Comandos

Linux

Introdução

- Linux é uma família de sistemas operacionais do tipo **Unix** de código aberto baseados no **kernel Linux**, um kernel de sistema operacional lançado pela primeira vez em 17 de setembro de 1991, por Linus Torvalds.
- O Linux normalmente é empacotado em uma distribuição Linux.

Unix

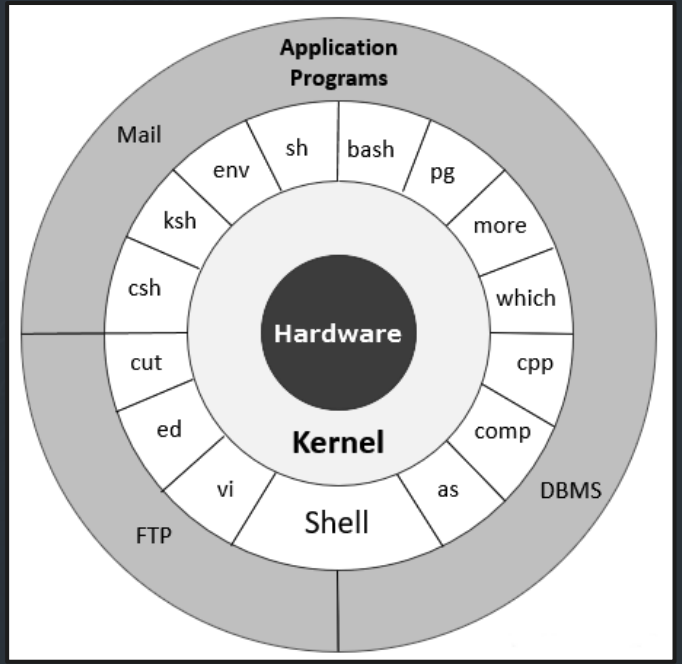
- Unix é uma família de sistemas operacionais de computador **multitarefa** e **multiusuário** derivados do **AT&T Unix** original, cujo desenvolvimento começou na década de 1970 no centro de pesquisa Bell Labs por Ken Thompson, Dennis Ritchie e outros.

Unix

- Inicialmente planejado para uso dentro do Sistema Bell, a AT&T licenciou o Unix para terceiros no final dos anos 1970, levando a uma variedade de variantes Unix **acadêmicas** e **comerciais**, incluindo:
  - University of California, Berkeley (BSD)
  - Microsoft (Xenix)
  - Sun Microsystems (SunOS/Solaris)
  - IBM (AIX)
  - HP/HPE (HP-UX)

Unix

- Os sistemas Unix são caracterizados por um design modular que às vezes é chamado de "filosofia Unix".
- De acordo com a filosofia Unix, o **Sistema Operacional** deve fornecer um conjunto de ferramentas simples, cada uma das quais desempenhando uma função limitada e bem definida.



Unix

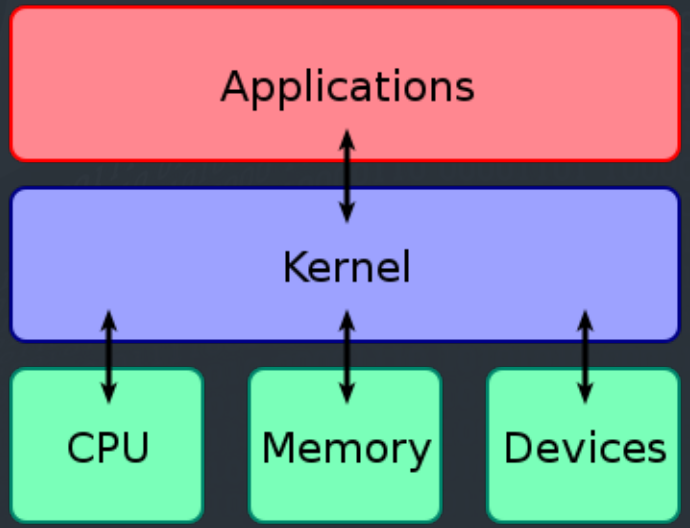
- Um sistema de arquivos unificado (o sistema de arquivos Unix) e um mecanismo de comunicação entre processos conhecido como "pipes" servem como o principal meio de comunicação, e um *shell scripting* e linguagem de comando (o shell Unix) é usado para combinar as ferramentas para executar fluxos de trabalho complexos .
- O Unix difere de seus predecessores como o primeiro **sistema operacional portátil**: quase todo o sistema operacional é escrito na linguagem de programação C, o que permite que o Unix opere em várias plataformas.

Componentes Unix

- O sistema Unix é composto de vários componentes que foram originalmente empacotados juntos.
- Ao incluir o ambiente de desenvolvimento, bibliotecas, documentos e o código-fonte portátil e modificável para todos esses componentes, além do **kernel** de um sistema operacional.
- Esta foi uma das principais razões pelas quais surgiu como uma importante ferramenta de ensino e aprendizagem e teve uma influência tão ampla.

Kernel

- O kernel é um programa de computador no núcleo do sistema operacional de um computador que tem controle completo sobre tudo no sistema.
- Ele é a "parte do código do sistema operacional que está sempre residente na memória" e facilita as interações entre os componentes de hardware e software.



Kernel

- Na maioria dos sistemas, o kernel é um dos primeiros programas carregados na inicialização (após o bootloader).
- Ele lida com o resto da inicialização, bem como com as solicitações de memória, periféricos e de Input/Output (I/O) do software, traduzindo-as em instruções de processamento de dados para a central processing unit (CPU).

Kernel

- A interface do kernel é uma **camada de abstração** de baixo nível.
- Quando um processo solicita um serviço para o kernel, ele deve invocar uma system call, geralmente por meio de uma **função wrapper** que é exposta a userspace applications por bibliotecas de sistema que incorporam o código assembly para inserir o kernel após carregar os **registros de CPU** com o **número syscall** e seus parâmetros (por exemplo, sistemas operacionais semelhantes ao UNIX realizam essa tarefa usando a biblioteca padrão C).

Kernel

- Existem diferentes designs de arquitetura de kernel.
- Kernels monolíticos são executados inteiramente em um único espaço de endereço com a CPU executando em modo supervisor, principalmente para velocidade.
- Os microkernels executam a maioria, mas não todos os seus serviços no espaço do usuário, como os processos do usuário fazem, principalmente para resiliência e modularidade.
- Este componente central de um sistema de computador é responsável por 'rodar' ou 'executar' programas.
- O kernel assume a responsabilidade de decidir a qualquer momento qual dos muitos programas em execução deve ser alocado para o processador ou processadores.

Random-Access Memory (RAM)

- A Random-access memory (RAM) é usada para armazenar **instruções** e **dados** do programa.
- Normalmente, ambos precisam estar presentes na memória para que um programa seja executado.
- Freqüentemente, vários programas desejam acessar a memória, frequentemente exigindo mais memória do que o computador tem disponível.
- O kernel é responsável por decidir qual memória cada processo pode usar e o que fazer quando não houver memória suficiente disponível.

Dispositivos de Input/Output

- Os dispositivos de I/O incluem periféricos como teclados, mouses, unidades de disco, impressoras, dispositivos USB, adaptadores de rede e dispositivos de exibição.
- O kernel aloca solicitações das aplicações para realizar Input/Output para um dispositivo apropriado e fornece métodos convenientes para usar o dispositivo (normalmente abstraídos ao ponto em que o aplicativo não precisa saber os detalhes de implementação do dispositivo).

Gerenciamento de Recursos

- Os principais aspectos necessários no **gerenciamento de recursos** são a definição do domínio de execução (address space) e o mecanismo de proteção usado para mediar o acesso aos recursos dentro de um domínio.
- Os kernels também fornecem métodos para sincronização e inter-process communication (IPC).
- Essas implementações podem estar dentro do próprio kernel ou o kernel também pode contar com outros processos que está executando.

Gerenciamento de Memória

- O kernel tem acesso total à memória do sistema e deve permitir que os processos acessem essa memória com segurança quando necessário.
- Freqüentemente, a primeira etapa para fazer isso é o endereçamento virtual, geralmente obtido por paginação e / ou segmentação.
- O endereçamento virtual permite que o kernel faça com que um determinado **endereço físico** pareça ser outro endereço, o **endereço virtual**.

Endereçamento Virtual

- Os espaços de endereço virtual podem ser diferentes para processos diferentes; a memória que um processo acessa em um determinado endereço (virtual) pode ser uma memória diferente daquela que outro processo acessa no mesmo endereço.
- Isso permite que cada programa se comporte como se fosse o único (além do kernel) em execução e, assim, evita que os aplicativos travem uns aos outros.
- Em muitos sistemas, o endereço virtual de um programa pode se referir a dados que não estão atualmente na memória.

Endereçamento Virtual

- A layer of indirection fornecida pelo **endereçamento virtual** permite que o sistema operacional use outros armazenamentos de dados, como um disco rígido, para armazenar o que de outra forma teria que permanecer na memória principal (RAM).
- Como resultado, os sistemas operacionais podem permitir que os programas usem mais memória do que o sistema fisicamente tem disponível. Quando um programa precisa de dados que não estão atualmente na RAM, a CPU sinaliza ao kernel que isso aconteceu, e o kernel responde gravando o conteúdo de um bloco de memória inativa no disco (se necessário) e substituindo-o pelos dados solicitados pelo programa.

Gerenciamento de Dispositivos

- Para executar funções úteis, os processos precisam de acesso aos periféricos conectados ao computador, que são controlados pelo kernel por meio de drivers de dispositivo.
- Um driver de dispositivo é um programa de computador que permite ao sistema operacional interagir com um dispositivo de hardware.
- Ele fornece ao sistema operacional informações sobre como controlar e se comunicar com uma determinada peça de hardware.
- O driver é uma peça importante e vital para a aplicação de um programa.

Gerenciamento de Dispositivos

- O objetivo do design de um driver é a abstração; a função do driver é traduzir as chamadas de função abstratas exigidas pelo sistema operacional (chamadas de programação) em chamadas específicas do dispositivo.
- Em teoria, o dispositivo deve funcionar corretamente com o driver adequado.
- Os drivers de dispositivo são usados para coisas como placas de vídeo, placas de som, impressoras, scanners, modems e placas de LAN.

Gerenciamento de Dispositivos

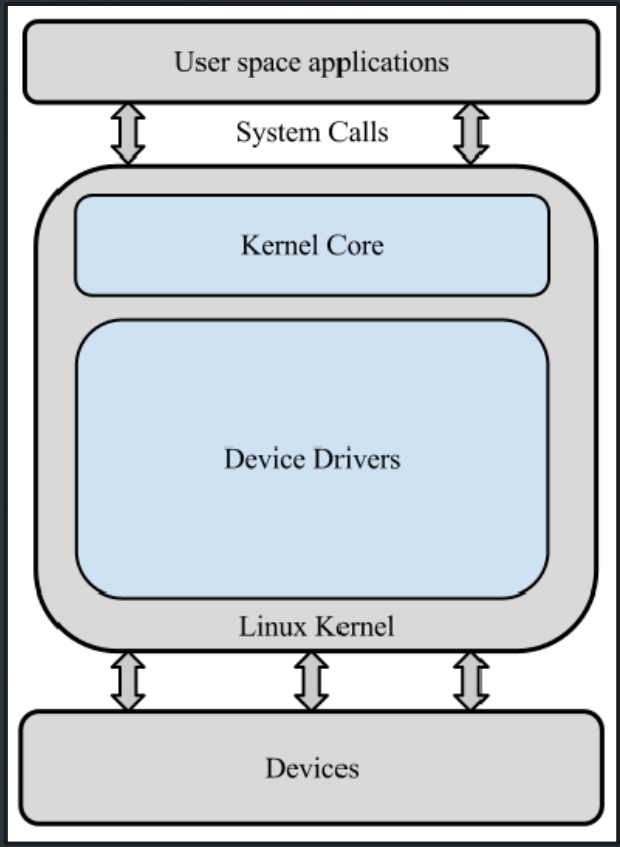
- No nível de hardware, abstrações comuns de drivers de dispositivo incluem:
  - Interface direta
  - Usando uma interface de alto nível (BIOS de vídeo)
  - Usando um driver de dispositivo de nível inferior (drivers de arquivo usando drivers de disco)
  - Simular trabalho com hardware, enquanto faz algo totalmente diferente

Gerenciamento de Dispositivos

- E no nível do software, as abstrações do driver de dispositivo incluem:
  - Permitir ao sistema operacional acesso direto aos recursos de hardware
  - Implementar uma interface para software sem driver, como TWAIN
  - Implementar uma linguagem (geralmente uma linguagem de alto nível, como PostScript)

Gerenciamento de Dispositivos

- Por exemplo, para mostrar ao usuário algo na tela, um aplicativo faria uma solicitação ao kernel, que encaminharia a solicitação ao seu driver de exibição, que seria então responsável por realmente plotar o caractere / pixel.



System Calls

- Na computação, um system call é como um processo solicita um serviço do kernel de um sistema operacional que normalmente não tem permissão para executar.
- System calls fornecem a interface entre um **processo** e o **sistema operacional**.
- A maioria das operações que interagem com o sistema requer permissões não disponíveis para um processo de nível de usuário, por exemplo, I/O realizada com um dispositivo presente no sistema, ou qualquer forma de comunicação com outros processos requer o uso de system calls.

System Calls

- O system call é um mecanismo usado pela aplicação de programa para solicitar um serviço do sistema operacional.
- Eles usam uma instrução de código de máquina que faz com que o processador mude de modo.
- Um exemplo seria do modo supervisor para o modo protegido.
- É aqui que o sistema operacional executa ações como acessar dispositivos de hardware ou a memory management unit.
- Geralmente, o sistema operacional fornece uma biblioteca que fica entre o sistema operacional e os programas normais do usuário.

Unix Development Environment

- As primeiras versões do Unix continham um ambiente de desenvolvimento suficiente para recriar todo o sistema a partir do código-fonte:
  - cc - Compilador da linguagem C
  - as - assembler de linguagem de máquina para a máquina
  - ld - linker, para combinar object files
  - lib - bibliotecas de código-objeto (instaladas em /lib ou /usr/lib)
  - make - build manager para automatizar efetivamente o processo de construção
  - include - header files para desenvolvimento de software, definindo interfaces padrão e invariantes de sistema

Comandos Unix

- Algumas categorias principais de comandos são:
  - sh - o interpretador de linha de comando programável "shell", a interface de usuário principal no Unix antes que os sistemas de janela aparecessem
  - Utilitários - o kit de ferramentas principal do conjunto de comandos do Unix, incluindo cp, ls, grep, find e muitos outros.
  - Formatação de documentos - os sistemas Unix foram usados desde o início para a preparação de documentos e sistemas de composição tipográfica

Documentação Unix

- O Unix foi o primeiro sistema operacional a incluir toda a sua documentação online em formato legível por máquina. A documentação incluiu:
  - man - páginas de manual para cada comando, componente de biblioteca, system call, header file, etc.
  - doc - documentos mais longos detalhando os principais subsistemas, como a linguagem C.

Características Unix

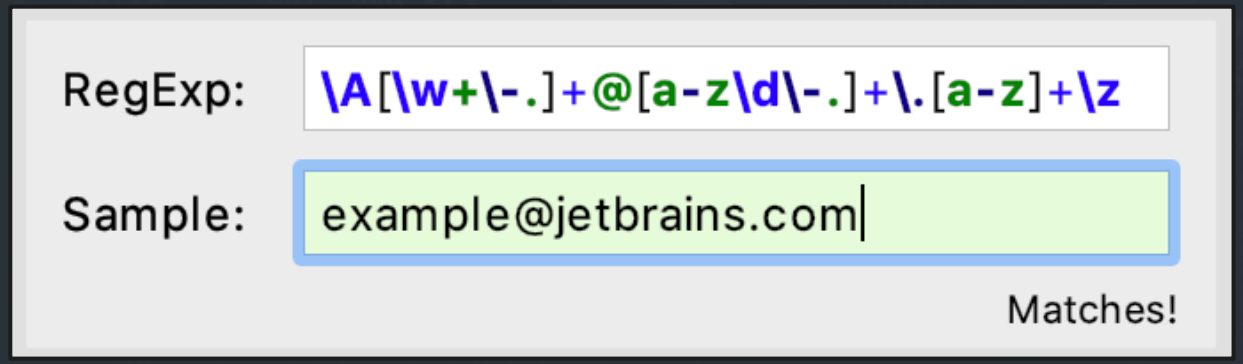
- Unix possui recursos **multitarefa** e **multiusuário** em uma configuração time-sharing.
- Os sistemas Unix são caracterizados por vários conceitos:
  - O uso de texto simples (plain text) para armazenar dados;
  - Um sistema de arquivos hierárquico;
  - Dispositivos e certos tipos de inter-process communication (IPC) são tratados como arquivos;
  - Uso ferramentas de software, pequenos programas que podem ser agrupados por meio de um interpretador de linha de comando usando pipes;

Impacto Unix

- No início dos anos 1980, os usuários começaram a ver o Unix como um sistema operacional universal em potencial, adequado para computadores de todos os tamanhos.
- O ambiente Unix e o modelo de programa cliente-servidor foram elementos essenciais no desenvolvimento da Internet e na reformulação da computação centrada em redes, em vez de em computadores individuais.
- Os protocolos de rede TCP/IP foram rapidamente implementados nas versões Unix amplamente utilizadas em computadores relativamente baratos, o que contribuiu para a explosão da conectividade em tempo real mundial da Internet.

Impacto Unix

- O Unix popularizou uma sintaxe para expressões regulares que encontraram uso generalizado.
- A linguagem de programação C logo se espalhou além do Unix e agora é onipresente na programação de sistemas e aplicações.



Unix Livre e Unix Variantes

Em 1983, Richard Stallman anunciou o projeto GNU (abreviação de "GNU's Not Unix"), um esforço ambicioso para criar um sistema de software livre semelhante ao Unix; "livre" no sentido de que todos que receberam uma cópia seriam livres para usá-la, estudar, modificar e redistribuí-la.

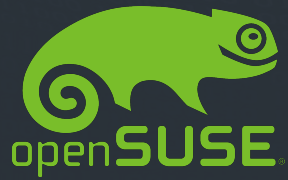
Unix Livre e Unix Variantes

- O próprio projeto de desenvolvimento de kernel do projeto GNU, GNU Hurd, ainda não havia produzido um kernel funcional, mas em 1991 Linus Torvalds lançou o kernel Linux como software livre sob a **GNU General Public License**.
- Além de seu uso no sistema operacional GNU, muitos pacotes GNU - como o GNU Compiler Collection (e o resto da cadeia de ferramentas GNU), a biblioteca GNU C e os utilitários básicos GNU - passaram a desempenhar papéis centrais em outros sistemas Unix gratuitos também.

Distribuições Linux

- As distribuições Linux, consistindo no kernel Linux e grandes coleções de software compatível, tornaram-se populares tanto para usuários individuais quanto para empresas.
- Distribuições populares incluem: Red Hat Enterprise Linux, Fedora, SUSE Linux Enterprise, openSUSE, Debian GNU/Linux, Ubuntu, Linux Mint, Mandriva Linux, Slackware Linux, Arch Linux e Gentoo.







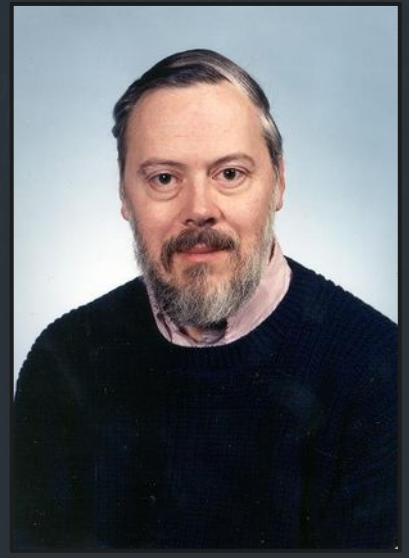
Berkeley Software Distribution (BSD)

- Um derivado gratuito do BSD Unix, 386BSD, foi lançado em 1992 e levou aos projetos NetBSD e FreeBSD.
- Com o acordo de 1994 de um processo movido contra a Universidade da Califórnia e Berkeley Software Design Inc. por Unix System Laboratories, foi esclarecido que Berkeley tinha o direito de distribuir BSD Unix gratuitamente se assim o desejasse.
- Desde então, o BSD Unix foi desenvolvido em vários ramos de produtos diferentes, incluindo OpenBSD e DragonFly BSD.

Dennis Ritchie

- Em uma entrevista de 1999, Dennis Ritchie expressou sua opinião de que os sistemas operacionais Linux e BSD são uma continuação da base do design do Unix e são derivados do Unix:

> “I think the Linux phenomenon is quite delightful, because it draws so strongly on the basis that Unix provided. Linux seems to be the among the healthiest of the direct Unix derivatives, though there are also the various BSD systems as well as the more official offerings from the workstation and mainframe manufacturers.”



Desenvolvimento Linux

- O Linux foi originalmente desenvolvido para computadores pessoais baseados na arquitetura Intel x86, mas desde então foi portado para mais plataformas do que qualquer outro sistema operacional.
- Por causa do domínio do Android baseado em Linux nos smartphones, o Linux também tem a maior base instalada de todos os sistemas operacionais de uso geral.
- Linux é o sistema operacional líder em servidores (mais de 96,4% dos principais 1 milhão de sistemas operacionais de servidores da web são Linux), lidera outros grandes sistemas, como computadores mainframe, é também o único sistema operacional usado em supercomputadores TOP500.

Embedded Systems

- O Linux também é executado em sistemas embarcados, ou seja, dispositivos cujo sistema operacional é normalmente integrado ao firmware e é altamente adaptado ao sistema.
- Isso inclui roteadores, controles de automação, smart home technology (como o Google Nest), televisores (Samsung e LG Smart TVs usam Tizen e WebOS, respectivamente), automóveis (por exemplo, Tesla, Audi, Mercedes-Benz, Hyundai e Toyota, todos dependem no Linux), gravadores de vídeo digital, consoles de videogame e smartwatches.

Linux Open-Source

- Linux é um dos exemplos mais proeminentes de colaboração de software livre e de código aberto.
- O código-fonte pode ser usado, modificado e distribuído comercialmente ou não comercialmente por qualquer pessoa sob os termos de suas respectivas licenças, como a GNU General Public License.
- É possível encontrar o código-fonte do kernel Linux no GitHub: https://github.com/torvalds/linux

Componentes de um Sistema Linux

- Os componentes instalados de um sistema Linux incluem o seguinte:
  - Um bootloader, por exemplo GNU GRUB, LILO, SYSLINUX ou Gummiboot. É um programa que carrega o kernel do Linux na memória principal do computador, sendo executado pelo computador ao ser ligado e após a inicialização do firmware.
  - Um programa init, como o sysvinit tradicional e o mais recente systemd, OpenRC e Upstart. Este é o primeiro processo lançado pelo kernel Linux e está na raiz da árvore de processos: em outros termos, todos os processos são iniciados por meio do init. Ele inicia processos como serviços do sistema e prompts de login (seja gráfico ou em modo de terminal).

Componentes de um Sistema Linux

- Os componentes instalados de um sistema Linux incluem o seguinte:
  - Bibliotecas de software, que contêm código que pode ser usado por processos em execução. Em sistemas Linux que usam arquivos executáveis no formato ELF, o dynamic linker que gerencia o uso de bibliotecas dinâmicas é conhecido como ld-linux.so. Se o sistema estiver configurado para o próprio usuário compilar o software, os header files também serão incluídos para descrever a interface das bibliotecas instaladas. Além da biblioteca de software mais comumente usada em sistemas Linux, a GNU C Library (glibc), existem inúmeras outras bibliotecas, como SDL e Mesa.

Componentes de um Sistema Linux

- Os componentes instalados de um sistema Linux incluem o seguinte:
  - Comandos Unix básicos, com GNU coreutils sendo a implementação padrão. Existem alternativas para sistemas embarcados, como o copyleft BusyBox e o Toybox licenciado por BSD.
  - Os Widget toolkits são as bibliotecas usadas para construir graphical user interfaces (GUIs) para aplicações de software. Vários Widget toolkits estão disponíveis, incluindo GTK e Clutter desenvolvidos pelo projeto GNOME, Qt desenvolvido pelo Projeto Qt e liderado pela Digia e Enlightenment Foundation Libraries (EFL).
  - Um package management system, como dpkg e RPM. Alternativamente, os pacotes podem ser compilados a partir de tarballs binários ou de origem.
  - Programas de interface de usuário, como shells de comando ou ambientes de janelas.

Interface de Usuário

- A interface do usuário, também conhecida como shell, é uma command-line interface (CLI), uma graphical user interface (GUI) ou controles anexados ao hardware associado, o que é comum para sistemas embarcados.
- Para sistemas de desktop, a interface de usuário padrão geralmente é gráfica, embora a CLI esteja comumente disponível por meio de janelas do emulador de terminal ou em um console virtual separado.

Command-Line Interface

- Os shells CLI são interfaces de usuário baseadas em texto, que usam texto para input e output.
- O shell dominante usado no Linux é o Bourne-Again Shell (bash), originalmente desenvolvido para o projeto GNU.
- O CLI é particularmente adequado para automação de tarefas repetitivas ou atrasadas e fornece inter-process communication muito simples.

Command-Line Interface

- Uma interface de linha de comando (CLI) processa comandos para um programa de computador na forma de linhas de texto.
- O programa que lida com a interface é chamado de interpretador de linha de comando ou processador de linha de comando.
- Os sistemas operacionais implementam uma interface de linha de comando em um shell para acesso interativo às funções ou serviços do sistema operacional.
- Tal acesso foi fornecido principalmente aos usuários por terminais de computador a partir de meados da década de 1960 e continuou a ser usado ao longo das décadas de 1970 e 1980 em VAX/VMS, sistemas Unix e sistemas de computador pessoal, incluindo DOS, CP/M e Apple DOS.

Shell

- Shell é um programa utilitário Unix.
- Ele possui uma identidade dupla: como uma interface do usuário para os utilitários Unix e como uma linguagem de programação, facilitando o uso e a combinação dos utilitários Unix.
- Quando abrimos o terminal, o programa shell é carregado na memória do computador.
- Quando digitamos comandos no terminal, o shell lê os comandos e os converte em um formato legível pelo kernel a ser executado.
- Ele fornece uma instância interativa para iniciar programas, gerenciar arquivos e processos em execução no computador.

Shell

- Como o shell é apenas um programa, muitas variações foram criadas desde 1969, quando Ken Thompson desenvolveu a primeira implementação do Unix no Bell Labs.
- O shell Unix original foi escrito por Steve Bourne em 1970 e é conhecido como Bourne shell ou sh.
- O Bourne shell não estava disponível gratuitamente na época, o que limitava seu uso por outros programadores.
- Para aliviar esse problema, em 1988, a Free Software Foundation encarregou Brian Fox de desenvolver uma reimplementação de código aberto do Bourne shell, o chamado Bourne again shell ou bash.

Bash

- O Bash é um processador de comandos que normalmente é executado em uma janela de texto em que o usuário digita comandos que causam ações.
- O Bash também pode ler e executar comandos de um arquivo, chamado de shell script.
- Como todos os shells do Unix, ele suporta filename globbing (wildcard matching), piping, here documents, command substitution, variáveis, e estruturas de controle para condition-testing e iteration.

Comandos Básicos

- Iniciaremos abrindo o terminal para assim podermos executar os nossos primeiros comandos, vamos utilizar o comando CTRL + ALT + T para inicializá-lo.
- Imprimindo Hello World:
  - O comando echo no linux é usado para exibir o texto/string que é passada como argumento.

```
$ echo “Hello World”
```

Comandos Básicos

- Imprimindo Variáveis de Ambiente:

```
$ echo $HOME # Diretório Home do usuário atual
$ echo $PWD # Imprime o diretório de trabalho atual
$ echo $TERM # Terminal sendo utilizado
$ echo $USER # Usuário atual do sistema
$ echo $SHELL # Shell sendo utilizada
$ echo $HOSTNAME # O nome do host do computador no momento
$ echo $LANG # Linguagem atual em uso
$ echo $BASH_VERSION # Versão do Bash
```

Comandos Básicos

- O comando history exibe ou manipula a lista do histórico de comandos:

```
$ history
```

- O comando clear ou CTRL + L limpa a tela do terminal:

```
$ clear
$ CTRL + L
```

Comandos Básicos

- O comando exit ou CTRL + D nos permite fechar o terminal:

```
$ exit
$ CTRL + D
```

- O comando man é uma interface para os manuais de referência online:

```
$ man python
$ man libc
$ man cat
```

Comandos Básicos

- Muitos programas executáveis suportam uma opção "--help" que exibe uma descrição da sintaxe e das opções suportadas pelo comando:

```
$ php --help
$ chmod --help
$ ls --help
```

- Muitos programas suportam uma opção "--version" que exibe a versão atual do programa que estamos utilizando.

```
$ bash --version
$ python --version
```

Informações do Sistema

- Exibir informações do sistema Linux:

```
$ uname -a
```

- Exibir informações de lançamento do kernel:

```
$ uname -r
```

- O arquivo `/etc/os-release` contém dados de identificação do sistema operacional, incluindo informações sobre a distribuição. Este arquivo é parte do pacote systemd e deve estar presente em todos os sistemas que executam o systemd.

```
$ cat /etc/os-release
```

Informações do Sistema

- Mostrar há quanto tempo o sistema está funcionando + carregado:

```
$ uptime
```

- Mostrar o hostname do sistema:

```
$ hostname
```

- Exibe todos os endereços IP locais do host:

```
$ hostname -I
```

- Mostrar histórico de reinicialização do sistema:

```
$ last reboot
```

Informações do Sistema

- Mostra a data e hora atuais:

```
$ date
```

- Mostra o calendário deste mês:

```
$ cal
```

- Apresenta quem está online:

```
$ w
```

- Apresenta com quem você está logado:

```
$ whoami
```

Informações de Hardware

- Exibir mensagens no kernel ring buffer:

```
$ dmesg
```

- Apresenta informações do CPU:

```
$ cat /proc/cpuinfo
```

- Apresenta informações da Memória:

```
$ cat /proc/meminfo
```

- Exibir memória livre e usada (-h para leitura humana, -m para MB, -g para GB):

```
$ free -h
```

Informações de Hardware

- Exibir dispositivos PCI:

```
$ lspci -tv
```

- Exibir dispositivos USB:

```
$ lsusb -tv
```

- Exibir DMI/SMBIOS (informações de hardware) do BIOS:

```
$ dmidecode
```

- Mostrar informações sobre o disco sda:

```
$ hdparm -i /dev/sda
```

Informações de Hardware

- Relatar informações sobre a CPU e unidades de processamento:

```
$ lscpu
```

- O mount é usada para montar/desmontar e visualizar sistemas de arquivos montados:

```
$ mount | column -t
```

- Executar um teste de velocidade de leitura no disco sda:

```
$ hdparm -tT /dev/sda
```

- Teste para blocos ilegíveis no disco sda:

```
$ badblocks -s /dev/sda
```

Monitoramento de Desempenho

- Exibir e gerenciar os principais processos:

```
$ top
```

- Visualizador de processo interativo (alternativa ao top):

```
$ htop
```

- Exibir estatísticas relacionadas ao processador:

```
$ mpstat 1
```

- Exibir estatísticas de memória virtual:

```
$ vmstat 1
```

Monitoramento de Desempenho

- Exibir estatísticas de I/O:

```
$ iostat 1
```

- Exibir as últimas 100 mensagens syslog (Use `/var/log/syslog` para sistemas baseados em Debian):

```
$ tail -100 /var/log/messages
```

- Capture e exiba todos os pacotes na interface eth0:

```
$ tcpdump -i eth0
```

- Monitore todo o tráfego na porta 80 (HTTP):

```
$ tcpdump -i eth0 'port 80'
```

Monitoramento de Desempenho

- Listar todos os arquivos abertos no sistema:

```
$ lsof
```

- Listar arquivos abertos pelo usuário:

```
$ lsof -u user
```

- Execute "df -h", mostrando atualizações periódicas:

```
$ watch df -h
```

- Para mostrar estatísticas de todos os protocolos:

```
$ netstat –s
```

Informação e Gerenciamento de Usuário

- Exibir os IDs de usuário e grupo de seu usuário atual:

```
$ id
```

- Exibe os últimos usuários que se conectaram ao sistema:

```
$ last
```

- Mostra quem está logado no sistema:

```
$ who
```

- Crie um grupo chamado "test":

```
$ groupadd test
```

Informação e Gerenciamento de Usuário

- Crie uma conta chamada akira, com um comentário de "Usuário Akira" e crie o diretório home do usuário:

```
$ useradd -c "Usuário Akira" -m akira
```

- Exclua a conta akira:

```
$ userdel akira
```

- Adicione a conta akira ao grupo test:

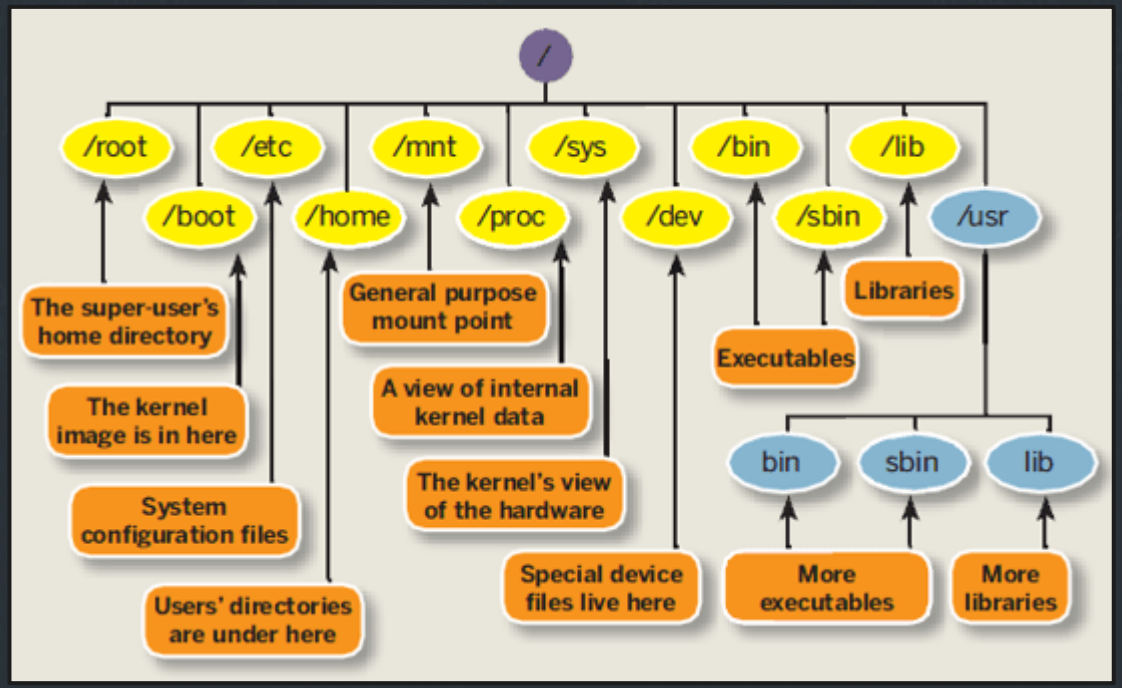
```
$ usermod -aG test akira
```

- Altere a senha da sua própria conta de usuário:

```
$ passwd
```

Filesystem Hierarchy Standard

- O Filesystem Hierarchy Standard (FHS) define a estrutura e o conteúdo do diretório nas distribuições Linux. É mantido pela Linux Foundation.



Comandos de Arquivos e Diretórios

- Liste todos os arquivos e diretórios em um formato de lista longa (detalhada):

```
$ ls -al
```

- Mostre o diretório de trabalho atual (print working directory):

```
$ pwd
```

- Crie um diretório:

```
$ mkdir fotos
```

- Remover (deletar) um arquivo:

```
$ rm arquivo
```

Comandos de Arquivos e Diretórios

- Remova o diretório e seu conteúdo recursivamente:

```
$ rm -r scripts
```

- Forçar a remoção do arquivo sem solicitar confirmação:

```
$ rm -f script.py
```

- Remover diretório recursivamente à força:

```
$ rm -rf projetos
```

- Copiar arquivo1 para arquivo2:

```
$ cp arquivo1 arquivo2
```

Comandos de Arquivos e Diretórios

- Copie source_directory recursivamente para o destination. Se o destination existir, copie source_directory para dentro do destination, caso contrário, crie o destination com o conteúdo do source_directory:

```
$ cp -r source_directory destination
```

- Renomeie ou mova arquivo1 para arquivo2. Se o arquivo2 for um diretório existente, mova o arquivo1 para o diretório arquivo2:

```
$ mv arquivo1 arquivo2
```

- Crie um link simbólico para linkname:

```
$ ln -s /path/to/file linkname
```

Comandos de Arquivos e Diretórios

- Crie um arquivo vazio ou atualize os horários de acesso e modificação do arquivo:

```
$ touch arquivo
```

- Escreva o conteúdo “Hello World” no arquivo:

```
$ echo "Hello World" > arquivo
```

- Visualize o conteúdo do arquivo:

```
$ cat arquivo
```

- Navegue por um arquivo de texto:

```
$ cat arquivo
```

Comandos de Arquivos e Diretórios

- Mostre as primeiras 10 linhas do arquivo:

```
$ head arquivo
```

- Mostre as últimas 10 linhas do arquivo:

```
$ tail arquivo
```

- Mostre as últimas 10 linhas do arquivo e "siga" o arquivo à medida que ele cresce:

```
$ tail -f arquivo
```

- Liste o conteúdo dos diretórios em formato de árvore:

```
$ tree
```

Navegação de Diretórios

- Navegar para o diretório home:

```
$ cd ~
```

- Para subir um nível na árvore de diretórios (mudar para o diretório pai):

```
$ cd ..
```

- Mudar para o diretório /etc:

```
$ cd /etc
```

- Navegando para o diretório anterior que estávamos:

```
$ cd -
```

Gerenciamento de Processos

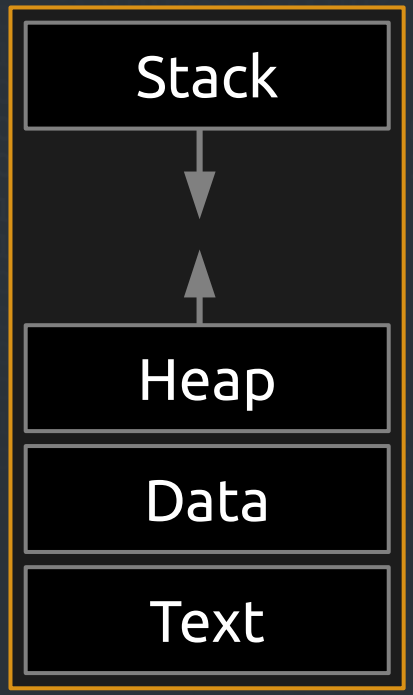
- Um processo é um programa em execução.
- Por exemplo, quando escrevemos um programa em C ou C++ e o compilamos, o compilador cria um código binário.
- O código original e o código binário são ambos programas.
- Quando realmente executamos o código binário, ele se torna um processo.

Gerenciamento de Processos

- Um processo é uma entidade "ativa", ao contrário de um programa, que é considerado uma entidade "passiva".
- Um único programa pode criar muitos processos quando executado várias vezes.
- Por exemplo, quando abrimos um .exe ou um arquivo binário várias vezes, várias instâncias começam (vários processos são criados).

Gerenciamento de Processos

- Qual é a aparência de um processo na memória?



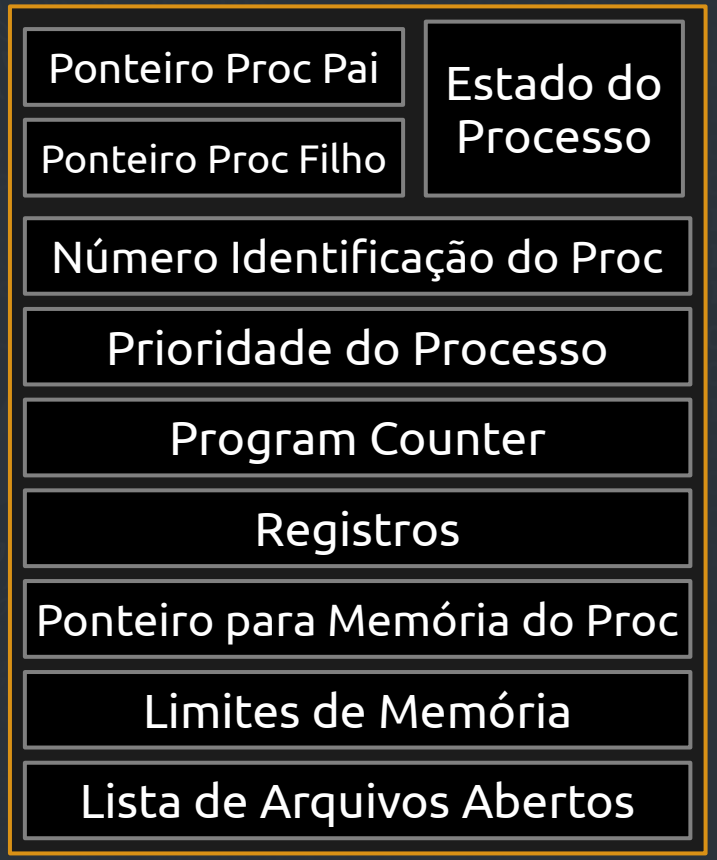
- Text Section: Um Processo, às vezes conhecido como Seção de Texto, também inclui a atividade atual representada pelo valor do Program Counter.
- Stack: A pilha contém os dados temporários, como parâmetros de função, endereços de retorno e variáveis locais.
- Data: Contém a variável global.
- Heap: Memória alocada dinamicamente para processar durante o tempo de execução.

Atributos dos Processos

- Um processo possui os seguintes atributos:
  1. ID do Processo: um identificador único atribuído pelo sistema operacional.
  2. Estado do Processo: pode estar pronto, em execução, etc.
  3. Registros da CPU: como o contador do programa (os registros da CPU devem ser salvos e restaurados quando um processo entra e sai da CPU)
  4. Informações de Contas.
  5. Informações de status de I/O: Por exemplo, dispositivos alocados para o processo, arquivos abertos, etc.
  6. Informações de agendamento da CPU: Por exemplo, Prioridade (processos diferentes podem ter prioridades diferentes, por exemplo, um processo curto pode ser atribuído a uma prioridade baixa no primeiro agendamento do trabalho (job) mais curto)

Atributos dos Processos

- Todos os atributos de um processo citados anteriormente também são conhecidos como o contexto do processo.
- Cada processo tem seu próprio process control block (PCB), ou seja, cada processo terá um PCB exclusivo. Todos os atributos anteriores fazem parte do PCB.

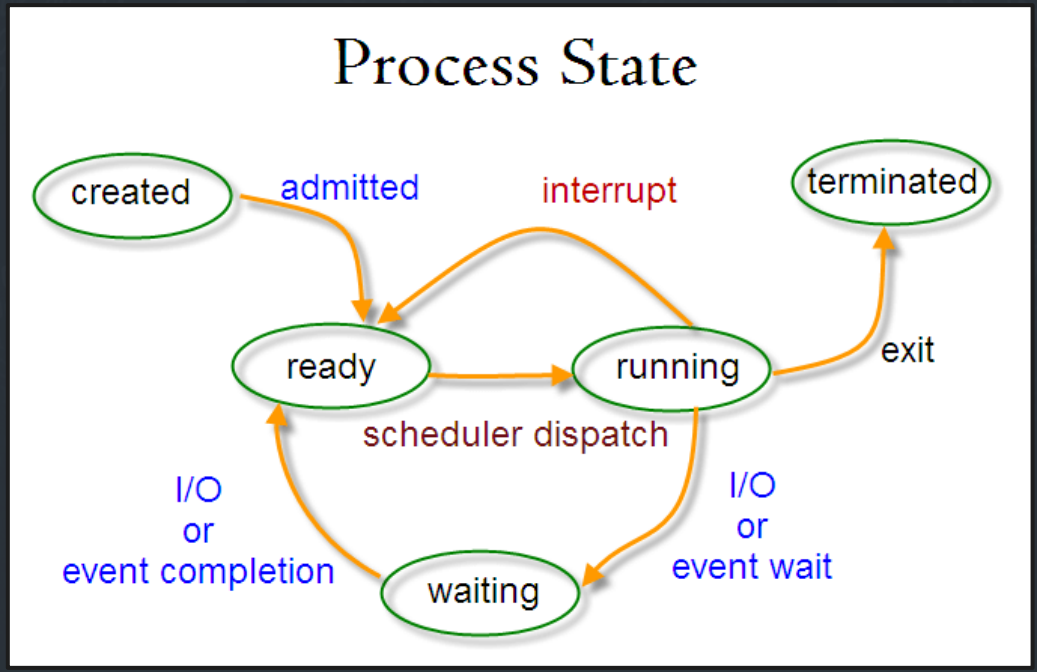


Estados de um Processo

- Um processo está em um dos seguintes estados:
  1. New: Processo recém-criado (ou) processo sendo criado.
  2. Ready: Depois da criação do processo, ele passa para o estado Ready, ou seja, o processo está pronto para execução.
  3. Run: Processo atualmente em execução na CPU (apenas um processo por vez pode estar em execução em um único processador).
  4. Wait (ou Block): Quando um processo solicita acesso de I/O.
  5. Complete (ou Terminated): O processo completou sua execução.
  6. Suspended Ready: Quando a fila de prontidão fica cheia, alguns processos são movidos para o estado de prontidão suspensa.
  7. Suspended Block: Quando a fila de espera fica cheia.

Estados de um Processo

- O diagrama a seguir mostra uma ideia simplificada dos estados de um processo:



Context Switching

- O processo de salvar o contexto de um processo e carregar o contexto de outro processo é conhecido como Context Switching. Em termos simples, é como carregar e descarregar o processo do running state para o ready state.
- Quando ocorre a troca de contexto?

    1. Quando um processo de alta prioridade chega a um ready state (ou seja, com maior prioridade do que o processo em execução).
    2. Ocorre uma interrupção.
    3. Alternância de modo de usuário e kernel.
    4. Agendamento de CPU preemptivo usado.

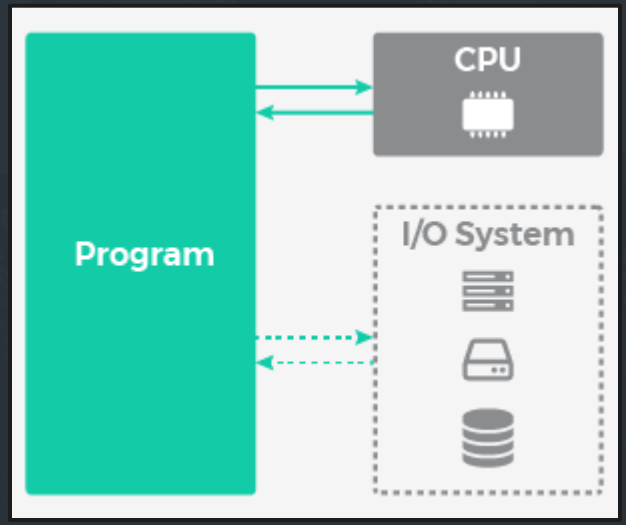
Context Switch vs Mode Switch

- Uma Mode Switch ocorre quando o nível de privilégio da CPU é alterado, por exemplo, quando uma system call é feita ou ocorre uma falha.
- O kernel funciona em um modo mais privilegiado do que em uma tarefa de usuário padrão.
- Se um processo do usuário deseja acessar coisas que são acessíveis apenas ao kernel, um Mode Switch deve ocorrer.
- O processo atualmente em execução não precisa ser alterado durante o Mode Switch.
- Um Mode Switch normalmente ocorre para que uma Context Switch de processo ocorra. Apenas o kernel pode causar uma Context Switch.

Processos CPU-Bound vs I/O-Bound

- Um processo CPU-Bound requer mais tempo da CPU ou gasta mais tempo no running state.
- Um processo I/O-Bound requer mais tempo de Input/Output e menos tempo de CPU. Um processo I/O-Bound passa mais tempo no waiting state.

CPU-Bound:
Operações passam a maior parte do seu tempo fazendo lógica e cálculo intensivo no CPU.



I/O-Bound: Operações que esperam a maior parte do tempo pela rede, sistema de arquivos ou banco de dados.

Gerenciamento de Processos (Comandos)

- Exibir seus processos atualmente em execução:

```
$ ps
```

- Exibir todos os processos em execução no sistema:

```
$ ps -ef
```

- Exibir informações do processo para processname:

```
$ ps -ef | grep processname
```

- Mate o processo com o ID do processo (pid):

```
$ kill pid
```

Gerenciamento de Processos (Comandos)

- Mate todos os processos nomeados processname:

```
$ killall processname
```

- Iniciar o programa em background:

```
$ program &
```

- Apresentar jobs “stopped” ou em “background”:

```
$ bg
```

- Traz o mais recente background job para o foreground:

```
$ fg
```

Processos Foreground vs Background

- Os processos de primeiro plano (foreground) referem-se aos aplicativos que você está executando e com os quais está interagindo no momento, e que se aplicam igualmente às interfaces gráficas do usuário e à linha de comando.
- Os processos em segundo plano (background) referem-se a aplicativos que estão em execução, mas não estão sendo interagidos pelo usuário. Como tal, os servidores Linux geralmente passam muito tempo com todos os seus processos sendo executados como processos em segundo plano.

O Comando Jobs

- Você provavelmente está familiarizado com a inicialização de aplicações na linha de comando, digitando o comando para iniciá-los.
- Isso normalmente deixa você com o programa em execução, sendo que você está interagindo por meio da tela e do teclado.
- Você também deve estar familiarizado com a combinação CTRL + C para encerrar o programa atual e, com sorte, colocá-lo de volta na linha de comando.
- A combinação de teclas CTRL + Z também coloca você de volta na linha de comando, mas em vez de encerrar o programa, ele o interrompe.

O Comando Jobs

- Isso efetivamente pausa o programa, da mesma forma que pausar um vídeo ou música em um reprodutor de mídia.
- Ele permanecerá neste estado de pausa até que você faça algo com ele.
- Para ver uma lista de programas que você está executando ou pausou, você pode usar o comando jobs:

```
$ sleep 100 &
$ jobs -l
```

Permissões no Linux

- Linux é um clone do UNIX, o sistema operacional multiusuário que pode ser acessado por vários usuários simultaneamente.
- O Linux também pode ser usado em mainframes e servidores sem nenhuma modificação.
- Mas isso levanta questões de segurança, pois um usuário não solicitado ou maligno pode corromper, alterar ou remover dados cruciais.
- Para uma segurança eficaz, o Linux divide a autorização em 2 níveis:
    1. Ownership (Propriedade)
    2. Permission (Permissão)

Ownership de Arquivos Linux

- Cada arquivo e diretório em seu sistema Unix/Linux são atribuídos a 3 tipos de proprietário (owner):
  - User: Um usuário é o proprietário do arquivo. Por padrão, a pessoa que criou um arquivo se torna seu proprietário. Portanto, às vezes, um usuário também é chamado de owner.
  - Group: Um grupo de usuários pode conter vários usuários. Todos os usuários pertencentes a um grupo terão as mesmas permissões de grupo Linux de acesso ao arquivo. Suponha que você tenha um projeto em que várias pessoas requeiram acesso a um arquivo. Em vez de atribuir permissões manualmente a cada usuário, você pode adicionar todos os usuários a um grupo e atribuir permissão de grupo ao arquivo de forma que apenas os membros desse grupo e ninguém mais possa ler ou modificar os arquivos.

Ownership de Arquivos Linux

- Other: Qualquer outro usuário que tenha acesso a um arquivo. Esta pessoa não criou o arquivo, nem pertence a um grupo de usuários que poderia ser o proprietário do arquivo. Praticamente, significa todo mundo. Portanto, quando você define a permissão para outras pessoas, também é referido como definir permissões para o mundo.

Permissões

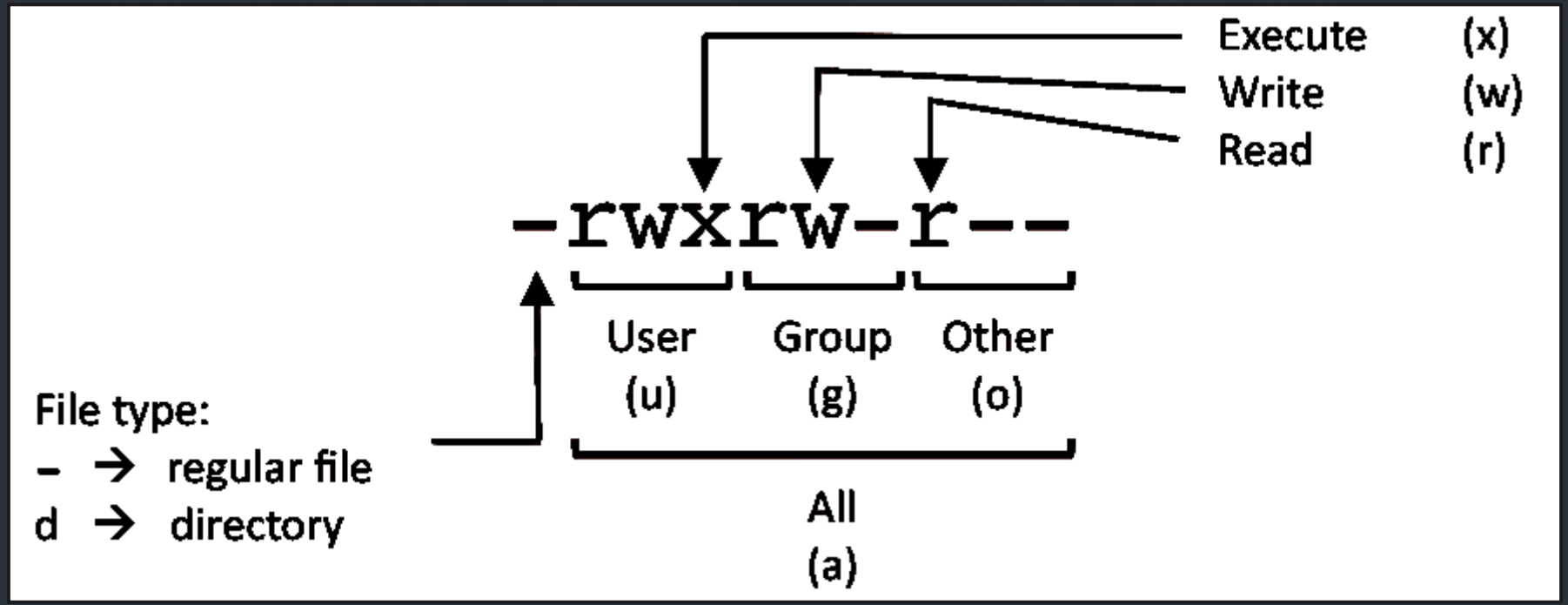
- Todo arquivo e diretório em seu sistema UNIX/Linux tem as seguintes 3 permissões definidas para todos os 3 proprietários discutidos anteriormente:
  - Read: Esta permissão dá a você autoridade para abrir e ler um arquivo. A permissão de leitura em um diretório permite listar seu conteúdo.
  - Write: A permissão de gravação dá a você autoridade para modificar o conteúdo de um arquivo. A permissão de gravação em um diretório lhe dá autoridade para adicionar, remover e renomear arquivos armazenados no diretório. Considere um cenário em que você precisa ter permissão de gravação no arquivo, mas não tem permissão de gravação no diretório onde o arquivo está armazenado. Você poderá modificar o conteúdo do arquivo. Mas você não poderá renomear, mover ou remover o arquivo do diretório.

Permissões

- Execute: No Windows, um programa executável geralmente tem uma extensão ".exe" e que você pode executar facilmente. No Unix/Linux, você não pode executar um programa a menos que a permissão de execução seja definida. Se a permissão de execução não estiver definida, você ainda poderá ver/modificar o código do programa (desde que as permissões de leitura e gravação estejam definidas), mas não executá-lo.

Permissões

- A imagem a seguir ilustra as permissões para os diferentes proprietários de um arquivo ou diretório:



Permissões

- Alterando as permissões com o comando chmod:

Permissão			Comando
User	Group	Other	r = Read, w = Write, x = Execute
rwx	rwx	rwx	$ chmod 777 arquivo
rwx	rwx	r-x	$ chmod 775 arquivo
rwx	r-x	r-x	$ chmod 755 arquivo
rw-	rw-	r--	$ chmod 664 arquivo
rw-	r--	r--	$ chmod 644 arquivo

Networking

- O Linux gerencia facilmente vários adaptadores de interface de rede.
- Os laptops geralmente incluem interfaces com e sem fio e também podem oferecer suporte a interfaces WiMax para redes celulares.
- Os computadores desktop Linux também suportam várias interfaces de rede e você pode usar seu computador Linux como um cliente de várias redes ou como um roteador para redes internas

Comandos de Rede

- Exibir todas as interfaces de rede e endereço IP:

```
$ ip a
```

- Exibir endereço eth0 e detalhes:

```
$ ip addr show dev eth0
```

- Consultar ou controlar o driver de rede e as configurações de hardware:

```
$ ethtool eth0
```

- Enviar echo request ICMP ao host:

```
$ ping host
```

Comandos de Rede

- Exibir informações whois para o domínio:

```
$ whois domain
```

- Exibir informações de DNS para o domínio:

```
$ dig domain
```

- Pesquisa reversa de Endereço IP:

```
$ dig -x ENDEREÇO_IP
```

- Exibir endereço IP DNS para o domínio:

```
$ host domain
```

Comandos de Rede

- Exibe o endereço de rede do hostname:

```
$ hostname -i
```

- Exibe todos os endereços IP locais do host:

```
$ hostname -I
```

- Download de arquivos de um endereço (URL):

```
$ wget http://domain.com/file
```

- Exibir portas TCP e UDP em escuta e programas correspondentes:

```
$ netstat -nutlp
```

Archives (Arquivos TAR)

- Crie o tar chamado archive.tar contendo o directory:

```
$ tar cf archive.tar directory
```

- Extraia o conteúdo de archive.tar:

```
$ tar xf archive.tar
```

- Crie um arquivo tar compactado com gzip com o nome archive.tar.gz:

```
$ tar czf archive.tar.gz directory
```

- Extraia um arquivo tar compactado com gzip:

```
$ tar xzf archive.tar.gz
```

Instalando Pacotes

- Pesquise um pacote por palavra-chave:

```
$ apt search keyword
```

- Instale um pacote:

```
$ apt install package
```

- Exibir descrição e informações resumidas sobre o pacote:

```
$ apt info package
```

- Instale o pacote do arquivo local denominado package.rpm:

```
$ rpm -i package.rpm
```

Instalando Pacotes

- Removendo / desinstalando um pacote:

```
$ apt remove package
```

- Instalando software através do source code:

```
$ tar zxvf sourcecode.tar.gz
$ cd sourcecode
$ ./configure
$ make
$ make install
```

Buscas

- Pesquise por um padrão em um arquivo:

```
$ grep pattern arquivo
```

- Pesquise recursivamente pelo padrão no diretório:

```
$ grep -r pattern directory
```

- Encontre arquivos e diretórios por nome:

```
$ locate name
```

- Encontre arquivos maiores que 100MB em /home:

```
$ find /home -size +100M
```

SSH

- Conecte-se ao host como seu nome de usuário local:

```
$ ssh host
```

- Conecte-se ao host como usuário:

```
$ ssh user@host
```

- Conecte-se ao host usando uma porta:

```
$ ssh -p port user@host
```

- Exemplo de conexão SSH:

```
$ ssh bandit0@bandit.labs.overthewire.org -p 2220 # senha = bandit0
```

Transferências de Arquivos

- Cópia segura do arquivo.txt para a pasta /tmp no servidor:

```
$ scp file.txt server:/tmp
```

- Copie os arquivos *.html do servidor para a pasta local /tmp:

```
$ scp server:/var/www/*.html /tmp
```

- Copie todos os arquivos e diretórios recursivamente do servidor para a pasta /tmp do sistema atual:

```
$ scp -r server:/var/www /tmp
```

- Sincronizar /home com /backups/home:

```
$ rsync -a /home /backups/
```

Uso de Disco

- Mostra o espaço livre e usado em sistemas de arquivos montados:

```
$ df -h
```

- Exibe tamanhos e tipos de partições de discos:

```
$ fdisk -l
```

- Exibir o uso do disco para todos os arquivos e diretórios em formato legível por humanos:

```
$ du -ah
```

- Exibir o uso total de disco do diretório atual:

```
$ du -sh
```

Projeto Comandos Bash

- Para mais detalhes específicos sobre Linux e Comandos Bash, visite o projeto:

https://github.com/the-akira/Comandos-Bash

Bons Estudos!

“Magic is believing in yourself, if you can do that, you can make anything happen.”

“Knowing is not enough; we must apply. Willing is not enough; we must do.”

Johann Wolfgang von Goethe

Referências

- https://en.wikipedia.org/wiki/Unix
- https://en.wikipedia.org/wiki/Linux
- https://en.wikipedia.org/wiki/Command-line_interface
- https://en.wikipedia.org/wiki/Kernel_(operating_system)
- https://www.guru99.com/file-permissions.html
- https://www.geeksforgeeks.org/introduction-of-process-management/
- https://blog.100tb.com/getting-started-with-linux-foreground-vs-background-processes