IMERSÃO

DevOps & Cloud

Aquecimento

E-BOOK

Linux

Básico

E-BOOK

Linux

O que é um sistema operacional

Um sistema operacional (SO) é o software mais fundamental executado em um computador. É responsável por gerenciar o hardware do computador, fornecer serviços para outros softwares e executar aplicativos essenciais, desde sistemas centrais a funções básicas como administrar o teclado e enviar vídeo para a tela.

Funções Principais:

- **Gerenciamento de Hardware**: Controla o hardware do computador, incluindo CPU, memória, discos rígidos e periféricos, como impressoras e monitores.
- **Execução de Software:** Carrega e executa aplicativos, assegurando-se de que eles não interfiram uns com os outros.
- **Interface de Usuário:** Oferece uma interface para os usuários interagirem com o computador, seja por meio de uma interface gráfica (GUI) ou de uma linha de comando.
- **Comunicação de Dados:** Facilita a comunicação entre os componentes do computador.
  Segurança: Garante a proteção das informações do usuário contra acessos não autorizados.

Os sistemas operacionais desempenham um papel fundamental, agindo como um intermediário entre o usuário e o hardware físico do computador. Sem um SO, usar um computador seria extremamente complicado e altamente ineficiente.

No contexto da computação em nuvem e DevOps, entender como os sistemas operacionais funcionam, em especial o Linux, é básico. Isso porque muitas infraestruturas em nuvem são baseadas em Linux, e as práticas de DevOps frequentemente envolvem a manipulação direta de sistemas baseados em Linux.

Linux

O que é o Linux

O Linux é um sistema operacional de código aberto baseado no Unix, que proporciona uma alternativa livre para os sistemas operacionais proprietários, como o Windows. Inicialmente, foi desenvolvido por Linus Torvalds no início dos anos 1990. Esse diferencial permite que qualquer pessoa possa ver, modificar e, se desejar, distribuir o código.

Características Principais:

- **Código Aberto:** Uma das características mais marcantes do Linux é sua natureza open source, permitindo que o código-fonte seja alterado e distribuído por qualquer um.
- **Multiusuário:** O Linux é um sistema multiusuário, o que significa que mais de uma pessoa pode usar o sistema ao mesmo tempo.
- **Multitarefa:** É capaz de executar vários programas ao mesmo tempo.
- **Portabilidade:** O Linux foi escrito em C, o que o torna portável, ou seja, pode ser usado em diferentes tipos de hardware.
- **Segurança:** Oferece um sistema robusto com características de segurança inerentes, como separação de usuário e superusuário e mecanismos de permissão para arquivos.

Por que o Linux é tão popular em DevOps e Computação em Nuvem?

- **Flexibilidade:** Como é de código aberto, o Linux pode ser personalizado de acordo com as necessidades específicas.
- **Custo:** Muitas distribuições Linux são gratuitas, reduzindo custos em ambientes de TI.
- **Comunidade:** Uma vasta comunidade de desenvolvedores e entusiastas apoia o Linux, oferecendo suporte robusto e contribuições constantes.
- **Performance:** O Linux é conhecido por sua estabilidade e performance, tornando-o uma escolha ideal para servidores e sistemas que exigem alta disponibilidade.
- **Integração com Tecnologias Emergentes:** O Linux se integra bem com tecnologias emergentes, como containers (por exemplo, Docker) e orquestração (por exemplo, Kubernetes), que são amplamente usados em ambientes de DevOps e nuvem.

Linux e suas Distribuições

O universo Linux é vasto e variado, composto por várias “distribuições”. Uma distribuição, ou “distro”, é uma versão específica do sistema operacional Linux que contém o kernel Linux (o núcleo do sistema operacional) juntamente com um conjunto de softwares adicionais.

Por que existem diferentes distribuições?

A natureza de código aberto do Linux permite que qualquer pessoa ou organização modifique e redistribua o sistema conforme suas necessidades ou objetivos específicos. Ao longo dos anos, isso levou ao surgimento de diversas distribuições, cada uma com suas próprias características, focos e comunidades.

Por que algumas distribuições Linux são pagas?

Apesar do Linux ser de código aberto e muitas distribuições serem gratuitas, existem versões pagas. Essas distros geralmente são cobradas por conta de alguns fatores:

- **Suporte Profissional:** Empresas por trás de algumas distribuições oferecem suporte técnico especializado, garantindo que problemas sejam resolvidos rapidamente.
- **Certificações e Garantias:** Algumas organizações precisam de certos padrões e garantias de software para seus sistemas (normalmente empresas públicas e bancárias), algo que distribuições pagas muitas vezes oferecem.
- **Atualizações e Manutenção:** Oferecem atualizações regulares, patches de segurança e manutenção contínua do sistema.
- **Softwares Adicionais:** Além do sistema operacional em si, algumas distribuições pagas vêm com um conjunto de softwares proprietários ou licenciados inclusos.
- **Treinamento:** Algumas distribuições oferecem treinamento e recursos educacionais como parte do pacote.

Algumas distribuições populares incluem:**

- **Ubuntu:** Uma das distribuições Linux mais populares para desktops e servidores. É conhecida por sua facilidade de uso e comunidade ativa.
- **Fedora:** Uma distribuição inovadora que frequentemente introduz as mais recentes tecnologias no mundo Linux.
- **Debian:** Conhecida por sua estabilidade, é a base de muitas outras distribuições, incluindo o Ubuntu.
- **CentOS:** Derivado do Red Hat Enterprise Linux, é focado em estabilidade e desempenho para ambientes empresariais.
- **Arch Linux:** Adotado por usuários que desejam um sistema altamente personalizável.
- **Red Hat Enterprise Linux (RHEL):** Uma distribuição comercial focada em ambientes corporativos com suporte e treinamento oferecidos pela Red Hat.

Como escolher a distribuição certa?

A escolha da distribuição Linux depende em grande parte das necessidades e preferências do usuário. Algumas questões devem ser consideradas:

- **Finalidade:** O uso será para desenvolvimento, produção ou uso pessoal?
- **Estabilidade vs. Atualizações:** Prefere ter as últimas tecnologias ou uma plataforma mais estável e testada?
- **Suporte:** Precisa de suporte comercial ou uma comunidade ativa é suficiente?
- **Habilidades e conhecimento:** Está à procura de uma experiência de aprendizado profundo ou de algo mais user-friendly?

No contexto de DevOps e computação em nuvem, é comum ver distribuições como Ubuntu, CentOS e RHEL sendo amplamente adotadas por conta da sua estabilidade, desempenho e suporte.

Sistema de Arquivos Linux

O sistema de arquivos é a forma como um sistema operacional organiza e gerencia dados no disco. No mundo do Linux, essa organização é significativamente diferente daquela encontrada em sistemas baseados no Windows. Vamos explorar a estrutura do sistema de arquivos do Linux e comparar com o que muitos usuários podem estar familiarizados no Windows.

Estrutura Básica:

No Linux, a organização dos arquivos começa a partir da raiz, denotada como `/`. Já no Windows, a organização começa com letras de unidade, como `C:\\`.

Dentro do diretório raiz do Linux, encontramos:

- **/bin:** Assim como o `System32` no Windows, contém binários e comandos fundamentais necessários para o funcionamento básico do sistema.
- **/etc:** Semelhante com registro do Windows (`regedit`), armazena arquivos de configuração do sistema.
- **/home:** Semelhante ao diretório `Users` no Windows, é onde os diretórios pessoais dos usuários são armazenados.
- **/var:** Armazena arquivos variáveis, como logs, semelhante ao `C:\\Windows\\Temp` no Windows.
- **/usr:** Contém binários, bibliotecas, documentação e código-fonte para programas no sistema, algo como o `Program Files` no Windows.
- **/boot:** Assim como a pasta `Boot` no Windows, contém arquivos necessários para o processo de inicialização do sistema.
- **/dev:** Representa os dispositivos no sistema. No Windows, os dispositivos são geralmente gerenciados pelo Gerenciador de Dispositivos.
- **/proc:** Sistema de arquivos virtual que fornece informações sobre processos e recursos do sistema, um pouco parecido com o Monitor de Recursos no Windows.

Montagem de Dispositivos:

Diferente do Windows, onde um novo dispositivo (como um pendrive) é automaticamente atribuído a uma nova letra de unidade (ex: `E:\\`), no Linux, ele precisa ser “montado” em um diretório existente para se tornar parte da árvore do sistema de arquivos.

Comandos Básicos do Linux

Conhecer os comandos essenciais e seus parâmetros no Linux é crucial para qualquer profissional. Vamos explorar em detalhes alguns dos comandos mais comuns e suas opções mais úteis.

Identificando Informações do Sistema:**

- **`uname`**: Fornece informações sobre o sistema.
  - **Parâmetros comuns:**
    - `a`: Mostra todas as informações disponíveis.
    - `s`: Nome do kernel.
    - `r`: Versão do kernel.
  - **Exemplo:**

```
Input: uname -a
Output: Linux myserver 5.4.0-70-generic #78-Ubuntu SMP Fri Mar
19 13:29:52 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

Limpando a Tela:

- **`clear`**: Limpa todo o conteúdo da tela do terminal.
  - **Exemplo:**

```
Input: clear
Output: (Tela limpa)
```

Executando Comandos como Superusuário:

- **`sudo`**: Executa comandos com privilégios de superusuário.
  - **Exemplo:**

```
Input: sudo apt update
Output: [sudo] password for user: (Após inserir a senha, o sistema começará a
atualizar a lista de pacotes.)
```

Identificando Usuários Logados:

- **`who`**: Mostra informações sobre usuários logados.
  - **Parâmetros comuns:**
    - `a`: Mostra todas as informações, incluindo os processos de inicialização.
  - **Exemplo:**

```
Input: who -a
Output:
system boot  2021-04-10 07:10
jane       tty1         2021-04-10 07:22
```

Verificando a Memória:

- **`free`**: Fornece informações sobre a memória.
  - **Parâmetros comuns:**
    - `h`: Mostra os números de forma humanamente legível (ex: “G” para gigabytes).
  - **Exemplo:**

```
Input: free -h
Output:
               total        used        free      shared  buff/cache
available
Mem:           7.7Gi       1.2Gi       5.8Gi       112Mi       738Mi       6.2Gi
Swap:          2.0Gi          0B       2.0Gi
```

Desligando ou Reiniciando o Sistema:

- **`shutdown`**: Desliga ou reinicia o sistema.
  - **Parâmetros comuns**:**
    - `r`: Reinicia o sistema.
    - `h`: Desliga o sistema.
    - `now`: Executa a ação imediatamente.
  - **Exemplo:**

```
Input: shutdown -r now
Output: (O sistema irá reiniciar imediatamente.)
```

Consultando o Manual:

- **`man`**: Fornece o manual detalhado de qualquer comando.
  - **Exemplo:**

```
Input: man ls
Output: (Será exibida uma página de manual detalhando
o comando `ls`, suas opções e exemplos de uso.)
```

Visualizando Comandos Anteriores:

- **`history`**: Mostra a lista dos comandos previamente executados.
  - **Exemplo:**

```
Input: history
Output:
1  ls
2  cd /home
3  sudo apt update
4  man uname
5  clear
```

Criação e Manipulação de Arquivos e Diretórios

A habilidade de criar, manipular e navegar através de arquivos e diretórios é fundamental quando você está trabalhando com Linux. Aqui, vamos detalhar alguns dos comandos mais usados, assim como suas variações.

1. Determinando o Diretório Atual

- **`pwd`** (print working directory): Exibe o caminho completo do diretório atual.
  - **Exemplo:**

```
Input: pwd
Output: /home/usuario/projeto
```

2. Mudando de Diretório

`cd` (change directory): Usado para mudar de diretório.

`cd ..`: Retorna ao diretório pai.

`cd`: Retorna ao diretório home do usuário.

- **Exemplo:**

```
Input: cd /home/usuario/documentos
Output: (Você será levado ao diretório 'documentos'.)
```

3. Listando Arquivos e Diretórios

- **`ls`**: Lista os arquivos e diretórios no diretório atual.
  - `ls -l`: Exibe os detalhes dos arquivos/diretórios, como permissões, número de links, dono, \ tamanho e data da última modificação.
  - `ls -a`: Lista todos os arquivos, incluindo arquivos ocultos.
  - `ls -la`: Combina as opções anteriores.
  - **Exemplo:**

```
Input: ls -l
Output:
drwxr-xr-x 5 usuario usuario 4096 Jan  1 10:00 projeto/
-rw-r--r-- 1 usuario usuario  512 Jan  1 09:45 arquivo.txt
```

4. Criando Diretórios

- **`mkdir`** (make directory): Cria um novo diretório.
  - `mkdir -p caminho/para/diretorio`: Cria diretórios pais conforme necessário.
  - **Exemplo:**

```
Input: mkdir -p /home/usuario/novo/diretorio
Output: (Os diretórios 'novo' e 'diretorio' serão criados.)
```

5. Removendo Arquivos e Diretórios

- **`rm`**: Remove arquivos ou diretórios.
  - `rm -r`: Remove diretórios e seus arquivos.
  - `rm -f`: Força a remoção, sem perguntar.
  - `rm -rf`: Combina as opções acima, use com cautela.
  - **Exemplo:**

```
Input: rm -rf diretorio/
Output: (O diretório e seu conteúdo serão removidos sem confirmação.)
```

6. Copiando e Movendo Arquivos e Diretórios

- **`cp`**: Copia arquivos ou diretórios.
  - `cp -r`: Copia diretórios inteiros.
  - `cp -i`: Pergunta antes de substituir.
- **`mv`**: Move ou renomeia arquivos ou diretórios.
  - `mv -i`: Pergunta antes de substituir.
  - **Exemplo:**

```
Input: cp -r projeto/ backup_projeto/
Output: (Copia todo o diretório 'projeto' para 'backup_projeto'.)

Input: mv -i arquivo.txt novo_nome.txt
Output: (Se 'novo_nome.txt' já existir, o sistema perguntará antes
de substituir.)
```

Criação e Manipulação de Arquivos no Linux

1. Navegando no Sistema de Arquivos

- **pwd**: Exibe o diretório atual.

```
Input: pwd
Output: /home/usuario
```

- **cd:** Muda o diretório atual.

```
Input: cd /var/log
Output: (Muda para o diretório /var/log.)
```

2. Listando Arquivos e Diretórios

- **ls**: Lista os arquivos e diretórios no diretório atual.
  - l: Formato longo, mostrando permissões, número de links, dono, grupo, tamanho e data de modificação.
  - a: Inclui arquivos ocultos.
  - lh: Mostra o tamanho dos arquivos de forma legível (KB, MB).

```
Input: ls -lh
Output: -rw-r--r-- 1 usuario usuario 5M Jan 1 12:00 arquivo1.txt
```

3. Criando Diretórios e Arquivos

- **mkdir**: Cria um novo diretório.
  - p: Cria diretórios pais conforme necessário.

```
Input: mkdir -p novo_diretorio/sub_diretorio
Output: (Cria a estrutura de diretórios especificada.)
```

- **touch:** Cria um novo arquivo vazio.

```
Input: touch novo_arquivo.txt
Output: (Cria 'novo_arquivo.txt'.)
```

4. Exibindo Conteúdo de Arquivos

- **cat**: Mostra o conteúdo completo de um arquivo

```
Input: cat novo_arquivo.txt
Output: Este é o conteúdo do novo_arquivo.txt.
```

- **head**: Exibe as primeiras linhas de um arquivo.
  - n: Especifica o número de linhas.

```
Input: head -n 5 arquivo_grande.txt
Output: (Exibe as 5 primeiras linhas.)
```

- **tail**: Mostra as últimas linhas de um arquivo.
  - n: Especifica o número de linhas.

```
Input: tail -n 5 arquivo_grande.txt
Output: (Exibe as 5 últimas linhas.)
```

- **more** e **less**: Ferramentas para visualizar e navegar pelo conteúdo dos arquivos.

```
Input: more arquivo_grande.txt
Output: (Exibe o conteúdo página por página.)
```

5. Buscando no Conteúdo dos Arquivos

- **grep**: Busca por padrões em arquivos.
  - i: Ignora a distinção entre maiúsculas e minúsculas.
  - r: Busca recursiva em diretórios.
  - n: Mostra o número da linha junto ao resultado.

```
Input: grep -in "conteúdo" arquivo_grande.txt
Output: 42: Linha com a palavra "conteúdo".
```

6. Movendo, Copiando e Removendo

- **mv**: Move ou renomeia arquivos/diretórios.

```
Input: mv arquivo1.txt pasta/
Output: (Move 'arquivo1.txt' para 'pasta/'.)
```

- **cp**: Copia arquivos ou diretórios.
  - r: Copia diretórios recursivamente.

```
Input: cp -r pasta_origem/ pasta_destino/
Output: (Copia todo o conteúdo de 'pasta_origem/' para 'pasta_destino/'.)
```

- **rm**: Remove arquivos ou diretórios.
  - r: Remove diretórios e seus conteúdos recursivamente.
  - f: Força a remoção sem pedir confirmação.

```
Input: rm -rf pasta_perigosa/
Output: (Remove 'pasta_perigosa/' e seu conteúdo sem confirmação.)
```

Execução de Comandos, Redirecionamentos e Pipes

1. Execução de Comandos

No Linux, qualquer ação, desde a mais simples até a mais complexa, pode ser realizada através de comandos. Cada comando pode ter uma ou várias opções e argumentos.

○ Comando com Argumentos

```
Input: echo "Olá, Mundo!"
Output: Olá, Mundo!
```

2. Redirecionamentos

Os redirecionamentos permitem que a saída de um comando seja utilizada como entrada para outro comando ou salva em um arquivo.

○ `>`: Redireciona a saída de um comando para um arquivo, sobrescrevendo seu conteúdo.

```
Input: echo "Isto é um teste." > teste.txt
Output: (Salva a string no arquivo teste.txt.)
```

○ `>>`: Redireciona a saída de um comando para um arquivo, anexando ao conteúdo existente.

```
Input: echo "Outra linha." >> teste.txt
Output: (Anexa a string ao arquivo teste.txt.)
```

○ `<`: Usa um arquivo como entrada para um comando.

```
Input: sort < desordenado.txt
Output: (Exibe o conteúdo de desordenado.txt ordenado.)
```

3. Pipes (|)

Pipes permitem usar a saída de um comando como entrada para outro. Isso permite a combinação de vários comandos para realizar tarefas complexas.

○ Usando Pipes

```
Input: cat teste.txt | grep "teste"
Output: Isto é um teste.
```

Neste exemplo, `cat` lê o conteúdo do arquivo `teste.txt` e `grep` filtra linhas contendo a palavra "teste".

4. Comandos em Sequência

Usando `;` é possível executar múltiplos comandos em sequência.

```
Input: echo "Um"; echo "Dois"
Output:
Um
Dois
```

O `;` permite que comandos sejam executados um após o outro, independentemente do sucesso ou falha do comando anterior.

5. Executando Comandos em Background

Adicionando `&` ao final de um comando, é possível executá-lo em background, permitindo que você continue usando o terminal enquanto o comando é processado.

```
Input: long_command &
Output: [1] 12345
```

Neste exemplo, long_command é um comando fictício que leva muito tempo para ser executado. Ao adicionarmos &, ele é executado em background e o terminal retorna imediatamente.

Comandos para Gerenciamento de Processos

No Linux, assim como em outros sistemas operacionais, os processos representam as atividades em execução no sistema. Pode ser um programa em execução, um script ou até mesmo o próprio sistema operacional realizando suas tarefas de rotina. O gerenciamento eficaz dos processos é crucial para manter o sistema estável, rápido e seguro.

1. ps - Exibir processos em execução

O comando `ps` fornece informações sobre os processos em execução em sua sessão.

```
Input: ps
Output:
  PID TTY          TIME CMD
 1234 pts/1    00:00:00 bash
 5678 pts/1    00:00:00 ps
```

Opções comuns:

- `e`: Mostra todos os processos no sistema.
- `f`: Exibe uma saída detalhada.

```
Input: ps -ef
Output:
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 07:31 ?        00:00:01 /usr/lib/systemd/systemd
root           2     0  0 07:31 ?        00:00:00 [kthreadd]
...
```

2. top - Monitoramento dinâmico dos processos

O comando `top` fornece uma visão em tempo real dos processos em execução.

```
Input: top
Output:
Tasks: 100 total,   2 running,  98 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.5 us,  0.7 sy,  0.0 ni, 96.5 id,  0.2 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   2000.0 total,    987.5 free,    567.2 used,    445.3 buff/cache
...
```

3. `kill` - Terminar processos

Para encerrar um processo, você pode usar o comando `kill` seguido do ID do processo (PID).

```
Input: kill 1234
Output: (nenhuma saída, mas o processo com PID 1234 será encerrado)
```

Opções comuns:

- `9`: Mata o processo imediatamente.

4. `htop` - Monitoramento avançado

O `htop` é uma versão melhorada do `top`, com uma interface mais amigável e colorida.

```
Input: htop
Output:
(Exibe uma visão colorida e interativa em tempo real dos processos)
```