103 Comandos GNU e Unix

103.1 Trabalho na linha de comando

Lição 1

- **pwd** - informa caminho para diretório atual
- **touch** - criar novos arquivos
- **ls** - listar arquivos do diretório atual
- **uname --all** - informações sobre o sistema Linux, versão, distribuição
- **man [comando]** - obter informações sobre o comando
- **apropos [comando]** - explorar os nomes e descrições das páginas man
- **type [comando]** - obtém localização do comando no sistema
- **hash -d** - limpa a tabela de hash
- **which [comando]** - localização absoluta de um comando
- **history** - exibe comandos mais recentes em ordem de execução
- **history | grep texto** - localizar comandos que contém o texto
- **lsblk** - lista os dispositivos de bloco

Lição 2

- **env -** identificar os valores atuais para cada uma das variáveis de ambiente
- **set -** exibe todas as variáveis e funções. | grep texto - filtrar busca
- **echo [texto/$variável] -** exibe na tela mensagem/variável
- **myvar=texto -** cria sua própria variável de ambiente.
- **bash -** cria um novo shell, filho do original
- **exit -** sai do shell e retorna para o pai
- **export [nome variável] -** exporta variável para quaisquer shells filhos
- **unset [nome variável] -** remove a variável de ambiente. (Pode remover também fechando o shell pai ou reiniciando o pc)

103.2 Processar fluxos de texto usando filtros

- **cat**
    - **cat -** lerá a entrada padrão (o que for digitado) e reproduz na saída padrão
    - **cat > mytextfile -** encaminha saída (texto digitado) para o arquivo mytextfile
    - **cat mytextfile -** mostra na saída padrão o conteúdo de mytextfile
    - **cat mytextfile > mynewtextfile -** copia mytextfile para mynewtextfile
    - **cat [arquivo]| grep [texto] -** filtra a busca do texto no arquivo
        - **-i -** ignora maiúsculas e minúsculas
        - **-v -** ignora as linhas que contém o texto
- **echo “texto” >> mynewtextfile -** adiciona o texto no arquivo mynewtextfile
- **diff mynewtextfile mytextfile -** compara os dois arquivos, saída vazia são arquivos iguais
- **grep [texto] [arquivo] -** também filtra o texto no arquivo

Lendo um arquivo compactado:

- bzip, xzcat para arquivos xz e **zcat** para arquivos **gzip** - visualizar o conteúdo de arquivos compactados.
- **gzip/gunzip -** compacta/descompacta pacote gzip
- **bzip2/bunzip2 -** compacta/descompacta pacote bzip2

Visualizando um arquivo no paginador:
- **less -** usado para canalizar a saída, para que os resultados sejam paginados.

Obtendo uma parte de um arquivo de texto:
- **head -** ler as primeiras dez linhas de um arquivo
- **tail -** ler as últimas dez linhas
    - **-n [número] -** exibe a quantidade de linhas informada
- **| nl -** enumera, na direita, cada linha de texto transmitida para o comando
- **| wc -** conta o número de palavras em um documento
    - **| wc -l -** imprime na tela o número de linhas lidas

Sed: editor de fluxo para filtrar e transformar texto
- **sed -n /[string]/p < [arquivo] -** lista linhas que contém a string
- **sed /[string]/d < [arquivo] -** exibe linhas que não contém a string
- **sed s/[termo1]/[termo2]/ < [arquivo] -** substitui termo1 por termo2 (não opera diretamente no arquivo).
    - **-i -** executa uma operação sed diretamente no arquivo original
- **sed -i.backup s/cat/dog/ ftu.txt** - cria arquivo backup e substitui nele

Integridade dos arquivos:
- Uma soma de verificação (checksum) é um valor derivado de um cálculo matemático em relação a um arquivo
- **sha256sum ftu.txt -** calcula o valor da função SHA256
- **sha256sum ftu.txt > sha256.txt -** redireciona para o arquivo sha256.txt
- **sha256sum -c sha256.txt -** verificar integridade

O comando octal dump (od) é usado para depurar aplicativos e diversos tipos de arquivos em formato octal. Para visualizar em formato hexadecimal, usa -x. Útil para ver caracteres que não são visíveis, como o \n, utilizando -c.

103.3 Gerenciamento básico de arquivos

Lição 1

Manipulação de arquivos
- **ls -** lista nomes de arquivos e diretórios >> **ls OPTIONS FILE**
    - **-l -** listagem longa. - arquivo regular. d diretório. c arquivo especial
    - **-h -** tamanho dos arquivos em formato legível
    - **-a -** lista todos os arquivos, inclui os .

Criar, copiar, mover e remover arquivos
- **touch -** criar arquivos novos e vazios >> **touch OPTIONS FILE_NAME(S)**

    - **-a -** altera apenas hora de acesso
    - **-m -** altera apenas hora de modificação
- **cp -** copiar arquivos >> **cp OPTIONS SOURCE DESTINATION**
    - Caminho absoluto possui /. Caminho relativo não possui /
- **mv -** mover e renomear arquivos >> **mv FILENAME DESTINATION_DIRECTORY / mv old_file_name new_file_name**
    - **-i -** exibe mensagem de confirmação e permissão
    - **-f -** será feito sem pedir permissão, à força
- **rm -** excluir arquivos >> **rm OPTIONS FILE**
    - -i - exibe mensagem
    - -f - exclui à força
- **mkdir -** cria diretórios
    - **-p -** cria diretório e subdiretórios, separados por /
- **rmdir -** excluir diretórios
    - **-p -** remove diretório junto com seu sub

Manipulação recursiva de arquivos e diretórios

- **-r -R --recursive -** usado para recursão
    - ls -R - usado para listar o conteúdo de um diretório junto com seus subdiretórios e arquivos
    - cp -r - copiar um diretório junto com todos os seus subdiretórios e arquivos
    - rm -r - remove um diretório e todo o seu conteúdo

Globbing de arquivos e caracteres curinga

- * - zero, uma ou mais ocorrências de qualquer caractere
- ? - uma única ocorrência de qualquer caractere
- [ ] - qualquer ocorrência do(s) caractere(s) inseridos nos colchetes.
- + - representar caractere(s) que aparecem uma ou mais vezes

Lição 2

Como encontrar arquivos

- **find** - pesquisar e localizar arquivos >> **find STARTING_PATH OPTIONS EXPRESSION**

Critérios para acelerar a pesquisa

- -type f - busca por arquivos
- -type d - busca por diretórios
- -type l - busca por links simbólicos
- -name - pesquisa com base no nome fornecido “”
- -iname - com base no nome e ignora maiúsculas e minúsculas
- -not - retorna resultados que não correspondem ao procurado
- -maxdepth N - pesquisa no diretório atual, até o subdiretório de nível N

Por hora de modificação

- -mtime N - baseado no número de dias da modificação

Por tamanho

- -size - exibe arquivos do tamanho de acordo com o argumento passado
    - 100b,+100k, -20M, +2G

Resultado

- -exec - realizar uma ação no conjunto de resultados. {} reserva o espaço para o resultados encontrados por find. Conclui o -exec com \;
- -print - imprime na tela os arquivos que atendem a condição
- -delete - exclui todos os arquivos correspondentes, coloca no final

Arquivos de pacote

- **tar** - "tape archive(r)" - usado para criar, mover, fazer backup, extrair arquivos >>

tar [OPERATION_AND_OPTIONS] [ARCHIVE_NAME] [FILE_NAME(S)]

Operação e opções:

- --create (-c) - cria um novo arquivo tar
- --extract (-x) - extrai pacote ou um ou mais arquivo de um pacote
- --list (-t) - lista os arquivos incluídos no pacote
- --verbose (-v) - mostra arquivos que estão sendo processados por tar
- --file=archive=name (-f archive-name) - especifica nome de arquivo do pacote

Archive_Name: nome do arquivo de pacote
File_Name: Lista com os nomes de arquivos a serem extraídos. Não presente, todo pacote é extraído.

Comandos:

- tar -cvf archive.tar stuff - salva diretório stuff no arquivo archive.tar
- tar -xvf archive.tar - extrai o pacote
- tar -xvf archive.tar -C /tmp - extrai o pacote para o diretório específico **-C**
- tar -czvf name-of-archive.tar.gz stuff - cria o arquivo e compacta com gzip (**-z**) ou (**-j**) cria arquivo bzip2

O comando cpio "copy in, copy out" é usado para processar arquivos de pacotes, podendo copiar arquivos para um pacote e extrair arquivo de um pacote.

- ls | cpio -o > archive.cpio - cria um arquivo cpio
- cpio -id < archive.cpio - extrair o arquivo de pacote. -i - extração; -d cria pasta de destino

O comando **dd** copia dados de um local para outro, com a sintaxe option=value

- dd if=oldfile of=newfile
- status=progress - exibe o andamento do trabalho no console

103.4 Usando streams, pipes e redirecionamentos

Lição 1

Os processos padrão do Linux têm três canais de comunicação abertos: o canal de entrada padrão (stdin, por exemplo, um teclado), o canal de saída padrão (stdout, por exemplo, um monitor) e o canal de erro padrão (chamado de stderr). Os descritores numéricos atribuídos a eles são 0,1,2, respectivamente, e estão localizados em /dev/.

O redirecionamento de uma saída padrão de um processo para um arquivo pode ser feito com o símbolo >, por exemplo:

- **cat /proc/cpuinfo >/tmp/cpu.txt -** equivalente a 1>
- **cat /proc/cpu_info 2>/tmp/error.txt -** equivalente 2>

O redirecionamento entre canais, usando o &, como por exemplo, 1>&2, redireciona stdout para stderr, e 2>&1 redireciona stderr para stdout. Para gravar stderr e stdout em um mesmo arquivo, pode utilizar >log.txt 2>&1, permitindo a análise de mensagens de depuração e erro. Os arquivos são substituídos pelos redirecionamentos de saída, a menos que a opção **noclobber** esteja habilitada no Bash. Usa-se **set -/+o noclobber** ou **set -/+C** para habilitar/desabilitar. Quando usa >>, mesmo com a opção noclobber habilitada, anexa dados redirecionados ao conteúdo existente. O símbolo de menor que < é usado para redirecionar o conteúdo de um arquivo para o stdin de um processo. Os métodos Here Document (<<) e Here String (<<<) também pode ser usados, de forma que o primeiro é indicado por <<, por exemplo: wc -c <<EOF, na qual poderá redirecionar várias linhas, até que a string EOF seja lida; e o segundo é indicado por <<<, por exemplo, wc -c <<<"How many characters in this Here string?", na qual direcionará somente o que está entre aspas.

Lição 2

Pipes

O pipe, representado pela barra vertical |, fazem com que os dados fluam da esquerda para a direita na linha de comando, possibilitando conectar a saída de um programa diretamente a entrada de outro programa. Podemos usar vários pipes ao mesmo tempo, por exemplo:

- cat /proc/cpuinfo | grep 'model name' | uniq

No exemplo, o conteúdo do arquivo é canalizado para grep e logo em seguida para uniq. Os pipes podem ser combinados com redirecionamentos na mesma linha de comando:

- grep 'model name' </proc/cpuinfo | uniq

Pode redirecionar uma saída para um arquivo e ainda vê-la na tela com **tee**:

- grep model\ name </proc/cpuinfo | uniq | tee cpu_model.txt
- grep model\ name </proc/cpuinfo | uniq >cpu_model.txt

Para capturar a saída de um comando, possibilitando inspeções futuras, também podemos usar:

- comando 2>&1 | tee log.txt

Isso faz com que o stderr e mensagens do comando seja direcionado ao stdout, de forma que o tee consiga capturar isso com um pipe, salvando-o no arquivo.

Substituição de comando

Outro método para capturar a saída de um comando. Quando colocado entre crases ou com $(), o bash o substitui por sua saída padrão. Ex:

- mkdir `date +%Y-%m-%d` - cria pasta com nome referente a data2

Esse método pode ser utilizado para armazenar a saída de um comando como uma variável.

- Hoje=`date +%Y-%m-%d`

Existe também um outro método para usar a saída de um programa como argumento de outro programa que emprega um intermediário chamado xargs. Ele usa o conteúdo que recebe via stdin para executar um determinado comando com o conteúdo como argumento

103.5 Criar, monitorar e eliminar processos

Lição 1

Jobs (trabalhos) são processos iniciados de forma interativa através de um terminal, enviados para o segundo plano e ainda não finalizados.
Comandos:
jobs - Opções: -l adiciona PID, **-p** lista somente PID, **-n** processos que mudaram de status, **-r** processos em running, **-s** processos stopped. **%n** especificar PID ou nome, **%% %+** job atual, **%-** job anterior
bg - job para background
fg - job para foreground
& - inicia em segundo plano
kill %n/PID - encerra job, sinal SIGTERM - sinal de encerramento (para usar com o nome: pkill)
kill -1 $(pgrep job name) - pela substituição de comando

Jobs pertencem somente aquela sessão do shell do usuário, para desvincular jobs de sessões e tê-lo em outras, usar o nohup: **nohup COMMAND &.**
Pode verificar a stdout, na saída nohup.out com **tail -f**

Monitoramento de processos podem ser feitos com **watch**. **uptime** exibe informações do pc em geral, **-n number** pode ser usado para alterar tempo de verificação de uptime. **free** exibe informações sobre uso da memória.

Pode verificar o PID de um processo com: pgrep job name // pidof job name
Pode encerrar um job com: pkill job name
Encerrar várias instâncias de um job: killall job name

Monitoramento de processos com job (resultados dinâmicos) e ps (resultados estáticos)

top: M - classifica por memória. N - número ID do processo. T - por tempo de exec. P - por porcentagem de uso CPU. R - ordem cres/decres
k - elimina processo. r - prioridade renice. u- lista jobs de um usuário. c- mostra caminho absoluto e diferença de processos kernel [] e de usuários. V- visão hierárquica/floresta. t e m- muda aparência da CPU e memória. W- salva definições de configuração em ~/.toprc.

A saída top é dividida em área de resumo e área de tarefas.
---
ps - Para ver todos os processos, usar ps a. As opções em ps seguem três padrões distintos:
BSD- opções usadas sem -
UNIX- opções usadas com -
GNU- opções usados com --
Ver saída semelhante a top é ps aux: a- processos tty. u- orientado ao usuário. x- não vinculados a tty.

Lição 2

Multiplexador (ou mux) é um dispositivo que permite que várias entradas sejam conectadas a uma única saída. Assim, um multiplexador de terminal proporciona a capacidade de alternar entre diferentes entradas conforme necessário. Exemplo de multiplexadores são o screen e o tmux.
screen: prefixo de comando é ctrl + a
Janelas:
screen -t nome - criar janela com nome desde início
w - see all windows
“ - ver lista de janelas
n/p - mover entre janelas next/previous
number - mudar para a janela correspondente
c - create a new window
A - rename a window
k - remover a janela atual
As janelas executam seus programas de forma totalmente independente umas das outras.

Regiões:
s / | - dividir janela horizontalmente/verticalmente
tab - passar para nova região
Q - encerrar região menos atual
X - encerrar região atual

Sessões
screen -list ou -ls - ver sessões
screen -S “name” - criar sessão
screen -S name/PID -X quit - encerrar sessão
d - desanexar sessão
screen -r name/PID - entrar na sessão desanexada

Modo de rolagem
[- entrar no modo de rolagem
espaço - começar a selecionar texto.
espaço- terminar de selecionar
] - colar texto

A personalização de screen pode ser feita em ~/etc/screen.rc: definir configurações gerais: defscrollback 1024; bindar uma tecla: bind l kill; terminal settings e startup screens: screen -t top top
---
tmux: prefixo de comando: ctrl + b
Possui modelo cliente-servidor, seleção interativa de sessões, janelas e clientes via menus, mesma janela anexada a várias sessões

tmux new -s “LPI” -n “Window zero” - criar janela Window zero e sessão LPI
Janelas:
c - criar nova janela
, - renomear janela
w - ver todas janelas

n/p - mover entre janelas next/previous
number - ir para janela number
& - eliminar uma janela
f - encontrar janela pelo nome
. - alter número do índice da janela
Painéis: São pseudoterminais completos anexados a janela
“ / %- dividir janela horizontalmente/verticalmente
x - destruir/encerrar um painel
setas - mover entre painéis
; - passar para o último painel ativo
ctrl/alt + seta - redimensionar painel em uma/cinco linhas
{ / } - trocar de painel para anterior/seguinte
z - aproximar/afastar painel
t - exibir relógio
! - transforma painel em janela

Sessões
s / tmux ls - listar sessões
:new - criar sessão
$ - renomear sessão
s - trocar de sessão
tmux kill-session -t name - eliminar uma sessão
tmux attach -t name - anexar sessão. Pode usar somente a ou at
d - desanexar sessão

Modo de rolagem
[- entrar no modo de rolagem
ctrl + espaço - marcar início da seleção
alt + w - copiar texto selecionado
] - colar texto

A personalização do tmux pode ser feita em /etc/tmux.cong ou ~/.tmux.conf. Além disso, pode-se fornecer um arquivo de configuração alternativo para ser lido com a opção -f. Ex: alterar tecla de prefixo, definir atalhos.

103.6 Modificar prioridades de execução de processos

Os sistemas operacionais são chamados de sistemas multitarefa ou multiprocessamento quando capazes de executar mais de um processo ao mesmo tempo. Mesmo acontecendo só quando há mais de um, os de processador único podem imitar isso alternando entre processos rapidamente.

Então, para ter aproveitamento máximo, esses sistemas mantêm uma fila dinâmica de processos ativos aguardando um slot de tempo da CPU. Assim, no Linux, que é um sistema operacional de multiprocessamento preventivo, há o agendador que organiza a fila de processos, de acordo com seus valores de prioridade.

Há duas políticas de programação: políticas em tempo real (e acordo com o valor de prioridade) e política normais (menos prioridade)

Os processos normais geralmente têm o mesmo valor de prioridade (120), mas as políticas normais podem definir regras de prioridade de execução usando outro predicado do processo: o valor nice. Mais baixo = mais prioridade

PRI para processos em tempo real - 0 a 99
PRI para processos em tempo real - 100 a 139

Pode encontrar a PRI de um processo ativo no arquivo /proc/PID/sched. Também pode-se utilizar o ps e o top

ps -Al ou ps -el - Coluna PRI, -40 a 99 por padrão, necessário somar 40 o valor.
top - prioridade em tempo real são negativas (rt), porque subtrai 100 do valor normal. prioridades normais vão de 0 a 39.

Todo processo começa com valor nice padrão de 0. Esse valor pode ir de -20 (PRI alta) a 19 (PRI baixa). Apenas root pode diminuir para menos de zero.

Modificar prioridade dos processos:

nice -n PRI comando - iniciar processo com valor nice PRI
renice PRI -p PID - alterar prioridade de processos em execução
Em top, pressionando r.

103.7 Pesquisar em arquivos de texto usando expressões regulares

Os algoritmos de pesquisa de string são muito usados por diversas tarefas de processamento de dados, e os sistemas Unix possuem sua própria implementação: as expressões regulares (ERs). Consistem em sequências de caracteres constituindo um padrão genérico usado para localizar e modificar uma sequência correspondente em uma sequência maior de caracteres.
. (ponto) - corresponde a qualquer caractere.

^ (acento circunflexo) - corresponde ao início de uma linha.

$ (cifrão) - corresponde ao fim de uma linha.

Classes de caracteres tradicionais:

[:alnum:] - caractere alfanumérico.

[:alpha:] - caractere alfabético.

[:ascii:] - caractere que pertence ao ASCII.

[:blank:] - caractere em branco, um espaço ou tabulação.

[:cntrl:] - caractere de controle.

[:digit:] - dígito (de 0 a 9).

[:graph:] - qualquer caractere imprimível, exceto espaço.

[:lower:] - caractere em minúsculas.

[:print:] - qualquer caractere imprimível, incluindo espaço.

[:punct:] - qualquer caractere imprimível que não seja um espaço ou um caractere alfanumérico.

[:space:] -caracteres de espaço em branco: espaço, alimentação de formulário (\f), nova linha (\n), retorno de carro (\r), tabulação horizontal (\t) e tabulação vertical (\v).

[:upper:] - letra maiúscula.

[:xdigit:] - dígitos hexadecimais (de 0 a F).

Quantificadores:
* - o átomo ocorre zero ou mais vezes. É literal quando precedido por \ ou se aparecer no início da expressão regular. {0,}
+ - seleciona as peças que contêm uma ou mais correspondências de átomos em sequência {1,}
? - o átomo correspondente aparece uma vez ou se não aparece {0,1}

Chaves - são quantificadores que permitem ao usuário especificar limites precisos de quantidade para um átomo.
{ i } - átomo aparece i vezes. Ex: [[:blank:]]{2}
{ i, } - átomo aparece pelo menos i vezes. Ex: [[:blank:]]{2,}
{ i , j } - átomo aparece no mínimo i e máximo j vezes.

Alternância e agrupamentos: a barra vertical | funciona como uma alternância, ou.

Em expressões básicas, +?(){}, necessitam ser precedidas por \ para serem interpretadas como tal em expressões estendidas.

Expressões regulares estendidas não são suportadas por padrão, mas find permite que sejam habilitadas com -regextype posix-extended or -regextype egrep:
find $HOME -regex
find /usr/share/fonts -regextype posix-extended -iregex

Lição 2

Usar as expressões regulares com grep (localizador de padrões) e sed (editor de fluxo)

Grep
Um dos usos mais comuns do grep é facilitar a inspeção de arquivos longos, usando a expressão regular como filtro aplicado a cada linha e usando também pipe redirecionando algum comando para grep.

Parâmetros importantes para grep:
-c / --count - exibe quantidade de linhas que ocorre a correspondência
-i / --ignore-case- ignora maiúsculas e minúsculas
-f FILE / --file=FILE - indica um arquivo contendo a expressão regular a ser usada
-n / --number - mostra o nº da linha
-v / --invert-match - mostra todas as linhas, exceto as que possui a correspondência
-H / --with-filename - mostra o nome do arquivo que contém a linha (usar número para exibir mais linhas acima ou abaixo)
-C number / --context=number - exibir mais linhas antes ou depois
-z / --null-data - passa a encarar a entrada ou saída como uma sequência de linhas

Programas complementares:
egrep - equivale a grep -E. Permite usar recursos de expressões regulares estendidas, como as |.
fgrep - equivalente a grep -F. Não analisa expressões regulares, interpretando caracteres especiais literalmente

Sed
Possui como objetivo modificar dados baseados em texto de forma não interativa
Exemplos de uso:
factor `seq 12` | sed "1,7d;11d"
factor `seq 12` | sed "1d;/:.*2.*/d" - (d remove)
factor `seq 12` | sed "1d;/:.*2.*/c CHANGED" - (c substitui)
/a TEXT - copia texto indicado por TEXT para uma nova linha após linha com a correspondência
/r FILE - faz o mesmo, mas copia o conteúdo do arquivo indicado por FILE
/w - faz oposto de r, linha será anexada ao arquivo indicado
sed s/old/new/ - substitui a primeira correspondência de old por new
sed s/old/new/g - substitui old por new em todas correspondências.

Comando last registra todos os logins e logouts . /var/log/wtmp
Comando lastb registra as tentativas de login falhadas /var/log/btmp