Fonaments dels Sistemes Operatius (FSO)

Departament d'Informàtica de Sistemes i Computadors (DISCA)
Universitat Politècnica de València

Bloc temàtic 1: Introducció

SUT02: El shell de UNIX

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

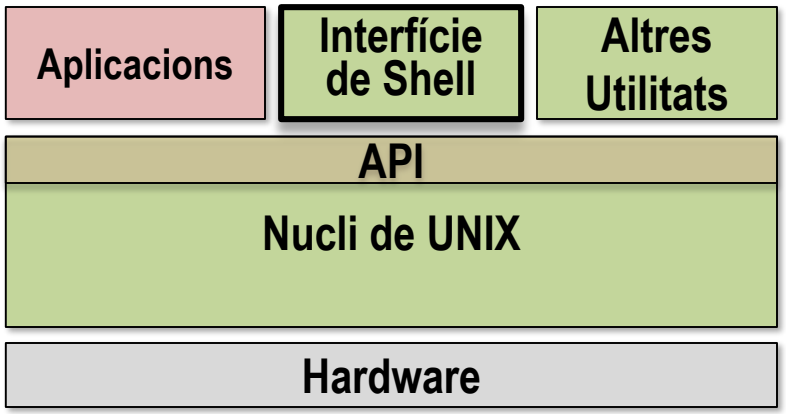
- **Objectius:**
  - Explicar la **utilitat i funcionament del shell** de UNIX
  - Descriure l'**estructura bàsica del sistema d'arxius en UNIX**
  - Exposar el **concepte de ruta (*path*)** de accés a arxius
  - Descriure **les variables i ordres del shell** mes utilitzats

- **Bibliografia clàssica**
  - Kernighan, Pike: The UNIX programming environment/El entorno de programación UNIX, Prentice Hall
- Accessible en Internet
  - Mark Burgess: The UNIX programming environment http://www.iu.hio.no/~mark/unix/unix_toc.html
  - Fernando Bellas: El entorno de programación UNIX http://www.tic.udc.es/~fbellas/teaching/unix/index.html

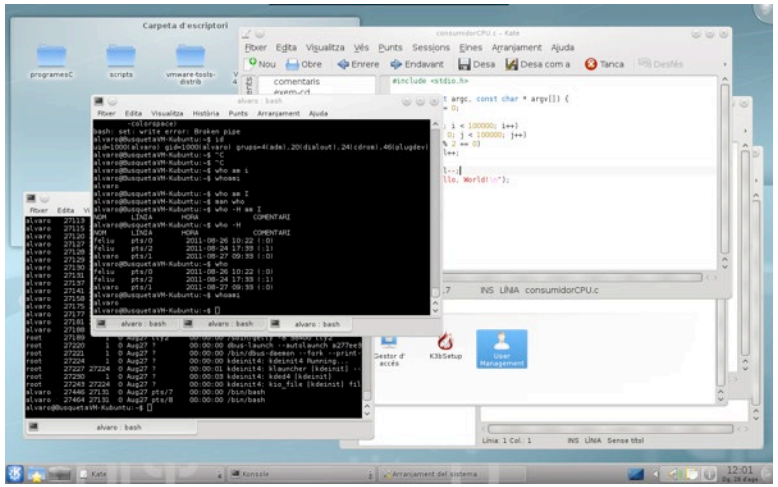
Contingut

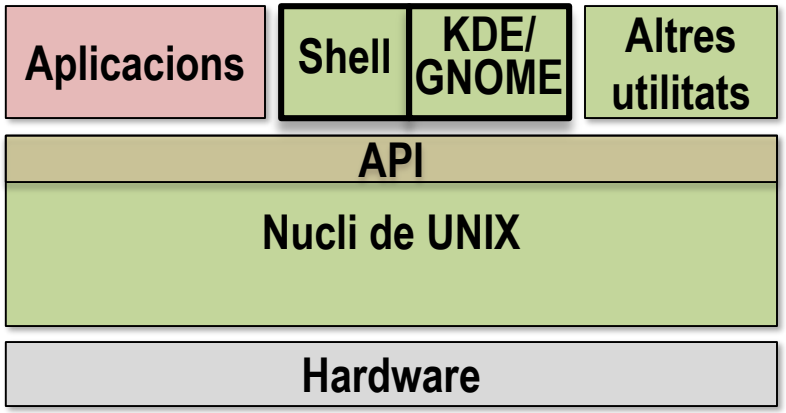
- **Introducció**
- Identitats en UNIX
- Variables del Shell
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- Maneig dels processos
- Programació amb el Shell

- **Les interfícies d'usuari (interfícies de text o gràfiques)**
  - Necessàries per a executar els programes, organitzar els arxius, administrar el sistema i altres
  - **Programa** que **tradueix les accions de l'usuari en crides al nucli de UNIX, fent servir la seua interfície als programes (API)**, i en mostra els resultats
  - **Interfície de text clàssica**: consoles formades per un teclat i un monitor en mode text
    - Dins de UNIX: dispositius *tty0*, *tty1,…*
    - Llenguatge: el **shell** de UNIX



- **Interfícies gràfiques de UNIX o GUI**
  - *X Window* (1984): *X11* o simplement *X*
    - Implementacions:
      - *XFree86*, *X.Org* (>= 2004)
      - *X Quartz*
      - *Cygwin/X*, *mobaXterm*, *Xming*, … *FreeNX*
      - *Android X Server*
  - Està pensada per a un monitor en mode gràfic, teclat i ratolí
    - **Emulen una o més consoles** de text dins d'una finestra com a pseudoterminals (dispositus pts/0, pts/1, etc)
    - Distribuïda: es possible connectar-se de de màquines diferents a la que iniciem
  - KDE, GNOME i altres **ofereixen les ferramentes d'ús i administració** mitjançant la GUI





Nomenclatura:
KDE: powerful graphical desktop environment for Unix workstations
GNOME: GNU Object Model Environment

- **Shell de UNIX**
  - **Intèrpret d'ordres: programa que llig línies, les interpreta i les executa**
  - La sintaxi admesa per un shell permet especificar seqüències d'ordres a la manera d'un llenguatge de programació
    - té comentaris, variables, control de flux, funcions (amb arguments i valors retornats), etc
    - ***Shell script***: el shell pot processar arxius de text formats per ordres
  - Hi ha diversos shells (pareguts però no iguals): cshell, kshell,…
    - anem a estudiar el ***bash* o *Bourne Again Shell***
  - El shell de UNIX permet:
    - gestionar el sistema d'arxius
    - executar programes
    - adreçar l'entrada/eixida dels programes
    - mantenir variables del sistema
    - monitoritzar i gestionar els processos

- **Diàleg del shell**

  repetir

    escriure el *prompt*

    llegir la línia escrita per l'usuari

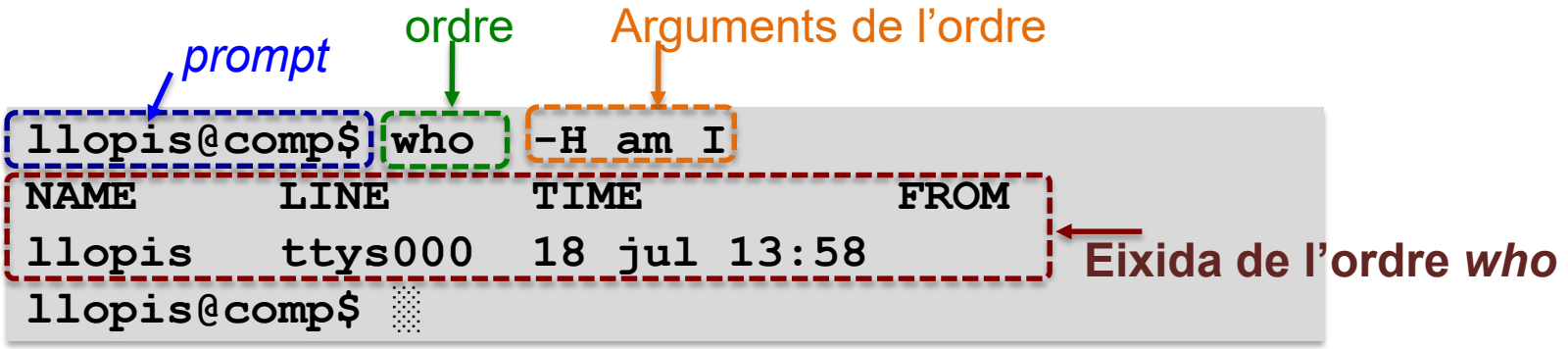
    processar la línia (desenvolupant metacaràcters)

    identificar l'ordre (el primer token de la línia)

    executar l'ordre transferint-li els arguments (la resta de tokens)

    rebre el codi de terminació del programa

  fins (final de l'arxiu d'entrada)



Metacaràcters: `$ * ? \ > < | & [ ]`

- **Ordres internes i externes**
  - El *shell* tracta dos tipus d'ordres:
    - Ordres **internes** o *built-in commands*:
      - El *shell* inclou internament el codi que cal i fa el treball
    - Ordres **externes**: el *shell* busca un programa en el disc i l'executa
      - Arxius amb codi binari (executable), p.ex. C i compilats
      - *Shell scripts*: i altres *scripts (PHP, Python, Perl,...)*
- **Ajuda en línia: l'ordre man**
    - Descripció d'una ordre concreta **`man ordre`**
    - Descripció del *shell*: **`man bash`**

```
LS(1)                    BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLPRTWZabcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory,
```

Contingut

- Introducció
- **Identitats en UNIX**
- Les variables del Shell
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- Maneig dels processos
- Programació amb el Shell

- **Usuaris i grups:** són la base de l'esquema de protecció de UNIX
  - La seua definició és una de les tasques administratives més importants de UNIX
  - Un **usuari** és una identitat coneguda pel sistema que té contrasenya (*password*), privilegis, directori *home*, etc
    - cada usuari té un nom (USERNAME) i un número (UID)
    - hi ha usuaris de sistema: *daemon*, *mail* i altres
    - *root* és un usuari predefinit *que* té els màxims privilegis
  - Un **grup** és un conjunt d'usuaris
    - cada grup té un nom i un número (GID)
    - hi ha grups predefinits: *adm*, *man*, etc
    - cada usuari està necessàriament en un grup primari

Nomenclatura
UID - User Identification
GID - Group Identification

- Ordres

Ordre	Descripció de la utilitat
`id`	mostra la identitat i els grups a què pertany l'usuari
`su`	canvia d'usuari
`who`	llista usuaris connectats actualment

```
…$ id
uid=1001(feliu) gid=1001(fso)
grups=4(adm),20(dialout),24(cdrom),…
…$ su llopis
Contrasenya:
```

Contingut

- Introducció
- Identitats en UNIX
- **Les variables del Shell**
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- Maneig dels processos
- Programació amb el Shell

Variables del shell

- Què són les variables del *shell*?
  - Són símbols que tenen assignat una cadena se caràcters
    - En la línia d'ordres: un símbol de la forma

      $*nom_de_variable* és substituit pel valor de la variable
  - = defineix i assigna una variable en la línia d'ordres

    Nombre_de_variable=valor *(Important: sense espais)*

```
…$ ElMeuNom=Feliu
…$ echo ElMeuNom
ElMeuNom
…$ echo $ElMeuNom
Feliu
…$
```

- L'entorn (*environment)*
  - El conjunt de variables accesibels des de programes esxterns es diu **entorn** (*environment*). Una variable d'entorn es defineix amb l'ordre **export**
  - **env** ordre que llista el valor de totes les variables de l'entorn

```
…$ env
TERM=xterm
SHELL=/bin/bash
USER=sterrasa
```

Variables del shell

- Ordres per a visualitzar variables del *shell* i definir variables

Ordre	Descripció d'utilitat
`=`	Crea una variable i li assigna valor (nom=valor)
`export`	Marca una variable com a variable d'entorn
`env`	Llista el valor de totes les variables de l'entorn
`echo`	Mostra una cadena de caracteres per la sortida estàndar

- Algunes variables del shell

Variables Shell	Descripció d'utilitat	Ordre mostrar valor
PATH	Conté la ruta de recerca de programes	`echo $PATH`
HOME	Conté ruta del directori de l'usuari	`echo $HOME`
HOSTNAME	Nom del computador en la xarxa	`echo $HOSTNAME`
PS1	Aspecte del *prompt*	`echo $PS1`
PWD	Directori actual de treball	`echo $PWD`
SHELL	Intérpret d'ordres per defecte	`echo $SHELL`
TERM	Tipus de terminal	`echo $TERM`

Variables del shell

- **La variable PATH**
  - Conté la llista de directoris on el *shell* busca els programes i scripts en processar la línia d'ordres:
    - els noms de directori estan separats per ":"
    - en buscar els programes, l'intèrpret fa recorregut per ordre

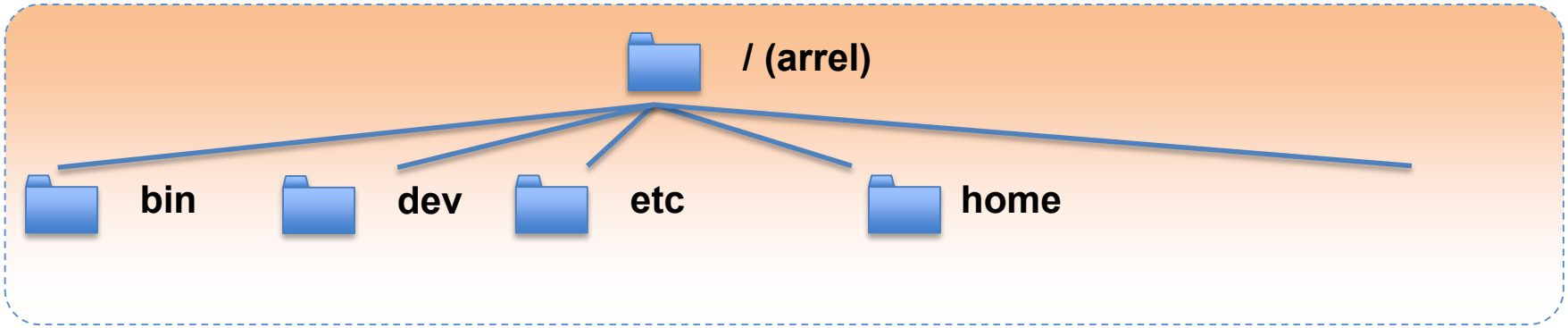
```
…$ echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin
…$ export PATH=$HOME/proves:$PATH
…$ echo $PATH
/Users/llopis/proves:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin
…$
```

Contingut

- Introducció
- Identitats en UNIX
- Les variables del Shell
- **Navegació pel sistema d'arxius**
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- Maneig dels processos
- Programació amb el Shell

Navegació pel sistema d'arxius

- Esquema del sistema d'arxius UNIX



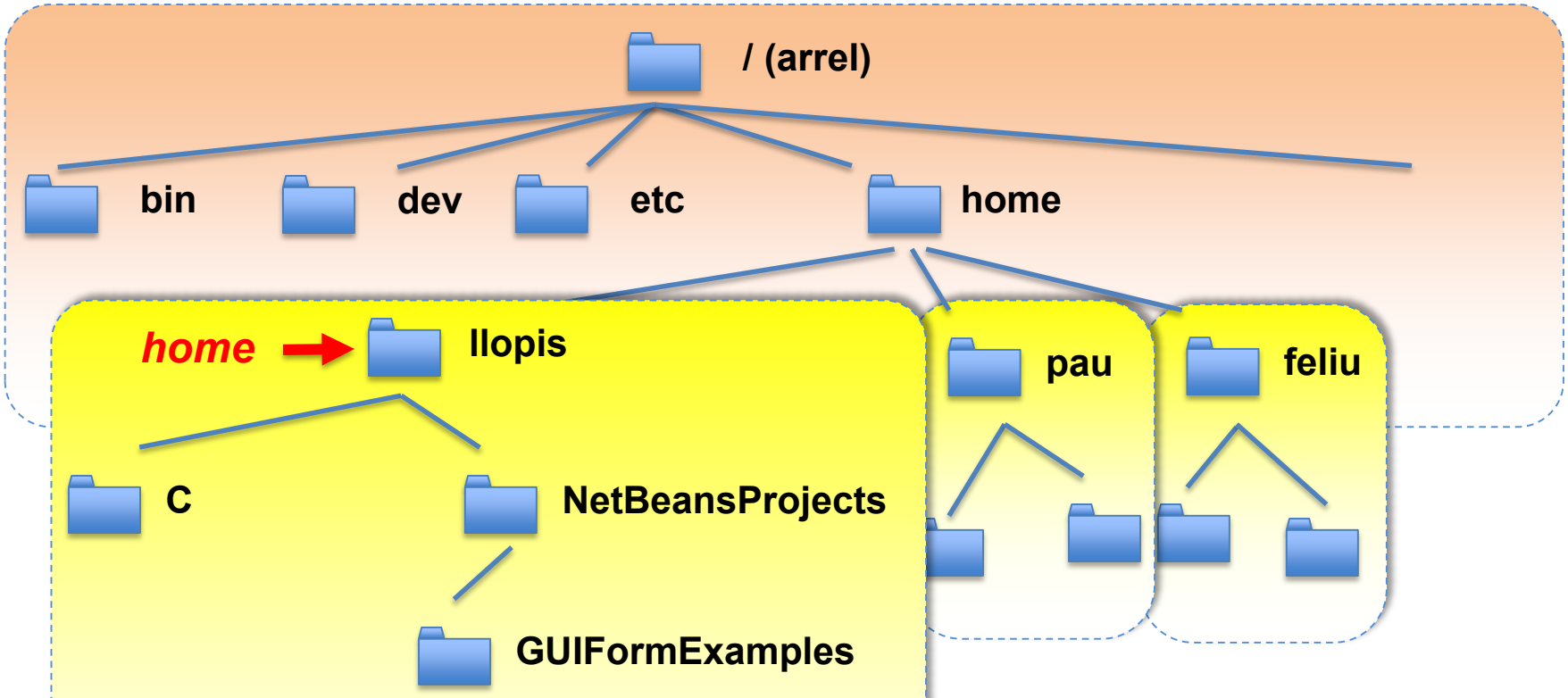
- Directoris d'ús general
  - Contenen els programes i les dades útils per al sistema i per a tots els usuaris
- Directori "."
  - Fa referència al directori actual de treball
- Directori ".."
  - Fa referència al directori "pare"

alguns exemples

directori	ús
/bin	ordres comuns
/dev	dispositius d'entrada/eixida
/etc	informació d'administració
/usr	aplicacions

Navegació pel sistema d'arxius

- **Esquema del sistema d'arxius UNIX (2)**



Cada usuari té un arbre de directoris propi que creix a partir del seu directori *home*. Habitualment, el directori *home* té el mateix nom que l'usuari

NOTA: En alguns sistemes UNIX el directori "home" pot estar en `/Users` **ó** `/user`

Navegació pel sistema d'arxius

- **Noms d'arxiu i directori en UNIX**
  - Els directoris formen un arbre (invertit) crescut des de l'arrel
  - El nom complet (camí o *path*) d'un arxiu o directori conté el nom de tots els directoris superiors
    - El separador és el caràcter "/"
    - El **directori arrel** és "/"
  - **Nom absolut** comença des de l'arrel del sistema d'arxius
    - Exemple: "/users/llopis/eclipse/p1/main.java"
  - **Nom relatiu** comença des d'un directori de treball.
    Casos especials:
    - "**.**" és el mateix directori
    - "**..**" és el directori **pare** del directori de treball
    - Exemples des de "/users/llopis/eclipse":
      - "p1/main.java"
      - ".." és el directori "/users/llopis"
      - "." és el directori "/users/llopis/eclipse"

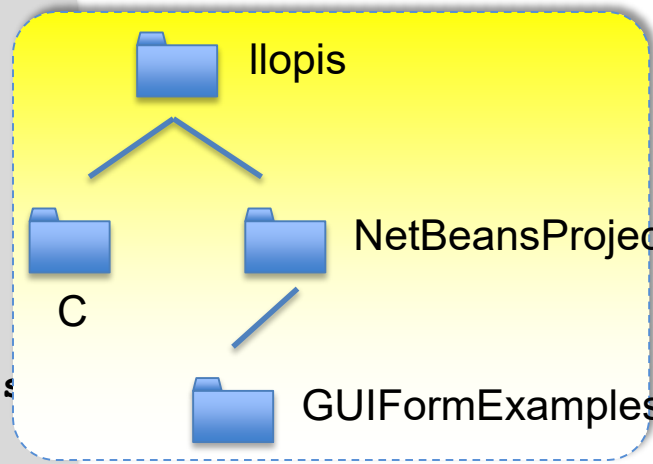
Navegació pel sistema d'arxius

- El directori de treball

Ordre	Arguments	Utilitat
`cd`	*directori*	Fixa el directori de treball
`pwd`		Mostra el directori de treball actual

  - ordre *cd [dir]*: canvia el directori de treball a *dir*
    - sense arguments: canvia el directori de treball al directori *home*
  - ordre *pwd*: mostra el camí absolut del directori de treball
  - al principi d'una sessió, el directori de treball és *home*

```
…$ pwd
/Users/llopis
…$ cd NetBeansProjects
…$ pwd
/Users/llopis/NetBeansProjects
…$ cd GUIFormExamples
…$ pwd
/Users/llopis/NetBeansProjects/GUIFormExamples
…$ cd
…$ pwd
/Users/llopis
…$
```



Contingut

- Introducció
- Identitats en UNIX
- Les variables del Shell
- Navegació pel sistema d'arxius
- **Maneig del sistema d'arxius**
- Maneig de l'entrada/eixida
- Maneig dels processos
- Programació amb el Shell

Maneig del sistema d'arxius

- **Ordres del shell per al maneig del sistema d'arxius**

Ordre	Opcions	Arguments	Utilitat
`cat`		*arxiu(s)*	bolca el contingut del(s) arxiu(s)
`ls`	`-l -a -d`	*arxiu(s)*	llista informació variada sobre l'arxiu
`file`		*arxiu(s)*	informa del tipus d'arxiu
`rm`	`-R`	*arxiu(s)*	elimina enllaços o arxius
`mkdir`		*directori*	crea un directori
`rmdir`		*directori*	elimina un directori
`mv`		*arxiu arxiu* *arxiu(s) dir*	reanomena/reubica arxius
`cp`	`-r`	*arxiu arxiu* *arxiu(s) dir*	còpia arxiu(s)
`ln`	`-s`	*arxiu arxiu* *arxiu dir*	crea nous enllaços a un arxiu existent
`chown`	`-R`	*usuari arxiu(s)*	fixa el propietari
`chmod`	`-R`	*mode arxiu(s)*	fixa permisos

- more, less, locate, which, find, df, du, …

Maneig del sistema d'arxius

- **L'ordre `cat`**
  - Imprimeix per pantalla el contingut d'un arxiu (siga o no de text)
  - No aplicable a directoris
- **L'ordre `ls`**
  - Llista noms i altres atributs dels arxius:

    Ús: *ls [opcions] [noms...]*
    - opcions: detalls que dóna de cada arxiu

      `-l` (de *long*): permisos, propietari, grup, …

      `-d` dóna informació dels directoris

      - **a** mostra tots els noms d'arxiu incloent els amagats
    - noms: de directoris o d'arxius

      si el nom és un directori, llista els arxius que conté

      (a no ser que s'utilitze l'opció –d)

      sense noms: llista el directori de treball
  - Tractament especial de **noms amagats**
    - Els arxius els noms dels quals comecen per punt (.) est´n amagats i no apareixen en el llistat (si apareixen amb l'opció`-a`)

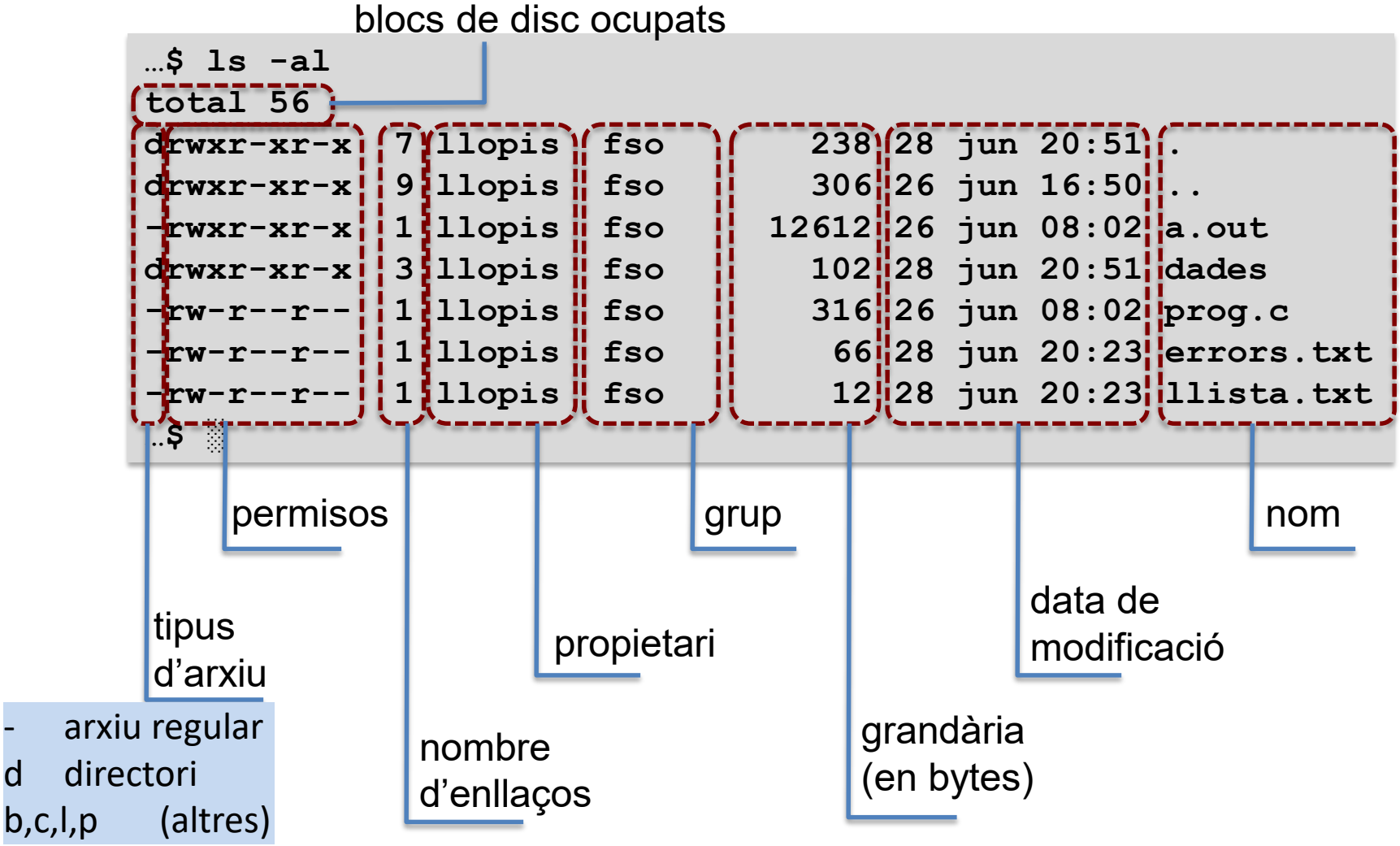
Maneig del sistema d'arxius

- **Comodins**
  - Són caracters que substitueixen a cadenes de caracters.
    - Serveixen per a nomenar diversos fitxers sense especificar-ho
  - Quan un token conté un comodí, el shell fa *pattern matching* amb les entrades d'un directori i substitueix el token per tots els noms que s'ajusten al patró
  - Exemples
    - **'*'** es correspon amb qualsevol substring
    - **'?'** es correspon amb qualsevol caràcter (només un)

```
…$ ls
salida.txt ejemplo.txt param param.c Param algo.
…$ ls [Pp]*
Param param param.c
…$ ls *.*
algo. salida.txt ejemplo.txt param.
…$ ls *.txt
salida.txt ejemplo.txt
…$ ls p*
param param.c
```

Maneig del sistema d'arxius

- El format llarg de l'ordre **`ls`** (opció **`-l`**)



Maneig del sistema d'arxius

- **Arxius: Propietaris i permisos**
  - Un arxiu és propietat d'un usuari i d'un grup
    - Inicialment, els propietaris són l'usuari (UID) que l'ha creat i el grup primari (GID) a què pertany
    - L'ordre **`chown`** (*change owner*) permet canviar els propietaris
  - Tres **permisos** independents**: r (*read*), w (*write*), x (*execute*)**

permís	arxius regulars	directoris
`r`	l'arxiu es pot llegir	el directori es pot llistar amb **`ls`** només noms
`w`	l'arxiu es pot escriure	s'hi poden eliminar o afegir arxius (necessita x)
`x`	l'arxiu és pot executar (si és binari o un script)	(junt amb el permís r) el directori es pot llistar amb **`ls -l`** i pot ser directori de treball

- Tres àmbits de **propietat**: ***user*, *group* i *other***
- Aplicació dels permisos en l'accés de l'usuari *U* a l'arxiu o directori *X*:

si (U es *root*) *té permís*
sino si (*U* és el propietari de *X*) aplica permisos *user*
sino si (*U* està en el grup propietari de *X*) aplica permisos *group*
sino aplica permisos *other*

Maneig del sistema d'arxius

- Gestió dels permisos amb **`chmod`** (*change mode*)
  - Són 9 bits
    - 3 àmbits (user, group, other) x 3 permisos
  - Mode octal o numèric:
    - Tres dígits. Cada dígit codifica els tres permisos d'un àmbit amb valors del 0 al 7
    - Exemple: **`chmod 640 nom`**
  - Mode simbòlic
    - Modifica permisos un a un
    - Exemples:

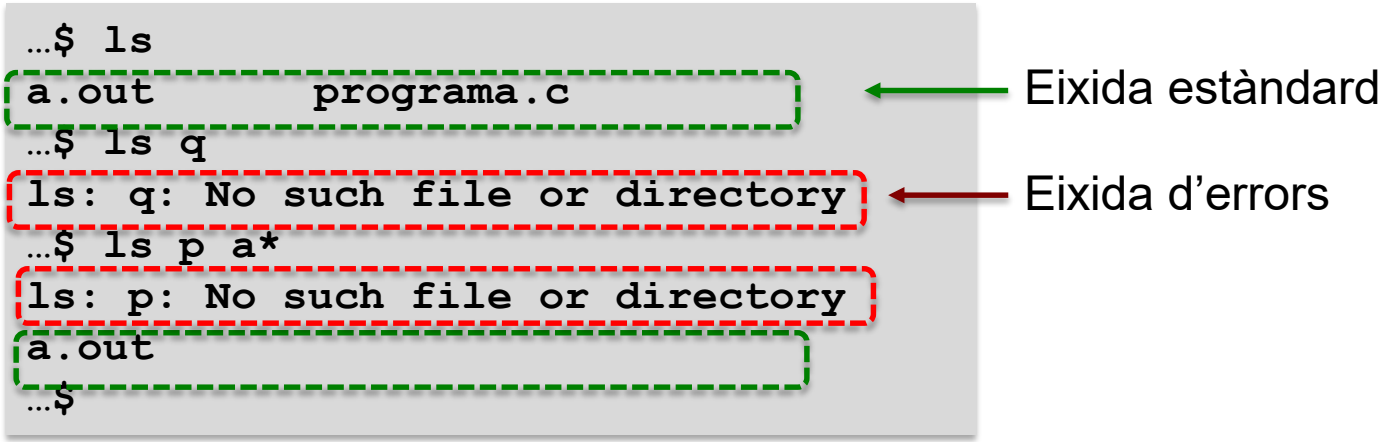
```
chmod u+w nom
chmod +x nom
chmod a+r,o-w nom
chmod ug=rw,o= nom
```

a qui	operació	permís
u (user) g (group) o (other) a (all)	+ (donar) – (llevar) = (fixar)	r w x

Contingut

- Introducció
- Identitats en UNIX
- Les variables del Shell
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- **Maneig de l'entrada/eixida**
- Maneig dels processos
- Programació amb el Shell

- **Dispositius estàndard**
  - El shell maneja tres canals de caracters, cadascun dels quals pot estar assignat a un dispositiu: la consola, un arxiu o altres
    - L'entrada estàndard ***stdin***
    - L'eixida estàndard ***stdout***
    - L'eixida d'errors estàndard ***stderr***
  - Per omissió, en una sessió interactiva, tots tres canals estan assignats a la consola
  - El shell escriu el prompt en *stdout* i llig la línia d'ordres de *stdin*
  - Els programes poden llegir de *stdin* i escriure en *stdout* i *stderr*
  - Exemple (ls):



- **Redireccionament**
  - Assignació de les entrades i eixides estàndard a un dispositiu concret (típicament un arxiu)
  - Algunes possibilitats des de la línia d'ordres:

forma	efecte
< dispositiu	redirecciona *stdin* al dispositiu
> dispositivo	redirecciona *stdout* al dispositiu (creant-lo nou si ja existeix)
>> dispositivo	redirecciona *stdout* al dispositiu (escrivint al final si ya existeix)
2> dispositivo	redirecciona *stderr* al dispositiu (creant-lo nou si ja existeix)
2>&1	redirecciona *stderr* al dispositiu associat a *stdout*

```
…$ ls p a*
ls: p: No such file or directory
a.out
…$ ls q a* > llista.txt 2> errors.txt
…$ cat llista.txt
a.out
…$ cat errors.txt
ls: q: No such file or directory
…$
```

stdout redireccionada a *llista.txt*

stderr redireccionada a *errors.txt*

Maneig de l'entrada/eixida

- **Filtres**
  - Ordes que lligen de *stdin*, fan operacions senzilles i escriuen el resultat en *stdout*

ordre	opcions	Arguments	Descripció
`head`	`-n lin`		transcriu les *n* primeres línies llegides
`tail`	`-n lin`		transcriu les *n* darreres línies llegides
`sort`			ordena les línies de text llegides i escriu el resultat
`tee`		*arxiu*	transcriu l'entrada en l'eixida i en fa còpia en un arxiu
`wc`	`-l -w -c`		compta línies, paraules i caràcters llegits i escriu l'estadística
`grep`		*regexp*	transcriu les línies que satisfan una expresió regular
`awk`			processa arxius buscant patrons
`cut`	`-f -d`	*regexp*	Selecciona components de cada línia que processa
`sed`		*Script arxiu*	Editor de fluix de caracters

- *"regexp" (e*xpressiones regulars) :descriuen un conjunt de cadenes que contenen un patró. Usa metacaracteres com \ ^ $. [ ] { } | ( ) * + ?
- Poden encadenar-se: en seqüència amb';' o en connexió amb'|'

Maneig de l'entrada/eixida

- Filtres (exemples)

```
…$ cat entrada
one
two
three
four
five
…$
```

```
…$ head -n 3 <entrada
one
two
three
…$ tail -n 4 <entrada
two
three
four
five
…$ wc <entrada
       5       5      24
…$
```

```
…$ grep fi <entrada
five
…$ grep t <entrada
two
three
…$ sort <entrada
five
four
one
three
two
…$
```

```
…$ head -n 3 < entrada; grep fi < entrada
…$ grep fi <entrada; echo "Hay`wc -l entrada` coincidencia/s"
```

Maneig de l'entrada/eixida

- **Tuberies**
  - Connecten l'eixida estàndard d'una ordre a l'entrada estàndard de l'ordre següent

```
…$ sort <entrada | head -3
five
four
one
…$ sort <entrada | tail -3 >eixida
…$ cat eixida
one
three
two
```

```
…$ ls -l
total 80
-rwxr-xr-x  1 feliu   fso     12612  3 jul 08:41 a.out
drwxr-xr-x  3 feliu   fso       102 13 ago 22:34 e-s
-rw-r--r--  1 feliu   fso         0  3 jul 08:24 eixida.txt
-rw-r--r--  1 feliu   fso        37  3 jul 08:24 exemple.txt
-rwxr-xr-x  1 feliu   fso     12612  3 jul 08:47 param
…$ ls -l | tail +2 | head -3
-rwxr-xr-x  1 feliu   fso     12612  3 jul 08:41 a.out
drwxr-xr-x  3 feliu   fso       102 13 ago 22:34 e-s
-rw-r--r--  1 feliu   fso         0  3 jul 08:24 eixida.txt
```

```
…$ df | sort -rnH|head -1
…$ ps -ax | grep firefox | cut -f 1 -d " "
```

Contingut

- Introducció
- Identitats en UNIX
- Les variables del Shell
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- **Maneig dels processos**
- Programació amb el Shell

Maneig dels processos

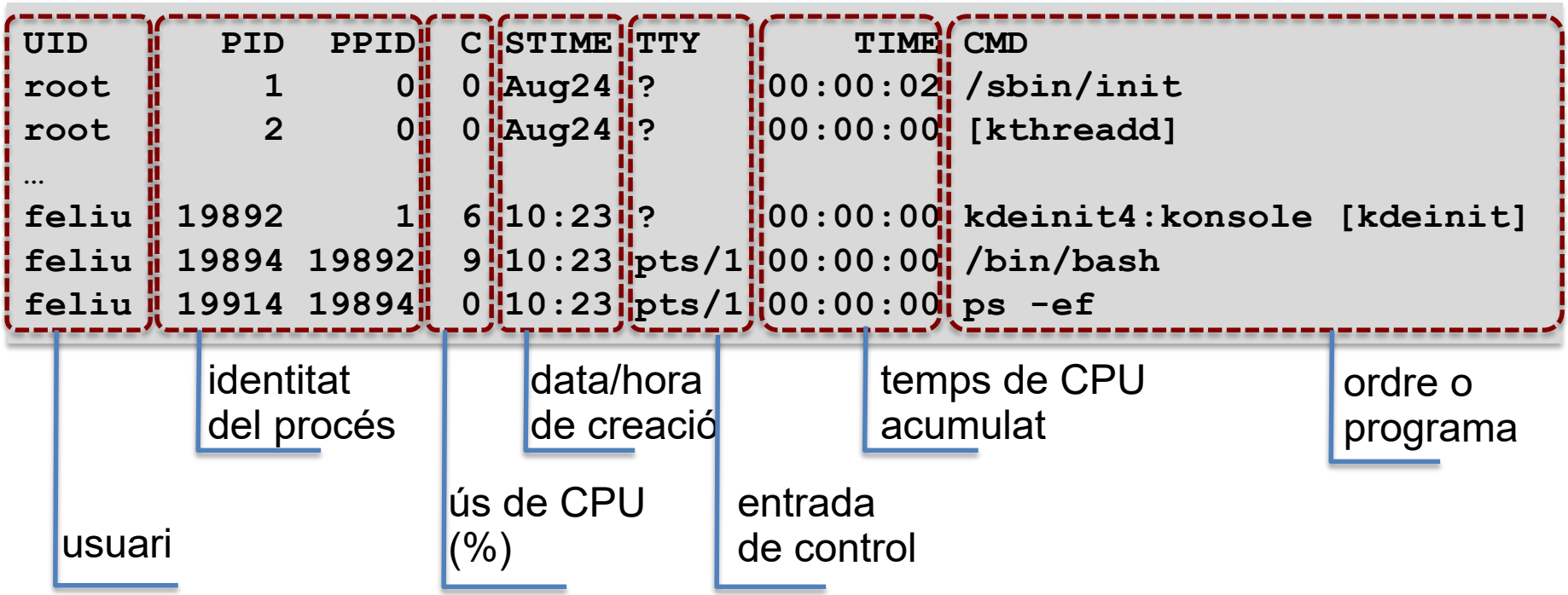
- **Processos en UNIX**
  - S'identifiquen pel seu **PID (*process identifier*)**
    - el shell mostra el seu propi PID amb `echo $$`
  - Cada procés està associat a un usuari amb UID donat
  - El conjunt de processos té estructura d'arbre
    - cadascun té un **procés pare** definit pel seu **PPID (*parent process identifier*)**
    - cadascun pot crear un **procés fill** (*child process*) o més
- Ordres:

ordres	opcions útils	arguments	utilitat
`ps`	`-e -f a f`	*pid(s)*	llista informació dels processos
`kill`	`-s -n`	*senyal pid(s)*	envia un senyal als processos
`sleep`		*temps*	suspen l'execució de l'intèrpret
`pstree`			mostra l'arbre de processos
`top` `htop`			mostra estadístiques dels processos en temps real

Maneig dels processos

- L'ordre **`ps -ef`**

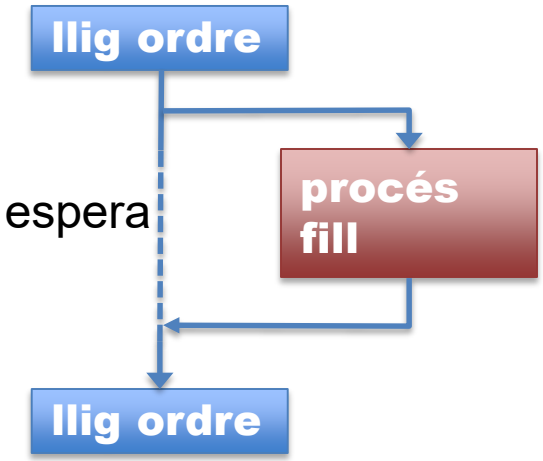
```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Aug24 ?        00:00:02 /sbin/init
root         2     0  0 Aug24 ?        00:00:00 [kthreadd]
…
feliu    19892     1  6 10:23 ?        00:00:00 kdeinit4:konsole [kdeinit]
feliu    19894 19892  9 10:23 pts/1    00:00:00 /bin/bash
feliu    19914 19894  0 10:23 pts/1    00:00:00 ps -ef
```

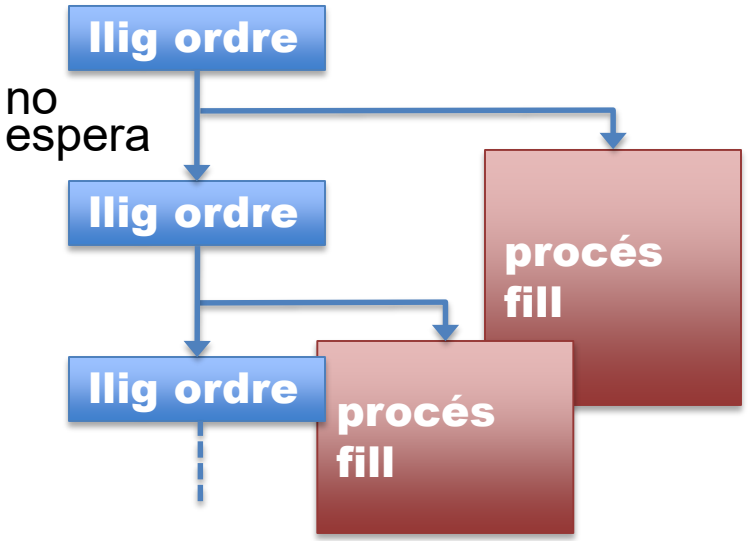


  - Mostra la relació pare➔fill entre els processos
    - arranque (PID 0) ➔ init (PID 1), kthread (PID 2)
    - init (PID 1) ➔ kdeinit (PID 19892) ➔ bash (PID 19894) ➔ ps -ef (PID 19914)

Maneig dels processos

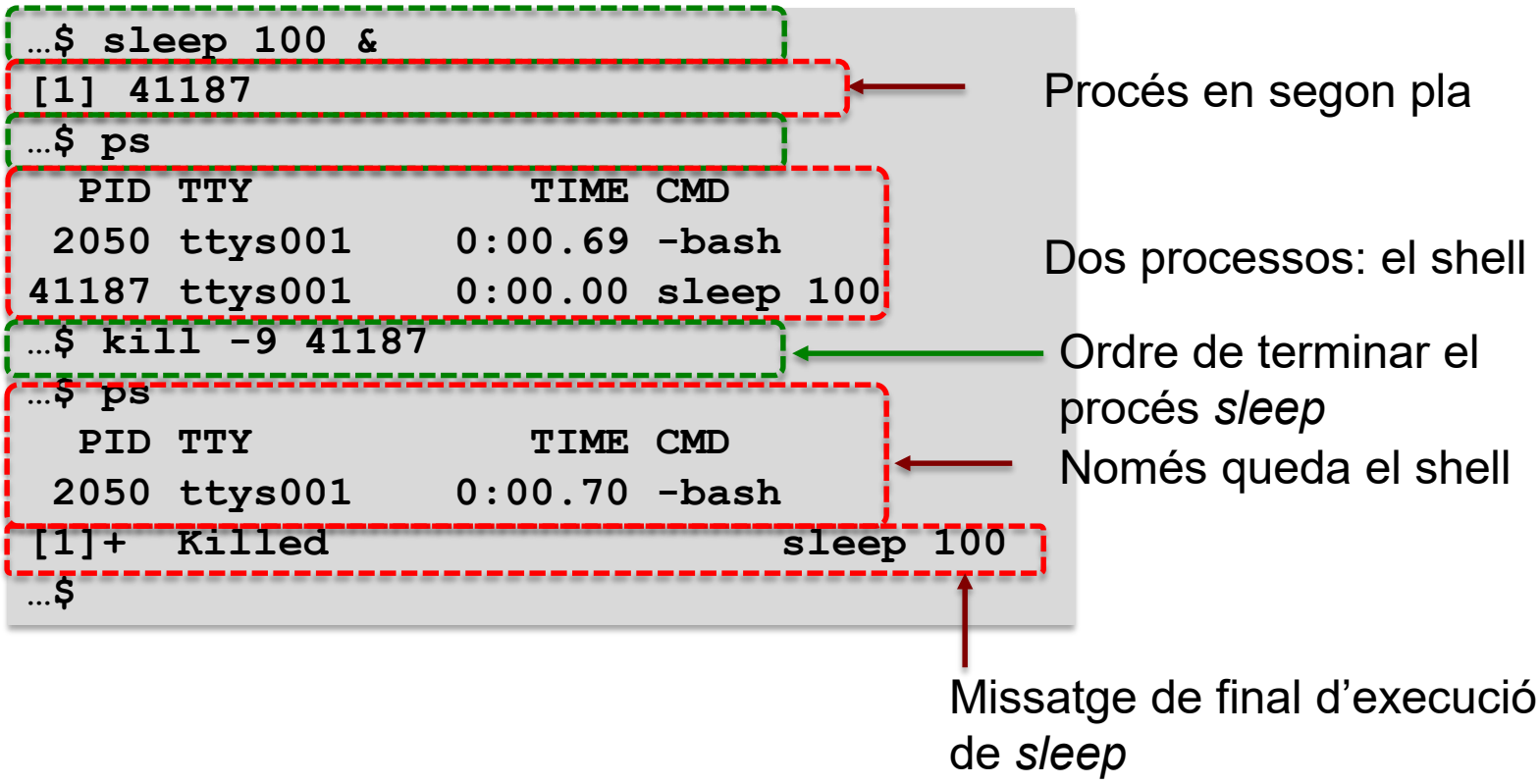
- **Processos en primer (interactiu o *foreground process)*r i en segon pla (*background process)***
  - El *SO* crea un nou procés (procés fill del *shell*) per a executar una ordre externa
  - Processos en **primer pla** (per omissió): el *shell* espera que acabe el procés fill abans de continuar la seua execució mostrant el *prompt*
    - El procés es pot aturar amb ctrl-C (^C)
  - Procés en **segon pla**: el *shell* i el procés fill continuen en concurrència
    - ***ordre &:*** ordre s'executa en *background*
    - **`kill -9 PID:`** El procés amb identificació PID finalitza





Maneig dels processos

- Exemples



Contingut

- Introducció
- Identitats en UNIX
- Les variables del Shell
- Navegació pel sistema d'arxius
- Maneig del sistema d'arxius
- Maneig de l'entrada/eixida
- Maneig dels processos
- **Programació amb el Shell**

Programació del shell

- *Shell scripts*
  - Són arxius de text, formats per ordres del shell
  - Accepten arguments
  - S'executen:
    - mitjançant l'ordre sh: **`sh`** *nom argument(s)*
    - invocant-los directament pel seu nom
      - si no estan ubicats en cap dels directoris llistats en PATH, caldrà donar la seua ruta (absoluta o relativa a PWD)
      - han de tenir permís d'execució *-x*
  - Hereten part de l'entorn del shell
  - Els arguments del script són accessibles com variables:

símbol	valor
`$0`	nom del script
`$1…$9`	1r … 9é argument
`$#`	nombre d'arguments

```
#!/bin/sh
if[$# -gt 0 ]
then
 echo "param1 es $1"
else
 echo "Us $0 param1"
fi
```

Programació del shell

- Codi de terminació (exit code) de les ordres
  - La variable denominada ? Conté el codi de finalització
    - El codi de terminació és un valor numèric que es mostra amb **`echo $?`**
  - Retorn = 0 correspon a l'execució de l'ordre sense errors
  - s'interpreta com a condició,
    0 = cert i qualsevol altre valor és fals
  - Retorn =valor entre 1..255 l'ordre ha fallat

```
…$ ls
lote     lote-llarg…
…$ echo $?
0
…$ ls altre-nom
ls: altre-nom: No such file…
…$ echo $?
1
…$
```

- Ordre **`exit`**: Acaba l'execució del script i en determina el codi de terminació
- Algunes ordres relacionades amb el codi de terminació

ordre	Descripció de la utilitat
`true`	no fa res i torna un 0
`false`	no fa res i torna un 1
`test`	avalua una condició (vegeu *man test*) i torna 0 si s'acompleix i 1 en cas contrari

- Exemples

```
## creadir: script amb un argument
## Crea un directori buit de nom donat per l'argument $1

# Si no existeix cap arxiu ni directori amb el nom, el crea
if ! test -e $1; then mkdir $1; exit 0;
# Si existeix el directori, n'esborra el contingut
elif test -d $1; then rm -r $1/*; exit 0;
# En altre cas, no fa res i senyala error
else echo $1 ja existeix i no és un directori; exit 1; fi
```

```
## allold: script sense arguments
## afegeix la terminació ".old" a tots els noms d'arxiu
## del directori actual

# i és una variable local del script que valdrà cada nom
# del directori actual
for i in * ; do mv $i $i.old; done
exit 0
```

```
$ for((i=1000;i--; i>0)); do echo "$i orxates. Nomes queden
$i orxates"; done
```

Exercicis

- **Exercici SUT2.1_** Execute una a una les següents línies d'ordres del shell en una màquina UNIX, analitze el resultat de l'execució de cadascuna i contesteu a les preguntes proposades

```
$cat  prova
$echo  "Hola sòc un alumne de FSO" >&1
$echo  "Hola sò un alumne de FSO" >prova
$cat  prova
$echo "Estic practicant el shell" >>prova
$cat   prova | wc   -l
```

Contesteu per a cada línia d'ordres

a) Quin és el resultat d'execució?

b) S'ha creat un nou arxiu?

c) S'ha executat correctament la línia d'ordres?

d) Quin codi de retorn ha tornat el sistema?

e) Quantes ordres intervenen en aquesta línia?

f) Quants arxius intervenen en aquesta línia?

¡Avis!: el `$` que encapçala cada ordre representa el prompt de la màquina UNIX