Juan Tonda.

Unix, **Guía de comandos básicos**

Para ejecutar un comando utilizaremos el TERMINAL (emulador terminal) para la puesta en marcha de cada uno de ellos.

El formato de salida del comando es el siguiente:

$ _ <comando> <nombre fichero> <parámetros opcionales>; no obstante para cualquier aclaración y comprobación correcta de ejecución del mismo utilizaremos:

$ _ man <comando>; para ver todas las opciones posibles de dicho comando.

LISTA DE COMANDOS GENERALES

passwd	Cambio de contraseña
banner [argumento]	Argumento en MAYUSCULAS
cal [[mes] [año]]	Muestra el calendario
calendar	Realizar una agenda
date [argumento]	Hora y Fecha actual
*Formatos de salida (comprobar manual: -**man**-)* %m(1-12);%d(1-31);%Y(0-99);%Dmm/dd/yy; %H(00-23);%M(00-59);%S(00-59);%Thh:mm:ss; %j(1-366);%w(0-6); %adia;%hmes;%rAM/PM;%%-%	
who	Averiguar quien hay en el sistema
man [comando]	Ayuda del comando

TRABAJANDO CON DIRECTORIOS

mkdir DIR1 DIR2 …	Crea directorios en el actual
-p, a partir del actual colgando uno tras otro	
cd [dir]	Cambia al directorio destino.
cd directorio de trabajo. **cd .** directorio actual. **cd ..** al directorio padre. **cd /** sale a la raiz.	
rmdir [-p -s] dir	Borra directorio suponiendo éste vacío
-p confirma -s cancela mensaje	
dircmp [-s,-wN] dir1 dir2	Compara dos directorios
-s ficheros diferentes -wN formato de columnas	
pwd	Directorio actual

Juan Tonda.

TRABAJANDO CON ARCHIVOS

En Unix al trabajar con archivos debemos tener en cuenta los permisos con que cuenta el archivo, además de darnos la información correspondiente al listado mostrado, para ello hemos de observar lo siguiente:

Atributos de archivos
1 - coloca atributo/permiso; 0 - quita atributo/permiso

ABSOLUTO
chmod 0412 r-- --x -w- umask 0750 --- -w- rwx
100 001 010 -> pasado a OCTAL 111 101 000 -> pasado a OCTAL.

SIMBOLICO
chmod u=r ... o-r ... u+x

a) **Afectado:** u usuario, g grupo, o otros, a todos
b) **Operación:** + añadir, - quitar, = quitar los actuales
c) **Permiso:** r lectura, w escritura, x ejecución, l bloqueo fichero

chmod: Modificar los permisos de los ficheros (propietario o supervisor) de forma ABSOLUTA o SIMBOLICA.

umask: Modificar la máscara de grabación del tipo de fichero. Cuando creemos un fichero se le asignará los permisos definidos en la máscara. (se realiza a la inversa de CHMOD), el cual será restado a la máscara actual. No permite la grabación en los de ejecución.

Generales	
Permisos: lectura; escritura; ejecución usuario grupo otros -rwx -rwx -rwx	**chmod** modo fichero
	umask [nº]
rm [-f r i] fich's	Borrar un fichero o varios, dando posibilidad de utilizar el comodin '*'.
-f no pide conformidad -r borra los directorios contenidos en el actual -i pide conformidad	
cp	Copia archivos
mv [-f] ficheros	Mover archivos
-f no pide conformidad	
ls [-l …]	Lista en pantalla árbol de ficheros
-l con sus permisos correspondientes	

Juan Tonda.

Manejo de archivos	
cat [opciones] fichero	Visualiza y une ficheros
-vt tabuladores [^I] -ve carácter '$' al final de línea -b , -n {ver nº de línea del archvo}	> salida simple >> añadir a uno existente
cmp [-l] fich1 fich2	Compara 2 ficheros visualizando el lugar dónde no coinciden los ficheros.
-l visualiza número de bytes y valores distintos	
sort [opciones]	Ordena uno o más ficheros por líneas mostrando el resultado en pantalla -k ordenar desde [campoN.caracterX hasta campoN.caracterX]
-o fichero (rederigido a un fichero de salida) -n campos numéricos -r sentido inverso -u quita líneas repetidas -k [x.y],[z.t]	
uniq [-c -u -d -n]	Se utiliza mediante filtro, quitando líneas repetidas.
-c número de ocurrencias de las líneas únicas precediendo a la propia línea -u líneas no repetidas -d líneas repetidas -n desecha los 'n' primeros campos de la línea	
diff [-b -e] fich1 fich2	Diferencia entre 2 archivos
-b ignora las líneas que son distintas en los espacios en blanco -e para poder utilizar con el editor “ed”	
cut [-cn -fn -dc] fichero	Recortar una parte del fichero.
-c nºcolumna (1,3) ... (1-3) -f nºcampos -d carácter (divisisor de campo)	
wc [-l -w -c] [fichero]	Contar lineas, palabras y caracteres
-l lineas -w palabras -c bytes -m caracteres	$ ls | wc -l ... nº de ficheros del directorio

Juan Tonda.

BÚSQUEDAS

<table>
<tr><th colspan="2">ARCHIVOS y DIRECTORIOS</th></tr>
<tr><td>find acceso expresión</td><td></td></tr>
<tr><td>- acceso:</td><td>directorio o directorios donde vamos a buscar los ficheros y expresión de búsqueda</td></tr>
<tr><td>- expresión de búsqueda: (tipo)</td><td><u>Tipo de búsqueda</u></td></tr>
<tr><td>b .ficheros de bloque
c .ficheros terminales
d .directorios
f .normales</td><td>– name “fichero”
– type “caracter”
– user “nombre”
– group “nombre”
...</td></tr>
<tr><th colspan="2">TEXTOS</th></tr>
<tr><td>grep [opciones] patrón fich's</td><td>Sirve para localizar un carácter o cadena de caracteres dentro de un fichero o ficheros.</td></tr>
<tr><td>PATRON: Puede ser una palabra, una cadena de caracteres o metacaracteres.</td><td></td></tr>
<tr><td colspan="2"><u>METACARACTERES</u>
Irán encerrados entre comillas.
<u>.</u> sustituye a cualquier carácter (? de MS-DOS) "pep."
<u>\</u> desactiva los significados especiales "\."
<u>[lista]</u> coincidirá con cualquier carácter de los que están incluidos en la lista. [A..Z] [ABC] [1..9] ...
<u>[^lista]</u> para todos los carácteres que no estén en la lista.
<u>^[lista]</u> el patrón tiene que coincidir con el príncipio de la línea.</td></tr>
<tr><td colspan="2"><u>OPCIONES</u>
-<u>v</u> muestra en pantalla todas las líneas no coincidentes.
-<u>c</u> muestra en pantalla sólo el nº de líneas que coinciden.
-<u>l</u> visualiza los nombres de fichero con líneas coincidentes, pero no visualiza las líneas.
-<u>n</u> visualiza el nº de línea antes de cada línea coincidente.
-<u>b</u> antepone a cada línea que se muestra el nº de bloque en el que está situado, el primer bloque es siempre el 0.
-<u>s</u> suprime la visualización de los mensajes de error en el caso de que no existan las líneas o no sea posible leer los ficheros.</td></tr>
</table>

IMPRESIÓN		
lp [opciones] ficheros Imprime uno o varios ficheros. La prioridad va en función del orden en el cual se han seleccionado.	Opciones -c , modificación del fichero sin alterar su salida. -d destino, destino de impresora distinta a la habitual. -m, correo electrónico. -nX, copias. -s, sin mensajes. -w, mensaje de finalización en el terminal.	
lpstat [opciones] El estado de impresión de las peticiones solicitadas.	Opciones -c, sobre impresoras. -d, impresora por defecto. -o, peticiones de salida. -t, información de estado. -u, usuario.	
cancel [id`s] [impresora]	Cancelará el trabajo de impresión por su id, por la impresora cancela el trabajo en curso.	
COMUNICACIÓN ENTRE USUARIOS		
mesg [y|n]	permite o impide el envio de mensajes, sin pasar ninguna opción visualizará por defecto el estado de permisos de mensajes.	

write usuario
$ write user22
jjjjj, mayo 16, 2008
$ write user22 < fichero_mensajes
$ banner HOLA | write user22
	Envia información al usuario especificado.	

TIPOS DE PROCESO		
ps [opciones] - e , información de todos los procesos del terminal.	**IDENTIFICACION** > PID - número de proceso > UID - número de usuario > GID - número de grupo	Muestra PID TTY TIEMPO COMANDO.
nice [-s] <comandos> - s , el proceso se ralentizará en las unidades especificadas en s segundos.	Los usuarios, en general tienen la misma prioridad en los procesos, pero puede verse alterada por este comando.	
$ nice --7 cat/etc/passwd SOLO PUEDE HACERLO EL SUPERUSUARIO.		
kill [-señal] PID $ kill -9 340 -> se eliminará el proceso SH para volver pedir Login:_	Elimina el proceso para que finalice su ejecución, en una determinada prioridad, y solo teniendo acceso el superusuario a la detención de dichos procesos.	

UTILIDADES	
pack [-][-f] fichs	Sirve para compactar ficheros. fichero mismo nombre, extensión .z
-, el sistema muestra información respecto al algoritmo de compresión (HUFFMAN). -f , obliga a que se produzca la compactación aunque no haya ahorro de espacio.	
pcat fichs	Visualiza un fichero compactado, no hace falta utilizar .z
unpack fichs	Para descomprimir ficheros empaquetaçdos por pack, elimina la extensión .z
NO AVISA DE LA EXISTENCIA DE FICHEROS	
FICHEROS	
/etc/passwd -> referencia al usuario /etc/group -> referencia al grupo	

Juan Tonda.

EL SHELL
(trabajando con órdenes para automatizar procesos)

PROGRAMACION SHELL

Proceso SHELL, Un fichero cuyo contenido consta de cualquier comando, llevando un orden permitido por sentencias condicionales, entrada de datos, tipos de variables, ...; para ejecutar dichos procesos podemos proceder mediante:

Ejecución: $_ ./------.sh ….

Invocar	$ _ sh fichero_comandos.sh
Permisos de ejecución	$ _ chmod u+x fichero_comandos
Comentarios	Caracter # Ojo: **#!bin/bash** (interprete de comandos)
Textos (redireccionamiento)	<< cadena << - cadena **visualizar hasta encontrar la cadena**
DEPURACION	Llamadas de procesos chmod u+x fichero_comandos (permiso de ejecución) sh fichero_comandos (ejecución de script)
sh -v , línea a línea **sh -x , línea a línea para corrección de errores**	

Variables de entorno	
HOME	directorio home del usuario.
PATH	directorios de búsqueda, separados por ":".
PS1	prompt PRIMARIO
PS2	prompt SECUNDARIO
MAIL	path del buzón de correo
MAILPATH	similar a MAIL, pero contiene más de un fichero.
MAILCHECK	número de chequeos entre MAIL y MAILPATH (tamaño de ficheros)
IFS	separador de campos
LOGNAME	nombre de usuario
SHELL	tipo de shell invocado
CDPATH	comando CD (path)
TERM	tipo de terminal de trabajo
EXINIT	set por defecto del vi

Parámetros sustituibles
$1 ... $9 ------> ***shift (para ir cambiando el parametro)*** $0 nombre de fichero echo $1 ... $* suma de todos los valores $# número de parametros (no tiene en cuenta $0) $? valor de ejecución de un comando: 0 - ejecución ok; 1 – error; 2 - error de dispositivo $@ la lista de todos los parametros pasados (for {VAR} in $@)

Secuencias de Escape
\c impide saltar una línea \n retorno de carro \t tabulación \f salto de página

Variables SHELL
$ variable=valor
- El valor de dicha variable se recuperará anteponiendo el símbolo '$' delante del nombre de la variable.
$ echo $variable
- Los caracteres {variable} sirven para identificar a una variable de un literal.
$ ped="${variable} tipeado"
- Dichas variables SHELL servirán para invocar MACROS, en las cuales se podrán introducir ordenes.
$ busca="find . -name"

COMANDOS

$ echo
$ echo (mensaje o variable)
Preferible encerrar los mensajes entrecomillas dobles.

Estructuras de órdenes

Bucle FOR

```
for variable [in lista]
do
 ordenes
done
```

- En lista se pueden utilizar comodines "* y ?"

IF-THEN-ELSE-ELIF-FI

- If existe condición ENTONCES ...

```
if orden_cond
 then
ordenes
fi
```

- If existe condición ENTONCES … SINO ...

```
if orden_cond
 then
ordenes
 else
ordenes
fi
```

- If ENCADENADO

posibilidad 1

```
if orden_cond
 then
ordenes
elif orden_cond
  then
ordenes
  [else
ordenes]
fi
```

posibilidad 2

```
if orden_cond
 then
ordenes
 else if orden_cond
  then
    ordenes
    [else
    ordenes]
 fi
fi
```

Estructura while (mientras)

```
while orden_condición
do
 ordenes
done
```

Estructura until (hasta que)

```
until orden_condición
do
 ordenes
done
```

- **exit (salida del bucle)**

exit 0 - proceso sin error; exit 1 - proceso con error

- **break/continue**

break -> sale del bucle.
continue -> vuelta al inicio del bucle. 'loop'

- **false/true**

Permite realizar bucles infinitos.
while true (do while .t.)

- **case (muy útil para crear menús)**

```
case var in
a) --- ;;
b) --- ;;
c|d) --- ;;
?) "error" ;;
esac
```

Se puede utilizar "*" y "?".

- **expr**

Permite ejecutar operaciones aritméticas en un proceso.

```
expr op1 op2 op3 ...
```

+ suma, **-** resta, **%** resto, **/** división, ***** multiplicación

```
expr $1 + $2 + $3
var="expr $1 + ... " --- var=`expr $1 + $2 ... `
```

- **read**

read variable1 var...

```
-> echo "Introduce ....!"
-> read variable
-> echo $variable
```

- **tput**

tput cup fila columna (Posicionar el cursor en una determinada posición de la pantalla)

- trap

Permite hacer inmune un comando de usuario a la desconexión del proceso o sistema.
ORDEN ENTRE COMILLAS y SEPARADAS POR ;
2 - INTERRUPCION DE UN PROCESO
1 - DESCONEXION DEL SISTEMA

Comando TEST (comparación)		
NUMERICOS	**FICHEROS**	**CADENAS**
test num1 <operador> num2	test <operador> fichero	test cad1 <operador1> cad2 test <operador2> cad1 cadena -> para cadenas nula " $cadena "
-eq (igual que) -ne (distinto) -gt (mayor que) -lt (menor que) -qe (mayor o igual que) -le (menor o igual que)	-s fichero no vacío -f fichero, no directorio -d directorio -w permiso escritura -r permiso lectura -x permiso ejecución	op1 ==,= igual que, != distintas, <, > op2 -z longitud 0, -n superior a 0 -a, -o y ! -a .AND., -o .OR., ! - negación -> .NOT.
test (comparar existencia de archivos) 0 - cierto, 1 - falso		

FUNCIONES

Las funciones tienen que estar definidas al principio del programa.

```
nombre_función()
{
... ordenes
reemplazando las variables generadas por $1, $2, ...
}
```

Para llamar a las funciones se pone el nombre y a continuación los parámetros que queramos pasar a dicha función, separados por espacios, y serán reconocidos mediante $1 ... $9.

LLAMADA función_x $var1 $var2 -> donde $var1, $var2, serán sustituidas por $1, $2, ...

AMPLIACIÓN DE COMANDOS

Comando: touch
– para realizar cambio 'fecha de modificación'
– para crear archivos vacíos

uso: touch {archivo1 archivo2 ...}

Comando: ln
– crear enlaces de archivos

uso: ln [-L] archivo_origen archivo_destino

-L (crear enlaces símbolicos)
en simbolico se ve como ... → ...
Comando: head y tail
relacionados con cut

Filtros en Unix
– *reciben entrada, produce una salida.*

sort (ordenar); grep (búsqueda de cadenas); find (búsqueda de archivos); tr (sustitución de caracteres de un archivo); wc (contar); cut (recortar archivos)

se puede utilizar | (tubería) para enlazar con más de un comando.

Find
find <directorio> opciones comparaciones [lógico comparaciones …] operación

lógico: -a (equivale a AND -y-) - las dos; -o (equivale a OR -o-) -una de las dos

*** OPCIONES**
- maxdepth N { nivel de directorio a recorrer }
- mindepth N { lo contrario }

*** COMPARACIONES**
- name “archivo”; - user “usuario”; - group “grupo”; - perm; -size , ….
- amin [+-] accedido en minutos
- cmin [+-] modificado en minutos
- atime [+-] días (N*24) accedido
- ctime [+-] … modificado
- empty → fichero vacío y regular

*** OPERACIÓN**
- print (por defecto)
- delete (borrado)
- exec (ejecutar comandos)

6. copiar los shells scripts realizados en el directorio chorradas
find -name "*.sh" -exec **cp {}** otros/chorradas **\;** → ejecutar el comando cp.
find -name "*.BAT" -exec **sh pp.sh {}** \; → pasar parametros al shell script.

Fichero: pp.sh
```
# mostrar el contenido en pantalla del fichero que pasamos como parametro
echo $1 >> salida
cat $1 >> salida
ls -la $1 >> salida
#
```

Otras opciones
printf
\b Retroceso
\n Nueva línea
\t Tabulación horizontal
\v Tabulación vertical
printf "Surname: %s\nName: %s\n" "$SURNAME" "$LASTNAME"

Manual
man [nº] comando
nº → página del manual

Rename

rename → ej. rename 's/\.sh/.bat/' *

`s/ – buscar \.sh → ficheros que al final tengan “.sh”, /.bat/ → sustituir con “.bat”

Find

find . -name “ --- “ - exec grep “ --- “ {} \; -print

Juan Tonda.

ACTUALIZACIÓN: FEBRERO-MAYO 2013

RENOMBRANDO UN GRUPO DE ARCHIVOS (referencias)

1. Herramienta RENAME
2. http://www.penguintutor.com/blog/viewblog.php?blog=937
3. http://www.garron.me/go2linux/rename-bulk-files-with-linux-console-command.html
4. http://vampird.wordpress.com/2010/07/07/el-shell-de-linux-comando-rename-renombrado-masivo-de-archivos/
5. http://www.htmlpoint.com/perl/perl_11.htm (expresiones perl)

** octubre 2017 **

SELECT (ejemplo)

```
echo "Vas a eliminar el identificador: " $dni " Nombre: " $nombre
  sed $x datos.txt > agendanueva.txt
  read -p "Pulsa una tecla para continuar " pausa
  OPCIONES="Si No"
   select opt in $OPCIONES
    do
     if [[ $opt = "Si" ]]
      then
       rm datos.txt
       mv agendanueva.txt datos.txt
       break
      elif [[ $opt = "No" ]]
       then
        #echo "no se procederá a borrar"
        break
     fi
    done
```