Terminal Unix

Comandos básicos

1

Comandos básicos

- **man** <nome comando>: apresenta o help associado ao comando
- **ls** : lista todos os ficheiros e diretorias presentes na diretoria atual
  - ls –la

```
-rwxrw-r-- 1 root root 2048 Jan 13 07:11 afile.exe
```

- file permissions
- number of links,
- owner name,
- owner group,
- file size,
- time of last modification, and
- file/directory name

- **pwd**: imprime a localização da diretoria atual
- **mkdir** <nome dir>: cria diretoria
- **cd** <path>: muda de directoria de acordo com a path

Comandos básicos

- **rm** <file name>: remove ficheiro
- **rm –r** <diretoria>: remove todos os ficheiros e sub-diretorias presents na diretoria especificada (cuidado com o uso da opção -f !!)
- **cat** <file name>: imprime conteúdo de um ficheiro
- **head/tail –n** <file name>: apresenta as primeiras/últimas *n* linhas do ficheiro
- **mv** <origem> <destino>: move ficheiro de origem para o destino
- **cp** <origem> <destino>: copia o ficheiro de origem para o destino

Comandos básicos

- **find** : localizar arquivos.
  - **find . -name *.txt –print** : para pesquisa de arquivos de texto na ditetoria atual
- **grep**: Procura de um padrão, por exemplo um **cat a.txt | grep ola** irá mostrar-nos apenas as linhas do arquivo a.txt que contenham a palavra “ola”
- **gzip**: Comprime ou expande arquivo
- **tar:** Cria ou extrai arquivos, muito usado como programa de backup ou compressão de arquivos

Redirecionamento de input/output

- Redirecionamento é normalmente implementado por meio da colocação dos caracteres **<** e **>**
  - **comando > arquivo1.txt** : o outuput da execução do comando é guardado no ficheiro aquivo1.txt. Se o ficheiro já existir será substituido.
  - **comando < arquivo2.txt**: o comando é executado tendo como input o conteúdo do ficheiro arquivo2.txt.

Uso de pipes

- Permite o encadeamento de comandos, onde o output de um é usado como input do seguinte

  cat nomes.txt | sort

  ordena as linhas do ficheiro alfabeticamente

  cat file.txt | more

  permite a paginação do conteúdo do ficheiro

vi – editor de texto

- criado por *Bill Joy* em 1976. Em 1991 foi lançado o editor vim, uma derivação melhorada do vi
- Comandos básicos:
  - **:wq** - Salva o arquivo e sai do editor
  - **:w nome_do_arquivo** - Salva o arquivo corrente com o nome especificado
  - **:q** - Sai do editor
  - **:q!** - Sai do editor sem salvar as alterações realizadas

vi – editor de texto

- Comandos de inserção:
  - **i** - Insere texto antes do cursor
  - **a** - Insere texto depois do cursor
  - **r** - Insere texto no início da linha onde se encontra o cursor
  - **A** - Insere texto no final da linha onde se encontra o cursor
  - **o** - Adiciona linha abaixo da linha atual
  - **O** - Adiciona linha acima da linha atual
- Comandos de procura:
  - **/palavra** - Procura pela palavra ou caracter em todo o texto
  - **?palavra** - Move o cursor para a ocorrência anterior da palavra

vi – editor de texto

- Comandos de substituição e deleção:
  - **x** - Apaga o caracter que esta sob o cursor
  - **dw** - Apaga a palavra, da posição atual do cursor até o final
  - **dd** - Apaga a linha atual
  - **D** - Apaga a linha a partir da posição atual do cursor até o final
  - **rx** - Substitui o caracter sob o cursor pelo especificado em x(é opcional indicar o caracter)
  - **Rx** - Substitui a palavra sob o cursor pela palavra indicada em x
  - **u** - Desfaz a última modificação
  - **U** - Desfaz todas as modificações feitas na linha atual
  - **J** - Une a linha corrente a próxima
  - **s:/Linux/Unix** - Substitui a primeira ocorrência de "Linux" por "Unix"
  - **s:/Linux/Unix/g** - Substitui a ocorrência de "Linux" por "Unix" em todo arquivo

Execução de processos em background

- Por vezes é necessário executar processos que requerem muito tempo de execução, como solução existem diversos comandos e ferramentas que possibilitam a execução de processos em background.
- Exemplo:

```
rrodrigues@mendel:~$ man man &
[2] 42527
rrodrigues@mendel:~$ fg
```

  - **&** no fim de um comando “desbloqueia” o terminal, criando um processo em background com um id;
  - Em alternativa o comando **ctrl+z** pode substituir o **&** após a execução do programa;
- Para recuperar o **último** processo executado em background pode-se usar o comando **fg**

Execução de processos em background

- Todos os processos estão associados ao tempo de vida de um terminal, ou seja, se um utilizador **fechar um terminal ou a conexão ao terminal falhar**, todos os processos associados ao terminal também são **fechados automaticamente**.
- Esta propriedade obriga a utilização de ferramentas como **nohup** que possibilitam a manutenção de processos que tenham um tempo de vida dependente somente do próprio processo.

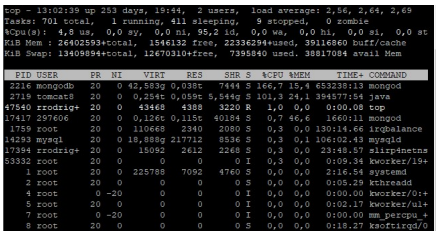
```
rrodrigues@mendel:~$ nohup man man &
[3] 42553
rrodrigues@mendel:~$ nohup: ignoring input and appending output to 'nohup.out'

[3]   Exit 2                  nohup man man
rrodrigues@mendel:~$
```


Execução de processos em background

- Outra alternativa passa por usar terminais (sessões) com tempo de vida próprio que são independentes do terminal executado na janela do utilizador.
- Exemplos:
  - **Byobu** https://medium.com/@aliartiza75/what-is-byobu-and-how-to-use-it-b09722008d65
  - **Screen** https://linuxize.com/post/how-to-use-linux-screen/

Gestão de processos

- Para gerir os processos que estão a ser executados e os recursos utilizados num computador, o Linux detém o gestor **top**.



- Para identificar os processos que estão a correr pode-se recorrer ao comando **ps -e.**
- É possível gerir um processo recorrendo ao comando **kill**. Por vezes quando queremos fechar um programa podemos obriga-lo a terminar, enviando um sinal "**SIGKILL**" com o seguinte comando: **kill -9** <PID>

https://stackabuse.com/handling-unix-signals-in-python/

https://likegeeks.com/killing-a-process-in-linux/

Lista de comandos

- http://linuxcommand.org/
- https://pplware.sapo.pt/linux/comandos-linux-para-totos-tutorial-no26/
- Pipes e redirecionamento:
- https://ryanstutorials.net/linuxtutorial/piping.php