UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS E PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

2 – Distribuições Linux

2.1 – Introdução ao Linux

Sistemas Operativos e Redes| Eng.º Milton Aguiar

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

Introdução ao Ambiente Unix/Linux

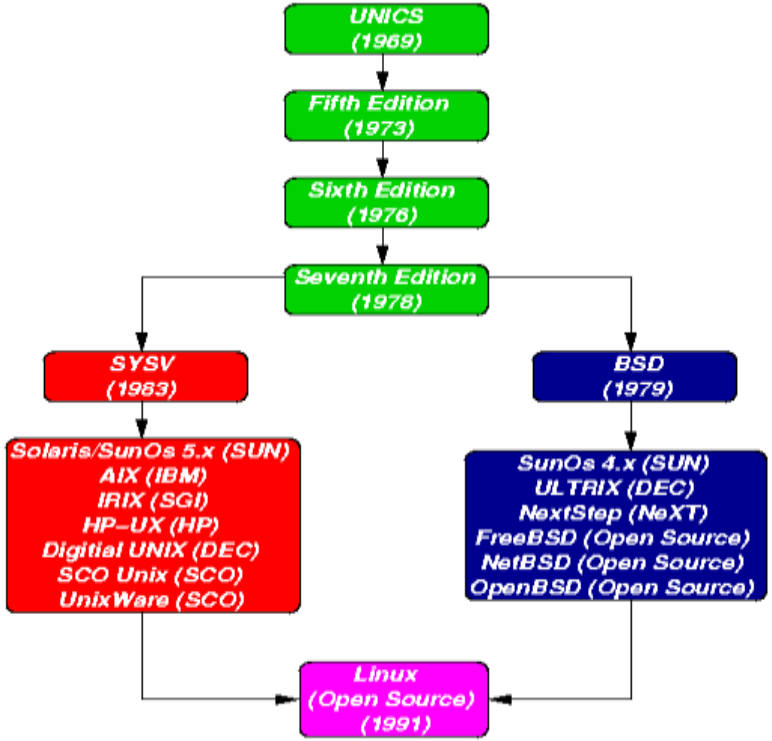
Objetivos

- Visão geral do sistema operativo Unix (e Linux) na ótica do utilizador
- Apresentação dos comandos mais usados

Introdução ao Ambiente Unix/Linux

- Introdução
- Sistema de Ficheiros
- Permissões
- Processos
- Processamento de Texto
- Agendamento
- Edição de texto
- *Shell Scripts*

Introdução - Visão Histórica



Fonte: http://www.doc.ic.ac.uk/~wjk/UnixIntro/Lecture1.html

Características principais (1/2)

Portabilidade

- Sistema Operativo
- Aplicações

Comunicações

- Entre programas – IPC (*Interprocess Communications*)
- Entre utilizadores...

Características principais (2/2)

Modularidade

- Adição e remoção de módulos muito facilitada

Multitarefa

- Execução de várias processos ao mesmo tempo

Multi-utilizador

- Possibilidade de vários utilizadores acederem ao sistema ao mesmo tempo

Kernel e Shell

O Kernel é o núcleo do sistema Operativo:

- Aloca tempo e memória para programas e faz a gestão de arquivos de comunicações em resposta a chamadas de sistema

A Shell é a interface entre o utilizador e o núcleo. Exemplos de Shell's:

- C Shell (CSH)
- Prompt %
- Bourne Shell
- Padrão do Unix
- Prompt $

- Kernel Shell
- Mais poderosa
- Prompt #

Utilitários de Sistema e Aplicações

Utilitários de sistema

- Os comandos, especificados pelo POSIX2, que permitem interagir com o Sistema Operativo (`ls, cp, grep, awk, sed, bc, wc, more`, ...)
- *deamons*, aplicações de sistema que permitem operações remotas e de administração (telnet, sshd, lpd, httpd, crond,...)
- Aplicações
- Instaladas por defeito (vi, emacs, gcc, latex,...)

Aceder a sistemas Unix

Terminais de texto

- Uso do `Telnet` (pouco seguro, e pouco usado) ou `ssh` (cifrado, e.g., Putty em Windows) em ambiente de consola de texto
- `login: <número de aluno>`
- `password: <pin>`

Ambientes gráficos

- Semelhante a outros sistemas gráficos (Windows ou MacOS)

Estrutura de um comando

Uma linha de comando segue a seguinte estrutura:

- **Comando** *Unix*: utilitário de sistema a invocar
- **Argumentos**:
- **Opções**: alteram o comportamento do comando
- **Destinatários**: ficheiros, diretórios e/ou outras entidades do Sistema sobre qual o comando atua

```
command -options targets
```

- Exemplos:

```
$ ls
$ cp files.txt documents/
$ ps -ax
$ cp -r documetns/ backups/
```

Sistema de Ficheiros

- Num sistema Unix toda a entidade persistente é guardada sob a forma de ficheiro. Existem 4 tipos de ficheiros:

- **Normais -** ficheiros comuns, com texto, dados ou programas. Não usam nenhuma estrutura (e.g., extensão) particular.

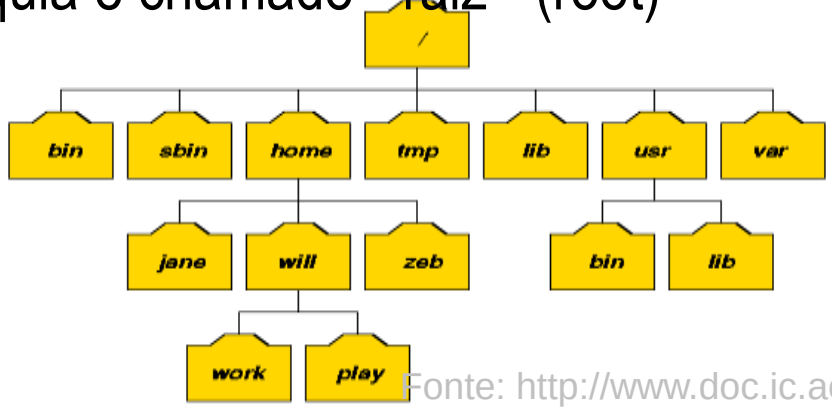
- **Diretorias -** Podem conter outras diretorias e/ou ficheiros

- **Dispositivos -** Apontadores para os dispositivos existentes no sistema. Existem dois tipos básicos de dispositivos: de blocos (e.g., discos rígidos) e de caracteres (e.g., modem e terminais)

- **Ligações -** Apontador para outro ficheiro. Existem dois tipos: *hard* e *soft*. (podemos assimilar a atalhos no windows)

Estrutura do Sistema de Ficheiros

- O sistema de ficheiros é organizado numa estrutura hierárquica, como uma árvore invertida.
- O topo da hierarquia é chamado " raiz " (root)



Fonte: http://www.doc.ic.ac.uk/~wjk/UnixIntro/Lecture2.html

Sistema de Ficheiros

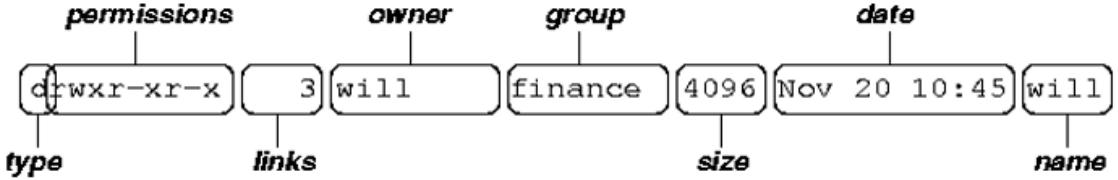
- **Listar arquivos e diretorias**
  - Após login no sistema a diretoria por defeito é HOME (`/home/user`)
  - Comando: **ls** (list schema). Exemplo **`ls -la`** :

```
$ ls -l -a                    (ou %ls -la)
drwxr-xr-x   5      aluno01  staff        512 Jul 10 09:44 .
drwxr-xr-x   13     root  root         512 Jul 10 09:37 ..
-rw-r--r--   1      aluno01  staff        137 Jul 10 09:42 .cshrc
-rw-r--r--   1      aluno01  staff        575 Jul 10 09:37 .login
-rw-r--r--   1      aluno01  staff        324 Jul 10 09:45 despesas
drwxr-xr-x   2      aluno01  staff        512 Jul 10 09:43 documentos
drwxr-xr-x   2      aluno01  staff        512 Jul 10 09:43 imagens
-rw-r--r--   1      aluno01  staff       3636 Jul 10 09:45 indice
```

Sistema de Ficheiros

- **Criar diretorias**
  - Comando: mkdir. Exemplo: `$ mkdir gsr`.
- Outros comandos para diretorias:
  - `cd` – mudar a diretoria atual
  - `pwd` – conhecer o caminho (*path*) atual

Sistema de Ficheiros



Fonte: http://www.doc.ic.ac.uk/~wjk/UnixIntro/Lecture2.html

- *type*
  - Carácter único que tem as seguinte opções: 'd' (*directory*), '-' (*ordinary file*), 'l' (*symbolic link*), 'b' (*block-oriented device*) or 'c' (*character-oriented device*).
- *permissions*
  - Conjunto de caracteres que descrever os direitos de acesso ao ficheiro (detalhe na secção de permissões).
- *links*
  - Refere o número de ligações no sistema de ficheiros para o ficheiro ou diretório (detalhe na secção de ligações *hard* e *soft*).
- *owner*
  - Normalmente o utilizador que criou o ficheiro ou diretório
- *group*
  - Grupo, ou seja, conjunto de utilizadores, que têm permissões de aceder ao ficheiro de acordo com definido no campo das permissões referente ao grupo.
- *size*
  - Dimensão do ficheiro ou o número de *bytes* usados pelo sistema operativos para guarda a lista de ficheiros de uma diretoria
- *date*
  - Data da última modificação do ficheiro ou diretorias
- *name*
  - Nome do ficheiro ou diretoria

Sistema de Ficheiros

Exibir o conteúdo de um arquivo

- Comandos: `cat, more, head, tail`. Exemplos:

```
$ cat /etc/passwd
$ more ~/.bash_history
```

Copiar e mover arquivos

- Comandos: `cp` (copy) e `mv` (move). Exemplo:

```
$ cp /etc/passwd ./passwd.bak
```

Remover arquivos e directorias

- Comandos: `rm` (remove) e `rmdir`. Exemplo:

```
$ rm /tmp/game.log
```

Sistema de Ficheiros

O comando ***ln*** permite criar links.

Existem dois tipos de links suportados pelo Linux, os ***hard links*** e os ***links simbólicos***.

Os links simbólicos têm uma função parecida com os atalhos do Windows, eles apontam para um arquivo, mas se o arquivo é movido para outro diretório o link fica inativo/inútil.

Os hard links são semelhantes aos atalhos do OS/2 da IBM, eles são mais "intimamente" ligados ao arquivo e são alterados juntamente com ele. **Se o arquivo muda de lugar, o link é automaticamente atualizado.**

Sistema de Ficheiros

O comando **ln** dado sem argumentos cria um ***hard link,*** como em:

$ ***ln /home/apel/arquivo.txt arquivo***

Onde será criado um link chamado "arquivo" no diretório corrente, que apontará para arquivo.txt dentro do diretório /home/apel

Para criar um link simbólico, basta acrescentar o argumento "-s", como em:

$ ***ln -s /home/apel/arquivo.txt arquivo***

Sistema de Ficheiros

- **Carateres *Wildcards***
  - São carateres que permitem substituir parte de caminhos e nomes de ficheiros e diretorias nos comandos, reduzindo a digitação e trazendo flexibilidade

* - substitui um qualquer número de caracteres no nome do ficheiro ou diretoria

? - substitui um e apenas um carácter no nome do ficheiro ou diretoria

[ ] - aceita intervalos de letras/inteiros (x-y,z,t,1-9)

{ } - expande usando o produto cartesiano (valores separados por vírgulas)

Sistema de Ficheiros

- Caracteres *Wildcards*. Exemplos:

`???`	aponta para todos os ficheiros com 3 caracteres no nome
`?ell?`	aponta para todos os ficheiros de 5 letras com 'ell' no meio.
`he*`	qualquer ficheiro iniciado por 'he'
`[m-z]*[a-l]`	qualquer ficheiro iniciado por letra entre 'm' e 'z' e terminado entre 'a' e 'l'

`{/usr,}{/bin,/lib}/file` expande as diretorias para `/usr/bin/file /usr/lib/file /bin/file & /lib/file`

- Nota: estas expansões são realizadas antes da execução dos comandos.

Sistema de Ficheiros

- **Caracteres especiais**

'\' para incluir caracteres especias/restritos

" " para evitar algumas expansões de *wildcards*

' ' para evitar todas as expansões de *wildcards*

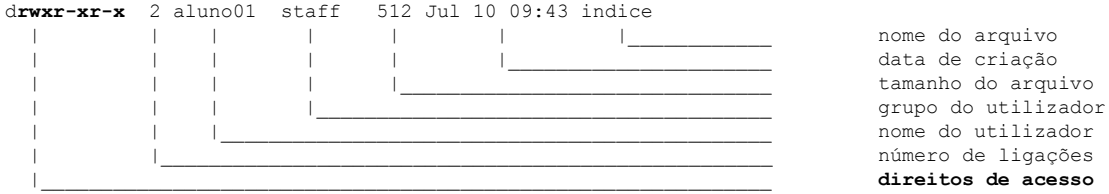
` para executar comandos dentro de outros comandos

- Exemplo do uso de ` :

```
$ hostname
rose
$ echo this machine is called `hostname`
this machine is called rose
```

Permissões de Ficheiros e Diretorias

- O Unix/Linux implementa um esquema de permissões que permite identificar quem, e de que forma, pode aceder a um dado recurso
  - A primeira coluna contém 10 caracteres onde temos o tipo de arquivo e 9 caracteres para definir as suas permissões



Permissões de Ficheiros e Diretorias

- Tipo de arquivos possíveis:
  - '-' - arquivo (ficheiro)
  - d - diretoria (pasta)
  - l - ligação (atalho)
  - b,c,p,s - especiais
- Os 9 restantes caracteres são divididos em 3 grupos de 3 caracteres cada para definir as permissões do:
  - **Dono do arquivo**
  - **Grupo do arquivo**
  - **Outros utilizadores**

Permissões de Ficheiros e Diretorias

- Em arquivos:
  - **r** indica permissão de leitura, o utilizador pode ler o conteúdo
  - **w** indica permissão de escrita, o utilizador por alterar o conteúdo
  - **x** indica permissão de execução, o utilizador pode usar o ficheiro como um comando Unix/Linux
- Em diretorias
  - **r** permite listar arquivos e diretorias presentes na diretoria
  - **w** permite apagar/mover/criar arquivos da diretoria
  - **x** permite ter acesso à diretoria, mas não define a permissão de listar (r)

Alguns exemplos:

`rwxrwxrwx` Um arquivo com estas permissões pode ser lido, gravado e executado por qualquer utilizador, independente de ser ou não o dono ou pertencer ou não ao grupo do dono do arquivo.

`rwx------` Um arquivo com estas permissões pode ser lido, gravado e executado somente pelo dono do arquivo.

Permissões de Ficheiros e Diretorias

- Alterar as permissões – apenas permitido ao dono do arquivo (e ao root)
  - Comando: `chmod` *`options`* `files`
- Exemplos:
  - `chmod go-rwx list01.txt`: Retirar todas as permissões ao *grupo* e aos *outros* sobre list01.txt
  - `chmod a+rw list01.txt`: Atribuir a *todos* permissões de leitura e escrita sobre o arquivo
- Outro comando importante:
  - `chown` ; alterar o dono do arquivo

Símbolo	Significando
u	utilizador [user]
g	grupo [group]
o	outros [other]
a	todos [all]
r	leitura
w	escrita (e apagamento)
x	execução
+	adiciona permissão
-	retira permissão

Permissões de Ficheiros e Diretorias

- Outros exemplos/ forma de configuração
  - Agora, suponha que queira dar a permissão de escrita para todos no arquivo index.html. Bastaria executar o comando:
    - *chmod 666 index.html*
    - Verificando:
      - *ls -l index.html*

        -rw-rw-rw- 1 daniduc webmasters 4632 2004-12-17 13:39 index.html

Símbolo	Significando
u	utilizador [user]
g	grupo [group]
o	outros [other]
a	todos [all]

```
rwx - Valor octal
000 - 0
001 - 1
010 - 2
011 - 3
100 - 4
101 - 5
110 - 6
111 - 7
```

Para retirar a permissão de escrita dos outros e do grupo fica:
chmod 644 index.html
ls -l index.html
-rw-r--r-- 1 daniduc webmasters 4632 2004-12-17 13:39 index.html

Ler conteúdo de arquivos

- `file <ficheiro(s)>`

```
$ file myprog.c letter.txt webpage.html
myprog.c:      C program text
letter.txt:    English text
webpage.html:  HTML document text
```

- Exibir o conteúdo de um arquivo
  - `cat, more, head, tail <ficheiros(s)>`

```
cat /etc/passwd
more ~/.bash_history
tail -5 /etc/services
```

Pesquisa de ficheiros

- Três formas para pesquisar ficheiros:

`find` *`directory`* `-name` *`targetfile`*

```
which <system command>
```

```
locate <string>
```

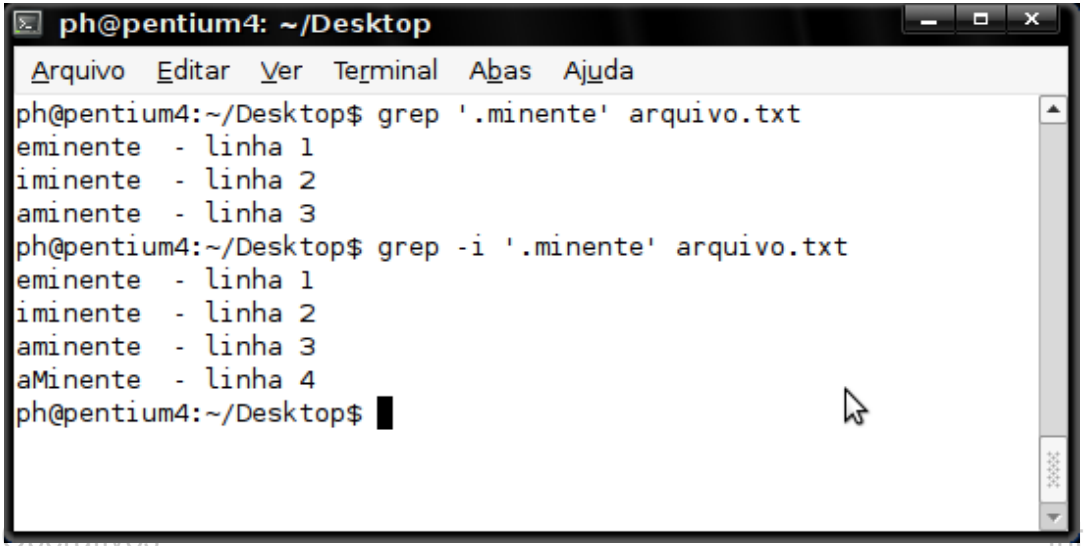
`find` é a mais usada. Exemplos:

```
$ find / -name printer
/usr/share/foomatic/db/source/printer
/usr/share/terminfo/p/printer
/sys/class/printer
```

Sendo que aqui ele está procurando no diretório raiz (/) pelo termo "printer".

Pesquisa de texto dentro dos ficheiros

- Comando `grep` (*General Regular Expression Print*). Exemplos:



Ordenação do texto dos ficheiros

- Existe o comando `sort` para, linha a linha, ordenar o texto de um ou vários ficheiros

```
sort texto.txt - Organiza o arquivo texto.txt em ordem crescente.

sort texto.txt -r - Organiza o conteúdo do arquivo texto.txt em ordem
    decrescente.
```

O comando `uniq` elimina linhas iguais

Compressão de Ficheiros e salvaguarda

- Em Unix/Linux existem várias ferramentas
  - `tar` (tape archiver)

Criar: `tar -cvf archivename filenames`
Extrair: `tar -xvf archivename`
Listar: `tar -tvf archivename`

  - `compress, gzip`

```
compress filename
gzip filename
```

A opção -d reverte a compressão

Processos

- **É um programa em execução identificado por um PID (identificador de processo) único. É filho da *shell* onde é criado.**
  - Comando: `ps`
  - `$ ps`
  - `  PID TTY          TIME CMD`
  - `17717 pts/10   00:00:00 bash`
  - `27502 pts/10   00:00:00 ps`
  - Outros exemplos:
  - **`ps -fae`** ; conhecer todos os processos do sistema
  - `ps -aeH` ; hierarquia de processos do sistema
  - Outro comando para conhecer os processos: `top`

s - set update frequency

k - kill process (by PID)

u - display processes of one user

q - quit

Processos - *Background*

- Por vezes é conveniente colocar um processo em segundo plano (processos longos e sem intervenção do utilizador)
  - Colocar em *backgroung*: `'comando &'`
  - ou então comando `bg` se o processo já está activo
  - Por exemplo:

`find / -name lista1.txt &` ; pode demorar algum tempo a encontrar todos os arquivos com este nome

Processos – *Foreground* e Terminar

- Listar processos em segundo plano: `jobs`
  - Exemplo de um resultado

```
[1]   Suspended      sleep 100
[2]   Running        netscape
[3]   Suspended      find
```

  - Passar um processo para primeiro plano: `fg`
    - Exemplo: `fg %1`
  - Terminar um processo
    - Comando: `kill`
    - Exemplo:

```
kill %1
```

kill 21587, neste caso utilizou-se o PID?

Processos - *Pipes*

- Permite que vários programas sejam concatenados, de forma que a saída de um programa é a entrada do próximo (muito poderoso).
  - Exemplos:

`who | sort` ; lista os utilizadores no sistema de forma ordenada.

`ps | grep a2012345` ; Lista os processos do utilizador a2012345

Informação do Utilizador e Comunicação

- Obter informação dos utilizadores

`who`, `finger`: permitem obter diversos dados dos utilizadores a usar o sistema

- Comunicação entre utilizadores

`talk` - comunicação direta entre dois utilizadores;

`write` - envia mensagem a utilizador ativo no sistema;

`wall` - envia a mensagem para todos os utilizadores ativos no sistema.

`mesg n/y` - cancela o recebimento de mensagens via os comandos anteriores.

Comunicação – e-mail

- `mail`: comando para gestão do e-mail. Exemplo:

```
$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/will": 2 messages 2 new
1 jack@sprat.com      Mon Dec 11 10:37 "Beanstalks"
2 bill@whitehouse.gov Mon Dec 11 11:00 "Re: Monica"
```

  - Comandos principais:
    - `?` — help
    - `q` — quit, saving changes to mailbox
    - `x` — quit, restoring mailbox to its original state
    - `t` *`messagelist`* — displays messages
    - `+/-` — show next/previous message
    - `d` *`messagelist`* — deletes messages
    - `u` *`messagelist`* — undelete messages
    - `m` *`address`* — send a new email
    - `r` *`messagelist`* — reply to sender and other receipients
    - `R` *`messagelist`* — reply only to sender

```
$ mail -s "Hi" wjk@doc.ic.ac.uk < message.txt
```

Processamento de texto – sed

Stream editor (`sed`)

- Permite a transformação de fluxos de texto. Exemplos:
  - Substituir o padrão 1 pelo 2, uma vez em cada linha

```
$ sed "s/pattern1/pattern2/" inputfile > outputfile
```

  - Substituir o padrão 2 pelo 1, sempre

```
$ sed "s/pattern1/pattern2/g" inputfile > outputfile
```

  - Apagar as linhas com o padrão 1

```
$ sed "/pattern1/d" inputfile > outputfile
```

  - Substituir os caracteres 2 pelos 1

```
$ sed "y/string1/string2/" inputfile > outputfile
```

Processamento de texto – awk

- *Aho, Weinberger and Kernigan* (`awk`)
  - Permite a manipulação de ficheiros com dados em colunas. Exemplo, considere os seguintes dados (ficheiro: `cricket.dat`):

```
1 atherton     0   bowled
2 hussain     20   caught
3 stewart     47   stumped
4 thorpe      33   lbw
5 gough        6   run-out
```

  - **Para obter os dados da 2ª e 3ª coluna**
  - `$ awk '{ print $2 " " $3 }' cricket.dat`
  - `atherton 0`
  - `hussain 20`
  - `stewart 47`
  - `thorpe 33`
  - `gough 6`

Ajuda

- Caso estejam instaladas, existem ferramentas de ajuda ao utilizador na utilização dos vários comandos:
  - `man <comando>` - apresenta uma descrição detalhada do comando, incluindo a explicação dos parâmetros possíveis
  - `whatis <comando>` - apresenta uma descrição curta do comando
  - `apropos <palavra>` - apresenta uma lista de comando onde a palavra surge
  - `info` – versão interativa da ajuda

Terminar um sistema

- **Shutdown: `shutdown, halt, reboot`**
  - Permitem desligar um sistema de forma controlada. Exemplos:

```
# /sbin/shutdown -r now   (desligar já e reiniciar)
# /sbin/shutdown -h +5    (desligar dentro de 5 minutos & halt)
# /sbin/shutdown -k 17:00 (simular um 'desligar' às 5pm)
```

  - **`halt` e `reboot` são equivalentes a `'shutdown -h'` e `'shutdown -r'`**
  - Em caso de urgência, usar `sync` para atualizar o sistema

Tarefas agendadas

- **O *crond* é um serviço que executa comandos agendados. Para agendar/listar os agendamentos usar `crontab -e/l`**
  - Formato de cada entrada:

minute hour day_of_month month weekday *command*

  - These fields accept the following values:

```
minute                  0 a 59
hour                    0 a 23
day_of_month 1 a 31
month                   1 a 12
weekday                 0 (Sun) a 6 (Sat)
command                 o comando a executar

30 6 * * 1,3,5 /usr/bin/clean (?)
* * * * * /home/user/script.sh (?)
```

Edição de texto - vi

- Para iniciar: `vi ficheiro`
- Dois modos de uso:
  - Modo **comando**, os caracteres indicam ações (mover cursor, gravar, copiar, colar, etc...)
  - Modo **entrada**, os caracteres são adicionados no documento
- Notas:
  - No arranque, está em modo comando
  - Para passar para modo entrada, digitar 'i'
  - Para sair do modo entrada, carregar 'Esc'

Edição de texto - vi

- **Edição básica**
  - `h` (left), `j` (down), `k` (up) e `l` (right), são os caracteres que permite navegar no documento (e as teclas de cursor em alguns sistemas)
  - `^` e `$` para colocar o cursor no início e fim da linha, respetivamente
  - `^F` e `^B` para colocar o cursor na página anterior e seguinte, respetivamente
  - `nGG` para ir para a linha n
  - `x` apaga o carácter atual, `dw` a palavra, `d4w` as seguintes 4 palavras, `dd` a linha completa, `4dd` as 4 linhas seguintes. `u` para retornar da última alteração (*undo*)
- **Mover e copiar:**
  - Após apagar, `p` cola o texto no local do cursor
  - Para copiar e colar, trocar o `d` de apagar por `y`, e.g., `5yy` copia e `p` cola

Edição de texto - vi

- Pesquisa e substituição
  - Em modo comando, usar `/<texto>` para pesquisar
  - `%s/pattern1/pattern2/g`, permite alterar 1 por 2 em todo o texto
- Outros comandos
  - `:w` para gravar o texto
  - `:q` para sair (`:q!` para sair sem gravar)
  - `:wq` para gravar e sair (`zz` é idêntico)
  - `:e` *`ficheiro`*, para editar outro ficheiro
  - `:!`*`shellcomand`*, para executar comando da *shell*

Edição de texto - vi

Inserir e editar Texto

i insert text (and enter input mode)

$a append text (to end of line)

ESC re-enter command mode

J join lines

Movimento do Cursor:

h left

j down

k up

l right

^ beginning of line

$ end of line

1 G top of document

G end of document

<n> G go to line <n>

^F page forward

^B page backward

w word forwards

b word backwards

Edição de texto - vi

Apagar e mover texto:

Backspace delete character before cursor (only works in insert mode)

x delete character under cursor

dw delete word

dd delete line (restore with p or P)

<n> dd delete n lines

d$ delete to end of line

dG delete to end of file

yy yank/copy line (restore with p or P)

<n> yy yank/copy <n> lines

Pesquisa e Substituição

%s/<search string>/<replace string>/g

Diversos:

u undo

:w save file

:wq save file and quit

ZZ save file and quit

:q! quit without saving

Edição de texto - outros

- emacs
  - Muito mais que edição de texto. Permite consultar vários ficheiro ao mesmo tempo, compilar e depurar aplicações, executar comandos da *shell*, ler páginas de manual, e-mail e ainda operar como navegador.
- pico (http://www.ece.uwaterloo.ca/~ece250/Online/Unix/pico/)
  - Editor de texto dos mais simples, com poucos comandos (semelhante ao *Notepad* do *Windows*)
  - Nota: ambos não são standard do Unix/Linux

Shell scripts

- Um *Shell Script* é um ficheiro de texto com comandos na linguagem da *shell* (*batch file* em *Windows*)
  - Será usada a *Bourne shell*, `sh`

Shell scripts

- Variáveis
  - Dados apontados por um nome. Exemplo, variável 'mundo':

```
$ mundo='hello world'
$ echo $mundo
hello world
```

  - No exemplo anterior, a variável é local à shell. Para poder ser usada em outras shells, devemos 'exportar'. Exemplo:

```
$ export mundo
```

  - `PATH`, `HOME`, `HOSTNAME`, `SHELL`, são exemplos de variáveis do ambiente (experimente `echo $<variável>`)

Shell scripts

- Pequeno exemplo, ficheiro `teste`:

```
#!/bin/sh
# Isto e' um comentário
echo "O numero de argumentos e' $#"
echo "Os argumentos sao: $*"
echo "O primeiro e' $1"
echo "O numero do processo e' $$"
echo "Digite um numero: "
read numero
echo "O numero que digitou foi o $numero"
```

#! informa o Unix para usar a Bourne shell (sh)

- Para executar o ficheiro deverá ter as permissões de execução. Exemplos de utilização:

```
$./teste hello world

$echo 5 | ./teste hello world
```

Shell scripts

- Nos próximos módulos irão abordar com mais pormenor…

Introdução aos Sistemas Operativos