# CURSO DE INTRODUÇÃO AO LINUX

## Distribuição openSUSE®

AULA 5 – Usuários e Segurança Local

PETEC
ENGENHARIA DE COMPUTAÇÃO

# Objetivos dessa aula

- Configurar contas e grupos de usuários;
- Definir variáveis de ambiente;
- Usar o histórico de comandos do terminal;
- Usar atalhos de teclado;
- Definir aliases;
- Definir permissões e posse de arquivos;

# Objetivos dessa aula

- Entender boas práticas e ferramentas para manter o Linux seguro;
- Entender os poderes e riscos do uso da conta de superusuário;
- Aprender a usar o sudo para operações privilegiadas restringindo poderes o quanto for factível;
- Explicar a importância do isolamento de processos e acesso ao hardware;
- Trabalhar com senhas;
- Proteger o processo de boot e recursos de hardware.

# Contas, Usuários e Grupos

## *Identificando o usuário atual*

Como discutido, o Linux é um sistema operacional multiusuário, ou seja, mais que um usuário pode acessar ao mesmo tempo. É possível usar o terminal para verificar os usuários utilizando o sistema.

- Para listar todos os usuários atualmente conectados:
  - $ who
- Para identificar o usuário atual:
  - $ whoami

Usar a opção **-a** com o comando **who** dará mais informações, como horário de inicialização do sistema, tempo de login do usuário, entre outros detalhes.

# Contas, Usuários e Grupos

## *Usuários e Grupos - Conceitos básicos*

O Linux usa **grupos** para organizar **usuários**. Grupos são coleções de contas com certas permissões compartilhadas. O controle de associações de grupos é administrado através do arquivo **/etc/group**, que mostra uma lista dos grupos e seus membros.

Por padrão, cada usuário pertence a um grupo padrão ou primário. Quando realiza login, a associação de grupo é definida pra seu grupo primário e todos os membros possuem o mesmo nível de acesso e privilégio. Permissões a vários arquivos e diretórios pode ser modificadas a nível de grupo.

# Contas, Usuários e Grupos

## *Usuários e Grupos – Conceitos básicos*

Todos os usuários no sitema são atribuídos ID único de usuário (**uid**), que é apenas um número inteiro, e também um ou mais IDs de grupo (**gid**), inclusive um padrão que é o mesmo que o ID de usuário. Esses números são associados a nomes através dos arquivos **/etc/passwd** e **/etc/group**.

Para verificar as atribuições de uid e gid de um determinado usuário, basta usar o comando **id**.

- $ id <usuario>
  - Se não for definido um usuário por argumento, o id retornará as atribuições do usuário atual.

# Contas, Usuários e Grupos

## *Adicionando e removendo usuários*

As distribuições possuem interfaces gráficas simples e diretas para criar e remover usuários e grupos, além de manipular associações de grupos. Porém, é útil fazê-lo a partir da linha de comando ou a partir de scripts de shell. Apenas o superusuário pode adicionar e remover usuários/grupos.

Adicionar um novo usuário é feito com o comando **useradd**. Por exemplo:

- # useradd teste

Isso vai criar um usuário chamado *teste*, com diretório home em **/home/teste**, populado com alguns arquivos básicos (copiados de /etc/skel) e adiciona uma linha a **/etc/passwd**, e define o shell padrão para **/bin/bash**.

Remover uma conta de usuário é igualmente fácil, através do comando **userdel** (e.g. # userdel teste). Isso porém, deixará o diretório **/home/teste** intacto. Para remover o diretório ao mesmo tempo, se faz necessário o uso da opção **-r**.

# Contas, Usuários e Grupos

## *Adicionando e removendo grupos*

A adição de um novo grupo é feita através do comando **groupadd**.

- # groupadd grupol

E a remoção através do comando **groupdel**.

- # groupdel grupol

# Contas, Usuários e Grupos

## *Gerenciando grupos*

Antes de manipular as associações de grupos de um usuário, é interessante ver a quais ele já pertence:

- $ groups <usuario>

Para adicionar/remover um usuário a/de grupos existentes usa-se o comando **usermod**.

- # usermod -G <grupo1>,<grupo2> <usuario>

Para manipular propriedades de grupos, usa-se o comando **groupmod**, junto de alguma opção, como **-g** para manipular o ID de grupo, ou **-n** para modificar o nome.

# Contas, Usuários e Grupos

## *Conta de Superusuário (root account)*

A conta **raiz** é muito poderosa, pois possui acesso total ao sistema. Alguns sistemas a chamam de conta de administrador, no Linux é comumente chamada de conta de superusuário.

Deve-se ter extremo cuidado antes de conceder acesso total a um usuário, raramente justificável. Ataques externos geralmente consistem em truques para obter elevação à conta raiz.

Você pode, porém, usar a função **sudo** para atribuir alguns privilégios limitados a contas de usuários:

- Em caratér temporário;
- Apenas para um conjunto específico de comandos.

# Contas, Usuários e Grupos

## *su e sudo*

Para atribuir privilégios elevados, você pode usar o comando **su** para lançar um novo shell como outro usuário (é preciso digitar a senha do usuário a se tornar). Muito geralmente esse usuário é o superusuário, e esse novo shell permite o uso de privilégios elevados até sua saída.

Quase sempre é uma prática ruim (por questões de segurança e estabilidade) usar o **su** para se tornar superusuário. Erros resultantes podem incluir a deleção de arquivos vitais para o sistema e brechas de segurança.

Conceder privilégios através do **sudo** é preferível, por ser menos perigoso.

# Contas, Usuários e Grupos

## *Elevação à conta raiz*

Para se tornar completamente superusuário, basta digitar em um terminal **su** e responder corretamente a senha de superusuário.

Para executar apenas um comando com direito administrativo, digite **sudo <comando>**. Ao término da execução do comando, você retornará a ser um usuário normal.

Os arquivos de configuração do **sudo** estão guardados no arquivo **/etc/sudoers** e no diretório **/etc/sudoers.d/**. Por padrão, o diretório **sudoers.d** é vazio.

# Contas, Usuários e Grupos

## *Arquivos de inicialização*

No Linux, o programa de shell de comando (tipicamente o **bash**) usa um ou mais arquivos de inicialização para configurar o ambiente.

Arquivos no diretório /**etc** definem configurações globais para todos os usuários, enquanto arquivos de inicialização no diretório home do usuário podem incluir e/ou substituir as configurações globais.

Os arquivos de inicialização podem fazer tudo que o usuário gostaria de fazer em um shell de comando, como:

- Personalizar o prompt de usuário;
- Definir atalhos e aliases de linha de comando;
- Definir o editor de texto padrão;
- Definir o caminho para encontrar programas executáveis.

# Contas, Usuários e Grupos

## *Ordem dos arquivos de inicialização*

Quando da realização de login no Linux, **/etc/profile** é lido e avaliado, e então os seguintes arquivos são vistos na seguinte ordem (caso existam):

1) ~/.bash_profile
2) ~/.bash_login
3) ~/.profile

O shell de login do Linux avalia qualquer arquivo de inicialização for encontrado primeiro e ignora os outros. Diferentes distribuições podem usar arquivos de inicialização diferentes.

# Contas, Usuários e Grupos

## *Ordem dos arquivos de inicialização*

Porém, toda vez que for criado um novo shell, ou janela de terminal, etc., você não realiza um login completo do sistema, apenas o arquivo ~/.**bashrc** é lido e avaliado.

Apesar deste arquivo não ser lido e avaliado junto do shell de login, a maioria das distribuições e/ou usuários o incluem a partirde um dos três arquivos de inicialização possuídos pelo usuário.

No openSUSE, o usuário deve fazer as mudanças apropriadas no arquivo ~/.**bash_profile** para incluir o arquivo ~/.**bashrc**.

# Variáveis de Ambiente

## *Conceito e visualização*

**Variáveis de ambiente** são atributos simples com valores específicos, que são entendidos pelo shell de comando. Algumas são predefinidas pelo sistema, enquanto outras são definidas pelo usuário através da linha de comando ou na inicialização ou ainda através de scripts.

Uma variável de ambiente é tão somente uma string de caracteres que contém informação utilizada por uma ou mais aplicações.

Há algumas maneiras de visualizar as variáveis de ambiente atualmente definidas:

- $ set
- $ env
- $ export

# Variáveis de Ambiente

## *Definindo as variáveis de ambiente*

Por padrão, variáveis criadas a partir de um script são disponíveis apenas ao shell atual; processos filhos não terão acessos a valores que foram definidos ou modificados. Permitir processos filhos visualizar os valores requer o uso do comando **export**.

- Para mostrar o valor de uma variável específica:
  - $ echo $VARIAVEL
- Para exportar um novo valor de variável:
  - $ export VARIAVEL=valor (ou $ VARIAVEL=valor; export VARIAVEL)
- Para adicionar permantente uma variável:
  1) Editar **~/.bashrc** e adicionar a linha **export VARIAVEL=valor**
  2) $ source ~/.bashrc

PET COMP
ENGENHARIA DE COMPUTAÇÃO

# Variáveis de Ambiente

## *Variável HOME*

**HOME** é uma variável de ambiente que representa o diretório home do usuário. O comando **cd** sem argumentos mudará o diretório ativo para o valor de HOME.

O caractere ~ é comumente usado como abreviação para **$HOME**. Logo, **cd ~** e **cd $HOME** são declarações equivalentes.

# Variáveis de Ambiente

## *Variável PATH*

**PATH** é uma variável de ambiente que representa uma lista ordenada de diretórios (o **caminho**) que é vasculhada quando um comando é dado para encontrar o programa ou script apropriado a ser executado.

Cada diretório no caminho é separado por : (dois-pontos).

# Variáveis de Ambiente

## *Variável PS1*

**Declaração de Prompt (Prompt Statement - PS)** é uma variável usada para personalizar sua string de prompt em suas janelas de terminal para mostrar a informação que você quiser.

**PS1** é a variável de prompt primária que controla como que sua linha de comando se parece. Os seguintes caracteres especiais podem ser incluídos em PS1:

- **\u** : Nome do usuário
- **\h** : Nome do host
- **\w** : Diretório ativo atual
- **\!** : Número deste comando no histórico
- **\d** : Data

Exemplo: $ export PS1='\u@\h:\w $ '

# Variáveis de Ambiente

## *Variável SHELL*

A variável de ambiente **SHELL** aponta para o shell de comando (o programa que gerencia tudo que você digitar em uma janela de comando) padrão do usuário, e contêm o caminho completo até o shell.

# Recuperando comandos anteriores

## *Histórico do Shell*

O **bash** mantém registro de comandos e declarações previamente feitos em um buffer de **histórico**. Você pode invocar comandos previamente utilizados de modo simples usando **as setas para cima e para baixo**. Para ver uma lista de comandos executados, digite **history** na linha de comando.

A lista de comandos é mostrada com o comando mais recente aparecendo por último. Essa informação é guardada em ~/.**bash_history**.

# Recuperando comandos anteriores

## *Usando variáveis de ambiente*

Várias variáveis de ambiente associadas podem ser usadas para obter informação sobre o arquivo de histórico.

- **HISTFILE** guarda a localização do arquivo de histórico;
- **HISTFILESIZE** guarda o número máximo de linhas no arquivo de histórico;
- **HISTSIZE** guarda o número máximo de linhas no arquivo de histórico para a sessão atual.

# Recuperando comandos anteriores

## *Buscando comandos anteriores*

- ↑/↓ : Vasculhar através da lista de comandos no histórico;
- !! : Executar o comando imediatamente anterior;
- **CTRL-R** : Buscar comandos usados previamente

  O comando CTRL-R vai iniciar uma busca reversa inteligente, buscando o último comando executado de modo a combinar com os caracteres conforme você digita. Tecle ENTER para confirmar o comando encontrado, CTRL-C para cancelar.

PET ENGENHARIA DE COMPUTAÇÃO

# Atalhos no terminal

## *Atalhos de teclado*

É possível utilizar atalhos de teclado para realizar diferentes tarefas rapidamente, a seguir alguns usos:

- **CTRL-L** : Limpa a tela;
- **CTRL-D** : Sai do shell atual;
- **CTRL-Z** : Põe o processo atual em plano de fundo suspenso;
- **CTRL-C** : Mata o processo atual;
- **CTRL-W** : Deleta a palavra que precede o cursor;
- **CTRL-U** : Deleta do começo da linha até o cursor;
- **TAB** : Auto-completar.

# Aliases de comando

## *Criando aliases*

Você pode criar comandos personalizados ou modificar o comportamento de comandos já existentes criando **aliases**. Geralmente esses aliases são colocados em seu arquivo ~/.**bashrc** para que estejam disponíveis para qualquer shell de comando que você crie.

Executar o comando **alias** sem argumento vai listar os aliases atualmente definidos. Para criar um alias, segue a sintaxe básica:

- $ alias <alias>='<comando>'

# Permissões de arquivos

## *Posse*

No Linux e em outros sistemas baseados em UNIX, cada arquivo é associado com um usuário que é o **proprietário**. Cada arquivo também é associado a um **grupo** que possui interesse no arquivo e certos direitos/permissões, como ler, escrever e executar.

Os utilitários a seguir trabalham a configuração de permissão e posse:

- **chown** : Usado para mudar a posse de usuário de um arquivo ou diretório;
- **chgrp** : Usado para mudar a posse de grupo;
- **chmod** : Usado para alterar as permissões de um arquivo, pose ser feito separadamente para proprietário, grupo e outros.

# Permissões de arquivos

## *Modos de Permissão*

Arquivos têm tres tipos de permissões: leitura (r), escrita (w) e execução (x), representadas geralmente por **rwx**. Estas permissões afetam três grupos de posse: usuário/proprietário (u), grupo (g) e outros (o). Como resultado, temos três grupos de três permissões:

- rwx : rwx : rwx
- u : g : o

# Permissões de arquivos

## *Modos de Permissão - chmod*

O comando **chmod** permite manipular essas permissões, mas possui uma sintaxe peculiar:

- Por exemplo, para dar ao proprietário e outros direito de execução e remover o direito de escrita ao grupo:
  - $ chmod uo+x,g-w <arquivo>

Essa sintaxe não é simples, mas há um método numérico que o resume, utilizando somas:

- 4 para direito de leitura;
- 2 para direito de escrita;
- 1 para direito de execução.
  - Exemplo: $ chmod 755 <arquivo>

PET COMP
ENGENHARIA DE COMPUTAÇÃO

# Permissões de arquivos

## *Mudança de posse de usuário - chown*

Para mudar o usuário proprietário de um arquivo ou diretório, basta usar o comando **chown**:

- $ chown <usuario> <arquivo>
  - Caso seja feita uma mudança de posse de/para o superusuário, será necessário usar o sudo para executar o comando.

# Permissões de arquivos

## *Mudança de posse de grupo - chgrp*

Para mudar o grupo proprietário de um arquivo ou diretório, basta usar o comando **chgrp**:

- # chgrp <grupo> <arquivo>

# Permissões de arquivos

## *Try-it-yourself: permissões*

1) Crie um arquivo vazio

    $ touch <nome>

2) Use a listagem longa para verificar os arquivos e suas permissões;

    $ ls -l

3) Mude a permissão do arquivo para **rwxr--r-x**, usando a notação **rwx**;

4) Use a listagem longa para verificar os arquivos e suas permissões;

5) Mude a permissão do arquivo para **rw-r-x-wx**, usando o método **421**;

6) Use a listagem longa para verificar os arquivos e suas permissões.

# Entendendo a segurança no Linux

## *Contas de usuário*

O Kernel do Linux permite a usuários propriamente autenticados acesso a arquivos e aplicações. Ao passo que cada usuário é identificado por um inteiro único (UID), uma database separada associa um nome de usuário a cada UID.

Quando da criação de uma conta, informação sobre o novo usuário é adicionada à database e o diretório home do usuário deve ser criado e populado com alguns arquivos essenciais.

# Entendendo a segurança no Linux

## *Contas de usuário*

Para cada usuário, os 7 campos seguintes são mantidos no arquivo **/etc/passwd**:

- Nome de usuário
- Senha
- UID
- GID
- Informação sobre o usuário
- Diretório home
- Shell

# Entendendo a segurança no Linux

## *Tipos de contas*

Por padrão, o Linux distingue entre vários tipos de contas para isolar processos e cargas de trabalho diferentes, com 4 tipos de contas:

- Superusuário, sistema, normal e de rede.

Para um ambiente seguro, é recomendado conceder o mínimo de privilégios possível e necessário às contas, e remover contas inativas. A ferramenta **last**, que mostra a última vez que cada usuário entrou no sistema, pode ser útil para identificar contas possivelmente inativas, candidatas à remoção.

# Necessidade de direito root

## *Operações que requerem privilégios*

Direitos de superusuário são necessários para realizar operações tais como:

- Criar, remover e gerenciar contas de usuário;
- Gerenciar pacotes de software;
- Remover ou modificar arquivos de sistema;
- Reiniciar serviços do sistema.

Contas regulares, em algumas distribuições, podem ser permitidas a instalar software, mudar algumas configurações e aplicar diversas alterações ao sistema.

# Necessidade de direito root

## *Operações que não requerem privilégios*

Uma conta de usuário regular pode realizar algumas operações que precisam de permissões especiais, porém a configuração do sistema deve permitir.

**SUID** (Set owner User ID upon execution – Definir UID do proprietário sob execução) é um tipo de permissão especial dada a um arquivo, que providencia permissões temporárias a um usuário para executar um programa com as permissões do proprietário (que pode ser o superusuário), ao invés das permissões próprias do usuário.

# Sudo

## *Comparando sudo e su*

Ao passo que tanto o **su** quanto o **sudo** dão acesso root temporário ao usuário, esses métodos possuem suas diferenças:

| su | sudo |
|---|---|
| Ao elevar privilégio, é necessária a entrada da senha raiz. Dar a senha raiz a usuários **JAMAIS** deve ser feito. | Ao elevar privilégio, usa-se a senha do usuário, não a senha raiz. |
| Estando elevado, o usuário pode fazer tudo que o superusuário pode ao longo que quiser, sem reentrada de senha. | O que usuário pode fazer é controlado e limitado. Por padrão o usuário precisa reentrar a senha a cada intervalo de tempo. |
| Possui recursos de log limitados. | Possui recursos detalhados de log. |

# Sudo

## *Recursos*

O **sudo** tem a habilidade de registrar tentativas mal-sucedidas de obter acesso root. A autorização dos usuários para usá-lo é baseada na informação de configuração guardada no arquivo **/etc/sudoers** e no diretório **/etc/sudoers.d**.

Sempre que alguém tentar executar um comando usando o **sudo** e não tiver sucesso na autenticação, uma mensagem aparecerá em um arquivo de log do sistema.

# Sudo

## *Arquivo sudoers*

Sempre que o sudo é invocado, um gatilho vasculhará o arquivo **/etc/sudoers** e os arquivos em **/etc/sudoers.d/** para determinar se o usuário possui o direito de usar o sudo e qual o escopo de seu privilégio.

Requisições de usuários desconhecidos e requisições para realizar operações não permitidas são reportadas.

O arquivo é bastante documentado sobre como personalizá-lo. A maioria das distribuições Linux prefere que você adicione um arquivo no diretório /etc/sudoers.d/ com o mesmo nome que o usuário. Esse arquivo contém a configuração individual do sudo para o usuário.

# Sudo

## *Log de comandos*

No openSUSE, o log do sudo é tratado pelo **journal**. Basta executar

- $ sudo journalctl -t sudo

para verificar toda a utilização recente do sudo, inclusive as mal-sucedidas.

Isso é uma salvaguarda importante para manter registro e contabilidade da utilização do sudo.

# Isolamento de processos

O Linux é considerado um sistema mais seguro que vários outros porque os processos são **isolados** uns dos outros. Um processo normalmente não pode acessar os recursos de outro processo, mesmo executando com os mesmos privilégios de usuário.

O Linux torna, então, difícil (mas não impossível) que vírus e falhas de segurança acessem e ataquem recursos aleatórios no sistema.

# Isolamento de processos

Mecanismos adicionais de segurança, que foram introduzidos recentemente para minimizar ainda mais os riscos, são:

- **Grupos de controle (cgroups)** : Permite a administradores de sistema agrupar processos e associar recursos finitos a cada cgroup;
- **Containers Linux (LXC)** : Torna possível executar múltiplos sistemas Linux isolados (containers) em um único sistema, usando cgroups;
- **Virtualização** : O hardware é emulado de tal maneira que não só processos podem ser isolados, mas sistemas inteiros são executados como máquinas virtuais isoladas em um host físico.

# Acesso a dispositivos de hardware

O Linux limita o acesso de usuários a dispositivos fora-de-rede de uma maneira extremamente similar ao acesso a arquivos regulares.

Aplicações interagem ao engajar a camada do sistema de arquivos. Essa camada vai então abrir um **arquivo especial de dispositivo** (geralmente chamado **nó de dispositivo**) sob o diretório **/dev/** que corresponde ao dispositivo sendo acessado. Cada arquivo especial de dispositivo tem os campos padrão de permissões **u/g/o**.

# Manter o sistema atualizado

Quando problemas de segurança no kernel Linux ou em aplicações e bibliotecas são encontrados, as distros Linux têm um bom histórico de resposta rápida e emitem soluções para seus sistemas através de atualizações de seus repositórios, emitindo notificações para atualização imediata.

Sabe-se, porém, que muitos sistemas não são atualizados com frequência e problemas que já foram sanados ainda persistem. Isso é particularmente verídico em sistemas proprietários onde os usuários desconhecem ou desconfiam da política de atualização do fabricante, onde atualizações podem causar novos problemas.

# Trabalhando com senhas

## *Como as senhas são armazenadas*

O sistema verifica autenticidade e identidade usando credenciais de usuário.

Originalmente, senhas encriptadas eram guardadas no arquivo **/etc/passwd**, que era legível por todo mundo. Isso tornava relativamente fácil quebrar senhas.

Em sistemas modernos, as senhas são armazenadas em um formato criptografado em um arquivo secundário chamado **/etc/shadow**. Somente usuários com acesso root podem modificar/ler esse arquivo.

# Trabalhando com senhas

## *Algoritmos de Criptografia*

Proteger senhas se tornou um elemento crucial de segurança. A maioria das distribuições Linux confiam em um algoritmo moderno de criptografia de senhas chamado **SHA-512** (Secure Hashing Algorithm 512 bits), desenvolvido pela NSA.

O algoritmo SHA-512 é vastamente usado para aplicações e protocolos de segurança, incluindo TLS, SSL, PHP, SSH, S/MIME e IPSec.

# Protegendo boot e hardware

## *Requisição de senha pelo bootloader*

É possível proteger o processo de boot com uma senha segura para prevenir que alguém ignore a etapa de autenticação de usuário.

Para sistemas que utilizem o GRUB2, não se altera diretamente o arquivo de configuração **/boot/grub/grub.cfg**, mas sim os arquivos de configuração do sistema em **/etc/grub.d/** e executando **update-grub** depois.

Para mais explicações, vide:
https://help.ubuntu.com/community/Grub2/Passwords

PET TECO
ENGENHARIA DE COMPUTAÇÃO

# Protegendo boot e hardware

## *Vulnerabilidade de hardware*

Quando o hardware é fisicamente acessível, a segurança pode ser comprometida por:

- Key logging
- Farejadores de rede
- Boot alternativo, e.g. via Live Media.

As diretrizes de segurança para proteger seus sistemas são:

- Trancar workstations e servidores;
- Proteger os nós de rede de modo a serem inacessíveis a pessoal não-autorizado;
- Proteger a BIOS através de senha.