LPI 105.1. Configuració a mida i ús de l'entorn shell

Wikis:

http://acacha.org/mediawiki/index.php/LPI_105.1

Autor: Sergi Tur Badenas

Objectius

105.1. Configuració a mida i ús de l'entorn shell

Objectiu: Els candidats han de ser capaços de configurar el entorn shell per tal d'acomplir els requeriments dels usuaris. Els candidats han de ser capaços de modificar perfils d'usuaris globals i d'usuari. **Pes**: 4

Àrees Clau de Coneixement: Establir variables d'entorn com per exemple PATH durant el login o quan s'executa una nova shell. Escriure funcions de bash per a les seqüències de comandes més utilitzades Mantenir els directoris esquelet per a les noves comptes d'usuari Establir l'ordre de cerca del path al directori adequat

La següent és una llista parcial de fitxers, termes i utilitats utilitzades: /etc/profile env export set unset ~/.bash_profile ~/.bash_login ~/.profile ~/.bashrc ~/.bash_logout function alias lists

Apunts: LPI 105.1. Configuració a mida i ús de l'entorn shell

Autor: Sergi Tur Badenas

Objectius

105.2. Adaptar o escriure guions d'intèrpret d'ordres simples	
	▪ **Objectiu**: Els candidats han de ser capaços d'adaptar guions de shell existents o escriure nous guions de BASH simples. ▪ **Pes**: 4
	**Àrees Clau de Coneixement**: ▪ Utilitzar la sintaxi estàndard de sh (bucles, tests). ▪ Utilitzar la substitució d'ordres. ▪ Comprovar valors de retorn d'èxit o d'error o altra informació proveïda per l'ordre. ▪ Enviar correus condicionals al superusuari. ▪ Seleccionar correctament l'interpret de guions mitjançant la línia shebang (#!). ▪ Gestionar la localització, la propietat, l'execució i els drets de suid dels guions.
	La següent és una llista parcial de fitxers, termes i utilitats utilitzades: ▪ for ▪ while ▪ test ▪ if ▪ read ▪ seq
	**Apunts**: LPI 105.2. Adaptar o escriure guions d'intèrpret d'ordres simples

Autor: Sergi Tur Badenas

Com s'invoca l'interpret d'ordres bash?

- **Intèrprets d'ordres segons com s'executin:**
  - **Interactive shell**: Si no s'indica cap paràmetre s'executa una shell interactiva. També es pot indicar implícitament amb -i
  - **Non-interactive shells**: **NO** llegeixen ni executen cap dels fitxers de configuració. S'utilitza l'opció -c.
- **Els interactius poden ser de dos tipus:**
  - **Login shell:** s'executa amb els paràmetres - (cap paràmetre) o --login. Normalment només s'executa durant el procés de login, p. ex. al iniciar un interpret d'ordres des d'una consola virtual (Ctrl+Alt+F1) o al iniciar una sessió gràfica amb un gestor de pantalla (Display Manager)
  - **Non-login shell**: la resta d'execucions d'intèrprets d'ordres. És el tipus d'intèrpret d'ordre que s'executa al executar una eina de terminal com xterm o gnome-terminal.

Autor: Sergi Tur Badenas

Exemples

- **Login shell:**
  - Iniciar un sessió de root:
    - $ sudo su -
  - Executar una login bash:
    - $ bash--login
    - $ bash -
  - Executar una sessió remota amb SSH:
    - $ ssh user@host
- **No-login shell:**
    - $ bash
    - $ gnome-terminal

Autor: Sergi Tur Badenas

Exemples

- **Intèrprets d'ordres no interactius:**
  - $ su user -c /path/to/command
  - $ bash -c /path/to/command
  - $ bash script.sh
- **Consulteu:**
  - Com s'invoca l'interpret d'ordres bash

Autor: Sergi Tur Badenas

Configuració de la shell

- **Configuració de sistema (per-system basis)**
  - Segons l'estàndard FHS els fitxers de configuració que afecten a tot el sistema s'han de trobar a la carpeta:
    - /etc
  - Els fitxers són:
    - **/etc/profile**: Fitxer de configuració de la bash a nivell de sistema. Només s'utilitza amb les bash que siguin de login.
    - **/etc/bash.bashrc**: Fitxer de configuració de la bash a nivell de sistema. Només s'utilitza amb les bash que NO siguin de login.

NOTA: Es poden veure els fitxers de configuració globals com fitxers que contenen les configuracions per defecte per a tots els usuaris del sistema. Cal però, tenir en compte que les configuracions de cada usuari poden sobreescriure les configuracions globals (a l'estil del que succeïx en llenguatges de programació amb les variables locals i les variables globals)

Autor: Sergi Tur Badenas

Configuració de la shell

- **Configuració d'usuari (per-user basis)**
  - Segons FHS els fitxers que són propietat dels usuaris del sistema es troben a la carpeta:
    - /home/usuari
  - Com que els fitxers de la carpeta HOME d'un usuari són propietat seva, l'usuari pot llegir, modificar o inclús eliminar els fitxers de la seva HOME.
  - Per raons de seguretat els fitxers de configuració solen ser fitxers ocults, és a dir comencen per "." (punt).
  - Recordeu que per veure aquesta fitxers cal utilitzar el paràmetre -a de l'ordre ls:

Autor: Sergi Tur Badenas

Configuració de la shell

```
$ ls -la /home/sergi
...
drwxr-xr-x 81 sergi sergi 212992 2010-03-06 17:33 .
drwxr-xr-x  6 root  root    4096 2010-02-10 08:55 ..
drwx------  3 sergi sergi   4096 2009-08-11 16:26 .adobe
drwxr-xr-x  3 sergi sergi   4096 2010-03-04 11:14 Baixades
-rw-------  1 sergi sergi  11494 2010-03-04 23:47 .bash_history
-rw-r--r--  1 sergi sergi    220 2009-07-27 09:33 .bash_logout
-rw-r--r--  1 sergi sergi   3233 2009-08-05 13:29 .bashrc
drwx------ 16 sergi sergi   4096 2010-02-21 12:22 .cache
drwx------  3 sergi sergi   4096 2009-08-29 16:59 .compiz
drwxr-xr-x 14 sergi sergi   4096 2010-02-25 09:23 .config
drwx------  3 sergi sergi   4096 2009-07-27 10:46 .dbus
drwx------  2 sergi sergi   4096 2009-08-05 12:18 Desktop
-rw-r--r--  1 sergi sergi     63 2010-03-06 17:32 .dmrc
drwxr-xr-x 10 sergi sergi   4096 2010-01-14 14:22 Documents
...
drwx------  2 sergi sergi   4096 2010-02-21 22:48 .filezilla
drwxr-xr-x  2 sergi sergi   4096 2009-11-02 11:37 .fontconfig
drwx------  5 sergi sergi   4096 2010-03-07 07:35 .gconf
```


Autor: Sergi Tur Badenas

Configuració de la shell

- **Els fitxers de configuració de bash a nivell d'usuari (per-user basis) són:**
  - ~/**.bashrc**: S'executa només al iniciar una bash de no login.
  - ~/**.bash_profile o** ~/**.profile**: S'executa només al iniciar una bash de login.
  - ~/**.bash_logout**: S'executa al finalitzar una bash.

Tipus de fitxer	Fitxer de configuració amb login	Fitxer de configuració sense login
Sistema	/etc/profile i els fitxers de la carpeta /etc/profile.d	/etc/bash.bashrc o /etc/bashrc
Usuari	~/.bash_login, ~/.profile, o ~/.bash_profile	~/.bashrc

Autor: Sergi Tur Badenas

Configuració de la shell

- **En quin ordre s'executen?:**
  - Login:
    - /etc/profile
    - Fitxers de la carpeta /etc/profile.d
    - /etc/bashrc o /etc/bash.bashrc
    - ~/.bash_profile o ~/.profile (si existeix)
    - ~/.bashrc
    - NOTA: s'executen els fitxers de configuració de no login per què són executats implícitament pels fitxers de login
  - No-login:
    - /etc/bashrc
    - ~/.bashrc

Autor: Sergi Tur Badenas

Configuració de la shell

- **Sortida d'un interpret de login**
  - S'executa el fitxer ~/.bash_logout

```
$ cd
$ joe ~/.bash_logout

            Afegiu al principi la línia:

   echo "sortint..."

        Inicieu un nou interpret d'ordres de no-login:
$ bash
$ logout
bash: logout: not login shell: use `exit'
$ exit
exit
        Ara inicieu un interpret d'ordres de login:
$ bash --login
$ logout
sortint...
```


Autor: Sergi Tur Badenas

Configuració de la shell

- **Podeu obtenir més informació al manual de bash:**
  - $ man bash
  - Consulteu la secció **INVOCATION**
  - Opcions d'execució de bash:
    - **--noprofile**: No llegeix cap dels fitxers de configuració de bash per a intèrprets de login: No llegeix ni /etc/profile ni ~/.bash_profile o ~/.bash_login o ~/.profile.
    - **--norc**: No llegeix cap dels fitxers de configuració de bash per a intèrprets de no-login: No llegeix ni /etc/bash.bashrc ni ~/.bashrc.
    - **--posix**: s'inicia una bash compatible amb POSIX
    - **--restricted**: S'executa una bash restringida.

Autor: Sergi Tur Badenas

Documentació de bash

- **A sistemes de la família Debian hi ha un paquet anomenat bash-doc que proporciona documentació i exemples de bash:**
  - $ sudo apt-get install bash-doc
- Fitxers:
  - $ dpkg -L bash-doc
- Manual de referència de bash en format pdf:
  - $ evince /usr/share/doc/bash/bashref.pdf

Autor: Sergi Tur Badenas

Paràmetres

Paràmetres

- Un paràmetre és una entitat que emmagatzema valors.
- Un paràmetre pot ser un nom, un número, un caràcter especial o una combinació dels anteriors.
- **Una variable és un paràmetre que es denota per un nom**. P. ex. no anomenem variables als paràmetres posicionals ja que es denoten per números
  - Una variable té un valor i zero o més atributs. Els atributs s'estableixen amb la comanda interna **declare**.
  - Un paràmetre està establert si se li ha assignat un valor.
  - El conjunt de caràcters (string) nul (null) és un valor vàlid.
  - Només es pot "desestablir" amb l'ordre interna unset (no és el mateix establir-li el valor =""").

Autor: Sergi Tur Badenas

Paràmetres

- **Les variables s'assignen de la següent forma:**
  - name =[value]
  - Si no es proporciona cap valor a la variable se li assigna el conjunt de caràcters null (null string).
    - NOTA: Cal tenir en compte que per defecte s'apliquen als valors tant les expansions, com les substitucions d'ordres, etc.... Per exemple:

```
$ nom=*
$ echo $nom
```

  - Quoting:

```
$ nom="*"
$ echo "$nom"
```


Autor: Sergi Tur Badenas

Variables locals vs globals

- **Dos tipus de variables:**
  - **locals**
    - Es defineixen només per a l'interpret d'ordres que es troba en execució:

```
$ nom="Sergi Tur"
$ echo $nom
Sergi Tur
```

  - **Globals**
    - aka variables d'entorn.
    - S'han d'"exportar"

```
$ nom="Sergi Tur"
$ echo $nom
Sergi Tur
$ bash
$ echo $nom
En canvi si s'exporta la
variable amb export:
$ export nom="Sergi Tur"
$ echo $nom
Sergi Tur
$ bash
$ echo $nom
Sergi Tur
```


Autor: Sergi Tur Badenas

Variables d'entorn

- **Environment variables**
  - Contenen informació sobre l'entorn d'execució:
    - Designen l'aspecte del teu prompt, el teu directori d'usuari, el teu directori de treball, el nom del teu interpret d'ordres, fitxers que has obert, funcions o comandes que has utilitzat...
  - Són utilitzades per la bash i per altres programes
- **Exemples**
  - Variables predefinides: PATH, EDITOR, PWD, etc..
  - Variables d'entorn definides pels usuaris.

Autor: Sergi Tur Badenas

Variables d'entorn predefinides

- **Algunes de les més conegudes són:**

Nom	Funció
USER	Nom de l'usuari en el sistema
UID	Identificador numèric de l'usuari
HOME	El directori home de l'usuari
PWD	El directori de treball actual
SHELL	El nom del shell
$	Identificador del procés que s'està executant
PPID	L'identificador del procés que ha iniciat el procés que s'està executant
?	El codi de retorn de la darrera instrucció executada

Autor: Sergi Tur Badenas

Variables d'entorn

- **Consultar una variable d'entorn**
  - Ordre echo
  - Totes les variables comencen per $ (incloses les d' entorn)

```
$ echo $PATH
$ env | grep PATH
```

- **Consultar totes les variables d'entorn**
  - Ordre env
  - Proporcionada per coreutils
  - Es pot utilitzar autocompletar:
    - Escriviu “$” i tabuleu dos cops.

```
$ env
...
TERM=xterm
SHELL=/bin/bash
...
USER=sergi
...
```


Autor: Sergi Tur Badenas

Variables d'entorn

- **Establir o modificar una variable d'entorn**

```
 $ SALUTACIO=Hola
 $ export SALUTACIO
O tot al mateix temps:
 $ export SALUTACIO=hola
```

- **Esborrar una variable d'entorn**

```
$ unset SALUTACIO
```


Autor: Sergi Tur Badenas

Variable d'entorn PATH

- **Cada cop que executem un ordre, l'interpret de comandes:**

  1) Primer es comprova si l'ordre és una ordre interna de l'intèrpret. Sí és així s'executa l'ordre interna.

  2) Si l'ordre no és un ordre interna, aleshores es busca l'ordre a les carpetes especificades a la variable d'entorn PATH.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

  - Les carpetes amb executables són normalment les especificades per l'estàndard FHS

Autor: Sergi Tur Badenas

Variable d'entorn PATH

- **Cada sistema o usuari pot tenir un PATH diferent**
  - P. ex. Hi ha diferències entre distribucions:
    - **Debian:** no posa la carpeta /sbin al PATH dels usuaris normals (només al superusuari)
    - **Ubuntu:** Posa /sbin al PATH del usuaris. Que un ordre estigui al path no vol dir que tinguem permisos per executar-la
- **Modificar el path--> Modificar una variable d'entorn**
  - Compte en no esborrar les carpetes ja definides al PATH!
  - Per exemple, per afegir al PATH la carpeta /home/sergi/bin
  - Per fer el canvi permanent poseu la línia a ~/**.bashrc**

```
$ export PATH=/home/sergi/bin:$PATH
```


Autor: Sergi Tur Badenas

Comanda interna declare

- **Sintaxi:**
  - **declare OPCIÓ(ns) VARIABLE=valor**
- Permet tenir un major control sobre la declaració de variables
- Bash és una llenguatge poc “typat”

Opció	Descripció
-a	La variable és un array
-f	Només utilitza noms de funcions
-i	La variable s'ha de tractar com un enter.
-p	Mostra els atributs i els valors de cada variable
-r	Crea una variable de només lectura (constant)
-r	Crea una variable de només lectura (constant)
-r	Crea una variable de només lectura (constant). No es poden modificat ni es pot utilitzar unset
-t	Assigna a cada variable el atribut trace
-x	Marca cada variable per tal de ser exportada a comandes subsegüents a través de l'entorn

Autor: Sergi Tur Badenas

Comanda interna declare

- **Sense declare**

```
$ n=6/3
$ echo "n = $n"
n = 6/3
```

- **Amb declare:**

```
$ declare -i n
$ n=6/3
$ echo "n = $n"
n = 2
```


Autor: Sergi Tur Badenas

Constants

- **Es pot utilitzar readonly:**
  - $ readonly OPCIÓ VARIABLE(s)
- **o**
  - $ declare -r VARIABLE=VALOR
- **Les constants no es poden modificar:**

```
$ readonly TUX=penguinpower
$ TUX=Mickeysoft
bash: TUX: readonly variable
```


Autor: Sergi Tur Badenas

Paràmetres de shell

- **De Bourne shell (sh):**

CDPATH	A colon-separated list of directories used as a search path for the cd builtin command.
HOME	The current user's home directory; the default for the cd builtin command. The value of this variable is also used by tilde expansion (see Section 3.5.2 [Tilde Expansion], page 19).
IFS	A list of characters that separate fields; used when the shell splits words as part of expansion.
MAIL	If this parameter is set to a filename and the MAILPATH variable is not set, Bash informs the user of the arrival of mail in the specified file.
MAILPATH	A colon-separated list of filenames which the shell periodically checks for new mail. Each list entry can specify the message that is printed when new mail arrives in the mail file by separating the file name from the message with a '?'. When used in the text of the message, $_ expands to the name of the current mail file.
OPTARG	The value of the last option argument processed by the getopts builtin.
OPTIND	The index of the last option argument processed by the getopts builtin.
PATH	A colon-separated list of directories in which the shell looks for commands. A zero-length (null) directory name in the value of PATH indicates the current directory. A null directory name may appear as two adjacent colons, or as an initial or trailing colon.
PS1	The primary prompt string. The default value is '\s-\v\$ '. See Section 6.9 [Printing a Prompt], page 85, for the complete list of escape sequences that are expanded before PS1 is displayed.
PS2	The secondary prompt string. The default value is '> '.

Autor: Sergi Tur Badenas

Paràmetres de shell

- **Variables bash:**
  - BASH, BASHPID, BASH_ALIASES, BASH_ARGC, BASH_ARGV, EUID, GLOB_IGNORE, HISTCMD, HISTCONTROL...
  - És una llarga llista...
  - Consulteu:
    - $ man bash
    - Paràmetres de shell

Autor: Sergi Tur Badenas

Paràmetres especials

- **Són els següents**

```
*   Mostra tots els paràmetres posicionals començant per 1.

@   Mostra tots els paràmetres posicionals començant per 1. Si no
    s'estableix la variable IFS "$@" s'expandeix a "$1" "$2"

#   Mostra el número de paràmetres posicionals.

?   Mostra el codi de sortida de l'última ordre executada en primer
    terme

-   hyphen) Mostra les opcions que s'estan utilitzant en el moment de
    la invocació

$   Mostra el PID del procés en execució. En una subshell mostra
    el procés/guió que ha invocat la subshell i no pas el PID de
    la subshell. En shells interactives mostra el PID de la
    Shell.

!   Mostra el PID de l'últim procés executat en segon terme

0   Mostra el nom l'interpret d'ordres que s'està utilitzant shell
    interactiu) o el nom de l'script en cas de no ser un shell
    interatiu.
```


Autor: Sergi Tur Badenas

Gramàtica de bash

- **Definicions**
  - **blank:** un espai o tabulador
  - **Espai en blanc (whitespace)**: Vegeu la següent transparència
  - **Paraula (word o token):** seqüència de caràcters
  - **Nom (name aka identifier)**: una paraula consistent només en caràcters alfanumèrics o underscores i que comença amb un caràcter alfabètic o guip baix
  - **Metacharacter:** un caràcter que quan es citat (quoting) separa paraules: | & ; ( ) < > space tab
  - **Control operator:** un token que realitza una funció de control
    - || & && ; ;; ( ) | |& <newline>

Autor: Sergi Tur Badenas

whitespace. Internal Field Separator ($IFS)

- **A bash quan es parla d'un espai en blanc es fa referència a tres opcions:**
  - espai en blanc
  - tabulador
  - nova línia
  - qualsevol combinació de les anteriors de forma consecutiva
- **Bash i word splitting**
  - Utilitzat contínuament (p. ex. amb l'ordre interna read, for, llistes...).
  - Es consideren paraules aquelles que estan delimitades per un espai en blanc (és a dir una de les 3 opcions indicades anteriorment!)

Autor: Sergi Tur Badenas

whitespace. Internal Field Separator ($IFS)

- **El caràcter especial $IFS determina quin és el separador de paraula.**
  - Valor per defecte. Per veure'l correctament cal fer:

```
$ echo "$IFS" | cat -vte
 ^I$
$
```

    - On podeu veure l'espai a la primera línia, després es mostra el tabulador (^I) i finalment el salt de línia (els $ indicant on s'acaben les línies)

- **Exemple:**

```
$ joe split.sh
data='x|y|z'
IFS='|'
for i in $data
 do
  echo $i
 done
$ bash split.sh
x
Y
z
```


Autor: Sergi Tur Badenas

Ordre simple

- **Sintaxi:**

```
$ ordre argument1 argument2 argument3 ... <operador_de_control>
```

  - Vegem les opcions possibles en un guió de shell:
    - ls -la <newline>
    - ls -la;
    - ls -la&

Autor: Sergi Tur Badenas

Canonades (pipelines)

- **És una seqüència d'ordres simples separades per l'operador de control "|" o "|&". La sintaxi és:**
  - **$ ordre1 | ordre2** o **$ ordre1 |& ordre2**
- Un conducte enllaça la sortida estàndard de l'ordre n-1 amb l'entrada estàndard de l'ordre n.
- Si s'indica |& s'enllaça la sortida estàndard i la sortida estàndard d'error.
- **IMPORTANT**: Cada ordre del conducte és executada en un procés diferent i de forma concurrent. Per exemple:

```
$ sleep 100 | wc -l
$ ps aux
...
sergi    12731  0.0  0.0   2956   628 pts/6    S+   20:09   0:00 sleep 100
sergi    12732  0.0  0.0   3276   700 pts/6    S+   20:09   0:00 wc -l
```


Autor: Sergi Tur Badenas

Llistes

- **Conjunt d'ordres simples o conductes**
  - Les llistes es poden fer amb algun dels següents operadors de control:
    - ; & && ||
  - **Llista seqüencial**: `$ ordre1 ; ordre2; ordre3`
    - Amb salts de línia o sense salts de línia!
  - **Llista AND**: `$ mkdir carpeta && cd carpeta`
    - L'ordre 2 només s'executa sí i només si l'ordre1 s'executa correctament (codi de sortida 0).
  - **Llista OR**: `$ mkdir carpeta || exit`
    - L'ordre 2 només s'executa sí i només si l'ordre1 s'executa incorrectament (codi de sortida <> 0).

Autor: Sergi Tur Badenas

Comandes compostes (Compound commands)

- **Procés a part:**
  - (list)
- **Seqüencial:**
  - { list; }
  - És el que fan les funcions -->
- **Expressions**
  - Aritmètiques
  - Condicionals
- **Estructures de control condicionals i de bucle**
  - **Condicionals:** if, case, select
  - **Bucle:** for, while, until

```
function nom {
 list;
}
```


Autor: Sergi Tur Badenas

Ordre test

- **Serveix per provar expressions**
  - El codi de retorn (exit status) és el resultat d'avaluar una expressió. Per exemple:

```
$ test -e /etc/network/interfaces
$ echo $?
0
$ test -e /etc/network/interfacesads
$ echo $?
1
```

- **Consulteu:**
  - **$ man test**

Autor: Sergi Tur Badenas

Expressions aritmètiques

- **Sintaxi:**
  - ((expression))

```
$ x=1
$ y=$((x+1))
$ echo $y;
2
```

- **Alternatives (ordres internes)**
  - **let**:

```
$ i=1
$ let i+=1
$ echo i
2
```

  - **expr: -->**

```
$ expr 1 + 1
2
$ expr 3 \* 2
6
$ expr 6 / 3
2
$ expr 6 % 3
0
$ expr 3 / 2
1
$ expr 3 / 6
0
$ expr 6 \* 3.5
expr: non-numeric argument
```


Autor: Sergi Tur Badenas

Operadors aritmètics

- **Operadors que es poden utilitzar en expressions aritmètiques**

```
id++ Post_increment. S'augmenta en un el valor després d'utilitzar-lo.
id-- Post_decrement. Es disminueix en un el valor després d'utilitzar-lo
++id Pre-increment. S'augmenta en un el valor abans d'utilitzar-lo.
--id Pre-decrement. Es disminueix en un el valor després d'utilitzar-lo
!~   Negació lògica i negació bitwise
**   Exponenciació
*    Multiplicació
/    Divisió
%    Resta de la divisió
+    Suma
-    Resta
<<   Left Bitwise Shift
>>   Right Bitwise Shift
<    Menor que
<=   Menor igual que
>    Menor que
>=   Major igual que
==   Igual
!=   No igual
&    Bitwise AND
^    Bitwise exclusive OR
|'   Bitwise OR
&&   Logical AND
'||  Logical OR
expr ? expr : expr Operador condicional
```

Assignacions aritmètiques

```
=
*=
/=
%=
+=
-=
<<=
>>=
&=
^=
|=
```


Autor: Sergi Tur Badenas

Expressions condicionals

- **Sintaxi:**
  - [[expression]]

```
$ if [[ "text" != "text2" ]]; then echo "diferents"; fi
diferents

En canvi:

$ if [[ "text" == "text" ]]; then echo "iguals"; fi
iguals
```


Autor: Sergi Tur Badenas

Expressions condicionals

- **Fitxers:**

```
-a fitxer        cert si el fitxer existeix.
-b fitxer        cert si el fitxer existeix i és un fitxer especial de bloc.
-c fitxer        cert si el fitxer existeix i és un fitxer especial de caràcter.
-d fitxer        cert si el fitxer existeix i és un directori.
-e fitxer        cert si el fitxer existeix.
-f fitxer        cert si el fitxer existeix i és un fitxer regular.
-g fitxer        cert si el fitxer existeix i el set-group-id està establert.
-h fitxer        cert si el fitxer existeix i és un enllaç simbòlic
-k fitxer        cert si el fitxer existeix i el sticky-bit està establert.
-p fitxer        cert si el fitxer existeix i és un conducte amb nom (named pipe (FIFO)).
-r fitxer        cert si el fitxer existeix i és llegible.
-s fitxer        cert si el fitxer existeix i té una mida diferent de zero.
-t fd            cert si el descriptor de fitxer existeix i està obert i es refereix a una terminal.
-u fitxer        cert si el fitxer existeix i el set-user-id està establert.
-w fitxer        cert si el fitxer existeix i és modificable.
-x fitxer        cert si el fitxer existeix i és executable.
-O fitxer        cert si el fitxer existeix i és propietat del effective user id.
-G fitxer        cert si el fitxer existeix i és propietat del effective group id.
-L fitxer        cert si el fitxer existeix i és un enllaç simbòlic
-S fitxer        cert si el fitxer existeix i és un endoll (socket).
-N fitxer        cert si el fitxer existeix i ha estat modificat des de l'últim cop que es va llegir.
file1 -nt file2  cert si el fitxer 1 és més nou que el fitxer2 (segons la data de modificació)
file1 -ot file2  cert si el fitxer 1 és més vell que el fitxer2 o sí el fitxer2 existeix i el fitxer1 no.
file1 -ef file2  sí es refereixen al mateix dispositiu o tenen el mateix inode.
```


Autor: Sergi Tur Badenas

Expressions condicionals

- **Altres expressions**
  - **-o optname**: cert si l'opció de shell optname està establerta.
  - **-z string:** cert si la mida de l'string és zero
  - **-n string:** cert si la mida de l'string és diferents de zero
  - **string1 == string2:** cert si dos strings són iguals (també se pot utilitzar =)
  - **string1 != string2:** cert sí dos strings no són iguals
  - **string1 < string2:** cert si l'string1 s'ordena lexicogràficament abans que l'string2
  - **string1 > string2:** cert si l'string1 s'ordena lexicogràficament després que l'string2

Autor: Sergi Tur Badenas

Expressions condicionals

- **Operadors de comparació**
  - arg1 OP arg2
- **On OP (operador):**
  - **-eq**: igual (equal)
  - **-ne**: no igual (not equal)
  - **-lt**: menys que (less than)
  - **-le**: menys que o igual (less than or equal)
  - **-gt**: més gran que (greater than)
  - **-ge**: més gran que o igual (greater than or equal)

Autor: Sergi Tur Badenas

Estructures de control

- **Condicionals**
  - **if**. La sintaxi és:

```
if test-commands ; then
      consequent-commands ;
[elif more-test-commands ; then
     more-consequents ;]
[else alternate-consequents ;]
fi
```

  - Exemple:

```
#!/bin/bash
T1="foo"
T2="bar"
if [ "$T1" = "$T2" ]; then
   echo expression evaluated as true
else
   echo expression evaluated as false
fi
```


Autor: Sergi Tur Badenas

Estructures de control

- **case**. Sintaxi:

```
case word in
   [ [(] pattern [| pattern ]...)

      Command-list ;;]
...
esac
```

- Exemple:

```
echo -n "Enter the name of an animal: "
read ANIMAL
echo -n "The $ANIMAL has "
case $ANIMAL in
        horse | dog | cat) echo -n "four";;
        man | kangaroo ) echo -n "two";;
        *) echo -n "an unknown number of";;
esac
echo " legs."
```


Autor: Sergi Tur Badenas

Estructures de control

- **Select**. Sintaxi similar a for:

```
select variable [in list]
 do
  command...
  break
 done
```

  - La construcció select permet generar fàcilment menús.

- Exemple:

```
$ joe selecciona.sh
#!/bin/bash
PS3='Choose your favorite vegetable: ' # Sets the prompt string.
                                       # Otherwise it defaults to #? .
select vegetable in "beans" "carrots" "potatoes" "onions" "rutabagas"
do
  echo
  echo "Your favorite veggie is $vegetable."
  echo
  break  # What happens if there is no 'break' here?
done
exit
```


Autor: Sergi Tur Badenas

Estructures de control

- PS3 és el PROMPT del menú select.
- Sortida de l'exemple anterior:

```
$ bash selecciona.sh
1) beans
2) carrots
3) potatoes
4) onions
5) rutabagas
Choose your favorite vegetable: 2
Your favorite veggie is carrots!
```

- Escollir un fitxer del directori de treball:

```
select fname in *;
 do
  echo you picked $fname \($REPLY\)
  break;
 done
```


Autor: Sergi Tur Badenas

Estructures de control

- **Bucles**
  - **for**. Sintaxi:

```
for name [in words ...];
 do
  commands ;
 done
```

  - Exemple:

```
$ for i in 1 2 3 4 5; do echo "Welcome $i times" ; done
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

  - Sintaxi a l'estil C:

```
for (( expr1 ; expr2 ; expr3 )) ;
 do
  commands ;
done
```


Autor: Sergi Tur Badenas

Estructures de control

- Exemple estil C:

```
$ for (( c=1; c<=5; c++ )); do echo
"Welcome $c times...";  done
Welcome 1 times...
Welcome 2 times...
Welcome 3 times...
Welcome 4 times...
Welcome 5 times...
```

- Bucle infinit:

```
#!/bin/bash
for (( ; ; ))
do
   echo "infinite loops [ hit CTRL+C to stop]"
done
```

- Es pot controlar el bucle amb break i continue:

```
for I in 1 2 3 4 5
done
  statements1
  statements2
  if (disaster-condition)
  then
        break
  fi
  statements3
done
```

```
for I in 1 2 3 4 5
done
  statements1
  statements2
  if (condition)
  then
       continue
  fi
  statements3
done
```


Autor: Sergi Tur Badenas

Estructures de control

- **Paràmetres posicionals i for:**
  - "in words" és opcional per què si no l'especifiqueu s'iteren els paràmetres posicionals:

```
$ joe for_pp.sh
for positional_parameter;
 do
   echo $positional_parameter;
 done
$ bash for_pp.sh 1 2 3
1
2
3
```

  - De fet és equivalent a:

```
for positional_parameter in
"$@";
 do
   echo
$positional_parameter;
 done
```


Autor: Sergi Tur Badenas

Estructures de control

- **while**. Sintaxi:

```
while test-commands ;
 do
  consequent-commands ;
 done
```

- Exemple:

```
#!/bin/bash
COUNTER=0
while [  $COUNTER -lt 10 ];
 do
   echo The counter is $COUNTER
   let COUNTER=COUNTER+1
 done
```

- Càlcul del factorial

```
#!/bin/bash
counter=$1
factorial=1
while [ $counter -gt 0 ]
do
   factorial=$(( $factorial * $counter ))
   counter=$(( $counter - 1 ))
done
echo $factorial
```


Autor: Sergi Tur Badenas

Estructures de control

- **until**. Sintaxi:

```
until test-commands ;
 do
  consequent-commands ;
 done
```

- L'exemple del comptador:

```
#!/bin/bash
COUNTER=20
until [  $COUNTER -lt 10 ];
 do
   echo COUNTER $COUNTER
   let COUNTER-=1
 done
```


Autor: Sergi Tur Badenas

Autor: Sergi Tur Badenas

Substitució d'ordres

- La substitució d'ordre permet que la sortida d'un ordre reemplaci a la pròpia ordre. Això succeïx quan s'utilitza:

```
$(ordre)
```

- L'anterior és la forma recomanada. També es pot utilitzar:

```
`ordre`
```

- Exemples:

```
$ date=$(date)
$ echo $date
dl mar 8 17:29:11 CET 2010

$ script_name=`basename $0`
$ echo "Nom:  $script_name."
```


Autor: Sergi Tur Badenas

Substitució d'ordres

- **Compte amb l'eliminació de salts de línia!**
  - Proveu:

```
$ dir_listing=`ls -l`
$ echo $dir_listing
```

  - Per tal que no apliqui “word splitting” cal utilitzar “quoting”:

```
$ dir_listing=`ls -l`
$ echo “$dir_listing”
```


Autor: Sergi Tur Badenas

Heredocs

- **Permeten les variables de text multilínia i literals**
  - El text EOF pot ser qualsevol (permet evitar conflictes amb textos que es vulguin utilitzar)

```
$ cat << EOF
Working dir $PWD
EOF
Working dir /home/user
```

  - Els espais i tabuladors a principis de línia es respecten. Es pot evitar aquest comportament amb **<<-**
  - **Permet l'execució d'ordres interactives (no executeu!):**

```
$ sudo fdisk /dev/sdd << EOF
n
p


W
EOF
```


Autor: Sergi Tur Badenas

Say yes! And run-parts

- **Hi ha un ordre per dir si a tot. Proveu:**
  - $ yes
  - Es pot utilitzar (en compte!) per contestar les típiques preguntes: segur que vols continuar (y/n)?
    - **$ yes | rm -r dirname**
- **run-parts**
  - Executa tots els guions d'un directori
    - $ run-parts directori_amb_guions
    - L'ordre és alfabètic. Es poden posar números davant dels scripts per controlar en quin ordre s'executen.

Autor: Sergi Tur Badenas

Seqüencies

- **Permet crear una seqüència de nombres enters**
  - El separador pe defecte és nova línia (es pot canviar amb -s)

```
$ seq 5
1
2
3
4
5
```

```
$ seq -s : 5
1:2:3:4:5
```

```
COUNT=80
#COUNT=$1
for a in $(seq $COUNT)
 do
  echo -n "$a "
 done
```


Autor: Sergi Tur Badenas

Obtenir informació de l'entrada. read

- **L'ordre interna read permet capturar dades de l'entrada estàndard.**
  - Sintaxi:

```
read [-ers] [-a aname] [-d delim] [-i text] [-n nchars] [-p
prompt] [-t timeout] [-u fd] [name ...]
```

  - Exemple bàsic:

```
$ read
Hola Mon!
$ echo $REPLY
Hola Mon!
```

  - A part de REPLY es pot guardar l'entrada a un altre variable

```
$ read HOLA
Hola Sergi!
$ echo $REPLY
Hola Mon!
$ echo $HOLA
Hola Sergi!
```


Autor: Sergi Tur Badenas

Obtenir informació de l'entrada. read

- Es poden llegir múltiples valors d'un sol cop amb:

```
$ joe hola.sh
#!/bin/bash
echo "Si us plau introdueix el teu nom i cognoms:"
read NOM COGNOMS
echo "Hola $COGNOMS, $NOM !"
```

  - El que fa realment es dividir l'entrada en paraules (separades per espais). Com que només especifiquem 2 variables i nosaltres especifiquem 2 paraules (nom + 2 cognoms) la segona variable conté tot el text fins al final de línia.
  - És dir el guió és equivalent a:

```
#!/bin/bash
echo "Si us plau introdueix el teu nom i cognoms:"
read NOM COGNOM1 COGNOM2
echo "Hola $COGNOM1 $COGNOM2, $NOM !"
```


Autor: Sergi Tur Badenas

Obtenir informació de l'entrada. read

- És molt utilitzat per a iterar un fitxer línia a línia

```
#!/bin/bash
cat filename | while read line;
do
  echo $line
done
```

- Es poden guardar els valors en un array

```
$ joe quedarbe.sh
echo "Quins són els teus
colors preferits?"
read -a colors
echo "Els meus colors
favorits també són: "
for i in "${colors[@]}"
 do
  echo $i
 done
echo ";-)"
```


Autor: Sergi Tur Badenas

Obtenir informació de l'entrada. read

- Altres opcions:
  - **-p prompt:** indica un prompt per cada cop que és demana una entrada.
  - **-d delim**: Especificar quin paràmetres s'utilitza per determinar que és una nova línia
  - **-n nchars**: read acaba un cop ha llegti n caràcters.
  - **-t timeout**: especificar un temps màxim d'espera d'una entrada.

Autor: Sergi Tur Badenas

Guions de l'interpret d'ordres

Guions de shell

- aka “shell scripts”. Fitxer de text que conté ordres de l'interpret.
- Quan el camí del fitxer que conté el guió s'utilitza com primer argument durant la invocació de bash:
  - **$ bash guio.sh**
- bash llegeix les ordres del fitxer, les executa i retorna l'execució a l'interpret d'ordres original.
- Aquest mode d'operació és anomenat **no interactiu**.
- Si no es proporciona el camí complet, primer es busca al directori de treball i aleshores es busca a les carpetes de $PATH.
- S'estableix el paràmetre 0 al nom del guió i els paràmetres posicionals passen a ser els paràmetres (si s'utilitzen) que es passen al guió.

Autor: Sergi Tur Badenas

Intèrprets. shebang

- **Permet indicar dins el guió quin és l'interpret**
  - El shebang és el conjunt de 2 caràcters #! que escrivim al principi d'un fitxer en entorns Unix/Linux.

```
#!/bin/bash
echo "Hola mon amb bash scripting!!!"
```

  - Script de Python (fitxer holamon.py):

```
#!/usr/bin/python
print "Hola mon amb python!!!"
```

  - Script de PHP (fitxer holamon.php):

```
#!/usr/bin/php5
<?php echo "Hola Mon!"?>
```


Autor: Sergi Tur Badenas

Intèrprets. shebang

- Si indiqueu el shebang els fitxer es poden executar de forma autònoma sí els feu executables:
  - $ chmod u+x nomFitxer
- Per executar-los:
  - ./nomFitxer
- Si no possessiu el shebang us caldria indicar quin és l'interpret que voleu utilitzar per executar l'script:
  - $ bash holamon.sh
  - $ python homamon.py
  - $ php5 holamon.php

Autor: Sergi Tur Badenas

Propietaris i permisos d'execució dels guions

- **Per executar un guió directament cal que sigui executable.**
  - **$ sudo chmod +x guio.sh**
  - Només si sou superusuari o teniu permisos de root
  - Només podreu canviar els permisos del fitxers que siguin de la vostra propietat amb:
    - **$ chmod +x guio.sh**
  - Si només voleu que el fitxer sigui executables per al vostre usuari:
    - **$ chmod u+x guio.sh**

Autor: Sergi Tur Badenas

Propietaris i permisos d'execució dels guions

- **Cal tenir en compte que els guions de shell s'executen en una subshell:**
  - $ guio.sh arguments
  - Equival a:
    - $ bash filename arguments
- **Es pot executar implícitament**
  - $ sh guio.sh
  - $ bash guio.sh
  - Cal tenir en compte el valor PATH:
    - $ echo $PATH
    - Normalment no inclou el directori de treball (PWD o .).
    - Slashdot (puntbarra) --> $ ./fitxer.sh

Autor: Sergi Tur Badenas

Bits especials de permisos

- **SUID (Set User ID)**
  - S'utilitza conjuntament amb **fitxers amb permisos d'execució**
  - Indica al sistema operatiu que el programa **s'ha d'executar amb els permisos del propietari del fitxer** i no pas amb els permisos de l'usuari que executa el fitxer.
  - Es pot utilitzar per tal d'**executar fitxers com a superusuari sense ser root**. El fitxer ha de pertànyer al superusuari, ser executable i tenir el bit especial d'execució SUID.
  - Aquesta opció es força utilitzada en alguns serveis i programes
  - Els programes que s'executen d'aquesta forma són anomenats SUID root.
  - Els fitxers amb aquest permís s'indiquen amb una **s al bit d'execució del propietari.** (o S majúscula si no és executable)

Autor: Sergi Tur Badenas

Bits especials de permisos

- **SUID exemple**
  - Per exemple, l'ordre ping en alguns sistemes com Ubuntu:

```
# ls -l /bin/ping
-rwsr-xr-x 1 root root 30856 2007-12-10 18:33 /bin/ping
```

  - La creació de paquets ICMP (protocol de ping) requereix accés de superusuari.
  - Activar el SUID en un directori no té cap sentit ni utilitat (s'ignora)
  - Per seguretat, només funciona amb fitxers executables binaris (no guions de llenguatges interpretats)

Autor: Sergi Tur Badenas

Bits especials de permisos

- **SGID (Set Group ID)**
  - És similar a SUID però estableix que el grup del programa executable és el grup del fitxer i no pas el grup de l'usuari que executa el fitxer.
  - S'indica amb una **s al bit d'execució del grup**.
  - **Directoris:** Quan el bit SGID s'estableix en un directori, els fitxers nous o directoris creats en aquest directori heretaran el grup del directori i no pas el grup de l'usuari que crea el directori o fitxer.

IMPORTANT: Tant el SUID com el GUID són bits potencialment insegurs. Només s'utilitzen quan no hi ha un altre remei. Cal que els programes que els utilitzin estiguin ben programats per tal d'evitar escalades de privilegis

Autor: Sergi Tur Badenas

Paràmetres posicionals

- **Al executar un guió tenim disponibles els següents paràmetres:**
  - Paràmetre especial 0 ($0): conté el nom del guió (normalment conté el nom de l'interpret)
  - Paràmetres posicionals:
    - $ ./guio.sh parametre1 parametre2 parametre3 parametre4...
    - És equivalent a:
    - $ bash guio.sh parametre1 parametre2 parametre3 parametre4...
    - Per referir-se al paràmetre N (1<=N < n) utilitzem: **${N} o $N**
    - Si n és major o igual de 10 --> obligatori ${10}

Autor: Sergi Tur Badenas

script

- **La sintaxi segons el manual és:**
  - $ man script
    - script [-a] [-c COMMAND] [-f] [-q] [-t] [file
  - Un cop iniciada guarda en un fitxer tot el que s'executa a la terminal. És útil per tal que els alumnes puguin guardar una còpia de les tasques que realitzen a la terminal segons uns deures que se'ls ha demanat.
  - Abans de començar cal executar script (iniciar una sessió). Teniu dos opcions:
    - $ script o $ script sessio.txt
  - Si no especifiqueu cap fitxer la sessió es guarda al fitxer typescript.

Autor: Sergi Tur Badenas

script

- Feu una prova:

```
$ script sessio.txt
S'ha iniciat l'execució de script, el fitxer és sessio.txt
$ ls
$ echo "prova"
```

- Per finalitzar una sessió cal prémer la combinació de tecles:

```
Ctrl+D
S'ha fet la seqüència, el fitxer és sessio.txt
```

  - Consulteu el fitxer: $ cat sessio.txt
  - El fitxer es pot anar creant a l'estil d'un fitxer de log (-f following) i es pot reaprofitar un fitxer (-a append):
    - $ script -f -a sessio.txt
    - $ tail -f sessio.txt

Autor: Sergi Tur Badenas

mail

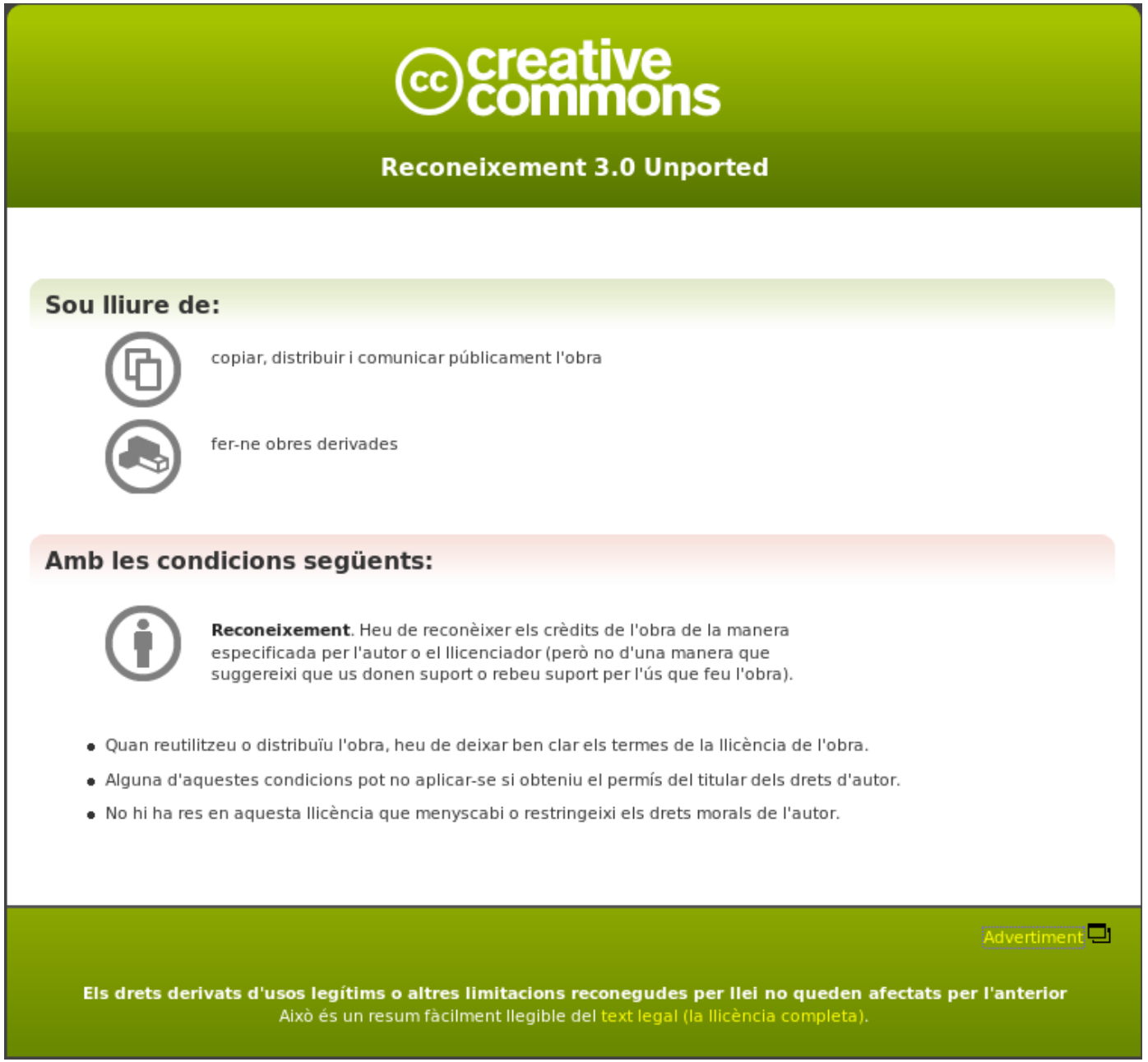
- mail és una ordre de línia de comandes que proporciona un client de correu electrònic bàsic per a sistemes Unix i GNU/Linux
- Si no indiqueu cap opció, aleshores mail el que fa és llegir el correu pendent de l'usuari:

```
$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/sergi": 1 message 1 unread
>U  1 sergi@portatil     Fri Apr  2 16:30   16/454
Prova
&
```

- Mail a la wiki del curs

Autor: Sergi Tur Badenas

Llicència Creative Commons



http://creativecommons.org/licenses/by/3.0/deed.ca

Autor: Sergi Tur Badenas