Linux estudos

apt é a ferramenta lá do Debian de gerenciamento de pacote, quando vamos baixar algo a gente baixa isso de um repositório, quando damos esse comando ele vai no repositório e faz o que pedimos

Sudo é para executarmos algo que precisa de direitos de administrador.

Comandos:

pwd: Identifica aonde estamos.
ls: Lista o conteúdo de onde estamos.
ls -a: Lista o conteúdo de onde estamos inclusive os arquivos ocultos.
ls -l: Lista o conteúdo de forma longa.
ls -al: Lista o conteúdo de onde estamos inclusive os arquivos ocultos só que em long list, ele da um detalhamento completo inclusive do diretório pai mostra os arquivos.
ll: É um atalho para executar o ls -al.
[comando] --help: Mostra os parâmetros possíveis para dar ao comando e o que eles fazem.
man [comando]: Abre o manual sobre o determinado comando.
cd (change dear): Faz a navegação entre os diretórios, ele sozinho sem especificação cai na home área do usuário, no caso /home/rubens
cd -: Volta para o último diretório que estávamos.
mkdir [nome]: Cria um diretório com o nome especificado.
mkdir -p [nome]: Cria o(s) diretório(s) passados no caminho se não existirem ou navega entre eles se existirem até o que não existe para cria-lo.
touch [nome]: Cria um arquivo zerado com o nome especificado. (se colocarmos touch .nome criamos um arquivo oculto por causa do ponto.)
rmdir [nome]: Remove diretório.
rm: Remove o que escolhermos.
rm -r [nome]: Remove com recursividade os diretórios e conteúdos (vai remover da pasta indicada em diante).
rm -rf [nome]: Remove com recursividade os diretórios e conteúdos (vai remover da pasta indicada em diante), porém de maneira forçada

- ```
  ricardo@ubuntu-server:~/labs/arqs_dirs$ mkdir diretorio\ 2 diretorio\ 3
  ricardo@ubuntu-server:~/labs/arqs_dirs$ ls
  1  2  Dir1  diretorio  'diretorio 1'  'diretorio 2'  'diretorio 3'
  ```

  Para criarmos algo com espaço colocamos a barra invertida, pois ela vai fazer com que o espaço não seja interpretado como um espaço e sim como um texto (string).
- ```
  ricardo@ubuntu-server:~/labs/copy_move$ cp -r dir1/* dir2
  ```

  Para copiarmos todos os arquivos de um diretório para outro usamos o comando acima.

  mv: move ou renomeia os arquivos ou diretório em forma de recortar e colar, se o diretório que quer receber os arquivos não existir o diretório anterior que tem os arquivos vai ser renomeado para o que irá receber.

  *: Pega tudo inclusive o nada.

  ?: Tem que vir algo no lugar dele.
- ```
  ls arq[1-5]
  ```

  O comando acima pega os arquivos arq1, arq2, arq3, arq4, arq5

  cat: mostra o conteúdo do arquivo desde que seja um arquivo de texto, se for binário ele mostra porém a gente acaba não compreendendo.

  grep: procura algo especifico nos arquivos.

  grep -i: procura algo especifico, porém sem fazer diferenciação entre letras maiúsculas ou minúsculas.
- ```
  rubensjvm@ubuntu-server:~/labs/filtrando_conteudo$ grep http services
  # Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml .
  http            80/tcp          www             # WorldWideWeb HTTP
  https           443/tcp                         # http protocol over TLS/SSL
  https           443/udp                         # HTTP/3
  http-alt        8080/tcp        webcache        # WWW caching service
  rubensjvm@ubuntu-server:~/labs/filtrando_conteudo$ grep -i http services
  # Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml .
  http            80/tcp          www             # WorldWideWeb HTTP
  https           443/tcp                         # http protocol over TLS/SSL
  https           443/udp                         # HTTP/3
  http-alt        8080/tcp        webcache        # WWW caching service
  hkp             11371/tcp                       # OpenPGP HTTP Keyserver
  ```

  grep -l: Informa em quais arquivos temos o que estamos especificando.

  grep -L: Informa em quais arquivos **não** temos o que estamos especificando.

  more: Lista o que tem no arquivo no terminal em forma de paginação, usa espaço para ir para a próxima página e enter de linha a linha

  less: Lista o que tem no arquivo no terminal em forma de paginação mas usa os atalhos normais para navegação.

  head: Mostra as 10 primeiras linhas do arquivo.

  head -n [n°]: Mostra as primeiras N linhas do arquivo de acordo com a quantidade.

  tail: Mostra as últimas 10 linhas do arquivo.

  tail -n [n°]: Mostra as últimas N linhas do arquivo de acordo com a quantidade.

find (diretório de início) -name [nome do arquivo para procurar]: Busca por um arquivo do diretório especificado em diante. Se colocarmos o i na frente do name ele ignora se é maiúsculo ou minúscula.
find (diretório de início) -maxdepth [n°] -name [nome do arquivo para procurar]: Busca por um arquivo do diretório especificado em diante até o limite de pastas para entrar, exemplo: se procurarmos um arquivo com o -maxdepth 3 ele vai procurar do caminho especificado e dois diretórios dentro dele.

- `find . -amin -5`

Com esse comando acima iremos encontrar o que foi modificado nos 5 minutos anteriores da pasta em questão em diante, por dias seria -atime, por tamanho usaríamos -asize.
grep 3389 services **>>** listagem.txt : Está redirecionando a linha que contém 3389 do arquivo services e criando se não existir e escrevendo em listagem.txt (usamos o >> ao invés de > para não criar o arquivo toda vez que executarmos isso e já tiver o arquivo com algo escrito não perdermos.
Pipe (*|*): A famosa barra em pé é usada para redirecionar o primeiro comando antecessor a ela e passar para o 2°, assim executando os 2 ao mesmo tempo com a condição do 2°.

-
```
└─ cat /etc/passwd | grep rubensjvm
rubensjvm:x:1000:1000:,,,:/home/rubensjvm:/usr/bin/zsh
```

Como estamos aprendendo já a utilizar o pipe e concatenar mais de um comando para a execução de um comando mais funcional e com uma só etapa ao invés de executar comando por comando para obter o mesmo resultado vamos explicar o comando abaixo

- `tail -n 5 syslog | grep systemd > ~/labs/redirecionamento/log5.txt`

O que vem antes do Pipe (|) serve de input para o que vem após, então o **tail -n 5 syslog** está pegando as últimas 5 linhas do arquivosyslog e concatena com o**grep systemd** que está procurando no arquivo aonde tem as aparições da palavrasystemd e o **>** está criando um arquivo no diretório**~/labs/redirecionamento**com o nome delog5.txt.
wc: Faz a contagem para nós de quantas vezes em linhas, palavras e bytes alguma ocorrência aparece em determinado arquivo.
wc -l: faz o comando acima, porém informa somente a quantidade de linhas.
cut -d "O que queremos usar como delimitador" -f[n° de delimitadores que iremos usar em diante]: o cut mostra algo especifico que queremos a partir de um delimitador e mostra na nossa tela.

- `cat logs | cut -d " " -f6-`

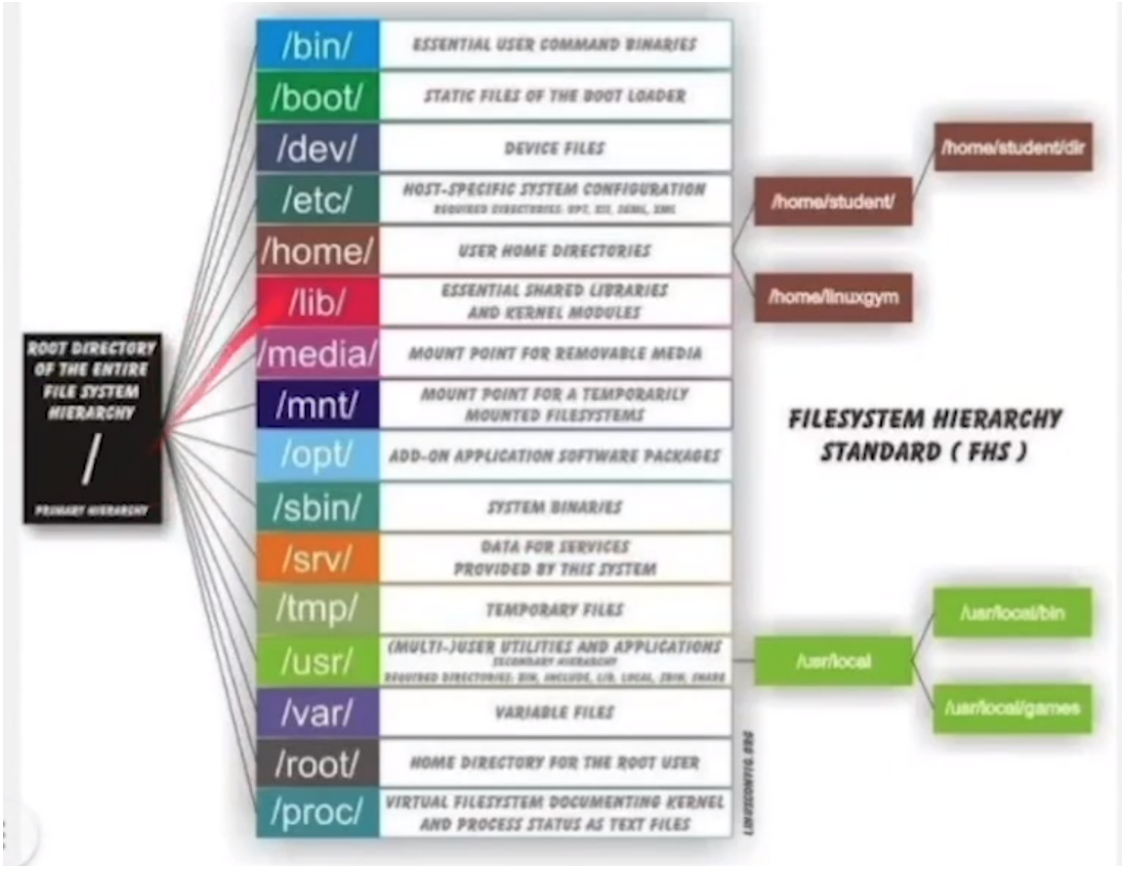
Nesse comando estamos mostrando no terminal o que tem no logs a partir do espaço 6 até o final do arquivo de todas as linahs.

- `cat logs | cut -d " " -f1-3,6-`

Já aqui estamos fazendo a mesma coisa porém do espaço 1 até o 3 e do 6 até o fim.

Locais:

/etc: local onde fica as configs do nosso sistema

- 

Precisamos voltar para a barra para ir a outro local a barra é como o começo da "arvore" a semente das raízes que estão todas ligadas por ela.

/etc/services: mostra sobre as portas que temos especificadas no sistema.

/etc/passwd: É a base de dados local dos usuários, quando fazemos o login/autenticação local usamos ele.

Regex:

Expressões regulares ou Regular expressions.

Em uma busca as vezes queremos restringir algo para ter um resultado específico e para facilitar de nós encontrarmos esse resultado usamos **Expressões regulares** que servem para indicarmos exatamente o que queremos receber.

Sem expressão regular nós podemos obter vários resultados indesejados, como abaixo:

```
ricardo@ubuntu-server:~/labs/expressoes_regulares$ cat american-english | grep computer
computer
computer's
computerization
computerization's
computerize
computerized
computerizes
computerizing
computers
microcomputer
microcomputer's
microcomputers
minicomputer
minicomputer's
minicomputers
supercomputer
supercomputer's
supercomputers
```

Já com uma expressão regular podemos eliminar as palavras que não começam com a palavra **computer**:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ cat american-english | grep -E "^computer"
computer
computerization
computerization's
computerize
computerized
computerizes
computerizing
computer's
computers
```

A expressão está dentro das aspas ("") e o sinal circunflexo (^) indica começo de linha.

Já para saber quais palavras terminam com **computer** usamos o sinal de dólar ($):

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ cat american-english | grep -E "computer$"
computer
microcomputer
minicomputer
supercomputer
```

E agora para chegarmos no resultado de começar e terminar com **computer** mesclamos as 2 expressões:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ cat american-english | grep -E "^computer$"
computer
```

Dessa maneira pegamos tanto maiúsculo como minúsculo:

```
ricardo@ubuntu-server:~/labs/expressoes_regulares$ cat american-english | grep -iE "^computer$"
computer
Computer
COMPUTER
```

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ cat american-english | grep -E "^computer$"
computer
```

A mesma coisa do comando acima, porém reduzido por causa que o egrep é a abreviação de grep -E:

```
ricardo@ubuntu-server:~/labs/expressoes_regulares$ egrep "^computer$" american-english
computer
```

Para pegar ou uma palavra ou outra nesse mesmo comando fica dessa maneira:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ cat american-english | grep -iE "^computer$|^smartphone$"
computer
smartphone
Computer
```

Sinal de (|) faz com que tenhamos o famoso OU.

Para pegar qualquer inicio de palavra por exemplo usamos o egrep com um ponto (.) para pegar qualquer caractere, ao invés de (?) como costumamos usar, comando de exemplo:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ egrep "^.oot" american-english
Booth
Booth's
Hooters
Hooters's
Nootka
Nootka's
Root
```

Podemos usar ele também no final:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ egrep "^.oot..$" american-english
Nootka
Root's
Wooten
booted
bootee
booths
```

Para pegar palavras que começam com letras específicas podemos usar o colchetes ([]) para delimitar, exemplo:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ egrep "^[fgil]oot..$" american-english
footed
foot's
looted
looter
loot's
```

Ou podemos também delimitar um range com o sinal de traço (-), exemplo:

```
rubensjvm@ItsJDK:~/labs/expressoes_regulares$ egrep "^[a-c]oot..$" american-english
booted
bootee
booths
bootie
boot's
cootie
coot's
```