta do Linux

```
.
```

(pode levar várias linhas, bem como, pode até não ter nenhuma - existem comandos que não dão respostas quando está tudo certo!!).

```
[prompt, de novo (esperando o próximo comando)]$
```

1) Comandos de Manipulação de Arquivos e Diretórios

1.1) ls (List - Listar): esse comando, semelhante ao **DIR** do **DOS**, serve para listar o conteúdo de um determinado diretório, apresentando arquivos e diretórios presentes no local especificado.

Veja, por exemplo um hipotético diretório provas, com quatro arquivos. Na interface "bonita" no Linux, o diretório provas seria visto assim:

Na interface "feia, mas funcional", a listagem seria apresentada pelo comando ls:

```
[joao@computer provas]$ ls

joao.antonio.txt
joao.antonio.txt~
espaco.juridico.doc
trf2005.txt
```

É, mas essa listagem não apresenta muitas informações úteis, não é? Portanto, vamos usar a opção "**l**"... Nos comandos do Linux, a maioria das opções é apresentada precedida do sinal de "-" (**menos**). Portanto, o comando ficaria:

```
[joao@computer provas]$ ls -l

total 16
-rw-rw-r-- 1 joao joao 7 Jul 1 13:24 joao.antonio.txt
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

A opção **-l** permite a apresentação dos arquivos com detalhamento. Esse detalhamento é o seguinte (em relação à primeira linha da listagem, mas serve para todas as outras, ok?):

-rw-rw-r--: Permissões do arquivo (veremos mais adiante);

1: número de links (atalhos) que apontam para esse arquivo;

joao joao: DONOS do arquivo (a primeira palavra é o USUARIO DONO e a segunda palavra é o GRUPO DONO)... Veremos depois também.

7: Tamanho (em bytes) do arquivo.

Jul 1 13:24: Data da última modificação que o arquivo sofreu (data/hora do último salvamento).

joao.antonio.txt: Nome do arquivo.

Tem outra opção do ls: a opção **-a** permite que sejam visualizados os arquivos ocultos também (arquivos que normalmente não são vistos pelos comandos ls normais - sem essa opção).

Então fica assim (caso queiramos listagem detalhada e vendo arquivos ocultos):

```
[joao@computer provas]$ ls -a -l
```

```
total 28
drwxrwxr-x 2 joao joao 4096 Jul 1 14:58 .
drwxrwx--- 5 joao joao 4096 Jul 1 13:24 ..
-rw-rw-r-- 1 joao joao 7 Jul 1 13:24 joao.antonio.txt
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
-rw-rw-r-- 1 joao joao 89892 Jul 1 13:24 .gabarito2005.txt
-rw-rw-r-- 1 joao joao 23 Jul 1 13:24 trf2005.txt
```

Para não termos que digitar **-l -a -qualquer coisa**, basta juntar todas as opções num **- (hífen)** só, como em **ls -la** (isso vale para quase todos os comandos).

Tem uma outra interessante, que é **-h** (h de "humano", ou "agradável para os humanos")... Essa opção permite que os valores em bytes dos arquivos sejam escritos com abreviações K para Kilo, M para Mega e G para Giga, facilitando a leitura do tamanho do arquivo... Só tem sentido usar essa opção associada ao **-l** (porque é quando aparecem os tamanhos dos arquivos).

```
[joao@computer provas]$ ls -lah

total 28K
drwxrwxr-x 2 joao joao 4,0K Jul 1 14:58 .
drwxrwx--- 5 joao joao 4,0K Jul 1 13:24 ..
-rw-rw-r-- 1 joao joao 7 Jul 1 13:24 joao.antonio.txt
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
-rw-rw-r-- 1 joao joao 90,0K Jul 1 13:24 .gabarito2005.txt
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

Aproveito o ensejo para mostrar algo interessante: na parte das permissões do arquivo, quando o primeiro caractere é um **"d"**, significa que o objeto em questão é um diretório (pasta), porém, quando um **"-"** (hífen) aparece ao invés do **"d"**, o objeto é um arquivo.

Uma última coisinha: o comando **ls** não é usado para somente listar o conteúdo do diretório atual... Ele pode ser usado para listar o conteúdo de qualquer outro diretório desde que seja informado qual será o alvo.

Um exemplo: vamos listar o conteúdo do diretório **/home/joao/downloads** (mesmo não estando nessa pasta).

```
[joao@computer provas]$ ls -lah /home/joao/downloads

total 7,3M
drwxrwx--- 3 joao joao 4,0K Jul 1 15:16 .
drwxr-xr-x 35 joao joao 4,0K Jul 1 14:53 ..
-rw-rw---- 1 joao joao 346K Jun 3 15:57 banrisul2005.pdf
-rw-rw---- 1 joao joao 323K Jun 21 18:27 banrisul.zip
-rw-rw-r-- 1 joao joao 213K Jun 28 17:52 gestor.mg.esaf.pdf
drwxrwxr-x 2 joao joao 4,0K Jul 1 15:16 programas
-rw-rw-r-- 1 joao joao 280K Jun 28 17:52 prova.pdf
```

```
-rw-rw-rw- 1 root root 38K Jun 28 18:11 prova.txt
-rw-rw-r-- 1 joao joao 6,1M Jun 22 17:08 skype-1.1.0.13-mdk.i586.rpm
```

Note que há um diretório (**programas**) dentro do diretório **downloads** (cujo conteúdo foi visualizado no comando acima). Caso se queira listar o conteúdo deste diretório também (ou seja, ao executar o **ls**, este ser usado para ler o conteúdo dos diretórios que estão dentro do diretório alvo), deve-se usar a opção **-R** ("R" de "recursivamente", ou seja, refazendo várias vezes até encontrar um diretório sem subdiretórios) - esse R tem que ser maiúsculo!!!...

Fica assim:

```
[joao@computer provas]$ ls -lahR /home/joao/downloads

downloads:
total 7,3M
drwxrwx--- 3 joao joao 4,0K Jul 1 15:16 .
drwxr-xr-x 35 joao joao 4,0K Jul 1 14:53 ..
-rw-rw---- 1 joao joao 346K Jun 3 15:57 banrisul2005.pdf
-rw-rw---- 1 joao joao 323K Jun 21 18:27 banrisul.zip
-rw-rw-r-- 1 joao joao 213K Jun 28 17:52 gestor.mg.esaf.pdf
drwxrwxr-x 2 joao joao 4,0K Jul 1 15:23 programas
-rw-rw-r-- 1 joao joao 280K Jun 28 17:52 prova.pdf
-rw-rw-rw- 1 root root 38K Jun 28 18:11 prova.txt
-rw-rw-r-- 1 joao joao 6,1M Jun 22 17:08 skype-1.1.0.13-mdk.i586.rpm

downloads/programas:
total 13M
drwxrwxr-x 2 joao joao 4,0K Jul 1 15:23 .
drwxrwx--- 3 joao joao 4,0K Jul 1 15:16 ..
-rw-rw-r-- 1 joao joao 6,1M Jun 22 17:08 jogos.i586.mdk.rpm
-rw-rw-r-- 1 joao joao 6,1M Jun 22 17:08 teste.tar.gz
```

Acho que sobre o ls já vimos o necessário... mas tem muito mais!!!

1.2) mkdir (Make Directory - Criar Diretório): o comando mkdir é usado para criar diretórios (ele é idêntico ao MD do DOS). Sua sintaxe de uso é assim:

mkdir <opções> nome

<opções>: opcionais (como no nome já diz)

nome: nome que será dado ao diretório que se deseja criar, como em:

```
[joao@computer provas]$ mkdir faceis
```

Nesse caso, será criado um diretório chamado faceis, dentro do diretório **provas** (diretório atual).

Caso se queira criar um diretório em outro local (que não seja o diretório atual), basta informar o caminho completo para isso.

Veja o exemplo a seguir, onde um diretório chamado **programas** será criado dentro de **/home/joao/downloads** note que o comando será executado tendo **provas** como diretório corrente:

```
[joao@computer provas]$ mkdir /home/joao/downloads/programas
```

Ainda tem uma muito boa: caso você deseje criar mais de um diretório, não é necessário criar um por vez, execute o comando `mkdir` com todos os diretórios que deseja criar separados por um espaço:

```
[joao@computer provas]$ mkdir medias dificeis esaf
```

Nesse caso serão criados os diretórios `medias`, `dificeis` e `esaf` dentro do diretório **provas** (que é o atual). O resultado do comando `mkdir` pode ser visto pelo `ls`. Veja no exemplo a seguir:

```
[joao@computer provas]$ ls -lah

total 28K
drwxrwxr-x 2 joao joao 4,0K Jul 1 14:58 .
drwxrwx--- 5 joao joao 4,0K Jul 1 13:24 ..
-rw-rw-r-- 1 joao joao 7 Jul 1 13:24 joao.antonio.txt
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxrwxr-x 2 joao joao 4,0K Jul 1 18:58 dificeis
drwxrwxr-x 2 joao joao 4,0K Jul 1 18:58 esaf
drwxrwxr-x 2 joao joao 4,0K Jul 1 18:53 faceis
-rw-rw-r-- 1 joao joao 90,0K Jul 1 13:24 .gabarito2005.txt
drwxrwxr-x 2 joao joao 4,0K Jul 1 18:58 medias
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

1.3) cd (Change Directory – Mudar de Diretório): esse comando tem a mesma função (e o mesmo nome) que tinha no DOS: permitir que o usuário “entre” em outro diretório.

Caso o usuário esteja no diretório `provas` e deseje “entrar” no diretório `esaf` (que criamos há pouco), é só digitar:

```
[joao@computer provas]$ cd esaf
[joao@computer esaf]$
```

Note a mudança no prompt depois do comando concluído: o diretório corrente é `esaf` agora (isso significa, em palavras fáceis: você está no diretório `esaf` agora!).

Para sair de um diretório, voltando ao diretório pai (ou seja, o diretório que contém aquele em que você está agora), basta digitar:

```
[joao@computer esaf]$ cd ..
[joao@computer provas]$
```

Note que o sinal de .. (ponto ponto) é separado do comando cd (não adianta digitar cd.. tudo junto, feito no DOS, o linux não entende!).

Caso o usuário queira ir para um diretório qualquer, mesmo que não tenha ligação com o diretório atual, basta digitar o caminho inteiro para ele (partindo do diretório raiz – a /). Veja:

```
[joao@computer provas]$ cd /home/joao/downloads
[joao@computer downloads]$
```

Caso o usuário queira ir direto ao diretório raiz (o nível mais alto da estrutura de diretórios do linux, basta digitar "cd /":

```
[joao@computer provas]$ cd /
[joao@computer /]$
```

Tem umas dicas legais aqui, que são diferentes do DOS: como cada usuário tem seu diretório pessoal (normalmente /home/usuario para usuários comuns e /root para o administrador), é possível saltar diretamente para esse diretório apenas digitando **cd ~** (**til**) ou simplesmente **cd**. Veja:

```
[joao@computer provas]$ cd ~
[joao@computer joao]$
```

1.4) pwd: esse comando informa ao usuário em que diretório ele está (parece inútil, mas de vez em quando é interessante). Veja o exemplo:

```
[joao@computer joao]$ pwd
/home/joao

[joao@computer joao]$
```

Eu não sei se é tarde demais para dizer isso, mas: todos os comandos do Linux são sucedidos da tela ENTER, para confirmação da ordem... (acho que vocês já haviam deduzido isso)...

1.5) rmdir (Remover Diretório): esse comando apaga diretórios vazios (é semelhante ao RD do DOS). Para remover diretórios com conteúdo, podemos usar outro comando, que será visto adiante.

DICA IMPORTANTE:

Aqui vai uma dica preciosa: a tecla TAB, em seu teclado, facilita, e muito, o trabalho dos usuários porque permite que os comandos sejam completados, dispensando a necessidade de digitar tudo.

*Exemplo: você pretende entrar num diretório chamado **tributario**, que está no diretório atual. Caso esse seja o único diretório que inicia com "t", digite o seguinte:*

```
[joao@computer joao]$ cd t<TAB>
```

... o Linux fará aparecer o seguinte:

```
[joao@computer joao]$ cd tributario
```

Claro que o TAB só sabia como completar o comando porque tributario era a única pasta com esse início. Caso haja mais de uma pasta com o início semelhante (exemplo: **tributario** e **tribunais**), ao pressionar TAB, o Linux irá preencher até a última letra idêntica e mostrará as opções... Veja:

```
[joao@computer joao]$ cd t<TAB>
[joao@computer joao]$ cd tribu
tributario tribunais
```

1.6) clear (Limpar a tela): esse comando esvazia todos os caracteres mostrados na tela e coloca o prompt sozinho na parte superior esquerda da mesma. É equivalente ao **cls** do DOS.

1.7) tree (Árvore): este comando mostra a estrutura de diretórios e arquivos em forma de uma árvore simples. É possível ver diretórios, os diretórios e arquivos dentro dos diretórios e assim sucessivamente. Para acionar o comando tree, basta digitá-lo na linha de comando.

A opção -F é usada para que a listagem de arquivos e diretórios possa ser apresentada com um caractere a mais sucedendo os nomes dos arquivos e diretórios (esse caractere adicional serve para identificar o tipo do objeto cujo nome o antecede):

/ = indica que o objeto é um diretório;

* = indica que é um arquivo executável;

@ = indica que é um link (atalho);

Essa opção (-F) também pode ser usada no comando ls!!! (para a mesma finalidade: apresentar um caractere adicional que indicará o tipo do arquivo/diretório).

Veja um exemplo do comando tree:

```
[joao@computer provas]$ tree -F

conursos/provas
|-- joao.antonio.txt*
|-- joao.antonio.txt~
|-- espaco.juridico.doc
|-- esaf/
|   |-- auditor/
|   |   |-- prova.doc
|   |-- casa
|   |-- gestor/
|   |-- prova.doc
|   |-- slack.pdf
|   |-- soft.pdf
|   |-- tecnico/
|       |-- prova.gestor.rtf*
```

```
^-- trf2005.txt
```

```
4 directories, 10 files
```

Note o resultado... Pudemos conhecer certas informações: joao.antonio.txt é executável; esaf, auditor, gestor e tecnico sao diretórios; dentro do diretório tecnico há um arquivo executável chamado prova.gestor.rtf (entendeu!?).

Só falta fazer um lembrete: o comando tree nao está presente em todas as distribuições ou formas de instalação do Linux, nao! Então, é perfeitamente possível que, ao tentar treinar esse comando, você se depare com um cenário como esse:

```
[joao@computer provas]$ tree -F
```

```
tree: command not found
```

1.8) mv (Mover): esse comando tem duas funções distintas: **mover** e **renomear** arquivos e diretórios. O comando mv substitui os comandos **MOVE** e **REN** do **DOS**.

A sintaxe (forma de escrever) do comando mv é:

```
mv [opções] <origem> <destino>
```

onde:

Origem: é o nome do arquivo a ser movido / renomeado.

Destino: é o nome do diretório para onde o arquivo vai (caso esteja sendo movido) ou o nome que o arquivo irá ter (caso esteja sendo renomeado).

Para usar o comando mv, deve-se tem em mente o que se quer fazer... Vamos começar pela ação de Renomear: veja a listagem do conteúdo do diretório esaf... Vamos renomear o arquivo chamado teste.doc para prova.doc:

```
[joao@computer esaf]$ ls -lah
```

```
total 23M
```

```
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:26 .
drwxrwxr-x 3 joao joao 4,0K Jul 2 18:24 ..
-rwxr-xr-x 1 joao joao 14K Jul 2 18:25 prova.gestor.rtf
-rw-r----- 1 joao joao 2,6M Jul 2 18:25 slack.pdf
-rw-r----- 1 joao joao 225K Jul 2 18:25 soft.pdf
-rw-r----- 1 joao joao 20M Jul 2 18:25 teste.doc
```

```
[joao@computer esaf]$ mv teste.doc prova.doc
```

```
[joao@computer esaf]$ ls -lah
```

```
total 23M
```

```
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:26 .
drwxrwxr-x 3 joao joao 4,0K Jul 2 18:24 ..
```

```
-rwxr-xr-x 1 joao joao 14K Jul 2 18:25 prova.gestor.rtf
-rw-r----- 1 joao joao 2,6M Jul 2 18:25 slack.pdf
-rw-r----- 1 joao joao 225K Jul 2 18:25 soft.pdf
-rw-r----- 1 joao joao 20M Jul 2 18:25 prova.doc
```

É importante saber como o Linux diferencia a ação a ser realizada pelo comando mv: Será mover se, no lugar do <Destino>, for escrito o nome de um diretório que existe. Caso no <Destino> seja descrito um nome qualquer que não existe, o comando automaticamente funcionará como RENOMEAR, atribuindo ao arquivo em questão o novo nome descrito na cláusula <Destino>.

Veja o comando mv sendo usado para Mover um arquivo:

```
[joao@computer esaf]$ ls -lh

total 23M
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 auditor
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 gestor
-rw-r----- 1 joao joao 20M Jul 2 18:25 prova.doc
-rwxr-xr-x 1 joao joao 14K Jul 2 18:25 prova.gestor.rtf
-rw-r----- 1 joao joao 2,6M Jul 2 18:25 slack.pdf
-rw-r----- 1 joao joao 225K Jul 2 18:25 soft.pdf
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 tecnico

[joao@computer esaf]$ mv prova.gestor.rtf tecnico

[joao@computer esaf]$ ls -lh

total 23M
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 auditor
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 gestor
-rw-r----- 1 joao joao 20M Jul 2 18:25 prova.doc
-rw-r----- 1 joao joao 2,6M Jul 2 18:25 slack.pdf
-rw-r----- 1 joao joao 225K Jul 2 18:25 soft.pdf
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 tecnico
```

O comando mv, mostrado acima, foi usado para mover o arquivo prova.gestor.rtf para o diretório tecnico. Note, no segundo ls, que o arquivo não se encontra mais no diretório corrente!

Muita gente vai perguntar: Joao, porque as operações de MOVER e RENOMEAR arquivos, que são tão diferentes entre si, são executadas pelo mesmo comando? Foi uma prova de que o pessoal do Linux é mão-de-vaca para criar comandos? Não! Não é isso: as ações de MOVER e RENOMEAR são, na verdade, a mesma coisa: quando se "move" um arquivo de um diretório para outro, o índice que aponta para o arquivo é atualizado, informando a nova localidade... é a mesma coisa que acontece quando se renomeia um arquivo (seu índice é atualizado, para conter o novo nome). Não há, portanto, alterações na localização física do arquivo no Disco (na maioria dos casos)... o que conta mesmo é a

alteração no índice que aponta para o arquivo (a tabela de alocação dos arquivos no disco).

Lembre-se: `mv <origem> <destino>` significa RENAME a <origem> se o nome colocado em <destino> não existir. Se o nome colocado em <destino> existir, e for um diretório (ainda tem isso), o comando `mv` assume imediatamente sua função de mover o objeto descrito em <origem>.

1.9) cp (Copiar): o comando `cp` copia arquivos e diretórios (copiar é criar um outro arquivo idêntico ao original).

A sintaxe do comando `cp` é:

```
cp [opções] <origem> <destino>
```

Lembro que, se <destino> for um diretório existente, uma nova cópia do arquivo descrito em <origem> será criada dentro daquele diretório. Contudo, caso o nome descrito em <destino> não exista, será criada uma cópia de <origem> com o nome <destino> (um segundo arquivo, idêntico em tudo, menos no nome, que será o que estiver descrito em <destino>). Veja que bonito... começaremos com um `ls` para verificar o conteúdo, depois, serão efetuados dois comandos `cp` seguidos (preste atenção neles) e, por fim, um comando `tree`, para mostrar a árvore de diretórios.

```
[joao@computer esaf]$ ls -lh
total 23M
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 auditor
drwxr-xr-x 2 joao joao 4,0K Jul 2 18:58 gestor
-rw-r----- 1 joao joao 20M Jul 2 18:25 prova.doc
-rw-r----- 1 joao joao 2,6M Jul 2 18:25 slack.pdf
-rw-r----- 1 joao joao 225K Jul 2 18:25 soft.pdf
drwxr-xr-x 2 joao joao 4,0K Jul 2 19:02 tecnico

[joao@computer esaf]$ cp prova.doc auditor

[joao@computer esaf]$ cp soft.pdf casa

[joao@computer esaf]$ tree -F

esaf
|-- auditor/
|   |-- prova.doc
|-- casa
|-- gestor/
|-- prova.doc
|-- slack.pdf
|-- soft.pdf
|-- tecnico/
|   |-- prova.gestor.rtf*

3 directories, 6 files
```

Como se pode perceber, existe agora um arquivo prova.doc dentro da pasta auditor (criado pelo primeiro comando cp) e existe um arquivo casa dentro da pasta esaf (este arquivo foi criado pelo segundo comando cp).

Um lembrete: por padrão, o comando cp omite os diretórios (não os copia). Portanto, se o usuário tentar executar o comando cp <diretorio> <destino>, o comando não fará nada! Para copiar um diretório e todo o seu conteúdo, use a opção -r no comando cp (esse r vem de "recursivamente" novamente, ou seja, copie o diretório, entre nele, vá copiando o que tem dentro (outros diretórios e/ou arquivos), entrando nesses diretórios, copiando seus conteúdos, e assim sucessivamente até chegar num ponto em que não haja mais diretórios para entrar)...

Veja esse cenário (dentro do diretório provas):

```
[joao@computer provas]$ ls -lhF

total 20K
-rwxrwxrwx 1 joao joao 7 Jul 1 13:24 joao.antonio.txt*
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt

[joao@computer provas]$ tree -F

concursos/provas
|-- joao.antonio.txt*
|-- joao.antonio.txt~
|-- espaco.juridico.doc
|-- esaf/
|   |-- auditor/
|   |   |-- prova.doc
|   |-- casa
|   |-- gestor/
|   |-- prova.doc
|   |-- slack.pdf
|   |-- soft.pdf
|   |-- tecnico/
|       |-- prova.gestor.rtf*
|-- trf2005.txt

4 directories, 10 files

[joao@computer provas]$ cp esaf fcc (prestem atenção aqui!!!)

cp: omitindo diretório 'esaf' (ops... errei, faltou o -r)
```

```
[joao@computer provas]$ cp -r esaf fcc

[joao@computer provas]$ ls -lhF

total 24K
-rwxrwxrwx 1 joao joao 7 Jul 1 13:24 joao.antonio.txt*
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
drwxr-xr-x 5 joao joao 4,0K Jul 3 00:55 fcc/
      -rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

Note que há mais um diretório agora (fcc). Vamos comprovar que seu conteúdo é idêntico ao conteúdo do diretório esaf com o comando tree -F:

```
[joao@computer provas]$ tree -F

conursos/provas
|-- joao.antonio.txt*
|-- joao.antonio.txt~
|-- espaco.juridico.doc
|-- esaf/
|   |-- auditor/
|   |   |-- prova.doc
|   |-- casa
|   |-- gestor/
|   |-- prova.doc
|   |-- slack.pdf
|   |-- soft.pdf
|   |-- tecnico/
|       |-- prova.gestor.rtf*
|-- fcc/
|   |-- auditor/
|   |   |-- prova.doc
|   |-- casa
|   |-- gestor/
|   |-- prova.doc
|   |-- slack.pdf
|   |-- soft.pdf
|   |-- tecnico/
|       |-- prova.gestor.rtf*
|-- trf2005.txt
```

```
8 directories, 16 files
```

Pois é, com isso chegamos ao fim do comando cp (tem mais coisa, mas acho que você descobrirá com o tempo).

1.9) rm (Remover arquivos e diretórios): o comando rm é usado para apagar arquivos e diretórios (incluindo os diretórios não-vazios).

Sintaxe: rm [opções] <alvo>

Note que o comando rm vai pedir uma confirmação do apagamento do referido arquivo.

```
[joao@computer provas]$ ls -lhF

total 24K
-rwxrwxrwx 1 joao joao 7 Jul 1 13:24 joao.antonio.txt*
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
drwxr-xr-x 5 joao joao 4,0K Jul 3 00:55 fcc/
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt

[joao@computer provas]$ rm joao.antonio.txt

rm: remover arquivo comum 'joao.antonio.txt'? s (digitei o "s")

[joao@computer provas]$ ls -lhF

total 24K
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
drwxr-xr-x 5 joao joao 4,0K Jul 3 00:55 fcc/
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

Caso se deseje apagar um diretório e tudo o que tem dentro dele, acione o comando rm com a opção -r ("r" de recursivamente, de novo!)... Mas aí, dou logo o aviso: o Linux vai perguntar por cada um dos arquivos a serem apagados (que estiverem dentro do diretório alvo), o que, por sinal, é um verdadeiro SACO!!!

Para que o Linux entenda que deve apagar sem perguntar (sem exigir confirmação para o apagamento de CADA ARQUIVO), use a opção -f ("f" de "forçadamente" - esse é minúsculo!!!).

Então veja as opções -f e -r em ação:

```
[joao@computer provas]$ ls -lhF

total 8K
```

```
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
drwxr-xr-x 5 joao joao 4,0K Jul 3 00:55 fcc/
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt

[joao@computer provas]$ rm -rf fcc

[joao@computer provas]$ ls -lhF

total 4K
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 joao.antonio.txt~
-rw-rw-r-- 1 joao joao 8 Jul 1 13:25 espaco.juridico.doc
drwxr-xr-x 5 joao joao 4,0K Jul 2 19:28 esaf/
-rw-rw-r-- 1 joao joao 2 Jul 1 13:24 trf2005.txt
```

2) Comandos para Permissões de Acesso a Arquivos e Diretórios

2.1) chmod (Change Mode - Mudar o Modo (!?!)): apesar deste comando ter um nome estranho, seu objetivo é mudar as permissões de um arquivo ou diretório. Esse "mudar as permissões" significa que o arquivo poderá ser acessado, modificado e executado por outras pessoas além daquelas que poderiam ter esses direitos. É um troço bastante esquisito, mas interessante, sem dúvida!

Sintaxe: `chmod <permissões> <alvo>`

onde:

chmod é o comando (claro!)

<permissões> é o dado destinado a informar quais os níveis de permissão que o arquivo vai ter...

<alvo> é o arquivo em si, que terá suas permissões alteradas (ou diretório, claro).

Vamos a algo bem interessante: um comando ls para listar o conteúdo de um diretório, primeiramente...

```
[joao@computer documentos]$ ls -lh
total 188K
drwxrwxr-x 2 joao users 4,0K Jul 8 18:40 apresentacoes
-rw-rw-r-- 1 joao users 4,7K Jul 8 18:40 avisos.doc
lrwxrwxrwx 1 joao users 14 Jul 8 19:07 dic.dic -> dicionario.dic
-rw-rw-r-- 1 joao users 61K Jul 8 18:40 dicionario.dic
-rw-rw-r-- 1 joao users 4,7K Jul 8 18:41 impressora_manual.doc
-rw-rw-r-- 1 joao users 4,2K Jul 8 18:40 instrucoes.doc
-rw-rw-r-- 1 joao users 31K Jul 8 18:40 jovem_galileu.mp3
-rwxr-xr-x 1 joao users 14K Jul 8 18:40 lattes.rtf
drwxrwxr-x 2 joao users 4,0K Jul 8 18:48 planilhas
-rw-rw-r-- 1 joao users 31K Jul 8 18:41 poesia.pdf
```

```
drwxrwxr-x 2 joao users 4,0K Jul 8 18:47 textos
drwxrwxr-x 2 joao users 4,0K Jul 8 18:45 web
```

Vamos tomar como exemplo o arquivo instrucoes.doc, que mostra o seguinte:

```
-rw-rw-r-- 1 joao users 4,2K Jul 8 18:40 instrucoes.doc
```

Como vimos anteriormente, esse arquivo tem as seguintes informações:

-rw-rw-r-- : permissões de acesso ao arquivo.

1 : quantidade de atalhos que apontam para esse arquivo

joao users: esse arquivo pertence a um usuário chamado joao e a um grupo chamado users

4,2K : tamanho do arquivo

Jul 8 18:40: data e hora da última alteração do arquivo.

Quanto às permissões, que são o nosso alvo de interesse, é bom que se saiba que há três tipos de permissões para arquivos e diretórios:

r (read - leitura): essa permissão diz que o arquivo pode ser lido (aberto);

w (write - escrita): informa que o arquivo pode ser escrito (modificado, salvo);

x (eXecute - executar): indica que o arquivo pode ser executado (ou seja, ele será considerado executável e poderá ser colocado na memória RAM como um programa, sem necessitar de outro programa qualquer para isso). Essa permissão, no caso de diretórios (pastas) é necessária para que a pasta seja acessada (ou seja, para que se possa ver seu conteúdo), então, em resumo é: para que um diretório seja acessado (seu conteúdo seja visto), é necessário que o indicador de x (execução) esteja ativado.

Outra coisa: há três pessoas (ou grupos de pessoas) que podem ter permissões distintas sobre um arquivo qualquer:

Usuário Dono (User): é o usuário cujo nome está descrito na primeira coluna do proprietário (no caso acima, joao)

Grupo Dono (Group): é o grupo de usuários ao qual o arquivo pertence, que no caso anterior é users (um arquivo pode pertencer a somente um grupo)

Outros usuários (Other): todos os demais usuários do sistema que não pertencem ao grupo dono.

Aquelas informações sobre permissões, localizadas na primeira coluna da descrição mostrada no comando ls, são justamente as informações sobre leitura, escrita e execução destes três grupos... Veja só:

- **Primeiro caractere**: Se for um "d", indica um diretório... Se for um "l", indica um atalho (link), se for um "-" (traço), indica que é um arquivo (isso não tem nada com as permissões, é apenas um indicativo do tipo do objeto).

rw- (primeiro conjunto de três caracteres): permissões dadas ao USUÁRIO DONO do arquivo.

rw- (segundo grupo de três caracteres): permissões dadas ao GRUPO DONO do arquivo.

r-- (terceiro grupo de três caracteres): permissões dadas aos OUTROS USUÁRIOS do sistema.

Cada um dos grupos de três caracteres pode ter rwx, onde, claro, r indica read, w indica write e x indica execute... O Traço (-) indica que aquela permissão não está dada (ou

seja, não se tem permissão de realizar aquela operação). Portanto, rw- significa direito de ler e modificar (escrever) um arquivo, mas não o direito de executá-lo

Então, um conjunto de permissões rw-rw-r-- no arquivo instrucoes.doc significa que este arquivo pode ser lido e modificado (rw-) pelo seu DONO (joao), também pode ser lido e modificado (rw-) pelos usuários que pertencem ao seu GRUPO (users) e pode ser apenas lido (r--) pelos OUTROS usuários do sistema. Esse arquivo não poderá ser executado (jogado na RAM como um programa) nem pelo DONO, nem pelo GRUPO nem pelos demais usuários.

É interessante entender que quando um arquivo é criado (um documento, como esse do exemplo, na primeira vez que é salvo), ele recebe imediatamente as permissões padrão do sistema (essas.. rw-rw-r--), mas essas permissões podem ser mudadas ao longo da utilização do sistema através do uso do comando chmod.

IMPORTANTE: o comando chmod só pode ser usado em um arquivo pelo root ou pelo DONO do arquivo.

Como indicamos as permissões no comando chmod?

Há várias maneiras de indicar as permissões que um arquivo vai ter no comando chmod... **Vamos começar pelo "modo texto".**

sintaxe: chmod detentor=permissões alvo

Seria assim: vamos supor que o arquivo instrucoes.doc vai ser alterado para ter permissão de ler, escrever e executar por parte do DONO, ler somente por parte do GRUPO e somente executar por parte dos DEMAIS USUÁRIOS... (ou seja, se você entendeu, sabe que ficará assim: rw-r---x)... O comando que faz isso é:

```
[joao@computer documentos]$ chmod u=rwx,g=r,o=x instrucoes.doc
```

```
[joao@computer documentos]$ ls -lh instrucoes.doc
-rw-r---x 1 joao users 4,2K Jul 8 18:40 instrucoes.doc
```

as letras antes do sinal de = (igual) identificam as pessoas detentoras dos direitos, sendo que: "u" é para USUÁRIO DONO, "g" é para GRUPO DONO e "o" é para OUTROS USUÁRIOS, caso se queira definir para todos os três grupos de uma vez só, usa-se "a" (de all - todos). Claro que você já entendeu que as letras depois dos sinais de = são para identificar as permissões em si. Só para completar, note que para separar os detentores de permissões foi usada uma vírgula... Não é para escrever com espaços... é vírgula mesmo! Outra coisa: não há ordem na colocação das cláusulas, o "o" pode vir primeiro, depois o "u" e depois o "g"... não tem problema!

Alguma dúvida em relação a esse comando? Calma! Não acabou ainda!

Se você quiser apenas adicionar ou retirar um tipo de permissão, fica simples: ao invés de usar o sinal de = (igual) use os sinais de + (mais) para colocar ou - (menos) para retirar permissões. Então, caso se deseje retirar o direito de modificar o arquivo para os usuários do GRUPO DONO, faz-se isso:

```
[joao@computer documentos]$ chmod g-w instrucoes.doc
```

Outro exemplo, caso se deseje adicionar as permissões de escrita e execução aos demais usuários do sistema para o arquivo instrucoes.doc, fazemos:

```
[joao@computer documentos]$ chmod o+rx instrucoes.doc
```

Entendido?

Comando chmod (Modo "Numérico")

Existe uma forma interessante de usar o chmod (eu prefiro essa), que é através de números octais (0 a 7) para identificar as permissões. Seria assim:

sintaxe: chmod ABC alvo

Onde A, B e C são números que podem variar entre 0 e 7. O Primeiro número (A) identifica as permissões do USUÁRIO DONO do arquivo, o segundo número (B) identifica as permissões dadas ao GRUPO DONO do arquivo e o terceiro número (C) identifica as permissões dadas aos DEMAIS USUÁRIOS (outros). Ou seja, cada número daqueles é para um DETENTOR de permissões.

Mas, cada número daqueles já identifica o estado das 3 permissões (rwx). Para isso, remeto a um cálculo simples de conversão de base, onde eu apresento os números de 0 a 7 em formato normal (Decimal) e em formato binário (escrito com 3 bits)...

Decimal	Binário
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Pense naqueles 3 bits dos números binários e leia-os como rwx (isso mesmo, cada 0 ou 1 dos números binários é uma permissão lida nessa mesma ordem, r (read - ler) depois w (Write - modificar), por fim x (executar). Agora saiba que quando o bit estiver definido como 1, aquela permissão será CONCEDIDA, e quando o bit estiver com 0, aquela permissão está sendo NEGADA. Isso significa que o número 5, que é escrito como 101 em binário, significa que foram concedidas as permissões de r (ler) e x (executar), mas a permissão para w (escrever) foi negada. A tradução dessa doidice seria:

Decimal	Binário		
	r	w	x
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Pronto, fica simples assim: vejamos o caso de quisermos que o arquivo instrucoes.doc fique assim: O USUÁRIO DONO pode fazer tudo (ler, escrever e executar), os usuários do GRUPO DONO poderão apenas ler e os DEMAIS usuários do sistema não farão nada (negação total). Isso vai ser possível mediante o comando:

```
[joao@computer documentos]$ chmod 740 instrucoes.doc

[joao@computer documentos]$ ls -lh instrucoes.doc
-rwxr----- 1 joao users 4,2K Jul 8 18:40 instrucoes.doc
```

Explicando:

7: O primeiro número é para os direitos do USUÁRIO DONO - 7 é 111 em binário (direitos totais - rwx CONCEDIDOS).

4: O Segundo número é para os direitos do GRUPO DONO - 4 é 100 em binário (somente direito de r CONCEDIDO).

0: O terceiro número é para as permissões dadas aos DEMAIS USUÁRIOS - 0 é 000 (nenhum direito CONCEDIDO).

Ufa... Espero sinceramente que tenham entendido (esse é um comando que pode fornecer inúmeras questões para concursos)...

Comando chmod em Diretórios

Prestem atenção: para que se possa acessar o conteúdo de um diretório qualquer (ou seja, conseguir ver os arquivos que estão presentes deste diretório), é necessário definir a permissão de x (execução) para ele. Se o x estiver NEGADO naquela pasta, não será possível ver os arquivos que estão dentro dela (nem por meio do comando ls... ficaria assim:

```
[joao@computer documentos]$ ls -lh
total 192K
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:40 apresentacoes
-rw-rw-r-- 1 joao joao 4,7K Jul 8 18:40 avisos.doc
drw----- 2 joao joao 4,0K Jul 16 13:17 bronca
lrwxrwxrwx 1 joao joao 14 Jul 8 19:07 dic.dic -> dicionario.dic
-rw-rw-r-- 1 joao joao 61K Jul 8 18:40 dicionario.dic
-rw-rw-r-- 1 joao joao 4,7K Jul 8 18:41 impressora_manual.doc
-rw-rwxr-x 1 joao joao 4,2K Jul 8 18:40 instrucoes.doc
-rw-rw-r-- 1 joao joao 31K Jul 8 18:40 jovem_galileu.mp3
-rwxr-xr-x 1 joao joao 14K Jul 8 18:40 latttes.rtf
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:48 planilhas
-rw-rw-r-- 1 joao joao 31K Jul 8 18:41 poesia.pdf
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:47 textos
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:45 web
```

Note a pasta bronca... ela não tem permissão para ninguém executar... Agora veja a continuação...

```
[joao@computer documentos]$ ls -lh bronca
```

```
ls: bronca/existo.txt: Permission denied
ls: bronca/eutambem.txt: Permission denied
total 0
```

Vamos liberar o acesso a essa pasta, atribuindo direito de execução a todos os usuários (DONO, GRUPO e OUTROS)... Esse comando poderia ser executado das seguintes maneiras:

chmod 711 bronca

chmod a+x bronca

chmod u+x,g+x,o+x bronca

Por que, no numérico, não foi feito 111? Porque isso iria tirar os direitos atuais da pasta (USUÁRIO DONO com rw seria retirado). Portanto, coloquei o 7 para o USUÁRIO DONO ter 111 ("rw" que já tinha e "x" que passou a ter). Se fosse colocado 111, a definição seria essa, sem considerar o que se tinha anteriormente (essa é a vantagem do modo texto com os sinais de + e -... eles apenas adicionam ou retiram aquele determinado direito, mantendo os demais como estavam antes.

Depois de liberada permissão de execução da pasta bronca, seu conteúdo poderá ser visualizado pelo usuário...

```
[joao@computer documentos]$ ls -lh bronca
total 4,0K
-rw-r--r-- 1 joao joao 29 Jul 16 13:17 eutambem.txt
-rw-r--r-- 1 joao joao 0 Jul 16 13:17 existo.txt
```

2.2) chown (Change Owner - Mudar o DONO): esse comando, que só pode ser executado pelo usuário ROOT, permite que um arquivo mude de proprietário. Pode-se definir o USUÁRIO DONO e o GRUPO DONO de um arquivo através deste comando.

Aposto que você acabou de franzir a testa e se perguntar: "pra que eu preciso saber disso?"... Pois é, amigo aluno... é lasca, né? Mas, se vão perguntar (e existe a chance disso acontecer), é melhor estar preparado e deixar a decepção de não passar para os concorrentes, não acha?!

Sintaxe: `chown <usuario:grupo> <arquivo>`

Exemplo: se quisermos que o arquivo `instrucoes.doc` passe a pertencer ao usuário "ana" e ao grupo "rh", fazemos o seguinte:

```
[root@computer documentos]# chown ana:rh instrucoes.doc

[root@computer documentos]# ls -lh instrucoes.doc
-rw-rwxr-x 1 ana rh 4,2K Jul 8 18:40 instrucoes.doc
```

Claro que o usuário e o grupo devem existir previamente (isso é bem óbvio!)

Esse comando não altera as permissões do arquivo: se ele estava com `rwxr--r--`, vai continuar assim. Esse comando só altera quais os donos do arquivo!

3) Comandos de Manipulação de usuários

3.1) useradd (adicionar usuário): esse comando permite criar uma nova conta de usuário no sistema. É um comando extremamente complexo, cheio de opções: vou listar apenas algumas delas:

sintaxe: `useradd <opções> conta`

Entre as opções, temos:

-u uid: uid (ou User ID - Identificação do Usuário) é um número que identifica o usuário de forma única. Esse número pode ser dado automaticamente pelo sistema Linux se não for especificado durante a execução deste comando.

-p password: Para definir a senha da nova conta.

-g group: Definir a que grupo principal o novo usuário vai pertencer.

-G group1, group2, etc.: Definir a quais outros grupos o usuário vai pertencer (o G é maiúsculo). Sim! Um usuário pode pertencer a vários grupos!

-d directory: Define qual o diretório pessoal do usuário (normalmente `/home/<nome>`)

Então, vamos testar: se for necessário criar um usuário chamado pedro, que fará parte do grupo diretoria, mas também fará parte dos grupos rh e informática, cuja senha inicial (depois ele pode mudar) será "casa123" e o diretório pessoal será `/home/pedro`, devemos digitar:

```
[root@computer documentos]# useradd -g diretoria  
-G rh,informática -d /home/pedro -p casa123 pedro
```

E pronto! Note que não definimos o UID (número que identifica o usuário perante o sistema), pois deixamos que o próprio sistema definisse isso... O UID, só para explicar, é um número (101, 304, 212...) que identifica o usuário. O Linux atribui os direitos aos usuários pelos seus UIDs... não pelos nomes que costumamos usar para nos identificar.

O comando `useradd` pode ser escrito como `adduser` (em algumas distribuições, há diferenças entre eles). **Esse comando só pode ser executado pelo root!**

3.2) groupadd (Adicionar Grupo): comando para a criação de um novo grupo de usuários. Esse comando também só pode ser executado pelo usuário administrador (root).

sintaxe: `groupadd -g GID nome`

Fácil de usar, esse comando pede algumas opções (novamente, só listo a importante: o GID, ou Group ID - Identificação do Grupo).

- g GID: é um número que identifica o grupo perante o sistema Linux. Se não for especificado, o Linux criará automaticamente o GID para o grupo.

Vamos criar um grupo chamado auditoria:

```
[root@computer documentos]# groupadd -g 80 auditoria
```

Com isso, os usuários criados a partir deste ponto já poderão fazer parte do grupo auditoria. Além disso, os arquivos do sistema já poderão ser atribuídos a esse grupo por meio do comando `chown`. Tudo certo?

Caso você não saiba qual GID usar, não especifique essa opção... O Linux vai usar o GID que estiver disponível!

3.3) passwd (password - senha): Esse comando altera a senha de um usuário. Ele pode ser executado por qualquer usuário (mas só terá efeito em sua própria conta). Se o

root executar o comando `passwd`, ele terá o direito de especificar qual será o usuário cuja senha será alterada.

Sintaxe:

`passwd` (para o usuário alterar sua própria senha) ou

`passwd` usuário (para o root alterar a senha de um outro usuário).

Lembre-se de que o root é o usuário mais poderoso do sistema: ele pode fazer basicamente qualquer coisa com o Linux e tem acesso a todos os arquivos do sistema! Root é o administrador do sistema Linux.

Caso o root queira alterar a senha do usuário pedro, segue:

```
[root@computer documentos]# passwd pedro
digite a nova senha para 'pedro':
redigite a nova senha:

senha alterada com sucesso.
```

3.4) su (Super User – Super Usuário): esse comando permite que um usuário qualquer se torne, momentaneamente, o root. Isso serve para que o usuário possa realizar alterações no sistema sem ter que reiniciar o computador.

Sintaxe: `su`

Depois de executado o comando, o usuário será cobrado pela senha do root (claro, ou você acha que seria assim tão fácil assumir o papel de root?). É para o administrador do sistema, que está trabalhando nele como um usuário normal, poder assumir seu lugar de direito naquele momento.

```
[joao@computer documentos]$ su
Senha do 'root':

[root@computer documentos]#
```

Note que o sistema permanece no mesmo diretório, apenas, agora, o “Clark Kent” tirou os óculos e virou o “Superman”.

Para tudo voltar ao normal (ou seja, o usuário deixar de ser root e voltar a ser quem era antes, basta digitar o comando **exit**.

4) Outros Comandos Interessantes

4.1) man (Manuais): esse comando é bastante útil para os usuários de Linux mais experientes (e para os mais curiosos). Esse comando permite o acesso às páginas dos manuais (de explicação) dos comandos Linux. Caso haja alguma dúvida em algum comando, talvez um:

sintaxe: `man <comando>`

Veja um exemplo do comando `man` sendo usado para verificar o manual do comando `tree`:

```
[joao@computer documentos]$ man tree
```

(Aqui aparecem as explicações do comando `Man`, que eu não coloquei para não consumir muitas páginas)

4.2) cat (GATO - tou brincando, cat vem de concatenar): esse comando permite que sejam unidos (concatenados) os conteúdos de dois arquivos de texto e que esse resultado seja mostrado na tela para o usuário. Muito normalmente se usa esse comando apenas para visualizar o conteúdo de um único arquivo de texto.

Sintaxe: `cat <arquivo1> <arquivo2>` (normalmente, apenas `cat <arquivo>`)

Veja isso:

```
[joao@computer documentos]$ ls -lh

total 200K
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:40 apresentacoes
-rw-rw-r-- 1 joao joao 4,7K Jul 8 18:40 avisos.doc
drwxrwxrwx 2 joao joao 4,0K Jul 16 13:17 bronca
lrwxrwxrwx 1 joao joao 14 Jul 8 19:07 dic.dic -> dicionario.dic
-rw-rw-r-- 1 joao joao 61K Jul 8 18:40 dicionario.dic
-rw-rw-r-- 1 joao joao 4,7K Jul 8 18:41 impressora_manual.doc
-rw-rwxr-x 1 joao joao 4,2K Jul 8 18:40 instrucoes.doc
-rw-rw-r-- 1 joao joao 31K Jul 8 18:40 jovem_galileu.mp3
-rwxr-xr-x 1 joao joao 14K Jul 8 18:40 lattes.rtf
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:48 planilhas
-rw-rw-r-- 1 joao joao 31K Jul 8 18:41 poesia.pdf
-rw-r--r-- 1 joao joao 161 Jul 18 11:25 prova.txt
-rw-r--r-- 1 joao joao 586 Jul 18 11:24 teste.txt
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:47 textos
drwxrwxr-x 2 joao joao 4,0K Jul 8 18:45 web
```

Reparou nos arquivos `prova.txt` e `teste.txt`? Vamos dar uma olhada em seu interior...

```
[joao@computer documentos]$ cat teste.txt

Este arquivo é só um teste para o comando cat.
Vamos concatenar seu conteúdo com o conteúdo de outro arquivo qualquer.
Esta é a terceira linha de texto deste arquivo.
Já estamos na quarta.
Estes arquivos de texto puro são muito usados em programação.
Nós, por outro lado, preferimos usar documentos mais bem elaborados, como os do Word.
Estou novamente escrevendo outra linha. Essa é a sétima.
Oi tava na peneira, oi, tava peneirando...
Nona, passa a pizza, per favore.
Desce mais, para ver a próxima...
Continuando a digitar...
Testando a Décima segunda...
Treze é um número de sorte.
```

```
[joao@computer documentos]$ cat prova.txt
```

Esse é o arquivo prova.txt.

Esse arquivo será concatenado...

Vamos testar o comando cat e os demais comandos de texto.

Essa é a quarta linha!

Final do arquivo.

Agora vamos "unir os dois" e mostrar o resultado na tela (Essa união não é estável, é somente na tela mesmo, ou seja, os dois arquivos não serão fundidos ou coisa parecida... eles permanecerão do mesmo jeito! A "concatenação" só acontece na tela do usuário.

```
[joao@computer documentos]$ cat prova.txt teste.txt
```

Esse é o arquivo prova.txt.

Esse arquivo será concatenado...

Vamos testar o comando cat e os demais comandos de texto.

Essa é a quarta linha!

Final do arquivo.

Este arquivo é só um teste para o comando cat.

Vamos concatenar seu conteúdo com o conteúdo de outro arquivo qualquer.

Esta é a terceira linha de texto deste arquivo.

Já estamos na quarta.

Estes arquivos de texto puro são muito usados em programação.

Nós, por outro lado, preferimos usar documentos mais bem elaborados, como os do Word.

Estou novamente escrevendo outra linha. Essa é a sétima.

Oi tava na peneira, oi, tava peneirando...

Nona, passa a pizza, per favore.

Desce mais, para ver a próxima...

Continuando a digitar...

Testando a Décima segunda...

Treze é um número de sorte.

E então? Tudo OK?

4.3) tail (cauda): esse comando permite visualizar as últimas partes de um arquivo de texto (o padrão é 10 linhas).

Sintaxe: tail <opções> arquivo

Entre as opções, estão:

-n X: onde X é o número de linhas que se deseja visualizar (se esta cláusula não for especificada, respeita-se o padrão).

-c X: onde X é o número de caracteres que se deseja visualizar (idem acima).

Exemplo:

```
[joao@computer documentos]$ tail -n 4 teste.txt
```

```
Desce mais, para ver a próxima...
```

```
Continuando a digitar...
```

```
Testando a Décima segunda...
```

```
Treze é um número de sorte.
```

```
[joao@computer documentos]$ tail -c 20 prova.txt
```

```
a!
```

```
Final do arquivo.
```

4.3) head (cabeçalho): permite, de forma análoga ao tail, visualizar a parte inicial de um arquivo de texto.

sintaxe: head <opções> arquivo

-n X: define o numero de linhas a serem vistas (o padrão, caso não seja especificado nada, são 10 linhas).

-c X: define o número de caracteres a serem vistos.

Exemplo:

```
[joao@computer documentos]$ head -n 2 prova.txt
```

```
Esse é o arquivo prova.txt.
```

```
Esse arquivo será concatenado...
```

```
[joao@computer documentos]$ head teste.txt
```

```
Este arquivo é só um teste para o comando cat.
```

```
Vamos concatenar seu conteúdo com o conteúdo de outro arquivo qualquer.
```

```
Esta é a terceira linha de texto deste arquivo.
```

```
Já estamos na quarta.
```

```
Estes arquivos de texto puro são muito usados em programação.
```

```
Nós, por outro lado, preferimos usar documentos mais bem elaborados, como os do Word.
```

```
Estou novamente escrevendo outra linha. Essa é a sétima.
```

```
Oi tava na peneira, oi, tava peneirando...
```

```
Nona, passa a pizza, per favore.
```

```
Desce mais, para ver a próxima...
```

Viram o uso do padrão de 10 linhas? (quando se usa o comando head ou tail sem as cláusulas de opções).

4.4) more (mais): é um programa chamado de "paginador", porque permite a leitura de um arquivo de texto longo de "página em página", ou melhor, de "tela em tela" no computador. É simples entender: imagine um arquivo de texto com uma quantidade excessiva de conteúdo (centenas de linhas)... Você acha que dá pra ler ele todo numa tela só? Claro que não, e o padrão do Linux é passar tudo de uma vez, até chegar ao final do arquivo (se usarmos o comando cat). Basta, portanto, usar o comando more ao invés do cat para visualizar tal arquivo.

sintaxe: more <arquivo>

Não dá pra mostrar ele aqui, mas quando se usa o comando more, o texto de um arquivo é colocado na tela até preencher a tela toda. Depois de preenchida por completo, o comando pára e fica esperado pelo usuário. Para passar de uma tela para outra, usa-se a BARRA DE ESPAÇO (para passar a tela toda) ou a tecla ENTER (para descer de linha em linha).

Um outro comando muito semelhante ao more (porém, com alguns recursos a mais) é o comando **less (menos - um trocadilho com o nome more)**. Esse comando é utilizado de forma muito semelhante ao more (basta less <arquivo> e usar ESPAÇO e ENTER também).

4.5)grep (???): Esse comando é usado para localizar trechos dentro de um arquivo de texto. Note, novamente, que quando se fala em arquivo de texto, não me refiro aos arquivos do Word, que são documentos, mas a arquivos de texto puro mesmo, como os que são criados pelo programa bloco de notas, do Windows (texto puro, sem formatação).

sintaxe: grep <opções> trecho arquivo

Como ficaria?

```
[joao@computer documentos]$ grep linha teste.txt
```

```
Esta é a terceira linha de texto deste arquivo.
```

```
Estou novamente escrevendo outra linha. Essa é a sétima.
```

O Padrão do comando grep não é retornar apenas a palavra que se procurou, mas a linha inteira, daí o fato de ter retornado a terceira e a sétima linhas inteiras.

4.5) find (encontrar): Esse comando permite encontrar arquivos em um determinado diretório do sistema de arquivos do Linux.

sintaxe: find diretório <opções>

Entre as opções, podemos citar:

-name <nome>: para apresentar o nome do arquivo (<nome>) como critério da pesquisa.

-user <usuário>: para localizar arquivos que pertencem ao usuário <usuário>.

-group <grupo>: para localizar arquivos que pertencem ao grupo <grupo>.

Exemplo: caso se queira encontrar um arquivo chamado **teste.doc**, localizado em algum lugar dentro do diretório **/home**, usa-se o seguinte:

```
[joao@computer documentos]$ find /home -name teste.doc

/home/joao/documentos/teste.doc
```

O nome do arquivo deve estar escrito por completo.

4.6) tar (Tape Archiver - Arquivador em Fita): é um programa que muitos confundem com o compactador, mas o que esse programa faz é EMPACOTAR vários arquivos em um só. O comando tar serve para transformar vários arquivos em um, mas não compactando-os (para isso, é necessário um outro programa, o Gzip, ou Bzip, compactadores comuns no Linux).

sintaxe: tar <opções> <arquivo.tar> arquivo1 arquivo2 arquivo3...

A lista de opções do comando tar inclui:

-c: criar o arquivo empacotado.

-x: extrair arquivos de um arquivo empacotado.

-f: forçar a operação.

-v: Verbose (ou seja, ficar "narrando" o que está acontecendo, enquanto vai sendo executado).

Vejamos a listagem dos arquivos do diretório cesppe:

```
[joao@computer cesppe]$ ls -lh

total 1,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 aula16.sxw
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 7,3K Jul 18 13:01 comandos.txt
-rw-r--r-- 1 joao joao 16K Jul 18 13:01 final.txt
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 modulos.doc
-rw-r--r-- 1 joao joao 15K Jul 18 13:01 quase.txt
-rw---xr-- 1 joao joao 4,0K Jul 18 13:01 teste
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

Agora, vamos criar um arquivo email.tar contendo os arquivos apostila.rtf, comandos.doc, teste.pdf e unico.pdf...

```
[joao@computer cesppe]$ tar -cf email.tar apostila.rtf comandos.doc teste.pdf
unico.pdf

[joao@computer cesppe]$ ls -lh

total 2,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 aula16.sxw
```

```
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 7,3K Jul 18 13:01 comandos.txt
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 16K Jul 18 13:01 final.txt
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 modulos.doc
-rw-r--r-- 1 joao joao 15K Jul 18 13:01 quase.txt
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

Note a presença do arquivo `email.tar`, com 1,1MB. O tamanho deste arquivo se deve à soma dos arquivos que o formaram.

Para compactar o arquivo em questão, utiliza-se a opção `-z` no comando `tar`:

```
[joao@computer cespe]$ tar -czf menor.tar.gz aula16.swx comandos.doc modulos.doc
```

```
[joao@computer cespe]$ ls -lh
```

```
total 2,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 aula16.swx
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 7,3K Jul 18 13:01 comandos.txt
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 16K Jul 18 13:01 final.txt
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 menor.tar.gz
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 modulos.doc
-rw-r--r-- 1 joao joao 15K Jul 18 13:01 quase.txt
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

A extensão `tar.gz` não é necessária (nem a `tar` no exemplo anterior), que podem ser quaisquer extensões de que o usuário goste. Apenas por questões de costume eu utilizei estas aqui. A extensão `tar` somente para indicar que o arquivo é um pacote sem compactação e a extensão `tar.gz` para indicar que este passou por um processo de compactação.

Vamos apagar todos os arquivos do diretório `cespe`, com exceção desses dois pacotes, criados nesses últimos exemplos...

```
[joao@computer cespe]$ rm -f apostila.rtf aula16.swx comandos.doc comandos.txt
final.txt modulos.doc quase.txt teste.pdf unico.pdf
```

```
[joao@computer cespe]$ ls -lh
```

```
total 1,1M
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 menor.tar.gz
```

Sim, antes que você pergunte: dá pra apagar mais de um arquivo, no comando `rm`, colocando os nomes dos arquivos separados por espaço (como no comando `mkdir`) e em outros vários.

Agora vamos usar a opção do comando `tar` que extrai arquivos de dentro de um arquivo-pacote.

```
[joao@computer cespe]$ tar -xvf email.tar
comandos.doc
unico.pdf
apostila.rtf
teste.pdf

[joao@computer cespe]$ ls -lh

total 2,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 menor.tar.gz
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

Como o arquivo `menor.tar.gz` foi compactado (com a opção `-z`), devemos usá-la para descompactar o conteúdo do pacote enquanto extraímos os arquivos dele. Daí a importância de diferenciar os arquivos com extensões variadas (`gz` para indicar a compactação, por exemplo), pois, sem isso, não seria possível distinguir se o arquivo estava ou não compactado.

A única forma de saber, fora pela extensão, é que, na tentativa de usar as cláusulas `-xvf` em um arquivo COMPACTADO, o comando `tar` retornará um erro, o que indicará a necessidade de usar as opções `-xzvf` (com "z" a mais). O mesmo vale para o contrário (usar `-xzvf` para um arquivo `tar` que não está compactado).

Vamos ao exemplo do uso do `z`:

```
[joao@computer cespe]$ tar -xzvf menor.tar.gz
aula16.sxw
comandos.doc
modulos.doc

[joao@computer cespe]$ ls -lh

total 2,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 aula16.sxw
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 menor.tar.gz
```

```
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 modulos.doc
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

A cláusula `-f`, como na maioria dos comandos, significa "fazer forçado", ou seja, ignorando qualquer inconveniente no comando (como as necessidades de perguntas para o usuário)... No caso, para o nosso comando, essa cláusula serve para que não se questione o usuário diante, por exemplo, de arquivos criados em duplicata (devido à descompactação de dois arquivos com o mesmo nome). O arquivo `comandos.doc`, por exemplo, lá em cima, foi sobrescrito na segunda descompactação, porque ele estava dentro dos dois arquivos (o `email.tar` e o `menor.tar.gz`), sem incomodar o usuário com tal pergunta.. Se você quiser ser incomodado com essas perguntas, então não use a opção `-f`.

4.7) alias (apelido): esse comando cria um "apelido" para vários comandos. Por exemplo, se você estiver com saudades do DOS e quiser usar comandos como o `DIR`, basta criar um alias para o `DIR`. Assim...

sintaxe: `alias apelido='comando desejado'` (tudo junto)

Como em:

```
[joao@computer cespe]$ alias dir='ls -lh'

[joao@computer cespe]$ dir

total 2,1M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 apostila.rtf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 aula16.sxw
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 comandos.doc
-rw-r--r-- 1 joao joao 7,3K Jul 18 13:01 comandos.txt
-rw-r--r-- 1 joao joao 1,1M Jul 18 14:43 email.tar
-rw-r--r-- 1 joao joao 16K Jul 18 13:01 final.txt
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 modulos.doc
-rw-r--r-- 1 joao joao 15K Jul 18 13:01 quase.txt
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 teste.pdf
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 unico.pdf
```

O comando `alias` não registra essa informação para sempre, mas apenas enquanto o sistema estiver ligado. Para atribuir o apelido de forma definitiva, deve-se alterar um arquivo de configuração do Linux.

Usando Caracteres Coringa

Caracteres Coringa são aqueles que usamos para representar outros caracteres, no intuito de facilitar a digitação de um comando. Os caracteres que mais conhecemos para essa finalidade são: `*` (asterisco) e `?` (interrogação).

A interrogação serve para substituir um único caractere em um comando qualquer. Veja, como exemplo, o conteúdo do diretório esaf, mostrado abaixo:

```
[joao@computer esaf]$ ls -lh

total 1,7M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova1.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova2.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova3.pdf
-rw-r--r-- 1 joao joao 800K Jul 18 14:43 prova36.pdf
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 coment.doc
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 ponto.pdf
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 aula.swx
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 teste.doc
```

Caso seja necessário excluir os arquivos prova1.pdf, prova2.pdf e prova3.pdf, pode-se fazê-lo do método antigo e doloroso (na verdade, é fácil fazer isso, mas vamos digitar muito!).

```
[joao@computer esaf]$ rm -f prova1.pdf prova2.pdf prova3.pdf
```

Que tal se usássemos uma característica comum aos nomes dos três condenados para podermos apagá-los? Sim, "prova", um número qualquer e depois ".pdf". Que tal agora?

```
[joao@computer esaf]$ rm -f prova?.pdf
```

A interrogação é usada para substituir um caractere na posição em que é colocada, logo, a expressão "prova?.pdf" abrangerá os arquivos prova1.pdf, prova2.pdf e prova3.pdf, mas não apagará o prova36.pdf, porque este tem 2 caracteres entre "prova" e ".pdf" (e a interrogação significa apenas um caractere).

Caso se deseje apagar todos os arquivos que comecem com a letra p e tenham extensão pdf (são quatro no exemplo acima), pode-se usar o caractere de *, que pode representar qualquer quantidade de caracteres (eu chamo o * de "qualquer coisa").

Então fica assim:

```
[joao@computer esaf]$ rm -f p*.pdf
```

Todos os 4 arquivos que respeitam o critério apresentado serão apagados.

Mas os caracteres coringa não servem somente para o comando rm, mas para grande parte dos comandos vistos até aqui. Exemplo, deseja-se copiar todos os arquivos que têm extensão "doc" para o diretório /home/pedro... O comando ficaria assim:

```
[joao@computer esaf]$ cp *.doc /home/pedro
```

Se você quiser executar um comando com todos os arquivos do diretório atual, use a referência * somente. Por exemplo, se deseja mover todos os arquivos do diretório atual para dentro do diretório /etc/users:

```
[joao@computer esaf]$ mv * /etc/users
```

O * sendo usado como alvo do comando significa TODOS OS ARQUIVOS daquele diretório. Então que tal compactar todos os arquivos do diretório cespe em um único arquivo chamado esaf.tar.gz?

```
[joao@computer esaf]$ tar -czf esaf.tar.gz *
```

O * representa todos os arquivos que serão adicionados ao arquivo esaf.tar.gz, ou seja, todos os arquivos daquele diretório. O resultado pode ser visto abaixo:

```
[joao@computer esaf]$ ls -lh

total 2,4M
-rw-r--r-- 1 joao joao 756K Jul 18 17:12 esaf.tar.gz
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova1.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova2.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova3.pdf
-rw-r--r-- 1 joao joao 800K Jul 18 14:43 prova36.pdf
-rw-r--r-- 1 joao joao 18K Jul 18 15:08 coment.doc
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 ponto.pdf
-rw-r--r-- 1 joao joao 24K Jul 18 13:01 aula.swx
-rw-r--r-- 1 joao joao 975K Jul 18 13:01 teste.doc
```

Tanto o * quanto a ? são coringas conhecidos dos usuários do DOS. Mas existe um outro coringa interessante no Linux (na verdade, existem diversos outros, mas que não são necessários, creio eu)...

Observe o resultado do comando abaixo: a listagem de conteúdo de um diretório chamado confuso...

```
[joao@computer confuso]$ ls -lh

total 2,9M
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova1.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova2.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova3.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova4.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova5.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf
```

Pense agora que se precisa apagar os arquivos prova1.pdf a prova5.pdf (ou seja, os arquivos prova 6, prova7, prova8 e prova9 serão mantidos). E aí?

Que tal esse?

```
[joao@computer confuso]$ rm -f prova?.pdf
```

Não funciona, não é? Esse comando vai apagar todos os arquivos, não é? Isso mesmo...

Então, vamos usar os colchetes!

```
[joao@computer confuso]$ rm -f prova[1-5].pdf

[joao@computer confuso]$ ls -lh

total 2,9M
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf
```

Dentro dos colchetes, colocamos expressões para o caractere que vai ser substituído: 1-5 indica "de 1 a 5". Se for necessário escolher os possíveis valores dos colchetes aleatoriamente (sem seqüência), usa-se a vírgula, como em:

```
[joao@computer confuso]$ rm -f prova[6,9].pdf
```

Esse comando apaga apenas os arquivos prova6.pdf e prova9.pdf.

O mesmo raciocínio funciona para letras ao invés de números (provaa.pdf, provab.pdf, provac.pdf, etc..)

Lembre-se, as expressões dentro dos colchetes representam um único caractere.

6) Redirecionamento de Comandos

Normalmente quando executamos um comando no Linux, a idéia é essa:

```
[prompt] comando Entrada do Comando <ENTER>
Saída do Comando
Saída do Comando
Saída do Comando
[Novo Prompt]
```

A idéia é simples, mas é mais fácil se virmos um exemplo prático:

```
[joao@computer documentos]$ ls -lh confuso

total 2,9M
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf

[joao@computer documentos]$
```

ls -lh: é o comando em si, com suas opções.

Confuso: é a entrada do comando, que significa, no caso do ls, a pasta que terá seu conteúdo visualizado. Já sabemos que, caso a entrada seja suprimida, será entendida como entrada a pasta atual (diretório corrente).

A entrada de um comando é tao-somente a (ou as) informação de que o comando necessita para ser executado.

As linhas restantes são a saída do comando, ou seja, sua resposta para o usuário. Normalmente, as saídas dos comandos acontecem na tela, no próprio shell (não seria inteligente de outra forma, porque o usuário tem que ler o que o comando quer dizer em resposta, não é?).

Vamos começar a redirecionar esses comandos. Primeiro, começaremos com os sinais de > e >>.

Veja isso:

```
[joao@computer confuso]$ ls -lh

total 2,9M
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf

[joao@computer confuso]$ ls -lh > lista.txt (note aqui!)

[joao@computer confuso]$
```

Note que no segundo comando ls não teve saída (na verdade, ele teve, mas ela não foi para a tela). Quando usamos o sinal de > (maior), estamos direcionando a saída de um comando para outro local. No caso do nosso exemplo, direcionamos a saída do comando ls -lh para um arquivo chamado lista.txt, que pode ser visto a seguir:

```
[joao@computer confuso]$ ls -lh

total 2,9M
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf
-rw-r--r-- 1 joao joao 345B Jul 18 17:55 lista.txt

[joao@computer confuso]$ cat lista.txt

total 2,9M
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova6.pdf
-rwxr-xr-x 1 joao joao 14K Jul 18 13:01 prova7.pdf
-rw-r--r-- 1 joao joao 12K Jul 18 13:01 prova8.pdf
-rw-r--r-- 1 joao joao 7,2K Jul 18 13:01 prova9.pdf

[joao@computer confuso]$
```

Note que o que está em vermelho não é o resultado de um ls, mas o conteúdo do arquivo lista.txt (que conseguimos através do comando cat). Esse arquivo tem, como conteúdo de texto, o resultado do comando ls -lh que direcionamos para ele no exemplo anterior.

Quando usamos o direcionador > para um arquivo que não existe, ele será criado. Caso o arquivo destino já exista (não era o caso do lista.txt) e tenha conteúdo, o conteúdo anterior é apagado para dar lugar ao novo conteúdo (ou seja, a saída do comando que foi direcionado).

O direcionador >> é semelhante, mas difere no que diz respeito a arquivos que já existem. Quando se usa o >>, o conteúdo anterior do arquivo não é apagado, mas o conteúdo novo é ADICIONADO depois do conteúdo antigo do arquivo. Ou seja, o >> adiciona a saída atual para um arquivo sem apagar o que havia no arquivo.

Ainda há mais um direcionador interessante de se conhecer o | (pipe – lê-se páipe), que é aquela barrinha vertical no teclado.

A função do pipe é muito interessante: ele redireciona a saída de um comando para a entrada do outro comando. Como assim?

```
[prompt]$ Comando1 | Comando2
```

A resposta que for dada ao comando1 não será apresentada na tela, mas será dada ao comando2, que processará essa entrada e dará seu resultado na tela.

Veja só: digamos que um comando ls -lh resulte em uma quantidade grande de linhas (imagine uns 400 arquivos). Como isso iria se apresentar? Em uma tela só, correndo feito louco, até atingir o fim do comando, não daria para ler nada a não ser o que estiver na última tela (arquivos finais).

Seria muito bom se pudéssemos fazer uso do more aqui, não é? Como...

```
[joao@computer confuso]$ more ls -lh
```

Mas isso não é possível (não dá certo) dessa maneira. A entrada do comando more tem que ser um texto, não um comando em si... Como fazer isso? Assim...

```
[joao@computer confuso]$ ls -lh | more
```

O "|" (pipe) faz a saída do comando ls -lh (que se apresenta como um texto, em que os vários arquivos aparecem em linhas uma em cima da outra) servir de entrada para o comando more, que paginará isso, apresentando na tela a listagem de arquivos aos poucos, de tela em tela.