1. QUE ÉS EL SISTEMA UNIX 2

2. CONCEPTES BÀSICS DE UNIX 2

2.1. Inici i acabament d'una sessió. 2

2.2. Comandes 2
2.2.1. Comandes generals 3
2.2.2. Comandes per directoris 3
2.2.3. Comandes per fitxers 4
2.2.4. Altres comandes 4

2.3. Proteccions en els fitxers i directoris 5

2.4. Comandes de la Shell 6

3. COMUNICACIONS EN UNIX 7

4.EDITORS DE UNIX 8

4.1. vi 8
4.1.1. Comandes bàsiques 8
4.1.2. Fitxer .exrc 10

5. EL F77(FORTRAN 77) I EL F90 (FORTRAN 90) 10

5.1. Compilar, linkar i executar 10

5.2. Llibreries 11

5.3. Errors 11

5.4. Opcions en la compilació 12

6. PROCESSOS INTERACTIUS 12

6.1. Processos en background 12

6.2. Cues de batch 13

1. QUE ÉS EL SISTEMA UNIX

El sistema operatiu UNIX és un sistema interactiu, multiusuari i multitasca desenvolupat per Bell Telephone Laboratories a finals de la dècada dels 60.

La **Shell** és el programa, escrit en llenguatge C, que interpreta i executa les ordres que l'usuari tecleja en el seu terminal, i a diferencia del MS-DOS, és pot modificar amb certa facilitat.

Una de les característiques més importants del UNIX és l**'estructura jeràrquica en arbre** de directoris i fitxers. Els seus noms poden estar formats per una cadena de fins a 256 caràcters i **no tenen extensió**. S'accepten noms amb un, o més, punts entre els seus caràcters, per exemple nom_dir, però el que realment distingeix un fitxer d'un directori és, com es veurà més endavant, la manera en que han estat creats.

Una altre característica que cal tenir molt en compte, i que podria ser un problema pels usuaris de VMS, és que el UNIX **distingeix entre majúscules i minúscules**. Les comandes que la Shell interpreta són majoritàriament en minúscula i dos fitxers són diferents si només difereixen en les majúscules o minúscules que formen el seu nom. Tampoc es guarden versions anteriors dels programes. Les opcions que tenen algunes comandes s'indiquen amb –opció en comptes de /opció com a VMS.

A diferència del sistema VMS (Silex i Alien), per aturar l'execució de qualsevol procès cal fer <Ctrl-c> o <Ctrl-y>. El <ctrl-d> indica a UNIX final de fitxer (a VMS és <ctrl-z>).

2. CONCEPTES BÀSICS DE UNIX

2.1. INICI I ACABAMENT D'UNA SESSIÓ.

Al establir connexió amb una màquina amb sistema operatiu UNIX (a l'escola hi ha, de moment, una màquina : titani), es sol·liciten el **username i password** de l'usuari (recordar la distinció entre majúscules i minúscules: Miquel i miquel NO serien el mateix usuari).

Per finalitzar una sessió en aquesta màquina es pot optar per : *logout* ó *<Ctrl-d> ó exit*
Es possible que al intentar sortir doni el missatge de que 'hi ha processos parats', cosa que no ens ha de preocupar. Per sortir definitivament, tornar a fer *logout*.

(Recordar que a efectes d'aquest manual els caràcters entre < > no s'imprimeixen per pantalla , sinó que representen una tecla que cal prémer).

2.2. COMANDES

Una de les comandes que us serà de gran utilitat és el ***man***, equivalent al help de VMS, amb la diferencia de que necessita d'una comanda, no dona un llistat general de les que disposa. També podem cercar ajuda sobre un tema relacionat amb la comanda ***apropos*** *text*.

2.2.1. Comandes generals

who	indica els usuaris connectats al sistema
whoami	indica el nostre username
yppasswd	per canviar el password d'accés al sistema
date	dona la data i l'hora
talk ***nom_usuari***	utilitat semblant al phone de VMS (Alien i Silex) que ens permet comunicació interactivament amb nom_usuari
echo ***text***	escriu 'text' per pantalla
banner ***text***	escriu 'text' per pantalla en lletres grans
lp -d ***nom_impressora nom_fitxer***	per imprimir un fitxer per la impressora especificada.
Lpstat -a	serveix per saber quines impressores tenim instal·lades al sistema

2.2.2. Comandes per directoris

Per facilitar la feina, la tcshell (shell per defecte que s'executa) disposa de l'opció del tabulador que permet escriure un path complert només donant la primera o primeres lletres del mateix. Per exemple si volem escriure : *cd programes* només cal escriure *cd pro* i prémer el tabulador, de manera que apareixerà el nom complet (si no tenim cap altre directori amb el mateix inici)

pwd	mostra el path complert (nom) del directori on ens trobem
ls	mostra els fitxers del directori actual
ls -l	igual que ls, però dona informació referent a proteccions, data de creació i mida, entre d'altres
ls -a	igual que ls, però a mes llista els fitxers ocults (a UNIX es consideren fitxers ocults, tots els que comencen amb un punt)
ls -R	llista els fitxers recursivament (tots els fitxers de cada subdirectori)
cd	canvia al nostre directori inicial (HOME DIR)
cd ***path***	canvia al directori donat per path (tant relatiu com absolut). path absolut vol dir tot el path, des de l'arrel, que es /
cd ..	canvia al directori pare del que estem ara
cd -	ens retorna a l'anterior subdirectori visitat.

mkdir ***nom_directori***	crea el directori de nom nom_directori (path absolut o relatiu)
rmdir ***nom_directori***	esborra el directori nom_directori, si aquest esta buit

2.2.3. Comandes per fitxers

cat ***fitxer1 ...***	mostra per pantalla els fitxers especificats, un rera l'altre.
more ***fitxer***	igual que l'anterior però la sortida per pantalla es paginada. Per veure la pagina següent prémer <ESPAI>, per sortir <q>
cp ***fitxer1 fitxer2***	copia el fitxer1 sobre el fitxer2, creant el segon si no existeix
mv ***fitxer1 fitxer2***	reanomena el fitxer1, també s'utilitza per moure fitxers a d'altres directoris
rm ***fitxer***	esborra el fitxer
rm -i ***fitxer***	demana confirmació abans d'esborrar el fitxer
rm -r ***directori***	esborra el contingut del directori i dels seus subdirectoris (deltree del DOS)
head -num ***fitxer***	mostra les num primeres línies del fitxer
tail -num ***fitxer***	mostra les num darreres línies del fitxer
paste ***fitxer1 fitxer2 ...***	concatena horitzontalment fitxers i els mostra per pantalla
comm ***fitxer1 fitxer2***	compara els dos fitxers i mostra resultats en tres columnes: la primera amb el que només hi ha al primer fitxer, la segona amb el que només hi ha al segon i la tercera amb les línies comunes

2.2.4. Altres comandes

quota -v	ens informa de la quota de disk que tenim i del espai que ens queda .
du (disk usage)	aquesta comanda permet esbrinar l'espai, en blocs de 512 bytes, que ocupen els nostres fitxers i directoris. amb la opció -k, la informació la dona en Kbytes.
gzip ***fitxer***	utilitat que permet comprimir fitxers per a que ocupin menys espai. el fitxer passa a tenir un .gz al final del nom.
? opcions: *-d*	per descomprimir fitxers, prèviament comprimits
-l	per veure'l sense tenir-lo que descomprimir

cal dona un calendari del mes corrent. Si s'especifica un mes (en número) i un any els dona també.

bc calculadora del sistema UNIX

2.3. PROTECCIONS EN ELS FITXERS I DIRECTORIS

A UNIX tots els fitxers (tant fitxers com directoris) tenen unes proteccions. Aquestes proteccions son a 3 nivells: usuari, grup, mon

Quan fem un *ls -l* , el llistat dels nostres fitxers i directoris és del tipus següent:

```
- rw- r-- r--  miquel etseccpb  fitxer
d rwx --- ---  miquel etseccpb  directori
```

?	columna	1	tipus de entrada -> fitxer(-) o directori(d)
?	columnes	2,3,4	proteccions a nivell usuari (r: lectura, w: escriptura, x: execució)
?	columnes	5,6,7	proteccions a nivell de grup al que pertany el fitxer
?	columnes	8,9,10	proteccions a nivell de la resta de usuaris
?	columna	11	user: propietari del fitxer
?	columna	12	group: grup al que pertany el fitxer
?	columna	13,14,15	data i hora de la darrera modificació
?	columna	16	nom del fitxer o directori

chmod ***proteccions fitxer*** canvia els permisos d'un fitxer/directori
Les proteccions es poden treure i posar molt fàcilment:

qui	**que**	**el que**	u : user	g: group	o: other	a: all
u/g/o/a	*+/-*	*r/w/x*	+: afegir	-: treure		
			r: read	w: write	x: execute	

(Ex.: *chmod u+r fitxer* permet lectura al propietari sobre fitxer
chmod g-w fitxer treu el dret d'escriptura al grup sobre el fitxer
chmod a+rw fitxer permet lectura i escriptura sobre el fitxer a tothom)

També es poden posar les proteccions amb un nombre de 3 dígits, on cada dígit conté les proteccions de cada nivell codificades en octal: r=4,w=2,x=1 .
Per exemple ***chmod 644 fitxer*** dona privilegis de lectura i escriptura al user (r+w=4+2), i només de lectura al group i a others.

chown ***nou_propietari fitxer*** permet canviar el propietari del fitxer
chgrp ***nou_grup fitxer*** permet canviar el grup al que pertany el fitxer

Tots els fitxers, quan es creen, tenen unes proteccions que per defecte que les posa el sistema automàticament; aquestes son les que estan definides amb la umask (màscara d'usuari). Canviant la umask, canviem les proteccions per defecte, que inicialment són

027 (o sigui, que quan creem un fitxer les proteccions son rw per l'usuari, r pel grup i res per la resta).

*umask **num*** a partir d'ara les proteccions per defecte seran (777 - num)

2.4. COMANDES DE LA SHELL

whereis comanda ens diu on es troba (path complet) la comanda (executable) demandada

find path genera una llista, en forma d'arbre, amb els continguts dels directoris a partir del directori, amb tot el camí d'accés

id dóna informació amb l'identificador d'usuari i dels grups als que pertany

newgrp grup permet a un usuari que pertanyi a diferents grups canviar de grup per poder accedir als fitxers del seu grup

sleep nsec suspèn l'execució de la shell durant nsec segons

wc fitxer comptador de paraules, línies i caràcters de la seva entrada estàndard o del fitxer

sort fitxer ordena la seva entrada estàndard o el fitxer especificat

grep patró fitxers busca el patró donat a la seva entrada estàndard o en els fitxers especificats

tee fitxer transcriu la entrada estàndard a la sortida estàndard fent una còpia en el fitxer. És com si bifurquèssim una pipe.

cut –fxxxx fitxer genera una sortida amb els camps seleccionats per xxxxx del fitxer especificat, on xxxxx indica quins camps de cada línia es mostren.

Les comandes anteriors són útils si les encadenem, mitjançant pipes, amb d'altres que generin una sortida.

Quan parlem d'entrada i sortida estàndard ens referim al teclat i a la pantalla, respectivament. Tot i així, a UNIX, existeixen mecanismes per redireccionar aquestes entrades i sortides estàndards. També podem redireccionar la sortida d'error.

La primera manera de redireccionar entrades i sortides és l'ús de "pipes". Les "pipes" representades per | , permeten unir cadenes de comandes, encadenant la sortida de la primera amb la entrada de la segona i així successivament. Alguns exemples útils:

cat jaja | more se'ns mostra el contingut de "jaja", però amb una pausa després de cada pantalla (more)

who | grep nom_usuari ens diu si l'usuari especificat està connectat

ls ~ | sort –r | lp –d ps_15 imprimeix per la ps_15 (impressora per defecte) el contingut del directori inicial de la compta ordenada alfabèticament. inversament

*who | tee –*a *jander* treu per pantalla el que genera el who i a més ho afegeix al

fitxer jander.

La segona forma, s'utilitza normalment per comandes aïllades. Per poder redireccionar l'entrada estàndard cal que posem un < al final de la comanda seguit del que volem que es llegeixi en comptes de l'estàndard. Per redireccionar la sortida, posarem un > en comptes del <. Si el que volem és redireccionar la sortida d'error, posarem un & darrera del > (en *tcsh*) o un 2 davant del > (en *ksh*). Si no volem que el redireccionament de sortida (tant estàndard com d'error) esborri les dades d'anterior redireccions posarem >> en comptes de >.

Ex:

mail elena < missate.txt	envia el contingut de missatge.txt a l'usuari elena via mail
ls ~ > llista	el llistat del directori el posa a *llista*
echo "Arbre de directoris....." > tree *find . >> tree*	posem al fitxer *tree* el títol i afegim la llista de fitxers que tenim a la compta.
cat pepe >& error	(si no existeix el fitxer *pepe*) el missatge d'error de que no troba el fitxer l'escriu al fitxer *error*

(en els casos en els que redireccionem qualsevol sortida, si el destí no existeix el crea)

Una comanda que pot resultar útil és *setenv* que ens permet consultar les variables d'entorn de la nostra sessió. Entre les més importants trobem PATH, HOME, DISPLAY... Per canviar alguna d'aquestes variables el que farem és: *setenv variable valor.*

Ex:

setenv DISPLAY *maquina.upc.es*	permet enviar la sortida gràfica d'un programa cap a la màquina especificada

3. COMUNICACIONS EN UNIX

El sistema operatiu UNIX disposa de sistemes de comunicació molt similars al de VMS. Ja s'ha mencionat la utilitat *talk*, que es totalment equivalent al phone i també existeix la utilitat de mail que funciona de manera molt similar a la que ja coneixem de SILEX o ALIEN:

mail ***nom_usuar****i*	envia correu a nom_usuari. Un cop finalitzem d'escriure el missatge, escrivim un punt en la darrera línia i premem <Enter>. També existeix la possibilitat de fer <Ctrl-d>.
mail	és la comanda que ens indica si tenim algun missatge nou,

i si en tenim, entrarem al mail.
Dins de mail, la comanda list us permetrà veure una mena de help molt reduït però útil, sobre les instruccions del mail com : esborrar, guardar, enviar fitxers, reenviar...

ftp ***maquina*** permet transmetre fitxers d'una màquina a una altra. El seu funcionament es similar al ftp, encara que sino sabem alguna cosa disposem d'un help per consultar.

telnet ***maquina*** amb aquesta comanda podem connectar-nos a una altra màquina per treballar amb ella. Un cop sortim de la màquina on ens hem connectat, tornem al lloc on erem abans de fer la connexió.

4.EDITORS DE UNIX

Actualment es disposa de diferents editors, el **vi** i el **emacs**. El vi el trobarem a qualsevol màquina UNIX, ja que és l'editor per defecte que s'instal·la.

4.1. VI

vi ***nom_fitxer*** accedim al nom_fitxer usant l'editor vi.

L'editor te tres modes d'operació: comanda, inserció i execució. Això ens permet, amés d'escriure, executar ordres de Shell. Accedim, sempre, en mode comanda.

4.1.1. Comandes bàsiques

NOTA: Les comandes admeten un nombre, amb la qual cosa fem que la comanda es repeteixi automàticament el nombre de cops indicat

? per passar de mode comanda a inserció, podem emprar una de les següents opcions:

a	afegeix text després del cursor
A	afegeix text al final de la línia
i	insereix text abans del cursor
o	afegeix una nova línia per sota l'actual (afegir línia)
O	afegeix una nova línia damunt de l'actual (inserir línia)
R	permet introduir text sobreescribint el que hi ha sota el cursor

? per passar de mode inserció a mode comanda
<ESC> ,ó
<ctrl-3>
(els caràcters entre < > no s'imprimeixen per pantalla, simbolitzen una tecla)

? sortir de l'editor (és important no oblidar els :)

ZZ ó :wq ó :x	grava el text al fitxer i surt de l'editor,
:q!	sortida sense gravar

? control del cursor

h	mou el cursor una posició a l'esquerra
j	el mou una línia avall
k	el mou una línia amunt
l	el mou una posició a la dreta
G	posiciona el cursor al final del fitxer
nG	el posiciona a la línia n (n és un nombre qualsevol)
w	el posiciona al principi de la paraula següent
b	el posiciona al principi de l'anterior
e	el posiciona al final de la següent
$	mou el cursor al final de la línia
L	col·loca el cursor a la ultima línia de la pantalla visualitzada
M	el col·loca a la línia del mig de la pantalla
H	el col·loca a la primera línia de la pantalla

? esborrar text

x	esborra el caràcter sobre el que esta situat el cursor
X	esborra el caràcter anterior al cursor
dw	esborra des de la posició actual fins al principi de la següent paraula (ho guarda dins un buffer que es pot usar després)
dG	esborra fins al final del fitxer
dd	esborra la línia sobre la qual esta situat el cursor

? canviar text

rX	substitueix el caràcter actual pel caràcter X
cw	substitueix fins al principi de la paraula següent, afegint a partir d'aleshores (aquesta comanda fa que passem a mode inserció)
cG	substitueix fins al final del fitxer (funcionant com l'anterior)
cc	substitueix tota la línia actual (funcionament com l'anterior)

? manipulació de fitxers

:*w nom_fitxer*	grava el text al fitxer indicat. Sense paràmetres ho fa al fitxer actual. Si el fitxer indicat existeix cal posar ! darrera w
:*w >> fitxer*	grava el buffer al final del fitxer especificat (concatenació)
:*r fitxer*	insereix el fitxer indicat a la posició del cursor
:*e fitxer*	edita el fitxer indicat, esborrant el que estàvem editant
:*M,Nw fitxer*	agafa les línies de la M a la N i les escriu al fitxer especificat. si el fitxer indicat existeix cal posar ! darrera w

? recerca de text

/text	cerca el text des de la posició actual fins al final de fitxer
? text	cerca des de la posició actual fins al principi del fitxer
n	repeteix la cerca en la mateixa direcció que originalment
N	repeteix la cerca en la direcció oposada a l'original
:*M,N /antic/nou*	canvia cada ocurrència del text antic pel nou en el tros de text compres entre les línies M i N

? copiar i moure

yX	copia l'objecte X, on X pot ser: w(paraula), $(línia) o G(final de text), a un buffer per després inserir-lo al text
p	insereix el buffer al text abans del cursor
P	insereix el buffer al text després del cursor

? comandes miscelànies

u :*u*	desfà l'últim canvi fet en el text (undo)
.	repeteix l'ultima acció que hem fet (redo)
J	uneix la línia següent al cursor amb l'actual (join)
:*ab expr1 expr2*	substitució intel·ligent. cada cop que al text s'escrigui la expr1, aquesta serà substituïda automàticament per expr2
:*!command*	executa una comanda de la Shell sense sortir de l'editor (p.e: un ls)

4.1.2. Fitxer .exrc

Aquest fitxer, situat en el directori HOME de cada usuari i que es llegeix abans d'entrar al vi. Està format per comandes tipus: *set* option i map tecla funció, on set option permet canviar la configuració del vi; i *map* tecla funció permet assignar a una tecla una funció especifica .

5. EL F77(FORTRAN 77) I EL F90 (FORTRAN 90)

5.1. COMPILAR, LINKAR I EXECUTAR

La **compilació i linkat** de programes fortran es du a lloc amb la comanda:

f77 nom_fitxer ó
f90 nom_fitxer

Es clar que el fitxer que compilem ha de ser un programa fortran, però no es necessari que tingui cap extensió especial; tot i que sovint s'els dona l'extensió *nom.f* El programa executable que es genera, si no es fan més especificacions, es diu *a.out*. Si es vol un fitxer executable amb un nom concret, aquest s´ha d'especificar amb l'opció -o, com segueix:

f77 -o nom_del _executable nom_del_compilable

D'aquesta manera s'aconsegueix un fitxer on l'extensió NO és necessària (tot i que podem posar-li nom.exe), i el nom del executable pot ser qualsevol. Una bona manera de distingir quins fitxers són executables, quins són directoris,etc...és amb l'opció ls –lF que assigna un * al final dels fitxers executables, un / al final dels directoris...

L'**execució** del programa, com a la majoria de llenguatges, es porta a terme escrivint el nom del fitxer executable.

nom_del_executable

5.2. LLIBRERIES

Si el nostre programa FORTRAN requereix de llibreries externes, podem compilar el programa i les llibreries conjuntament:

f77 -o programa programa.f llibreria1.f llibreria2.f llibreria3.f
(Amb això aconseguim un executable que en aquest cas es diu *programa*)

Una altra possibilitat, molt més emprada en UNIX és la creació d'un fitxer anomenat **Makefile** que és un programa on es recullen les accions que volem portar a terme amb el nostra programa FORTRAN : el seu nom, com compilar-lo amb les llibreries, diferents opcions de compilació, esborrar el fitxers objecte, etc. Com a petit exemple, editem el fitxer Makefile i dins d'ell:

Provem:
$f77 –o programa programa.f llibreria1.f llibreria2.f

Llavors quan fem *Make Provem* el que farem serà compilar programa.f conjuntament amb dues llibreries, i obtenir un fitxer executable anomenat programa.

5.3. ERRORS

De la mateixa manera que succeïa amb FORTRAN per a VMS existeixen **errors de sintaxi**, que apareixen al compilar:

warnings (quan oblidem de tancar un do, per exemple) i **errors** (quan realment no es un oblit si no una equivocació en la redacció que fa inintel·ligible el nostre programa al compilador). Un error de sintaxi en el programa té el format:

nom_programa, línia d'error: nº de l'error: tipus d'error amb breu explicació

Existeixen també errors al **executar** el programa, tot i que la sintaxi del mateix és correcte. Alguns dels exemples més comuns:

Si no existeix algun del fitxers que requereix el programa ho explicita clarament:
open(fitxer_de_dades): No such file or directory.

També passa si un fitxer de sortida te status =‘new’ i ja existeix, llavors:
nom.res: 'new' file exists

Quan els **errors** son més **greus**, com variar el número d’arguments a una subrutina entre el que li definim i el que rep, tema de mal dimensionament d’una variable, etc. el error en executar el programa seria:

segmentation fault (core dumped)

Aquest missatge és el que ens indica un error no de sintaxi sinó d'execució. El fitxer core. es genera automàticament i es un recull de totes les operacions que s'han dut a lloc fins que l'execució del programa ha fallat. Es un fitxer MOLT GRAN, per això es important esborrar-lo.

Un altre problema està els **formats** dels números: el FORTRAN de VMS sol donar ****** quan no pot representar un número. El FORTRAN de UNIX origina un *nan* (non a number). Això es prou comú en processos de convergència a una solució mal definits (mala aproximació inicial, valorar funcions en arguments no possibles...).
Si el problema es deu a un error en el format, llavors apareix ****, com a VMS; per exemple si donem un format de I3 i el nostre número és un enter de 4 dígits.

5.4. OPCIONS EN LA COMPILACIÓ

Tot i que és de carácter merament sintàctic, UNIX considera els backslash (“\”) com a principi de seqüència d’escapament. Per que siguin tractats com a “\” normal, cal usar l’opció de compilació

–backslash

Existeixen diferents opcions de compilat per a la optimització de codi, de manera que es guanya velocitat d’execució (també és possible optimitzar espai en el fitxer executable, però no és una opció tan interessant). Aquestes opcions estan disponibles en man f77 o man f90. Algunes d’aquestes opcions més importants son

-*64*	Aprofita millor l’arquitectura de 64 bits, que és la que correspon al nostre processador.
-mips4	Joc d’instruccions del processador R10000. També en la mateixa línia que l’anterior
-O0 a -O3	Són altres opcions que de menor a major grau optimitzen el codi màquina.

6. PROCESSOS INTERACTIUS

6.1. PROCESSOS EN BACKGROUND

Mentre executem un programa, podem fer que es pari momentàniament (stopped) fent un <Ctrl-z>. Per continuar més tard amb la seva execució tenim dues opcions : que continuï la seva execució en primer pla o que ho faci en segon pla. La comanda *fg* permet fer la primera opció ; per la segona, la comanda a executar es *bg*.

Un altra manera de enviar un programa a executar-se en segon pla és fer-ho en el moment d'invocar-lo. Per fer-ho hem d'afegir un & al final de la invocació. Per executar-se en primer pla el que farem és invocar-ho normalment, sense & al final.

Amb la comanda **jobs** podem veure quins processos tenim engegats (tant stopped com en background). Si tenim diversos processos en espera per passar-los a executar en primer o segon pla, podem especificar sobre quin volem actuar afegint %numjob després de la comanda adient (fg o bg).

Ex :

bg	posa l'últim procés a executar-se en background (segon pla)
fg %1	posa el job (treball) 1 a executar-se en foreground (primer pla)

La comanda **ps** ens dóna informació sobre els processos que tenim en execució a la màquina (dóna més informació que la comanda **jobs**). Entre la informació donada, a més del nom del programa, ens diu el seu PID o identificador de procés. Amb aquest identificador podem actuar sobre el procés modificant la seva execució.
Amb la comanda ***kill*** *-num PID* enviem signals (interrupcions) al procés amb identificador pid. Entre els num més utilitzats trobem el 9 (per matar a un procés) i el 15 (per avisar-lo primer abans de matar-lo).

6.2. CUES DE BATCH

Igual que en el sistema VMS (ALIEN i SILEX), a UNIX també existeix la possibilitat de enviar treballs a cues de batch per la seva execució. Per treballar sobre les cues utilitzarem les següents comandes :

qsub -q queue script envia l'script donat a la cua queue.

No podem enviar un programa tal qual, s'ha de fer un **"shell script"** per poder-ho enviar. El cos bàsic d'un "shell script" és :

```
# !/bin/sh
path/programa a executar
```

(Recordar que el path és el nom complert del directori on està el nostre programa)

Cada treball que enviem a la cua generarà dos fitxers: un d'error i un amb els resultats obtinguts.

El format d'aquests dos arxius és:

nom_script.**e***request*
nom_script.**o***request*

(exemple: si l'script que enviem es *obra.sh* i és el treball 57 que ha entrat a la cua, els fitxers creats seran: obra.sh.e57 i obra.sh.o57, aquests es crearan al directori des del que s'ha fet el qsub)

qdel request	per esborrar un treball encuat previament
qstat queue	ens informa del estat de les cues de batch (quins treballs tenen, quines cues hi ha, etc....). Si posem el nom de la cua, ens dóna la informació referent a aquella cua.

Actualment, hi ha 3 cues on es poden enviar processos : short, medium, xlarge. Les diferències són primordialment el temps de CPU de que disposa cada cua.