1 – Unix

Em 1965 formou-se um grupo de programadores, incluindo Ken Thompson, Dennis Ritchie, Douglas McIlroy e Peter Weiner, num esforço conjunto da AT&T (Laboratórios Bell), da General Electric (GE) e do MIT (Massachussets Institute of Technology) para o desenvolvimento de um sistema operacional chamado Multics.

O Multics deveria ser um sistema de tempo compartilhado para uma grande comunidade de usuários. Entretanto, os recursos computacionais disponíveis à época, particularmente os do computador utilizado, um GE 645, revelaram-se insuficientes para as pretensões do projeto. Em 1969, a Bell retirou-se do projeto. Duas razões principais foram citadas para explicar a sua saída. Primeira: três instituições com objetivos díspares dificilmente alcançariam uma solução satisfatória para cada uma delas (o MIT fazia pesquisa, AT&T monopolizava os serviços de telefonia americanos e a GE queria vender computadores). A segunda razão é que os participantes sofriam da síndrome do segundo projeto e, por isso, queriam incluir no Multics tudo que tinha sido excluído dos sistemas experimentais até então desenvolvidos.

Ainda em 1969, Ken Thompson, usando um ocioso computador PDP-7, começou a reescrever o Multics num conceito menos ambicioso, batizado de Unics, usando linguagem de montagem (*assembly*). Mais tarde, Brian Kernighan rebatizou o novo sistema de Unix.

Um marco importante foi estabelecido em 1973, quando Dennis Ritchie e Ken Thompson reescreveram o Unix, usando a linguagem C, para um computador PDP-11. A linguagem C havia sido desenvolvida por Ritchie para substituir e superar as limitações da linguagem B, desenvolvida por Thompson. O seu uso é considerado uma das principais razões para a rápida difusão do Unix.

Marcos importantes:

https://en.wikipedia.org/wiki/Dennis_Ritchie
https://en.wikipedia.org/wiki/Ken_Thompson
https://en.wikipedia.org/wiki/BSD
https://en.wikipedia.org/wiki/Comparison_of_BSD_operating_systems

Finalmente, ao longo dos anos 70 e 80 foram sendo desenvolvidas as primeiras distribuições de grande dimensão como os sistemas BSD (na Universidade de Berkeley na Califórnia) e os System III e System V (nos Bell Labs).

Marcos importantes:

https://en.wikipedia.org/wiki/Richard_Stallman
https://en.wikipedia.org/wiki/IBM_AIX
https://en.wikipedia.org/wiki/HP-UX

Em 1977, a AT&T começou a fornecer o Unix para instituições comerciais. A abertura do mercado comercial para o Unix deve muito a Peter Weiner - cientista de Yale e fundador da Interactive System Corporation. Weiner conseguiu da AT&T, então já desnudada de seu monopólio nas comunicações e liberada para atuação no mercado de software, licença para transportar e comercializar o Unix para o computador Interdata 8/32 para ambiente de automação de escritório. O Unix saía da linha das máquinas PDP, da Digital Equipament Corporation (DEC), demonstrando a relativa facilidade de migração (transporte) para outros computadores, e que, em parte, deveu-se ao uso da linguagem C. O sucesso da Interactive de Weiner com seu produto provou que o Unix era vendável e encorajou outros fabricantes a seguirem o mesmo curso. Iniciava-se a abertura do chamado mercado Unix.

Com a crescente oferta de microcomputadores, outras empresas transportaram o Unix para novas máquinas. Devido à disponibilidade dos fontes do Unix e à sua simplicidade, muitos fabricantes alteraram o sistema, gerando variantes personalizadas a partir do Unix básico licenciado pela AT&T. De 1977 a 1981, a AT&T integrou muitas variantes no primeiro sistema Unix comercial chamado de System III. Em 1983, após acrescentar vários melhoramentos ao System III, a AT&T apresentava o novo Unix comercial, agora chamado de System V. Hoje, o Unix System V é o padrão internacional de fato no mercado Unix, constando das licitações de compra de equipamentos de grandes clientes na América, Europa e Ásia.

Marcos importantes:

http://en.wikipedia.org/wiki/Linus_Torvalds
https://en.wikipedia.org/wiki/History_of_Linux
https://en.wikipedia.org/wiki/Linux
https://lwn.net
http://www.levenez.com/unix/

Atualmente, Unix (ou *nix) é o nome dado a uma grande família de Sistemas Operativos que partilham muitos dos conceitos dos Sistemas Unix originais, sendo todos eles desenvolvidos em torno de padrões como o POSIX (Portable Operating System Interface) e outros. Alguns dos Sistemas Operativos derivados do Unix são: BSD (FreeBSD, OpenBSD e NetBSD), Solaris (anteriormente conhecido por SunOS), IRIXG, AIX, HP-UX, Tru64, SCO, Linux (nas suas centenas de distribuições), e até o Mac OS X (baseado em um núcleo Mach BSD chamado Darwin). Existem mais de quarenta sistemas operacionais *nix, rodando desde celulares a supercomputadores, de relógios de pulso a sistemas de grande porte. (http://pt.wikipedia.org/wiki/Unix)

2015
Atualmente, 97% dos supercomputadores do mundo, incluindo os top 10, mais de 80% de todos os *smartphones,* 70% dos servidores web, muitos milhões de computadores desktop e tablets, uma infinidade de *appliances* como *dvd players, washing machines, dsl modens, routers, sefl-driving cars, space stations laptops*, rodan GNU/Linux.

2 – Padrão POSIX

O IEEE cria a especificação para padronizar o desenvolvimento das duas linhas de UNIX existente na época, através do padrão IEEE 1003.1. Este padrão garante uma nomenclatura comum de muitos comandos básicos, utilitários, subsistemas e bibliotecas compartilhadas. Assim, uma vez aprendido o conceito do sistema, este conhecimento é aplicado nos demais sistemas.

3 – Minix

Até 1983, o UNIX possuia o código-fonte aberto e as universidades o utilizavam amplamente para estudos acadêmicos. A AT&T, detentora dos direitos do UNIX, fechou o código tornando-o estritamente comercial e proprietário. Assim, as universidades começaram a ensinar somente na área teórica. O MINIX foi então criado pelo professor Andrew Tanenbaum e sua equipe para estudos acadêmicos.

4 – Projeto GNU

Criado em 1983 por Richard Stallman através da FSF (Free Software Foundation) em um momento que as empresas estavam fechando seus códigos, tirando a liberdade e a interação da qual os programadores gozavam nas décadas de 60 e 70. As mudanças na legislação Norte Americana, que então coibia o monopólio, acabou por permitir (ou estimular) para que as grandes corporações fechassem seus códios tornando-os proprietários.

O GCC (GNU Compiler Collection) e o GNU Emacs foram os primeiros softwares lançado pelo projeto, ambos desenvolvidos e mantidos por Stallman e seu grupo até hoje.

Em 1999 a UNESCO juntamente com a FSF, declara os programas da FSF como patrimônios cientificos da humanidade.

What is Free Software?
“Free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech”, not as in “free beer”. (http://www.gnu.org/)

5 – Fundação Apache

Na década de 90, a internet deixa de ser uma tecnologia somente para especialistas e passa a ser oferecida para qualquer usuário, momento que surgem os primeiros navegadores web. O httpd criado inicialmente por Rob McCool tem contribuições de outros especialistas tais como Brian Behlendorf e Cliff Skolnick que culminou em uma lista de 8 contribuidores que criavam e mantinham os patches, dando nome ao projeto de Apache. Todo o desenvolvimento são feitas em cima do conceito da licença livre do GNU, até que o servidor web Apache se tornou o sistema mais utilizado no mundo, ocupando 65% do mercado mundial.

6 – GNU/Linux

Em 1991, já existiam muitos softwares mantidos pelo projeto GNU. Faltava no entanto, o Kernel (núcleo do sistema que interage com o hardware e os processos). Neste contexto se encaixa Linus Torvalds que iniciou um projeto para construir um Kernel inspirado no Minix com o objetivo de fazer um Minix melhor do que o Minix. Com algum tempo, Linus desenvolve uma versão mais estável de seu kernel que conseguia reconhecer vários comandos do UNIX e rodar um Shell.
Assim, o Linux é formado por um kernel (o código base que gerencia os recursos de hardware e software) e os aplicativos de usuários (como bibliotecas, gerenciadores de janela e aplicativos).

7 – Software Livre

São considerados softwares livres, aqueles softwares que mantêm o código-fonte aberto e disponível aos usuários, permitindo a sua utilização, alteração e distribuição, sem a necessidade de autorização do fornecedor. Isso significa que o software pode ser modificado para adequar-se melhor a determinada realidade, atividade ou trabalho, sem que haja necessidade do usuário comunicar, justificar ou solicitar autorização do desenvolvedor do programa. Além do que, este mesmo usuário pode compartilhar suas versões modificadas em rede e, com isso, beneficiar toda a sociedade com seu conhecimento.

O software livre é desenvolvido a partir de um modelo coletivo e cooperativo, auxiliado pelas tecnologias da informação e comunicação. Em contraponto ao software proprietário, que comumente é desenvolvido por grupos restritos e sob sigilo industrial, o software livre permite,

desde a sua criação, a participação voluntária de diversos colaboradores que enriquecem o processo e, por conseguinte, o produto final (ZANAGA, 2006). Ademais, por razões óbvias, este modelo colaborativo poupa esforços e evita o trabalho duplicado, vindo a possibilitar a redução dos custos no seu desenvolvimento.

A própria terminologia software livre, muitas vezes ocasiona certa confusão com o que se entende por livre. Neste caso específico, livre não significa gratuito, ou que não possa ter uso, desenvolvimento ou distribuição comercial, mas sim que oferece certas liberdades de expressão. Conforme definição da Free Software Foundation um software livre dispõem de quatro liberdades essenciais aos seus usuários, sendo elas:

0-liberdade de executar o programa, para qualquer propósito;
1-liberdade de estudar como o programa funciona e alterá-lo para ele fazer o que quiser;
2-liberdade de redistribuir cópias, de modo que você possa ajudar ao seu próximo;
3-liberdade de distribuir cópias de suas versões modificadas para os outros.

Segundo Lemos (2005), no caso de um software em regime livre a violação ao direito do autor acontece quando se impõe algum obstáculo à realização de uma dessas liberdades e, principalmente, se impede o acesso ao código fonte. Essas liberdades devem, pois, ser irrevogáveis.

Licenças

Assim como qualquer outro material, ou mesmo softwares, os conteúdos educacionais abertos também requerem o suporte de licenças que resguardam o direito de autor e determinam as possibilidades de uso. No entanto, no contexto dos REA as licenças utilizadas são mais flexíveis que as leis nacionais e internacionais dos direitos autorais e constituem-se alternativa legítima para o autor ou detentor dos direitos indicarem como pretendem que suas obras sejam utilizadas, referenciadas e até mesmo modificadas.

Desta maneira, transcende-se o modelo tradicional de propriedade intelectual e direito autoral onde “todos os direitos estão reservados” ao autor, para um modelo onde apenas “alguns direitos estão reservados”. Pode-se assim dizer que estas licenças contribuem para tornar os conteúdos gerados, bens públicos e acessíveis (ZANAGA, 2006).

A Open Content License e Open Publication License são licenças, desenvolvidas por David Wiley, pesquisador de desenvolvimento de objetos de aprendizagem, inspiradas nas referências do software livre, porém direcionadas a disponibilização e compartilhamento de conteúdos educacionais digitais. Atualmente estas licenças caíram em desuso e tem-se destacado a GNU Free Documentation License e, sobretudo, a Creative Commons License (DUTRA e TAROUCO, 2007).

A Creative Commons é uma organização não-governamental com representações em vários países, inclusive o Brasil, que dispõe de uma série de licenças que permitem ao próprio autor estabelecer as “condições de uso de sua obra”, a partir da combinação de cláusulas específicas que garantem determinadas liberdades relacionadas principalmente a: exigência do reconhecimento da autoria; permissão de cópias sem consulta prévia ao autor da obra, ou portador do direito autoral; permissão, ou não, de uso comercial da obra.

8 – Arquitetura GNU/Linux

A figura 1 mostra os principios mais importantes [1] do Kernel do Linux, e ao final desta pilha, ha um conjunto de códigos dependente da arquitetura que ativa o Linux em diversos arrays de plataformas de hardware (x86 (Intel, AMD), x86-64 (Intel EM64T, AMD64), ARM, PowerPC, Alpha, SPARC, etc) com grande penetração em sistemas embarcados, Tvs e centro de multimídia, relógios de pulso, etc.

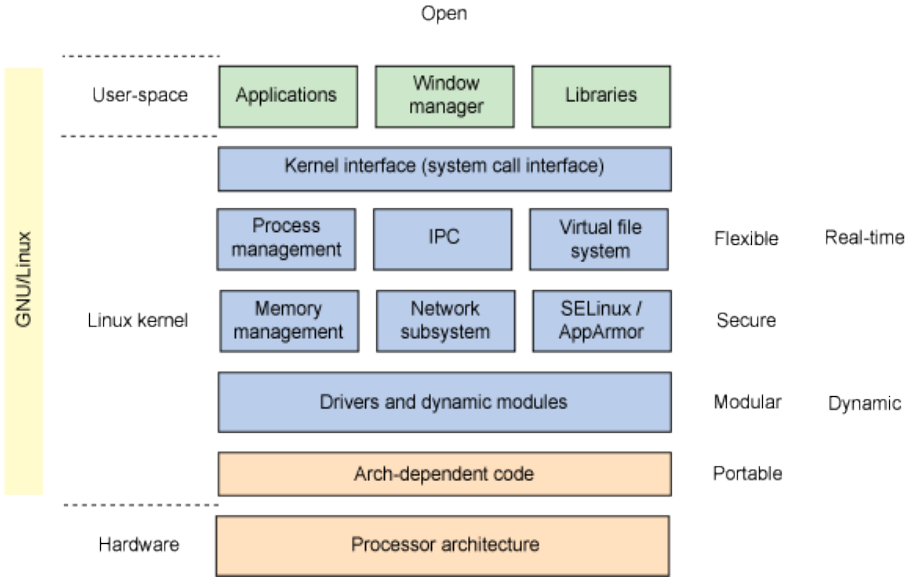
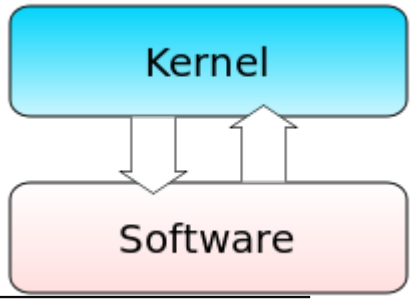


Figura 1

9 – O que é o Kernel?

É preciso imaginar um sistema operacional a partir de duas perspectivas. A primeira da perspectiva do usuário e a segunda da perspectiva do kernel. Ou seja, espaço de usuário e espaço do kernel. Este último é o responsável em conectar o espaço de usuário com todo o hardware fazendo a abstração nas chamadas de dispositivos, como leitura e gravação por exemplo. Resumidamente, o kernel não é nada mais do que um gerenciador de recursos, mesmo que seja um processo, uma memória, ou um dispositivo de hardware, intermediando assim a concorrência entre vários usuários.

10 – Arquitetura Monolítica



"Hoje o Linux é um kernel híbrido monolítico. Ao contrário dos kernels monolíticos padrão, os drivers de dispositivo são facilmente configurados como módulos, e carregados e descarregados enquanto o sistema está rodando. Também ao contrário de kernels monolíticos padrão, drivers de dispositivo podem ser pré-inseridos sob certas condições."[2]

1 http://www.ibm.com/developerworks/br/library/l-linuxuniversal/
2 http://br-linux.org/2008/01/faq-linux.html acessado em 26/05/2014

11 – Interação com Hardware

Drivers de dispositivo e extensões do kernel tipicamente rodam com acesso total ao hardware, embora alguns rodem em espaço de usuário; A característica que permite carregar e configurar drivers de dispositivos sob demanda, ou seja, enquanto o sistema está rodando, foi adicionada para corrigir o acesso a interrupções de hardware e para melhorar o suporte a multiprocessamento simétrico.

12 – Processo de inicialização do Kernel

O processo de inicialização do sistema depende do hardware no qual o linux está efetuando o boot. Vamos abstrair aqui as configurações de BIOS/CMOS e vamos considerar que o sistema pode ser inicializado de várias formas, tais como a partir de um dispositivo USB, CD/DVD, rede, etc. Os processo de inicialização inicia-se pelos seguintes estágios:

1) O loader de boot primário, que reside no MBR, é uma imagem de 512 bytes contendo o código do programa e uma pequena tabela de partição. Os primeiros 446 bytes são o loader de boot primário, que contém o código executável e um texto de mensagem de erro. Os 64 bytes seguintes formam a tabela de partição, que contém um registro para cada uma das quatro partições (cada uma com 16 bytes). O MBR termina com dois bytes, que são definidos como o número mágico (0xAA55). O número mágico funciona como verificação de validação do MBR.

2) O loader de boot secundário, ou de segundo estágio, pode ser chamado de loader de kernel. Neste estágio, sua tarefa é carregar o kernel Linux e o disco RAM inicial opcional.

a) Loaders de Boot de Estágio GRUB

O diretório /boot/grub contém os loaders de boot stage1, stage1.5, e stage2, bem como diversos loaders alternativos (por exemplo, os CR-ROMs usam iso9660_stage_1_5).

Os loaders de boot de primeiro e segundo estágios combinados são chamados de Linux Loader (LILO) ou GRand Unified Bootloader (GRUB) no ambiente x86 do PC. Como o LILO apresenta algumas desvantagens que foram corrigidas no GRUB, vejamos o GRUB.

O ponto alto do GRUB é que ele inclui o conhecimento dos sistemas de arquivo Linux. Em vez de utilizar setores brutos no disco, como o LILO faz, o GRUB pode carregar um kernel Linux a partir de um sistema de arquivos ext2 ou ext3. Isso é feito transformando o loader de boot de dois estágios em um de três estágios. O estágio 1 executa boot em um loader de boot de estágio intermediário (1,5), que compreende o sistema de arquivos específico, contendo a imagem do kernel Linux. Como exemplos, temos reiserfs_stage1_5 (para carregar a partir de um sistema de arquivos com registro de mudanças Reiser) ou e2fs_stage1_5 (para carregar a partir de um sistema de arquivos ext2 ou ext3). Quando o loader de boot do estágio intermediário estiver carregado e em execução, o loader de boot do estágio 2 poderá ser carregado.

Quando o estágio 2 estiver carregado, o GRUB poderá, mediante solicitação, exibir uma lista dos kernels disponíveis (definidos em /etc/grub.conf, com soft links de /etc/grub/menu.lst e /etc/grub.conf). É possível selecionar um kernel e até aditá-lo com parâmetros adicionais de kernel. Opcionalmente, é possível utilizar shell de linha de comandos para obter um maior controle manual sobre o processo de boot.

Com o loader de boot de segundo estágio na memória, o sistema de arquivos é consultado e a

imagem de kernel padrão e a imagem initrd são carregadas na memória. Com as imagens prontas, o loader de boot de estágio 2 chama a imagem de kernel.[3]

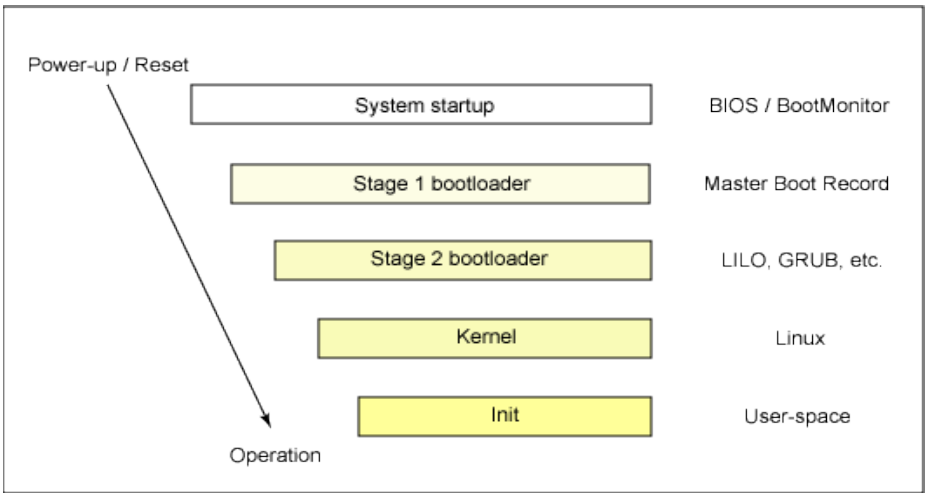
grub> **kernel /bzImage-2.6.14.2**
  [Linux-bzImage, setup=0x1400, size=0x29672e]

grub> **initrd /initrd-2.6.14.2.img**
  [Linux-initrd @ 0x5f13000, 0xcc199 bytes]

grub> **boot**

Descompactando Linux... Ok, executando boot no kernel.

Após o initrd (que carrega em memória um pequeno kernel com drivers e módulos carregáveis) o sistema pode então ter acesso aos discos e aos dispositivos e até mesmo drivers e recursos de hardware. O init é o primeiro programa no espaço de usuário carregado no sistema. O Init é compilado com a biblioteca C padrão e os processos podem ser configurados em /etc/inittab; Um sistema Linux em servidor por exemplo, pode apresentar somente um shell e em outros casos, como um dispositivo embarcado, carregar outras funções não presentes no /etc/inittab;



http://www.ibm.com/developerworks/br/library/l-linuxboot/

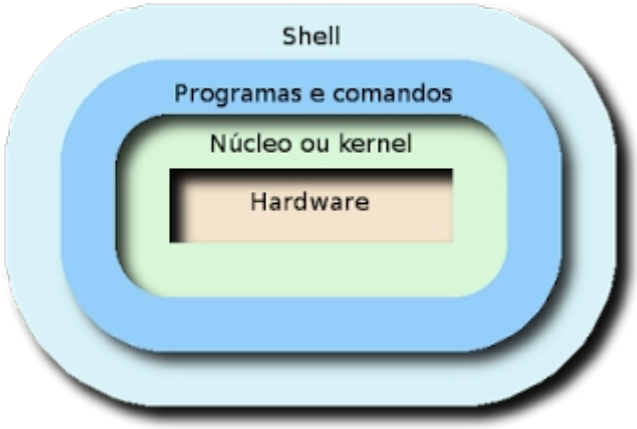
13 – O que é shell

O shell é um interpretador de comandos que está entre o usuário e o kernel do sistema operacional, fazendo todas as análises dos comandos, checando por exemplo sua sintaxe, se o comando é válido e enviando as requisições para execução. O Shell mais conhecido no mundo Linux é o bash − GNU Bourne-Again SHell. O Bash é um interpretador de comando compatível

3 http://www.ibm.com/developerworks/br/library/l-linuxboot/ acessado em 26/05/2014

com sh capaz de executar comandos pela entrada padrão ou através de um arquivo. Também incorpora características úteis do ksh e csh, fielmente seguindo as especificações POSIX (IEEE Standard 1003.1). [4]

Caso necessite informações sobre o bash, instale o bash-doc e vefifique sua documentação em /usr/share/doc/bash-doc ou sua man page (man bash);



Crédito imagem:http://apoie.org/JulioNeves/PapoI.htm

14 – Classificação dos Usuários do Sistema

O Linux é um sistema multi-usuário, portanto, permite mais de um usuário conectado simultaneamente e com mais de uma sessão por usuário. Em um sistema típico, os usuários são armazenados em /etc/passwd estes também relacioados aos seus grupos que ficam em /etc/group.
Vejamos um arquivo /etc/passwd:

```
 1 root:x:0:0:root:/root:/bin/bash
 2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
 3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
 4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
 5 sync:x:4:65534:sync:/bin:/bin/sync
 6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
 7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
 8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
 9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

O arquivo é separado por “:” delimitando cada campo. O primeiro é o nome, o segundo campo “x” indica que a senha está armazenada em /etc/shadow, o terceiro e quarto se refere ao id e grupo do usuario, quarto é um comentário sobre o usuario e seu path para o diretório home. Finalmente, o último se refere ao shell oferecido ao usuário quando fizer login ou negação do login, como por exemplo o daemon indicado na linha 2.
As senhas armazenadas em /etc/shadow somente são lidas pelo sistema ou pelo usuário root. Por padrão, a criptografia das senhas são feitas pelo MD5.

---

4 http://blog.silva.eti.br/2013/04/em-uma-aula-de-software-livre-abordei.html

Os usuários são classificados como os usuários comuns, isto é, usuários do sistema operacional e usuários de sistema cuja função é carregar serviços, como lp, mail, proxy, daemon, etc.
Os usuários pode estar também a grupos para facilitar sua administração. Os grupos, ficam armazenados em /etc/group. Vejamos um arquivo exemplo:

```
1 root:x:0:
2 daemon:x:1:
3 bin:x:2:
4 sys:x:3:
5 adm:x:4:
6 tty:x:5:
7 disk:x:6:
```

Na primeira linha, obeserve o primeiro campo. Indica o nome exclusivo do grupo, neste caso “root”. Em seguida, o “x” indica que uma senha é requiria. No próximo campo, indica poderes de “root”. Um grupo pode conter diversos usuários.

15 – O superusuário[5]

Ser root em um sistema Linux ou Unix não é somente um "direito" e sim um "privilégio". O "direito" não necessáriamente está relacionado com o privilégio e sim com diretivas a serem seguidas, enquanto "privilégio" é o "poder" investido na conta de acesso. Isto parece confuso? pode-se dizer que diretiva é o que você segue (ou deveria seguir) enquanto o privilégio é o que você tem.

Em um sistema Linux ou Unix, o usuário root é a principal conta do sistema com direitos supremos, podendo ter acesso a qualquer dispositivo, arquivos e comandos [1], inclusive para remover o sistema. Por esta razão, ao proteger esta conta do sistema, cria-se uma camada adicional na sua segurança. O interessante aqui, é o conceito de segurança do sistema operacional, seja ele um servidor ou um desktop. Note no entanto, que cada ambiente precisa de cuidados diferenciados, por exemplo, um servidor tem muitos serviços rodando, usuários com diferentes perfis e necessidades, logo, sua falha comprometerá atividades que podem significar a exposição de dados importantes e prejuizos institucionais, enquanto no desktop pode-se conviver com regras mais flexíveis (embora não recomendadas).

Em sistemas Debian, a definição de senha de root durante a instalação do sistema é opcional, ou seja, caso não seja informada, o primeiro usuário do sistema tem privilégio de sudo para atividades administrativas.

A lógica da segurança nesta questão, coloca luz para outro ponto cujo entendimento não tem tanta profundidade para iniciantes e estudantes do sistema, que é a questão de que a "segurança não é um produto e sim um conceito", por isso que vale o investimento do tempo para entender e melhorar a segurança do sistema.

Algumas recomendações de proteção para a conta root:

a-Não existem dois "root" no sistema;
b-Utilize a conta "root" para administrar, não para trabalhar;
c-Não permita acesso como "root" via acessos remotos;

---

5 http://blog.silva.eti.br/2012/07/root-direitos-privilegios-e-seguranca.html acessado 01/03/2014

d-Não permita rodar scripts não conhecidos como "root";
e-Não permita usuários com senha nula (em branco) no sistema;
f-Não permita acessos "root" ao console;
g-setuid e setgid também devem ser usados com cuidado, evitando-se a elevação de privilégios;
h-Cuide da "raiz" da sua árvore;
i-Id root = 0

a-Não existem dois "root" no sistema;
Na hipótese da necessidade de privilégios administradivos para outros usuários, utilize o sudo, dando atributos de acordo com a necessidade.

b-Utilize a conta "root" para administrar, não para trabalhar;
Não crie o hábito de logar e manter-se logado como root no seu sistema. Desta forma, cria-se a cultura de segurança em todas as atividades, desde as simples até as complexas.

c-Não permita acesso como "root" via acessos remotos;
Nunca permita que o root tenha acesso via ssh por exemplo. Ajuste seu sshd_config adequadamente:
PermitRootLogin no

d-Não permita rodar scripts desconhecidos como "root"

Diria ainda mais, não rode nem mesmo os scripts conhecidos como root, a não ser que tenha uma boa razão para isso.

e-Não permita usuários com senha nula (em branco, empty) no sistema;
Ao permitir isso, e estando o intruso no sistema, outras vulnerabilidades poderão ser alcançadas, inclusive a possbilidade de execução como sudo ou outro tipo de engenharia. Note que quem tem direitos de sudo, deve também cuidar de suas senhas. Para isso, ajuste seu sshd_config:

PermitEmptyPasswords no

f-Não permita acesso "root" ao console;
Ajustando seu securetty em /etc/securetty. Neste arquivo, são listados os terminais onde root pode logar-se. O recomendável, é que root não tenha acesso aos consoles, principalmente em sistemas firewall ou servidores onde a segurança seja uma questão suprema.

g-setuid e setgid também devem ser usados com cuidado, evitando-se a elevação de privilégios;
O sistema operacional checa estes bits identicadores a cada chamada de sistema. Então, pelo setuid e setgid permite-se ao usuário a execução com privilégios elevados temporáriamente, como por exemplo, a alteração de senha. Observe o executável passwd, cujo setgid permite a execução por qualquer conta do sistema, com privilégios temporários de root:

-rwsr-xr-x 1 root root 34740 Feb 15 2011 passwd

h-Cuide da "raiz" da sua árvore;
O sistema operacional possui uma hierarquia de diretórios onde tudo está relacionado ao / (raiz), formando a árvore de diretórios. Em alguns pontos desta árvore, é possível recuperar arquivos ou diretórios removidos, isto considerando experiência e tempo para tanto. Mas outros pontos, a

remoção pode ser fatal, como por exemplo o próprio /. A remoção (rm- rf /) por exemplo, destrói de forma irreversível a sua instalação.

i-id root = 0
O id 0 (zero) é reservado para o usuário root. Existem diversas razões para isso ser assim, incluindo a permissão dos processos do sistema. Portanto, nunca altere as permissões de um usuário para id 0, por exemplo, editando o /etc/passwd.

grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash

[1] http://www.linfo.org/root.html
man setgid
man setuid
man credentials

15.1 /etc/security

Se desejar "blindar" ainda mais seu servidor e torná-lo ainda mais seguro, observe este diretório. Nele temos por exemplo uma forma para controlar que pode ser acesso através de várias combinações: o acess.conf e group.conf oferecem uma tabela de controle para login extremamente eficiente com muitos tipos de permissões. Dentro de limits.conf, é possível fazer controle por usuário dos recursos do sistema, como número máximo de logins por usuário e o total no sistema inclusive tempo de processador e prioridade nos processos.

15.2 Reboot

Quando falamos em segurança, devemos prestar atenção em todos os detalhes. Na hipótese de um sistema estar comprometido, um reboot pode ser ainda mais desastroso, tornando os problemas piores para o syadmin e ainda podendo ter as evidências apagadas sem pensar em outros fatores. Em Linux, devemos dificultar o reboot proposital do sistema e até mesmo acidental.
Dentro de /etc/inittab basta comentar a linha (considerado Debian como exemplo):

```
# What to do when CTRL-ALT-DEL is pressed.

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

15.3 Proteção ssh

Alguns sysadmins acreditam que mudando a porta do ssh (22) para uma outra qualquer pode melhorar a segurança do serviço ssh. A isto chamamos de segurança por obscuridade e não acredite que alguém mal intencionado possa se satisfazer com pouco. O ideal é fazer pelo menos alguns ajustes no sshd como exemplificado abaixo:
•permita somente Protocol 2;
•não permita password em branco;
•não permita acesso como root;
•não permita X11 forward;
•ative timeout;

•ative AllowUsers;

Exemplo de /etc/ssh/sshd_config:

```
   Host *
   ForwardAgent no
   ForwardX11 no
   RhostsRSAAuthentication no
   RSAAuthentication yes
   PasswordAuthentication yes
   HostbasedAuthentication no
   BatchMode no
   CheckHostIP yes
   AddressFamily any
   ConnectTimeout 0
   StrictHostKeyChecking ask
   IdentityFile ~/.ssh/identity
   IdentityFile ~/.ssh/id_rsa
   IdentityFile ~/.ssh/id_dsa
   Port 22
   Protocol 2
   Cipher 3des
   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-
cbc,aes256-cbc
   Compression yes
   EscapeChar ~
   ForwardX11Trusted no
   ServerAliveInterval 160
   ServerAliveCountMax 3
   AllowUsers admin1 admin2 ...
   denyUsers ALL
   MaxStartups 5:90:10
```

(Após 5 conexões não autenticadas, 90% das conexões do ip serão recusadas e após 10 tentativas, o ip será bloqueado).

15.4 O arquivo /etc/securetty

Como o nome sugere, securetty é o terminal ou console onde o root tem acesso ao sistema. Por segurança, minha sugestão é que root não se logue em nenhum terminal. Caso você precise ter algum privilégio de root, poderá ainda usar su ou sudo para atividades administrativas. Basta comentar todas as entradas em /etc/securetty caso queira ter este tipo de segurança à conta root.
Leia este artigo e veja uma explicação humorada do que representa o root para o sistema operacional (e para você mesmo).

16 – Arquivos de inicialização

17 – Reiniciando e desligando o sistema

18 – Entrando e saindo do sistema

19 – Administração de arquivos e diretórios

20 – Hierarquia

21 – Caminhos (path)

22 – Tipos de arquivos em UNIX

23 **– Comandos Shell**

Para possibilitar que usuários possam interagir com o sistema operacional, são necessários recursos específicos para cada funcão do sistema, como um programa, um browser, dispositivo, etc. O shell oferece estes recursos através da cli (command line interface). Dentre os vários shells disponíveis, destacamos o *bash* shell, bastante compativel com o *sh*, mas contém muitas melhorias nos recursos de função e programação, incorporando recursos do shell *Korn* (ksh) e do shell *C* (csh) além de ser compatível com POSIX[6]; Podemos imaginar o shell como um programa que aceita e executa comandos e permite a construção de scripts complexos apesar de usar comandos simples.
O shell têm alguns comandos integrados, como o cd, break, exec. Outros comandos são externos. Quanto ao fluxo de E/S, o shell utiliza três tipos:

stadin: fluxo de entrada padrão, que fornece entrada para comandos;
stdout: fluxo de saída padrão, que mostra a saída dos comandos;
stderr: fluxo de erro padrão que mostra a saída de erro dos comandos.

Todo comando que escreve no vídeo (stdout) , pode ter sua saída redirecionada (para arquivo, impressora, |) e todo o comando que lê o teclado, (stdin) pode ter sua entrada redirecionada. Os sinais utilizados pelo shell para fazer o estes redirecionamentos são:

1> ou >	redireciona a saída padrão
2>	redireciona a saída padrão de erros
<	Redireciona a entrada padrão
>>	Redireciona e adiciona a saida padrão para o arquivo
2>>	Redireciona e adicona ao arquivo a saída padrão de erros

6 O padrão POSIX garante uma nomenclatura comum de muitos comandos básicos, utilitários, subsistemas e bibliotecas compartilhadas. Assim, uma vez aprendido o conceito do sistema, este conhecimento é aplicado nos demais sistemas.

A execução de comandos no shell é feita através do prompt, que pode ser de usuário ou superusuário (root); A diferença visual entre os dois tipos de prompt se dá através do caracter $ para usário ou # para o superusuário; Esta convenção é presente na maioria dos sistemas Linux;

Exemplo de prompt de usuário:

Um dos comandos mais conhecido em shell é o echo que tem como principal função mostrar algo na tela, como por exemplo:

$echo mundo
mundo

O bash tem alguns metacaracteres que tem a função de operadores como por exemplo:

- |
- &
- ;
- (
-)
- <
- >

Estes metacaracteres ao serem utilizados em partes de um texto por exmeplo, deverá ser colocado entre aspas ou com escape, utilizando-se para isso uma barra invertida (\). Nova linha e determinados metacaracteres ou pares de metacaracteres também funcionam como *operadores de controle* :

- ||
- &&
- &
- ;
- ;;
- |
- (
-)

Alguns destes operadores permitem criar uma sequência de comandos ou listas de comandos. Uma lista de comandos simples é composta por pelo menos dois comandos sepaprados por (ponto e virgula) (;), sendo executados em sequência. Em qualquer ambiente programável, os comandos retornam um status de sucesso ou falha; Os comandos do bash em Linux geralmente retornam um valor igual a zero para sucesso e um valor diferente de zero em caso de falha, possibilitando assim utilizar processamento condicional na lista usando os operadores de controle && e ||. Se dois comandos forem separados com o operador de controle &&, o segundo será executado somente se o primeiro retornar um valor de saída igual a zero. Se dois comandos forem separados com ||, o segundo será executado somente se o primeiro retornar um código de saída diferente de zero.

Exemplo com ; $ echo linha1;echo linha2; echo linha3
Exemplo com || $ echo linha1 || echo linha2 || echo linha3
Exmplo com && $ echo linha1 && echo linha2 && echo linha3

Durante a execução de um bash shell, muitas coisas constituem o ambiente, como a forma do prompt, o diretório inicial, o diretório de trabalho, o nome do shell, os arquivos abertos, as funções definidas, etc. O ambiente contém variáveis que podem ter sido definidas pelo usuário ou mesmo pelo bash. Estas variáveis também podem ser exportadas para o ambiente, para serem usadas por outros processos em execução no shell, até mesmo podem ser criadas a partir do shell atual.

As varíaveis de amviente de shell têm um nome e as referências para estas variáveis deve ser feitas pela prefixação de seu nome com '$';

Tabela	Algumas variáveis de ambiente bash comuns
USER	O nome do usuário conectado
UID	O ID de usuário numérico do usuário conectado
HOME	O diretório inicial do usuário
PWD	O diretório de trabalho atual
SHELL	O nome do shell
$	O ID de processo (ou PID) do processo de bash shell em execução (ou outro processo)
PPID	O ID de processo do processo que iniciou este processo (ou seja, o ID do processo pai)
?	O código de saída do último comando

Tecnicamente, $$ e $? são parâmetros de shell em vez de variáveis. Eles podem apenas ser mencionados; não é possível atribuir um valor a eles. Para definir uma nova variável, basta indicar um nome seguido por (=) imediatamente, lembrando que estas variáveis são sensíveis a maiúsculas e minúsculas, portanto, var1 e VAR1 são diferentes para o shell.

Para encerrar o shell é usado o comando exit; Se estiver executando o shell em uma janela de um desktop gráfico, esta janela sera fechada. Se estiver em uma sessão ssh ou telnet, a conexão será fechada; Alternativamente, pode-se usar o CTRL+d para encerrar a sessão shell.

Os comandos Linux são poderosos e precisos, seguindo o conceito de programas específicos para tarefas específicas; O shell pode facilitar muito a vida do administrador do sistema. No entanto, uma certa fluência nestes comandos podem levar algum tempo. O estudante nunca deve parar de observar os comandos e lembrar que o uso e costume é que trás a abilidade;

Abaixo alguns comandos e sua sintaxe mais usada:

ls	O ls é o comando mais básico de um shell. Ele serve para listar o conteúdo de um diretório, mostrando os arquivos que estão no mesmo. Ex: $ ls /etc
pwd	Mostra o diretório atual
cd	Muda de diretório
mkdir	Cria um novo diretório
rmdir	Remove um diretório quando fazio (rm -rf força excluir o diretório mesmo com conteúdo)
rm	Remove um arquivo/diretorio
du	Mostra tamanho do(s) diretório(s) e sub-diretório(s) (du -h)
df	Mostra a ocupção do disco rígido (df -h)
free	Mostra o tatal de meória RAM e swap;
find	Localiza arquivos em um diretório (find . -name teste.txt)
whoami	Mostra o nome do usuário corrente
who	Mostra usuários logados e seus terminais
hostname	Mostra o nome da máquina
su	Muda para um usuário especificado (su root)
echo	Echo na tela (echo “ola mundo)
cat	Imprime um arquivo na tela (cat teste.txt)
more	Imprime um arquivo na tela paginadamente
top	Apresenta
top	Apresenta recursos do sistema, processos rodando, etc

Outros comandos, considere consultar o Guia Foca[7] na seção iniciantes;

24 – Manipulando e atualizando arquivos

Um arquivo representa um conjunto de informações e é a unidade mínima de armazenamento de dados, em qualquer tipo de dispositivo. Desta forma, qualquer tipo de informação (texto, imagem, som, etc) armazenada em um destes tipos de dispositivo eletrônico estará na forma de um ou mais arquivos. Cada arquivo possui, entre outras propriedades, um nome que o identifica e um tamanho que mostra o número de caracteres (bytes) que ele contém. Importante ressaltar que o Linux distingue letras maiúsculas de minúsculas e nomes de arquivos longos, com até mais de uma centena de caracteres. Os arquivos podem ter qualquer tipo de caracteres e o último ponto, indica a extensão do arquivo, evidenciando a natureza do seu conteúdo e com qual aplicativo ele é utilizado. Importante ressaltar que no Linux não é obrigatório que um arquivo possua uma extensão, sendo portando uma convenção feita por usuários e desenvolvedores de aplicativos

7 http://www.guiafoca.org/

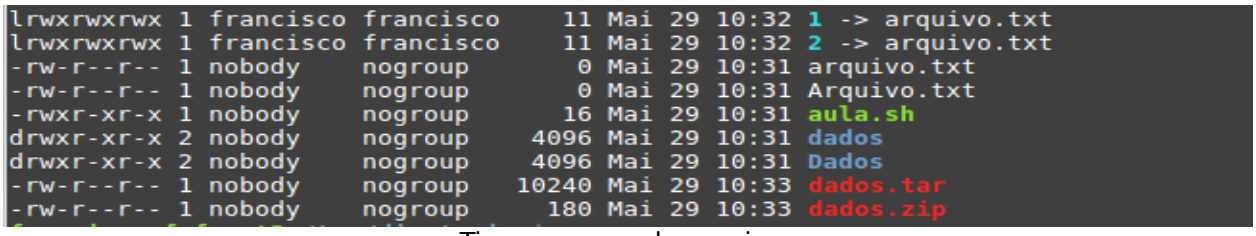
para facilitar a manipulação destes arquivos. Abaixo, algumas extensões de arquivos mais comuns de serem encontradas:

txt	Indica que o arquivo contém apenas texto (ASCII)
bak	Cópia de segurança (backup)
jpg	Arquivo que contém uma imagem em jpeg
mp3	Arquivo de som
html	Arquivo contendo hipertexto
odt	Opendocument
pdf	Documento pdf
sh	Indica um script shell

O aplicativo “file” também pode ser usado para consultar um determinado arquivo ou diretório, retornando com grande precisão informações a respeito do arquivo/diretório indicado. O aplicativo file é usado em linha de comando, com a seguinte sintaxe: “file arquivo”, ou “file diretório”.

Os arquivos também mostram uma visualização quando usados no prompt do shell, em cores diferentes, facilitando a identificação visual quando é um diretório, um executável (shell script), link simbólico, etc. Observe as seguintes cores mais comuns:

azul	O arquivo é um diretório
vermelho	Um arquivo compactado (tar, zip, rar, gzip, etc)
verde	O arquivo é um executável (shell script, executável de outro sistema operacional) etc
ciano	O arquivo é um link simbólico



Tipos comuns de arquivos

25 – Paginação, Pipe e Redirecionamento

26 – Estruturas de diretórios padrão nos Sistemas Posix

27 – Localização e Substituição

28 – Editor de Textos VI

28.1 EDICAO DE TEXTO COM VIM

o vim é um fork do vi e está presente na maioria dos sistemas operacionais modernos, por isso o dominio básico vai ajudar muito o administrador de sistema a ter condições de trabalhar sem dificuldade em todos eles.
o vim faz uso intensivo de teclado, o que na verdade, faz com que seja muito mais elegante e

funcional.
Abaixo as principais formas de utilização e dicas[8]:

28.2 MODOS DE EDICAO DE ARQUIVOS

```
vi -b arquivo -> abre arquivo no modo binario
vi -R arquivo -> abre no modo somente leitura
vi +10 arquivo -> abre o aquivo na linha 10
vi +/palavra arquivo -> abre o arquivo e procura palavra
vi +%s/velho/novo arquivo -> abre arquivo e troca palavra velho por novo
```

28.3 APAGAR TEXTOS

```
x -> apaga texto sobre o cursor
a -> insere texto
dd -> apaga uma linha inteira
dw -> apaga palavra sobre o cursor
d$ -> apaga tudo do cursor atéo final da linha
dG -> apaga tudo do cursor ate o final do arquivo
```

28.4 ALTERANDO TEXTOS

```
R -> pressionando antes de uma palavra, basta digitar a palavra nova
r -> substitui um unico caractere no texto
~ -> altera um caractere para Mm (maiusculo/minusculo)
cw -> altera palavra ou parte da palavra a esquerda do cursor
cc -> altera linha
C -> altera parte da linha a direita do cursor
J -> junta linha corrente com a proxima linha
xp -> muda o caracter que o cursor esta posicionado com o caracter a direita
u -> desfaz o comando anterior
U -> desfaz todas as alteracoes da linha
:u -> desfaz o comando anterior da linha
:g/.*/mo0 -> inverte o texto
:g/^/mo0 -> coloca o texto invertido na ordem correta
```

28.5 COMANDOS DE MOVIMENTACAO NO TEXTO

```
h -> esquerda
j -> para baixo
k -> para cima
l -> para direita
w -> move cursor para inicio da proxima palavra
b -> move cursor para comeco da palavra anterior
CTRL-F -> move o cursor uma pagina a frente
CTRL-B -> move o cursor uma pagina para tras
G - > move o cursor para o final do arquivo
55G -> move o cursor para a linha 55
|1| para navegar diretamente para uma posicao em um arquivo texto, use | n |, onde
n eh o numero da posicao que deseja ir
```

29 – Administração de usuários

30 – Gerenciamento de processos na memória

8 Encontre mais informações em http://wiki.silva.eti.br/doku.php/vim acessado em 05/08/2014.

31 – Compactação e backup

32 – Distribuição Linux

33 – Cron e crontab

A tarefas cron do Linux (ou Unix) podem ser planejadas com o comando crontab. Esse comando ativa uma sessão de edição que permite criar um arquivo crontab. Os cron jobs e os horários apropriados são definidos no crontab. As tarefas do crontab podem ser simples avisos até automação de rotinas complexas e chamadas de scripts do sistema. O sistema operacional mantém um arquivo com o nome do usuário em /var/spool/cron/crontabs (com acesso restrito).

`$crontab -l`
Lista as tarefas do usuário atual;
`$crontab -e`
Edita as tarefas do usuario atual usando o editor padrão;

Existe uma ordem lógica no crontab onde é preciso posissionar os momentos e comandos a serem executados pelo cron, conforme a seguir:

```
<minutos>: número entre 0 e 59;
<horas>: número entre 0 e 23;
<dias do mês>: números entre 1 e 31 (quando quiser informar mais de um, separe por vírgula);
<mês>: número entre 1 e 12;
<dias da semana>: números entre 0 e 6 (quando quiser informar mais de um, separe por vírgula);
<usuário>: usuário com o qual o sistema deve executar o comando agendado, opcional considerado o próprio usuário logado;
<comando>: é o comando a ser executado.
```

Exemplo de um agendamento cron:
05 6 * * * /home/usuario/backup.sh

Minuto 5	Hora 6	Durante todo o mês	Durante toda a semana	Usuario corrente	Executar o comando backup.sh no caminho indicado	

33.1 Usando data no crontab

Uma vez que você usa o crontab para automatizar tarefas, surgem dúvidas de como utilizar datas (comando date) sem ter que recorrer a scripts externos. Por exemplo, imagine que no crontab tenha uma rotina para compactar um determinado diretório /srv/samba/, guardando para cada dia um arquivo.

No crontab ficaria assim:

```
4 10 * * * tar -cvzf /bkp/samba/diario/`date +\%d-\%m-\%Y`.samba.tar.gz
/srv/samba/*
```

Perceba que uso um escape(\) para todo % para que a execução do trabalho seja realizada sem erros.

No shell, você pode testar e comparar o resultado:

```
date +\%d-\%m-\%Y  usando escape para todo %
date +%d-%m-%Y     sem usar escape
```

34 – dispositivos fisicos e módulos

35 – logs

36 -

Referências:
http://pt.wikipedia.org/wiki/Minix
Linux Para Profissionais: Do Básico à Conexão em Redes
George Leal Jamil & Bernardo Andrade Couvêa

http://blog.silva.eti.br/2012/07/root-direitos-privilegios-e-seguranca.html acessado 01/03/2014