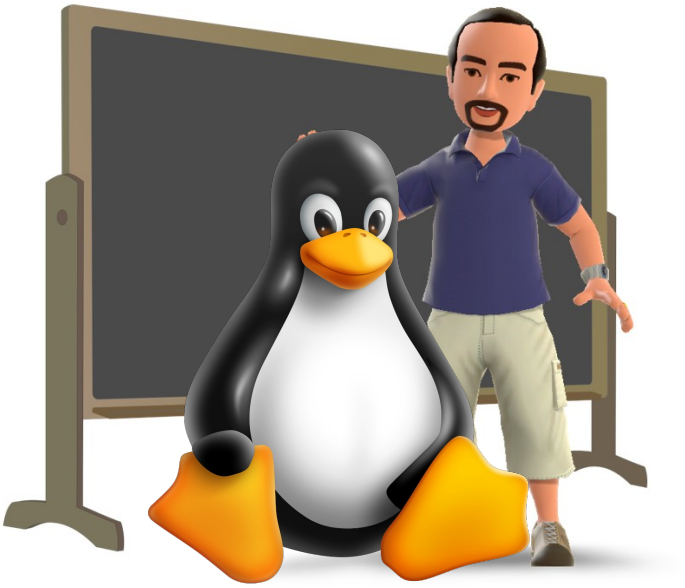


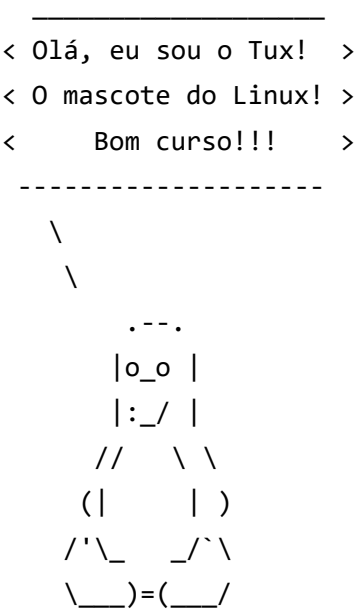
“Nanocurso” de Linux

Versão 4.0.2

Prof. Filippo Valiante Filho

Sumário

Apresentação



Este é um curso de Linux destinado principalmente a estudantes de Engenharia, Computação e Tecnologia, acompanhando disciplinas de Sistemas Operacionais e outras da área de Infraestrutura Computacional. Nada impede que profissionais e outros interessados possam acompanhá-lo, ele apenas tem uma preocupação maior em relacionar a teoria com a prática e apresentar alguns "porquês" do sistema. De toda forma o curso não possui pré-requisitos e avança até pontos mais intermediários, fornecendo uma base sólida para o estudante se aprofundar posteriormente.

O curso está formatado como um material de apoio para sala de aula, tanto deste professor, como de qualquer outro colega que possa achá-lo útil. Mas possui indicações e comentários de forma a viabilizar a autoinstrução. Ao longo do curso há propostas para pesquisa, ou discussão em sala de aula, bem como links úteis. Há material adicional no site e no canal do YouTube.

A primeira versão deste curso era apenas um roteiro de uma ou duas aulas práticas ainda em 2007 e evoluiu para esta versão, amadurecida a partir de 2018, que acompanha como parte prática uma disciplina regular de Sistemas Operacionais ao longo de um semestre.

Bom estudo!

Prof. Filippo Valiante Filho
Maio de 2020 (versão 4.x)

Atenção!

Caso não tenha uma máquina com Linux
para acompanhar o curso, veja primeiro o
Apêndice: Como acompanhar este curso?.

Introdução

O Linux é o sistema operacional mais usado do planeta e presente inclusive nos computadores da estação espacial internacional, dos foguetes das SpaceX e do Rover Curiosity da NASA em Marte. É o único sistema operacional na lista do Top500 que lista os 500 maiores supercomputadores em atividade. É o sistema operacional mais usado nos sistemas embarcados. É a base dos sistemas Android e Chrome-OS. É o sistema mais usado pelos provedores de nuvem. A infraestrutura da Amazon AWS, Google Cloud Platform, IBM Cloud, Oracle Cloud, DigitalOcean e AliBaba são baseadas em Linux, a exceção é a Microsoft Azure, porém a Microsoft é patrocinadora Platinum da Linux Foundation. De fato o único mercado onde o Linux não é dominante é o de sistemas operacionais desktop onde vidraças podem ser observadas com muito mais frequência e também avistam-se algumas macieiras…

Sendo o sistema mais usado, torna-se imprescindível ao profissional de computação, engenharia e tecnologia da informação ter bons conhecimentos sobre esse sistema operacional.

O Linux é um sistema operacional "Unix-like", open source (código aberto), criado pelo finlandês Linus Torvalds em 1991 (a versão 1.0 saiu apenas em 1994). O nome Linux significa algo como o Unix do Linus. Torvalds até hoje coordena o Kernel Linux.

Alguns assuntos para uma conversa inicial em aula, ou pesquisa inicial caso esteja seguindo este material sozinho, antes de passarmos à prática são:

- Mais sobre a história do Linux; Linus Torvalds; motivos para conhecer Linux/Unix; qual o sistema operacional mais usado no mundo?; onde é aplicado; cultura open source (open source não é necessariamente grátis); padrão POSIX; conceito de distribuição...
- Distribuições (distros) gerais normalmente disponíveis em versão desktop e server. As duas principais "famílias", que fornecem uma base segura para iniciar no mundo Linux, são lideradas pelo:
  - **Ubuntu**
    -  www.ubuntu.com.
    - **É a que usaremos neste material!**
    - Os comandos foram (re)testados na versão 20.04 LTS do Ubuntu.
    - "Aparentada" com Debian (a origem), Mint e outras.
  - Fedora
    - "Aparentada" com RedHat (a origem), CentOS, Oracle Linux e OpenSuse (parente mais distante nesse caso).
- Há também uma série de distribuições especializadas:

    - Slax (para instalar em um pendrive), Kali Linux (segurança), OpenWRT (wireless), etc.
- Principais ambientes gráficos (desktops): Gnome, KDE, Xfce, etc.
    - E suas implicações nas distribuições...
- LTS (Long Term Support) e ciclo de desenvolvimento.
- Familiarização com o desktop escolhido.
    - Menu do sistema e barra de tarefas. Aplicações e utilitários que acompanham a distro (monitor do sistema, navegadores, LibreOffice, gerenciador de arquivos, etc.). Como organizar janelas.

Ao conhecer o Linux você automatiamente se sentirá muito à vontade em outros sistemas Unix, ou "Unix-like", como o BSD, FreeBSD, Solaris e até o MacOS.

 Conheça mais sobre as distribuições Linux em: www.distrowatch.com

 Teste distribuições Linux diretamente no navegador: www.distrotest.net.

 Procure vídeos com o título "evolution of desktop" para entender melhor o conceito de sistema desktop.

Terminal – Comandos básicos para familiarização

Para iniciar o terminal, o programa que permite a interação com o sistema operacional através da digitação de comandos, procure terminal no menu, ou use o atalho <ctrl> + <alt> + <t> que funciona na maior parte das distribuições.

O sinal de $ é chamado de prompt e indica que o terminal está pronto para receber um comando.

Nosso padrão neste material será:

- **comando1 comando2 comando opção**
    - Comentários e explicações. <Tecla> a ser pressionada.

Atenção, o Linux é case sensitive! Isto é, diferencia MAIÚSCULAS e minúsculas.

Abra um terminal para testar alguns comandos básicos.

- **date cal**
    - Verificam data e hora e o calendário.
- **hostname**

    - Mostra o nome do host
- **whoami**
    - Mostra o usuário atual.
    - Observe as informações que constam no prompt de comando!
- **w**
    - Usuários logados.
    - Compare com o comando **who**.
    - Muitas vezes há diversas formas de se fazer a mesma coisa, ou se obter a mesma informação!
- **pwd**
    - Exibe o caminho do diretório atual. Você deve estar no diretório do usuário.
- **cd** **cd /** **cd ~** **cd .** **cd ..**
    - Movimentação entre diretórios. Execute cada um deles e use o pwd para testar.
    - ~ atalho pra “home” do usuário
        - Teste **cd** sem nenhuma opção quando estiver fora de “home”.
    - / = root, raiz. O diretório /root é o diretório do usuário root.
    - O que significam "." e ".." na navegação de diretórios?
- **ls**
    - Lista arquivos e diretórios.
    - Verifique a estrutura de diretórios com: **ls /**
    - Cada diretório da estrutura pode ser montado em uma partição ou disco.
    - Cada diretório tem sua função.
        - P.ex. /etc armazena arquivos de configuração, /lib as bibliotecas e /bin aplicações e utilitários.
        - 🔍 Pesquise o papel dos demais diretórios!
    - Teste a opções: **ls -l** **ls – lh** **ls -lha** **ls -lahS**
    - Verifique a mudança nas informações apresentadas a cada opção.
        - Varie com **ls -l -h**, ou **ls -hl** e note que funciona independentemente da ordem das opções fornecidas ou do fato de terem sido agrupadas.
    - O que significa o "." no início do nome do arquivo ou diretório?
        - Dica: em um sistema Unix-like é possível se esconder atrás de um simples ponto!

Mas com tantas opções nos comandos, como saber o que faz cada uma delas? Precisamos de ajuda!

Em tempo, o comando **exit** sai do terminal, ou faz “logoff” conforme o caso.

Obtendo ajuda no terminal

- *comandoqualquer* **--help**
  - Por exemplo: **ls --help**
  - Observe que os argumentos de comando abreviados são usados com "-" e as opções "verborrágicas" (verbose), mais legíveis, mas mais compridas, são usadas com "--".
- **man** *comandoqualquer*
  - Por exemplo: **man ls**
  - Abre a página de manual do comando.
  - Observe a parte de baixo da tela. Você verá como sair do manual.
  - **man man** abre o manual do manual! Observe as seções.
- **man -k** *termo_de_busca*
  - Por exemplo: **man -k usb**
  - Permite pesquisar no manual.
- **help** **info**
- **help** e **info** também são comandos úteis para se obter ajuda.
  - Note que o comando **help** menciona o GNU bash (Bourne-Again Shell), ou simplesmente **bash**, o shell (interpretador de comandos) mais usado no Linux.
  - Siga as dicas apresentadas pelo comando **help**...

Ajuda de linhas de comando bash, powershell, bancos de dados, etc.:

https://ss64.com/

Digite um comando e obtenha uma explicação detalhada:

https://explainshell.com/

Desligando e (re)inicializando a máquina

- **reboot**
  - Reinicializa a máquina.
- **shutdown now**
  - Desliga a máquina imediatamente.
  - Veja a ajuda para saber como desligar a máquina depois de algum tempo.

Ah, sim! Para iniciar a máquina aperte o botão de energia, seja físico, seja virtual!8^P

Dicas práticas de terminal

Há alguns atalhos muito úteis usando as teclas...

- <tab> para autocompletar.
  - A tecla <tab> é sua grande amiga e a digitadora mais rápida do mundo! Ela completa comandos, caminhos, nomes de arquivo, etc.
  - Teste também pressioná-la duas vezes ao completar.
- <q> é usada para sair de determinados comandos.
- <shift> + <pg up/down> para percorrer a tela para cima e para baixo, principalmente quando logado sem a interface gráfica.
- <ctrl> + <l> para limpar a tela mantendo sua digitação.
  - O comando **clear** limpa a tela, mas você tem que digitá-lo…
- <ctrl> + <u> recorta a linha digitada no prompt e <ctrl> + <y> cola de volta.
- <ctrl> + <alt> + <Fn> para alternar entre os terminais.
  - O número do terminal da interface gráfica varia entre as distribuições. 1, 2 ou 7 são os mais comuns. Os demais normalmente são terminais em modo texto.

Além desses atalhos...

- **history**
  - Exibe o histórico dos comandos invocados.
  - 🔍 Use os comandos de ajuda para descobrir como limpar o histórico e como repetir um comando já usado no histórico.
- A \ permite quebrar o comando em várias linhas!

Obtendo informações sobre o sistema (hardware e software)

Estes comandos talvez não sejam os mais usados no dia a dia, mas são muito úteis para verificar as configurações de uma máquina em que se esteja trabalhando pela primeira vez, ou checar algumas dessas configurações e informações conforme a necessidade. Também permitem visualizar um pouco melhor alguns detalhes do sistema operacional e do hardware.

- **uname** **uname -a**
  - Mostra informações sobre o sistema e a versão do kernel.
- **cat /etc/*release**
  - Informações da distribuição.
  - O que é /etc/...?
  - 🔍 Pesquise sobre os curingas "*" e "?".

- **cat /proc/cpuinfo** **lscpu**
  - Informações da CPU.
  - As configurações nos sistemas Linux (e Unix) ficam em arquivos texto. Até os periféricos são acessados como se fossem arquivos...
- **lshw**
  - Lista as configurações gerais de hardware.
  - Você reparou no aviso que o comando emitiu?
  - Logo mais você aprenderá como contornar esse aviso e poderá testar um comando que mostra informações sobre o sistema básico de inicialização do computador por meio do comando **dmidecode -q**
  - E também aprenderá a instalar aplicações, podendo testar o comando **lstopo**
- **lsusb** **lsusb -t** **lsusb -v**
  - Lista informações sobre os dispositivos USB.
- **lspci** **lspci -t** **lspci -v**
  - Lista informações sobre os dispositivos PCI.
- **lslogins** **lslogins -u**
  - Lista os usuários do sistema.
- **last -x** **w**
  - Últimos logins, reinicializações e desligamentos.
- **uptime**
  - Informa há quanto tempo o sistema está ligado.
- **upower -d**
  - Exibe as informações sobre bateria/energia.
- **df -h** **lsblk** **fdisk -l**
  - Exibe informações sobre a memória secundária (dispositivos, partições e sistemas de arquivos).
  - A nomenclatura padrão para os dispositivos de bloco, ou seja, dispositivos de memória secundária, vulgos “discos” mesmo que possam não ser exatamente redondos, é sda, sdb, etc. Com as partições sendo indicadas como sda1, sda2, etc. Em vez de volumes lógicos c:, d:, etc., como seriam conhecidos nas janelas da vida.
  - Embarque na discussão entre o uso das bases 2 (1ki = 2^{10} = 1024 = 1 kibi) e 10 (1k = 10^3 = 1000 = 1 kilo) comparando **df -h** com **df -H** e veja quantos GiB e quantos GB há na memória secundária do sistema.
    - Quer entender melhor? Leia o manual! **man units** esclarecerá.
  - Provavelmente houve um erro de “permissão negada” em um desses comandos. Como resolver?
    - Veja o próximo item.

Há uma série de comandos que, além de fornecer informações sobre o sistema, permitem administrar alguns detalhes do sistema. São comandos que finalizam com

“ctl”, remetendo a controle, e que estão ligados à inicialização do sistema operacional. Alguns deles que vocês pode testar:

- **timedatectl**
- **hostnamectl**
- **systemctl**
  - Teste a opção **systemctl status.**

Há alguns comandos que permitem visualizar um pouco mais dos bastidores do sistema, ou se aproximar um pouco mais do núcleo:

- **lsmod**
  - Mostra quais os módulos do kernel atualmente carregados.
- **dmesg**
  - Mostra mensagens do kernel do sistema. Isso inclui as mensagens que aparecem na inicialização do sistema e as posteriores
- **systemd-analyze**
  - Oferece informações sobre a inicialização do sistema operacional. Na imensa maioria das distribuições Linux o systemd é o processo responsável por isso.
    - Consulte **man systemd**
  - Teste **systemd-analyze plot > carregamento.svg**
    - Abra o arquivo de imagem SVG gerado e analise.
    - Entenderemos esse > mais adiante na seção Redirecionando e conectando processos.

O superusuário (root) e os outros usuários

O usuário “administrador” é um usuário privilegiado, que pode realizar modificações no sistema. Nos sistemas “Unix-like” ele é o usuário “root”, ou superusuário. Em servidores muitas vezes é requerido que haja um usuário root, com senha, que possa efetuar login normalmente. Porém em ambientes desktop é mais seguro não ter uma conta administrativa com senha atribuída, tendo apenas a conta de usuário comum. Mas é preciso um mecanismo para executar comandos como administrador, então pedimos “Super User DO...”.

- **sudo** *comando*
  - Por exemplo: **sudo lshw**
  - A opção **sudo -i** inicia um terminal como root, eliminando a necessidade de digitar sudo antes de cada comando, mas isso tem suas desvantagens e não pode ser usado indiscriminadamente.
    - Note que o prompt mudou para # no lugar do $. Em boa parte das distribuições Linux isso é um indicativo de que você está atuando como usuário root.

    - 🔍 Leia sobre as vantagens e desvantagens do sudo e também sobre como habilitar um usuário a executar comandos com sudo digitando: **man sudo_root**
- Se o sistema estiver com o usuário root configurado com senha pode-se logar como root usando o comando **su**.
- O gerenciamento de usuários envolve a criação, exclusão, troca de senha, atribuição de permissões, etc. É uma tarefa essencial em ambientes corporativos.
    - 🔍 Os comandos **adduser**, **addgroup**, **deluser**, **delgroup**, **passwd**, **usermod**, **id** e **groups** resolverão seus problemas! E os arquivos /etc/passwd e /etc/shadow farão parte disso. Confira em **man sudo_root** como habilitar um novo usuário a executar comandos utilizando sudo.

Gerenciamento de pacotes (software)

Pacotes são a forma de manipular (instalar, remover, atualizar) software (aplicações) no Linux.

No Ubuntu e demais derivados do Debian (Mint, etc.) o comando chave é o apt.

- **apt**
    - Opções: install, remove, purge, update, upgrade, autoremove, autoclean, full-upgrade, show e, por último, mas não menos importante, a opção moo!
    - Teste com uma aplicação dos repositórios!
        - P.ex.: **sudo apt install sl**
            - Depois execute o novo programa!
        - Tente instalar e executar o **lstopo** indicado anteriormente.
    - Teste com um download (Google Chrome, ou Atom, por exemplo)!
        - P.ex.: **sudo apt install ~/Downloads/nomedoarquivo.deb**

Faça uma atualização do sistema. Somente quando você quiser, mas faça com frequência!

- Execute: **apt update** e depois **apt upgrade**
- Complete o serviço liberando espaço e fazendo uma pequena limpeza com **apt autoremove** e **apt autoclean**

Os comandos **dpkg** e **apt-get** podem ser úteis em algumas situações.

No Fedora, RedHat, CentOS e derivados deve-se usar o comando **dnf** (similar ao apt) e, eventualmente, os comandos **rpm** (similar ao dpkg) e **yum** (substituído recentemente pelo **dnf**).

As ferramentas de gerenciamento de pacotes são das diferenças mais marcantes entre as distribuições Linux. Há novos formatos de distribuição de software através de containers de aplicação, que consistem basicamente em se encapsular uma aplicação com todas as suas dependências, de forma que ela possa ser instalada em qualquer distribuição e conviver com aplicações em um mesmo sistema mesmo que tenham dependências conflitantes, já que ficam isoladas umas das outras. Eles também possuem a vantagem de serem "multidistribuição", eliminando a necessidade dos desenvolvedores criarem uma infinidade de pacotes para distribuir seu software. Há principalmente três dessas soluções hoje: Snapcraft (padrão do Ubuntu), Flatpak e AppImage. Muitas distribuições estão migrando dos aplicativos tradicionais para um ou mais desses formatos.

- **snap**
  - Opções: list, install, remove, refresh e info.
    - **snap list**
      - Exibe os pacotes snap instalados
    - **snap install hello-world**
      - Instala o aplicativo hello-world
      - Execute seu novo programa normalmente, digitando hello-world no terminal.

Saiba mais sobre esses formatos em:

https://snapcraft.io - https://flatpak.org - https://appimage.org

Gerenciamento do sistema de arquivos

De volta ao diretório home (~)... Como esta é uma tarefa das mais corriqueiras em um sistema operacional, já vimos diversos comandos relacionados nas primeiras seções. Então recorde os comandos **cd**, **dir**, **pwd** e **ls** vistos na Introdução e que permitem navegar por entre os diretórios e listar seus conteúdos. E também dos comandos **df**, **lsblk** e **fdisk** e considerações sobre a memória secundária em Obtendo informações sobre o sistema (hardware e software).

Agora comandos adicionais para o gerenciamento do sistema de arquivos.

- **mkdir**
  - Cria diretório.
- **rmdir**
  - Remove diretório. Veja adiante como remover um diretório que não esteja vazio.
- **cat** **less** **more** **head** **tail** **tac**
  - Mostram o conteúdo de arquivos.

    - 🔍 Teste e/ou pesquise e compare as diferenças...
- 🔍 Veja o que faz o comando **tail -f**. Para que isso é útil?
- **file**
    - Mostra informações sobre um arquivo (formato).

Os arquivos de configuração ficam em /etc. Aproveite que você já conhece os comandos para visualizá-los! Nós já fizemos isto neste curso...

Outro detalhe que já utilizamos mas não custa ressaltar é o fato de que não precisamos ir ao diretório específico para executar um comando. A partir do diretório atual você pode usar, p.ex.: **ls /etc**, ou **cat /etc/nomedoarquivo**, ou **cd /etc/nomedodiretorio**.

- touch nomedoarquivo
    - Por exemplo: **touch arquivo1**
    - Cria arquivo (vazio).
    - Retome o conceito de usuário root criando um arquivo e depois criando outro arquivo com **sudo touch**. Veja com **ls -l** a diferença.
- **cp**
    - Copia arquivos
- **scp**
    - Faz cópias seguras através da rede.

🔍 Quer sincronizar diretórios? Pesquise o comando **rsync**.

- **mv**
    - Move arquivos.
    - Mover um arquivo dentro do mesmo diretório é renomeá-lo.
- **rm** **rm -r**
    - Remove arquivos e diretórios. O -r remove de forma recursiva, ou seja, remove diretórios com tudo o que tiver dentro.
    - 🔍 Você executaria o comando abaixo!?
        - **sudo rm -rf /**
        - Preste atenção no ícone, melhor pesquisar a resposta antes de testar!!!
- **du -h** **du -hd1**
    - Mostra uma estimativa do espaço ocupado pelos arquivos.
    - Compare com o comando **df -h** que mostra o espaço ocupado em disco pelo sistema de arquivos.

🔍 O comando **wget** permite fazer o download de arquivos, mas mais poderoso ainda é o comando **curl** que além de downloads e uploads pode ser usado para interagir com APIs e é extremamente flexível.

🔍 Pesquise sobre arquivamento e compactação de arquivos com **tar** e os diversos "zips" e "unzips" (**gzip** e **gunzip**, **bzip2** e **bunzip2**, **xz** e **unxz**, **zip** e **unzip**, **7z**).

🔍 Pesquise como realizar a montagem manual de partições com **mount**. Isso pode ser necessário quando você adicionar um dispositivo de bloco (HD, SSD, pen drive, etc.).

🔍 Em um servidor normalmente usa-se um arranjo redundante de discos independentes, o chamado RAID. Ele é um conjunto de unidades de armazenamento (HDs e/ou SSDs) agrupados de modo a aumentar a disponibilidade e o desempenho. O comando **mdadm** possibilita implementar RAID via software no Linux. Mais informações sobre RAID podem ser obtidas no próprio manual com **man md**

Editando arquivos, criando o primeiro script e conhecendo as permissões de arquivos

A maior parte dos arquivos em um sistema Unix e Linux é texto. E para editar o conteúdo desses arquivos é preciso, obviamente, um editor de texto...

- **vi**
  - O editor de texto mais poderoso (e chato!) do mundo.
  - <shift> + <:> ou <ctrl> + <c>
    - :w grava
    - :q sai (q! força a saída)
- **nano**
  - Um editor mais humano!:p E autoexplicativo, basta olhar o rodapé...

Crie um arquivo chamado meuprimeiroscript e digite alguns comandos, um em cada linha. Salve-o. Cheque as permissões desse arquivo com **ls -l**.

As permissões de arquivo seguem o padrão -rwxrwxrwx para arquivo e drwxrwxrwx para diretório. Ou seja, o "d" no início caracteriza um diretório. Também é possível encontrar um "l" aí no começo como referência a um link para outro arquivo ou diretório e algumas outras possibilidades bem específicas.

Cada uma das sequências "rwx" corresponde a usuário proprietário (user/owner), grupo (group) e outros (others/all/world). Às vezes essa sequência é referida como "ugo" (user group others). Já quanto ao "rwx" em si temos que "r" é read, "w" é write e "x" é execute.

Caso se depare com um “s” no lugar do x no bloco de controle do usuário significa que o programa será executado com o usuário do proprietário do arquivo e não com o usuário atual. Esse é um truque útil para executar programas como root, por exemplo.

Usa-se muito a notação decimal (a rigor octal) equivalente ao binário correspondente de cada grupo conforme os parâmetros ligados. Assim 777 é rwxrwxrwx (111 111 111) e 644 é rw-r—r-- (110 100 100).

Voltando ao script, agora é preciso tornar o arquivo executável:

- **chmod +x meuprimeiroscript**

Cheque com **ls -l**. E finalmente pode-se executar o script (sim, tem um ponto no início mesmo!):

- **./meuprimeiroscript**

Brinque um pouco com a atribuição de permissões. Você pode usar as opções +x, -x, +w, -w, +r e -r para mudar as permissões para o usuário atual. Usar chmod o+x para tornar atribuir a um arquivo a permissão de execução para “qualquer um”, o que é muito útil em um servidor por exemplo. Usar chmod g-w para retirar a permissão de escrita (edição e exclusão) para o grupo. E também usar algo como chmod 644 para atribuir de uma vez permissões para owner, group e other.

A rigor o arquivo do script deve começar com #!/bin/bash, para indicar que trata-se de um script a ser interpretado usando o bash (a shell, ou interpretador de comandos padrão da maioria das distribuições Linux), e não outra shell, ou perl, ou python, etc. Por curiosidade o #! é lido "shebang"!:s

O símbolo # sozinho em um arquivo de script ou configuração define um comentário. Teste um comando qualquer no terminal iniciando com #.

Embora não seja obrigatório o uso de extensão no nome de arquivo, nomear um arquivo usando a extensão ".sh" torna mais claro para os usuários que o arquivo se trata de um script.

Digite **help** no terminal e veja o que mais é possível usar em um script. Você encontrará comandos como case, if, for, while, etc. Comandos típicos de uma linguagem de programação. Não é à toa que se diz programação Shell Script. É uma poderosa ferramenta para automação de tarefas em um sistema.

Este professor mantém alguns modelos de script bastante úteis em seu GitHub, confira: https://github.com/filippovf/exemplos_de_scripts.

A propósito basta usar

git clone https://github.com/filippovf/exemplos_de_scripts

E então testar os scripts.

Gerenciamento de processos

Monitorando processos e threads

- **ps**
  - Exibe os processos em execução.
  - Para verificar mais detalhes tente as versões estendidas:
    - **ps au** **ps al** **ps aux** **ps alx**
      - Sim, sem o "-" na opção mesmo nesse caso. O comando ps aceita diversos “estilos” de opções (BSD, UNIX e GNU, com 0, 1 e 2 "-"s).
      - Notou que a opção x mostra todos os processos do sistema?
      - Verifique as informações fornecidas! Há detalhes de identificação, usos de recursos do sistema, estado do processo, prioridade, etc.
  - Consulte a man page do ps para encontrar mais opções e mais detalhes sobre as saídas do comando.
  - Para verificar as threads tente:
    - **ps m** **ps -Lf** **ps -eLf**
      - Na 2ª opção a coluna NLWP corresponde ao número de threads, enquanto LWP corresponde à identificação da thread.
- **pgrep**
  - Permite pesquisar pelo nome de um processo.
    - P. ex.: **pgrep bash** e **pgrep -l bash**
- **pstree**
  - Permite visualizar a hierarquia (árvore) de processos.
- **top** **htop**
  - No **top**, pressione h ou ? para ver as opções.
  - O **htop** é mais fácil de usar. Navegue pelos menus! Você provavelmente precisará instalá-lo se não estiver usando uma distribuição Linux para servidores.

Cada processo possui seu próprio pseudodiretório em /proc. É uma forma de acessar os blocos de controle de processo (PCBs). Verifique o número do PID do processo e acesse /proc/#PID para verificar seu conteúdo.

Sinais e chamadas de sistema

- **kill**
  - Envia um sinal para o processo.

- Se não definir o sinal, é enviado por padrão o sinal 15 (SIGTERM) que termina o processo. Use kill #PID.
  - **xkill** é útil para matar aplicações gráficas no desktop.
    - Use o atalho <Alt> + <F2> para rodar um comando na GUI.
  - Confira a lista de sinais com **kill -l**
  - E as respectivas explicações com **man signal**
    - Talvez você tenha que usar **man 7 signal** para abrir o manual na seção correta.
    - **xman** na GUI (interface gráfica) também é útil.
  - Veja também as ajudas de **killall** e **killall5**.
  - **pkill** permite enviar um sinal usando o nome do processo (de forma análoga a **pgrep**).
- **strace**
  - Rastreia system calls (chamadas de sistema) e sinais.
  - Tente **strace ps**, **strace ls**, etc.

Verifique o comando **nohup**.

Dica: ele pode ser útil para disparar tarefas em um servidor remoto.

Background e foreground

- *processo*&
  - O & após o nome do processo (comando) executa-o em background (segundo plano) com a consequência prática de manter o terminal liberado.
  - P.ex.: **nano&** ou **firefox&**
    - Note a diferença de comportamento entre o processo que executa em background no terminal e o que executa no desktop.
- **bg**
  - Move o processo para background ou lista o último processo em background.
- **fg**
  - Traz processos para foreground.
- <Ctrl> + <z> suspende um processo em primeiro plano enviando-o para estado de espera/bloqueado (parado).
- <Ctrl> + <c> termina (finaliza) o processo.
- **jobs -l**
  - Compare com **ps** e com **ps au**.
    - Abra outro terminal para verificar o resultado nele.
  - Lembre do conceito de batch (processamento em lote)!
  - Você pode usar os números entre [] com fg (fg 1, fg 2, etc.).

Execute um processo em segundo plano, verifique o número do processo (PID) e use kill #PID para terminar o processo.

- Use o comando **ps** e observe que o processo continua!
- Use **fg** para trazer o processo para o primeiro plano. O que acontece agora?
- Lembre que um processo só pode ser efetivamente terminado quando está no estado de execução.

Outro teste interessante para entender a diferença entre os processos em 1º e 2º plano é executar uma listagem recursiva dos arquivos com "**ls -R /**", usar <Ctrl> + <z> para suspender o processo (ele está parado), usar **bg** para fazê-lo executar em background. Neste ponto você provavelmente verá a listagem correr no seu terminal, mas mesmo que pressione <Ctrl> + <z> novamente não conseguirá suspendê-lo. Digite **fg** e tecle <Enter>, então você poderá suspender o processo novamente.

Prioridade

- **renice nice**
  - Permite ajustar indiretamente a prioridade dos processos definindo seu "nível de gentileza" (niceness).

Redirecionando e conectando processos (pipe)

O pipe "|" conecta o canal de saída (stdout) de um processo à entrada (stdin) de outro. Use conforme os exemplos:

- **ps aux | more**
- **ps aux | less**
- **ps aux | head**
- **ps aux | tail**

É possível também redirecionar os canais de saída (>) e entrada (<) de um processo. Repare também no >>.

- **sudo lshw -html**
  - Sim, este comando também sai do padrão com apenas "-" ao invés de "--".
  - Analise a saída. Vamos transformá-la em um arquivo a seguir.
- **sudo lshw -html > hardware.html**
  - Abra o arquivo no navegador.
    - Para abrir a partir da linha de comando digite **firefox hardware.html** (ou outro navegador que tenha disponível).
      - Se estiver trabalhando em uma distribuição servidor pode ativar um servidor web como o Apache ou o NGINX e utilizá-lo para servir o arquivo visualizá-lo no navegador de um computador com acesso a essa máquina.

Execute a sequência a seguir:

- **ls > lista**
- **cat lista**
- **sort -r < lista**
- **sort -r < lista > listainv**
- **cat listainv**
- **cat lista**
- **sort -r < lista >>lista**
- **cat lista**

Para perceber bem a diferença entre > e >>:

- Execute novamente **ls > lista**
- **cat lista**
- **ls >> lista**
- **cat lista**

Além do canal de saída padrão (stdout) temos um segundo canal de saída para o erro padrão (stderror). Normalmente vemos a saída do erro padrão no terminal mesmo, misturada à saída padrão, mas para compreender melhor as diferenças e como manipular ambos os canais de saída faça esses testes:

- Considerando um arquivo xyz inexistente…
- **ls xyz**
- **ls xyz > arquivo**
- **cat arquivo**
  - Isso era o que você esperava?
- **ls xyz > arquivo 2> erro**
- **cat arquivo**
- **cat erro**
- **ls xyz > arquivonovo 2>&1**
- **cat arquivonovo**

O 2 antes do > representa o 2º canal de saída que é o stderror. Já 2>&1 significa concatenar o stderror com o 1º canal de saída que é stdout. Esse é o comportamento padrão observado no terminal.

O diretório /dev contém os arquivos relacionados aos dispositivos. Há um "dispositivo" especial que pode ser acessado através de /dev/null que serve como uma espécie de "buraco negro" do sistema. Qualquer escrita direcionada para esse dispositivo será irremediavelmente descartada. Então tome muito

cuidado com qualquer “> dev/null”, mas ele às vezes pode ser um bom destino para os erros.

E já que ainda estamos cuidando da tubulação, muitas vezes nosso encanamento precisa de um “T”. O comando tee permite redirecionar a saída para a tela e um arquivo ao mesmo tempo. Ele se usa normalmente após um pipe:

- **ls |tee arquivolista**
- **cat arquivolista**
- **ls |tee -a arquivolista**
  - -a é a opção “append”.
- **cat arquivolista**

Teste o tee sozinho, sem o pipe, fornecendo o nome de um arquivo. Você precisará usar <Ctrl> + <d> para encerrar a leitura do terminal. Compare o que você viu na tela do terminal e o conteúdo armazenado no arquivo.

O "&&" permite encadear comandos (processos) na sequência, desde que nenhum deles dê erro. Tem gente que chama isso de “script de uma linha”.

- **echo Meu usuário é && whoami && echo na máquina && hostname**
  - **echo** "ecoa" o texto no terminal. Tente a linha acima usando **echo -n**. As aspas ''‘simples’'' ou “duplas” podem ser úteis em algumas situações.

Já "||" encadeia comandos, mas só realiza o seguinte se o primeiro apresentar erro.

- **apt update || echo “Esqueceu de usar sudo?”**

Pode-se inclusive combinar "&&" e "||"

- **apt update || clear && echo -e "\n\n\tEsqueceu de usar sudo?\n\n"**
- Note que inserimos opções extras no echo.

E já que dobramos caracteres, digitar **!!** repete o último comando, mas você já deveria ter descoberto isso quando pesquisou sobre como repetir comandos do histórico... É muito útil quando esquecemos o sudo e precisamos repetir o comando, basta usar “**sudo !!**”.

Gerenciamento de memória

- **free** **free -ht**
  - Checa a quantidade de memória livre utilizada.

🔍 Verifique também os comandos **lsmem**, **pmap**, **vmstat** e **getconf -a**

 Este professor mantém no GitHub um script que exibe informações e algumas dicas sobre a hierarquia de memória e memória virtual do sistema. Ele está comentado e é um bom exemplo de script avançado. Confira em: https://github.com/filippovf/memory-hierarchy.

Mais manipulações de arquivos e buscas

- **grep**
  - Busca dentro de um arquivo (que pode ser todos os arquivos do sistema, ou diretório) por um texto específico.
  - A flag -i ignora o "case" (considera maiúsculas e minúsculas).
  - A flag -r busca recursivamente (em todos os arquivos do diretório atual ou do caminho indicado).
  - A flag -a faz com que procure dentro de arquivos binários também.
  - A flag -v faz com que mostre o resultado inverso, isto é, excluindo o termo buscado.
  - A flag -n mostra o número da linha.
  - As flags -A, -B e -C mostram um determinado número de linhas depois (After), antes (Before) ou ambas (C por falta de opção).
  - Teste alguns exemplos:
    - **grep -r linux**
    - **grep -r linux /etc**
    - **grep -r Linux /etc**
    - **grep -ri Linux /etc**
    - **grep -ri "Linux kernel" /etc**
    - **grep -ari linux /etc**
  - Note que as aspas servem para buscar uma expressão.
  - O comando grep é extremamente útil quando usado após um pipe!!! Teste!
    - **man grep | grep case**
    - **man grep | grep -n case**
    - **man grep | grep -A 2 case**
    - **man grep | grep -B 2 case**
    - **man grep | grep -C 2 case**
    - **ps aux | grep bash**
    - **ps aux | grep – v root**

- **find**
  - Busca por arquivos.
  - Também pode ser usado para filtrar tamanho, tipo, data, acesso, etc.

- Exemplos (supondo que você tenha diversos arquivos cujos nomes comecem com arq e, portanto, correspondam a arq*)
  - **find . -iname arquivo**
  - **find /home -iname arquivo**
  - **find /home -iname 'arq*'**
  - **find / -iname arquivo**
    - -iname busca no nome do arquivo ignorando "case".
    - O curinga * precisa aparecer entre aspas simples (**'**).
  - Em versões mais antigas o comando **locate** faz a mesma função.
    - **locate nomedoarquivo**

- **whereis** *comando*
  - **whereis** localiza um programa (ou comando), seu código fonte e página de manual.
  - P. ex.: **whereis ls**
- **type** *comando*
  - Exibe informações sobre um comando. Geralmente usa-se a opção -a:
    - **type -a ls**
    - **type -a la**
    - **type -a ll**
    - **type -a l**
  - Caso ainda não tenha testando os ~~comandos~~ apelidos (alias) l, la e ll este é um bom momento. Muitas definições definem aliases de comandos para o comando original (como o ls) e alternativos (como l, la e ll). Leia bem as saídas dos comandos executados para entender melhor.
    - 🔍 É possível criar seus próprios aliases de comando e verificar os existentes com o comando **alias**

- **awk**
  - Em adição ao **grep**, o comando **awk** pode substituir texto dentro dos arquivos.
- **wc**
  - Apresenta a quantidade de linhas, palavras e bytes de um arquivo texto, ou diretório.
  - Após um | pode contar a quantidade de linhas da saída do comando anterior. P.ex.: **ps aux | wc -l**
- **diff**
  - Compara arquivos.
  - P. ex. **diff arquivo1 arquivo2**

Rede

O principal comando para rede é justamente o comando **ip**:

- **ip addr** exibe os endereços (e dispositivos) também ip address ou apenas ip a
- **ip route** mostra a tabela de roteamento
- **ip link** lista as interfaces
- **ip link set DISPOSITIVO down** desativa o dispositivo especificado
  - Use <tab> para mostrar a lista de dispositivos e completar.
- **ip link set DISPOSITIVO up** ativa o dispositivo especificado
  - Use <tab> para mostrar a lista de dispositivos e completar.

É bem provável que suas placas sejam referidas como enpXsY e wlpXsY, sendo en – Ethernet, wl – wlan e ww – wwan; o p corresponde ao barramento e o s ao slot.

Obs.: Os comandos **ifconfig**, **ifup** e **ifdown** não estão disponíveis nas distribuições mais recentes e não convém instalá-los, mas é bom saber de sua existência pois podem constar em alguma distro ou outros Unixes.

- **networkctl** **networkctl status** **networkctl status DISPOSITIVO**
  - Exibe informações sobre os links de rede.

Para obter informações mais avançadas e realizar testes e diagnósticos ao longo das diversas camadas de rede são muito úteis os comandos: **ping**, **mtr**, **tracepath** (e **tracepath -6** para testar com Ipv6), **nslookup**, **ss**, **tcpdump**, **nmap** e **whois**.

O arquivo de configuração de rede nas versões mais recentes do Ubuntu está em /etc/netplan/nomedoarquivo.yaml em outras versões, ou distribuições, ele pode estar em: /etc/network/interfaces. Se estiver em um servidor confira também /etc/cloud.

Antes de editá-lo, faça um backup. Assim como para a maior parte dos arquivos de configuração do sistema, é necessário privilégio administrativo para editar o arquivo. Para configurar DHCP, IP fixo, etc., consulte a documentação, especialmente os exemplos na parte final da man page, usando **man netplan**. Depois de editar o arquivo aplique as configurações com **sudo netplan apply**.

Pesquise o papel dos arquivos /etc/hosts e /etc/resolv.conf.

O kernel Linux incorpora o módulo de firewall chamado netfilter. O comando "padrão" para manipulá-lo é o **iptables**. O Ubuntu inclui o comando **ufw** (uncomplicated firewall) um pouco mais fácil de usar.

Curiosidades e diversão

O comando script pode ser usado para gravar em arquivo uma sessão do terminal.

Obtenha a previsão do tempo no terminal acessando a API do wttr.in com o comando curl: **curl wttr.in**, **curl pt.wttr.in** (para português), ou **curl v2.wttr.in** (nova versão), **curl pt.wttr.in/Avenida+Paulista** e **curl pt.wttr.in/moon**

Faça contas com o comando **bc**. Use **bc -l** para iniciar e digite quit para sair.

Sentindo falta de um gerenciador de arquivos visual, editor de texto e outros utilitários no terminal? Tente o Midnight Commander. Ele é acessado com o comando **mc**.

É possível navegar na Internet em modo texto com o **lynx**.

Também é possível tocar mp3 ou um CD de áudio no terminal!

O comando **youtube-dl** permite fazer o download de vídeos do YouTube e outros sites.

Instale e teste os divertidos comandos **sl**, **fortune**, **cowsay**, **xcowsay** (versão gráfica), **cmatrix** e **figlet**.

Se você estiver se sentindo assertivo teste o comando **yes**, do contrário teste **yes no**

E que tal assistir StarWars no terminal? Tente **telnet towel.blinkenlights.nl**

Há várias formas de acrescentar um toque de Linux ao seu Windows:

http://prof.valiante.info/disciplinas/especial-nanocurso-de-linux/sites-uteis-1

No Android é possível utilizar um terminal Linux com o Termux, mas como toda aplicação no Android, ele não possui uma visão geral do sistema, pois é executado de forma isolada, em seu próprio contêiner. Link:

www.termux.com

E já que é uma seção de curiosidades, este curso chama “nanocurso” por causa de seu tamanho inicial para cerca de 2 aulas de laboratório lá em 2007 e também porque em contraste com outros editores complicados e pouco amigáveis, o editor de textos nano surgia como uma alternativa nova, fácil e voltada ao usuário comum.

Outros – Ou pra não dizer que não falei dos peixes...

Afinal o Tux, mascote do Linux, come peixes. Há muitas outras coisas que poderiam ter sido abordadas aqui, mas você pode continuar a investigar…

- Acesso remoto através do protocolo SSH, usando o comando **ssh**... Mas você já deve ter praticado isso seguindo as atividades propostas!
  - 🔍 Investigue o que faz **ssh -X**.
  - Os programas **screen**, **byobu** e **tmux** permitem trabalhar com múltiplos terminais em um servidor, inclusive sobre conexões ssh, e podem ser bastante úteis em alguns casos.
- Configuração de servidores para ambiente de produção.
  - Você não ficou achando que um simples **sudo apt install nginx** realmente deixaria seu servidor web pronto para um ambiente de produção, ficou?
  - Veja que ele mesmo avisa que "requer configuração adicional" para um ambiente de produção!
- O comando **systemctl** (há o antigo comando service que está sendo subsituído) é bastante útil para gerenciar os serviços ativos no sistema.
- Monitoramento e execução de tarefas periódicas com **watch**, **at**, **cron** e **crontab**.
- Avaliação de desempenho com **vmstat**, **mpstat**, **iostat**, **hdparm**, **iperf**, **time** e **sysbench**.

Caso você tenha seguido este material por conta, é interessante que instale e teste a versão para servidores (Ubuntu Server). Crie uma rede virtual, ou utilize a rede local, para realizar seus testes.

Apêndice A – Como acompanhar este curso?

A melhor forma de acompanhar este curso é ter o Linux Ubuntu instalado em uma máquina virtual, ou mesmo diretamente no computador (bare-metal), o que pode também pode ser feito como dual-boot, isto é, mantendo o Windows, MacOS, ou outro sistema pré-instalado. Para executar confortavelmente a versão padrão do

Ubuntu, seja em uma máquina virtual, seja no computador, é recomendável 2 CPUs e 4 GB de memória principal, além de pelo menos 25 GB de espaço em disco.

Caso não goste muito da interface padrão do Ubuntu, a Gnome, sugiro que tente o Kubuntu, versão com o ambiente KDE. Na verdade é esta opção que uso. Download em www.kubuntu.org.

Caso não possua nenhuma ferramenta de virtualização instalada recomendo o VirtualBox que é open source e multiplataforma. Download em www.virtualbox.org.

Mas e se eu não puder instalar a versão padrão do Ubuntu, ou Kubuntu, o que fazer? Verifique algumas sugestões conforme o caso:

- Caso o computador tenha os pré-requisitos, mas você não possa instalar outro sistema no disco, ou não possa nem sequer instalar uma VM. Por exemplo em um computador da empresa.
  - Você pode inicializar o computador com um pen drive, HD externo, ou mesmo DVD do Ubuntu (selecionar a opção "experimentar", ao invés de instalar, o que lhe permitirá usar o sistema sem modificá-lo.
- Caso o problema seja não ter recursos suficientes no computador. Por exemplo um computador com apenas 4 GB de memória principal.
  - Use um "sabor" (flavour) do Ubuntu mais leve, isto é, com uma interface gráfica mais leve. A mais leve disponível é a Lubuntu, que usa o ambiente de desktop LXQt e pode ser bem executado com apenas 1 GB de memória principal. Download em www.lubuntu.me.
  - Opte por instalar apenas a versão Ubuntu Server. Ela executará bem com 1 GB de memória principal ou menos.
  - Veja também o próximo ponto.
- Caso possa executar uma máquina virtual mas tenha muito pouco espaço. Caso precise de uma máquina virtual rapidamente, sem perder tempo instalando. Caso possa dar boot por um pen drive ou HD, mas precise de um sistema mais leve e rápido.
  - A distribuição Slax foi feita pensando em ser "instalada" em pen drives (e HDs externos). Ela acaba sendo um canivete suíço para salvar computadores cujo sistema operacional "morreu", manutenção, etc. Eu a tenho em todos os meus HDs externos e alguns pen drives. Inicialmente era baseada em uma distribuição chamada Slackware, mas nas versões mais recentes passou a ser baseada na Debian, portanto apresenta a mesma base do Ubuntu. É bastante rápida e se não quiser abrir muitas abas no navegador é perfeitamente funcional mesmo com 512 MB de memória principal. Recomendável para as situações mencionadas e para quando precisar simular uma rede com várias máquinas virtuais. Download em www.slax.org.

 Para saber mais sobre virtualização visite http://prof.valiante.info/disciplinas/hardware/maquinas-virtuais-e-containers