

# BASH Shell

por Jesús Fernández (Universidad de Cantabria), Jesus.Fernandez@unican.es  
Este resumen de instrucciones para terminal bash y otros comandos útiles ha sido puesto en el dominio público. Contacte al autor para obtener la versión más reciente.  
→ 09-10-2018

## Lo + básico

**pwd** muestra la ruta actual  
**ls** lista el contenido del directorio actual  
  **-ltr** lista con detalle y de más antiguo a más nuevo  
  **-lsr** lista con detalle y de más pequeño a mas grande  
**cd** *dir* entra en el directorio *dir*  
  **-** Vuelve al último directorio  
**mkdir** *dir* crea un nuevo directorio *dir*  
**rmdir** *dir* borra el directorio *dir* (vacío)  
**rm** *fich* borra el fichero *fich*  
  **-r** (!Usar con cuidado!) borra recursivamente un directorio y todo su contenido  
  **-f** (!Usar con cuidado!) fuerza el borrado aunque no tengas permisos de escritura

**man** *com* muestra la documentación del comando *com* (Se sale pulsando q). Si no hay página de manual de un comando, muchos de ellos admiten la opción **--help** o **-h**  
**more** *fich* muestra por pantalla el contenido de un fichero de texto. Dar *return* para avanzar una línea o *espacio* para avanzar una pantalla entera. No se puede retroceder. Para ver el contenido de un fichero pudiendo retroceder, utilizar el comando **less** (no siempre disponible).

## Variables de entorno

**var=valor** crea la variable de entorno *var* y le asigna un determinado *valor*. Sólo es accesible a la shell actual.  
**export var=valor** exporta la variable de entorno *var*, de forma que es accesible no sólo en la shell actual, sino en sub-shells, ejecutables y scripts ejecutadas desde ésta.  
**env** Muestra todas las variables definidas en la shell en curso.  
**unset var** Elimina una variable previamente definida.  
**\${var}** la shell sustituye esta expresión por el valor de la variable. Existen numerosas variantes útiles:  
  **\${var//old/new}** el valor de la variable sustituyendo el texto (o ER) *old* por *new*.  
  **\${#var}** el número de caracteres de la variable.  
  **\${var:ini}**  
  **\${var:ini:long}** Muestra el valor de la variable comenzando en el carácter en la posición *ini* y mostrando únicamente *long* caracteres a partir de ese punto.  
  **\${var:-def}** Si existe la variable *var*, muestra su valor. Si no, muestra el valor por defecto *def*.

## Comandos útiles

**alias** *nombre=comando* Define un *nombre* abreviado para un *comando*. Lo normal es hacer que persista entre sesiones, por ejemplo, poniendolo en el .bashrc. se puede vitar caer en un alias poniendo por delante una contrabarra (e.g. \ls)  
**chmod** *opc fichero* Cambia los permisos de un fichero o directorio. *Opc*:  
  **g+rw** Añade permisos de lectura y escritura al grupo  
  **664** Permiso de lectura y escritura al dueño y al grupo, sólo lectura a otros  
**chown** *usuario:grupo fichero* Cambia el dueño de un fich. o directorio.  
**date** Muestra, en multitud de formatos, la fecha actual o cualquier otra que se le

indique.  
**+%Y-%m-%d %H:%M:%S** Da formato de salida a la fecha  
**-d** *fecha* Muestra esta fecha en lugar de la actual  
**-u** Muestra la hora UTC  
**diff** *fich1 fich2* Muestra las diferencias entre dos ficheros de texto y las envía a la salida estándar.  
**echo** *texto* Envía un *texto* a la salida estándar.  
  **-n** No introduce una nueva linea al acabar el texto.  
  **-e** Interpreta caracteres “escapados” en el texto (\n, \t, \a, \0nnn, \xHH)  
**file** Da información sobre el tipo de fichero.  
**find** *dir* [*opciones*] busca ficheros en el directorio *dir* y devuelve su ruta completa. Entre las opciones más útiles están:  
  **-type d|f|l** Localiza ficheros de un determinado tipo (**d**irectorios, **f**icheros, **l**inks, ...)  
  **-name ‘patrón’** busca ficheros con un determinado *patrón* en el nombre. Por ejemplo, find . -name ‘\*.py’ encuentra los ficheros con extensión py. La opción **-iname** es insensible a mayúsculas y minúsculas.  
  **-mtime -n** localiza ficheros modificados hace menos de *n* días. También se puede poner **+n**, para los que han sido modificados hace más de *n* días. Hay otra opción **-mmin** para indicar el tiempo en minutos, así como filtros por tiempo de acceso en lugar de modificación.  
  **-exec comando \;** A cada fichero encontrado le aplica el *comando* proporcionado. El comando debe incluir {} en el lugar del nombre del fichero.  
**hexdump -C** *fich* Muestra un fichero byte a byte (en hexadecimal)  
**paste** *fich1 fich2* Une dos ficheros línea a línea y los envía a la salida estándar.  
**read var1 [... varn]** Lee variables de la entrada estándar y las asigna en orden. Útil para bucles (**while read ... ; do ... done**) y “here documents” (**read ... <<< \$var**)  
**rsync -auv usuario@host:ruta** . Sincroniza el directorio actual con una ruta remota (ver ssh), copiando únicamente los ficheros que difieren y sun más recientes (-u) y preservando todos los permisos y propietarios (-a).  
  **-n** No realiza la copia. Sólo muestra los ficheros que movería.  
**scp usuario@host:ruta** . Copia segura de un fichero desde una máquina remota.  
  **-r** Copia un directorio de forma recursiva.  
  **-P** *puerto* Para usar un puerto que no sea el que usa por defecto este programa (el 22).  
**seq** *desde hasta* Genera una secuencia de valores numéricos desde un número inicial hasta otro. Si se dan 3 números, el central es el paso.  
  **-w** Iguala la anchura de la salida rellenando con ceros.  
  **-f** *formato* Controla por completo el *formato* de salida.  
**ssh** *usuario@host* Conexión segura (con un determinado nombre de *usuario*) a una terminal remota servida desde un *host* (puede ser una IP o un nombre válido).  
  **-X** Canaliza los gráficos a través de la conexión segura.  
  **-p** *puerto* Para conectarse a un puerto que no sea el que usa por defecto este programa (el 22).  
**strings** *fich* Muestra solo los caracteres ASCII imprimibles de un fichero.  
**tar** Des/empaqueta archivos y directorios en un solo archivo  
  **-cf** *fich.tar dir* crea el archivo *fich.tar* conteniendo el directorio *dir*  
  **-tf** *fich.tar* muestra el contenido del archivo *fich.tar*  
  **-xf** *fich.tar* desempaqueta el contenido del archivo *fich.tar* en el directorio actual  
  **-xzf** *fich.tar.gz* desempaqueta el contenido del archivo tar comprimido mediante gzip *fich.tar.gz* en el directorio actual (la opción **-z** no siempre está disponible)  
**wget** *url* Descarga de internet el recurso al que apunta la *url*  
**which** *ejec* muestra la ruta en la que se encuentra el fichero ejecutable *ejec*

## Tuberías y E/S

| Tubería (pipe). Conecta la salida estándar de un comando con la entrada estándar del siguiente.  
**comando** > **fichero** Envía la salida estándar de un comando a un fichero. Pisa el contenido de éste si existiese.  
**comando** >& **fichero** Envía la salida estándar y de error de un comando a un fichero. Pisa el contenido de éste si existiese.  
**comando** >> **fichero** Envía la salida estándar de un comando a un fichero. Añade el cotenido al que ya tuviera el fichero.

## Filtrando texto

Los comandos que se muestran a continuación trabajan con texto. Este puede venir por la entrada estándar o indicar un fichero como argumento de cada uno de los comandos.

**cat** Envía un fichero (o varios seguidos) a la salida estándar.  
**tail** muestra las 10 últimas líneas. **head** muestra las 10 primeras  
  **-num** muestra *num* líneas  
  **-f** muestra continuamente el final del fichero. Se ven nuevas líneas a medida que aparecen. Salir con Ctrl-C  
**sort** Ordena las líneas de un fichero alfabéticamente.  
  **-r** Ordena en orden inverso.  
  **-n** Ordena numéricamente en lugar de alfabéticamente.  
  **-k** *num* Ordena por la columna *num*.  
**cut** Recorta cada línea de un fichero y la envía a la salida estándar.  
  **-c** *desde-hasta* Indica desde qué carácter hasta cuál se recortará.  
  **-d** *delim* Indica el *delimitador* de campos (por defecto el TAB).  
  **-f** *desde-hasta* Indica qué campos recortar.  
**uniq** Elimina líneas duplicadas consecutivas.  
  **-c** Añade a la salida el número de veces que se repetía la línea.  
**wc** Cuenta el número de líneas, palabras y caracteres. Si sólo se quiere una de estas cuentas se puede especificar la opción **-l** (líneas), **-w** (palabras) o **-c** (caracteres).  
**nl** Añade números de línea.  
  **-v** *num* Empieza a numerar la primera línea por *num*.  
**tr** *origen destino* Reemplaza carácter a carácter cada elemento de *origen* por el correspondiente de *destino*.  
  **-d** *origen* Elimina estos caracteres (en lugar de reemplazarlos).  
**grep** *patrón* Selecciona sólo las líneas que cumplen con un determinado *patrón*. En el caso más sencillo, simplemente aquellas que contienen una palabra.  
  **-i** Opera de forma insensible a mayúsculas y minúsculas.  
  **-v** Selecciona las líneas que NO cumplen con el *patrón*.  
  **-E** Permite expresiones regulares complejas en el *patrón*.  
  **-l** Muestra el nombre del fichero si cumple el *patrón*.  
**awk** permite, no sólo filtrar líneas (como grep), sino también filtrar por valores numéricos, trabajar con columnas y programar de forma sencilla. Algunos casos de uso típicos:  
  **’/patrón/’** Comportamiento similar a grep.  
  **’/patrón/ {print \$3,\$5}’** Filtra las líneas que cumplen el patrón y muestra de ellas sólo las columnas 3 y 5.  
  **-F**, Esta opción permite elegir el separador de columnas (por defecto son los espacios).  
  **’\$1>=7 {print \$2}’** Muestra la segunda columna de aquellas líneas cuya primera columna es mayor o igual que 7.  
**sed** es un editor de un flujo de texto. Una operación habitual es la sustitución de texto más avanzada que con tr:  
  **-e ‘s/old/new/g’** sustituye (s) todas las apariciones (g) de un texto (old) por otro (new). el texto de origen (old) puede ser una expresión regular.

**-e '7iTEXTO'** (i)nserta un texto en la línea 7. Otros comandos: también puede (a)ñadir texto a una línea, borrar (d)elete una línea, mostrar (p)rint una línea concreta (en este caso, -n suprime el resto de salida del comando).  
**-i** (;Usar con cuidado!) Hace la sustitución directamente en el fichero, en lugar de enviarla a la salida estándar.  
**xargs comando** Hace que la entrada estándar sean los argumentos de un *comando*

## Sustitución automática en la shell

*Comillas dobles y simples*

~ Representa el directorio HOME del usuario

~**usuario** Se refiere al HOME de otro usuario

**\$(comando)** Reemplaza en este punto la salida estándar de un *comando*.

**PATH** Variable de entorno que contiene los directorios en los que la shell busca (en orden) el nombre de un comando (prevalecen alias, funciones y comando internos).

**type** Indica como un nombre sería interpretado por la shell

## Procesos

(*comando*) Ejecuta *comando* en una sub-shell

**source script** Ejecuta una *script* en la shell actual. Es como si escribiésemos los comandos directamente en la terminal o script que estemos usando.

**&** Al final de un comando, indica que este debe ejecutarse en el fondo (background), de forma que recuperamos el control de la terminal.

**jobs** lista las tareas que se están ejecutando actualmente en el fondo.

**\$?**  Muestra el código de salida del último comando.

**CTRL-Z** Detiene una tarea en ejecución. Se puede reanudar enviándola a ejecutar al fondo (bg) o recuperándola en el frente (fg).

**bg** y **fg** envían, respectivamente, una tarea al fondo (background) o al frente (foreground).

**ps aux** lista todos los procesos actualmente en ejecución. La segunda columna es el ID de proceso (PID), que nos puede servir para actuar sobre él.

**top** muestra de forma interactiva todas las tareas que se están ejecutando en el sistema. Ayuda: ? Salir: q

**kill pid** envía una señal para detener una tarea.

**wait pid** espera hasta que acabe una determinada tarea.

## Programación en *bash* (creación de *scripts*)

La shell es un lenguaje de programación completo.

```
#!/bin/bash
function mifuncion {
    sentencias
}
var1="texto"
var2=17 # Es tb.  texto
var3=$(comando)
for item in lista; do
    sentencias
done
# Comentario
if [ $var -eq valor ]; then
    # man test:  ver operadores de comparación
    sentencias
fi
case $var in
    valor1) sentencias;;
    ...
    valorn) sentencias;;
esac
echo "A salida estándar con sustitución de ${vars}"
```

## Otras herramientas

**git** es un sistema de control de versiones. Tiene un amplio abanico de subcomandos, que a su vez tienen opciones. Algunos de los más utilizados son:

**init repo** Crea un *repositorio* local nuevo (vacío, y llamado *repo*)

**clone url** Para replicar localmente un repositorio remoto (situado en *url*)

**status** Muestra el estado del directorio de trabajo.

**checkout -b nuevarama** Crea una *nuevarama* a partir del último commit

y situa en ella el directorio de trabajo

**checkout rama** Permite cambiar tu directorio de trabajo a una determinada *rama*.

**checkout commit fich** Copia un *fichero* según estaba en un determinado *commit* a tu directorio de trabajo

**commit -a** Añade todos los ficheros con modificaciones al stage y crea un nuevo commit a partir de ellos.

**push remoto rama** Actualiza una *rama* (la crea si es necesario) de un repositorio *remoto* para coincidir con la historia local.

**branch -a** Muestra todas las ramas de desarrollo

**remote -v** Muestra todos los repositorios remotos definidos

**log -num** Muestra los últimos *num* commits.

**tag -l -n** Muestra la lista de etiquetas (punteros a commits concretos) definidas

**tag -v etiqueta** Información sobre una *etiqueta* concreta

**conda** es la interfaz de línea de comandos de Anaconda o miniconda. Se trata de un sistema de gestión de paquetes de software, que permite aislar entornos de ejecución con diferentes versiones del mismo software. Inicialmente ideado para paquetes de Python, hoy en día es capaz de gestionar un creciente número de aplicaciones. Los subcomandos más habituales para conda son:

**env list** Muestra la lista de entornos disponibles. Para activar un entorno se utiliza **source activate entorno** (sin conda delante).

**create -n entorno paquetes** Crea un nuevo *entorno* e instala en él los *paquetes* que se indiquen separados por espacios. Para crear un entorno "vacío", se puede poner python como paquete.

**install paquete[=versión]** Instala un determinado paquete en el entorno activo. Se puede indicar una *versión* específica.

**install -c canal paquete** Instala un determinado paquete en el entorno activo desde un determinado *canal*. Por ejemplo,

**install -c r r-base r-essentials rstudio** nos instalaría R y Rstudio

**remove paquete** Desinstala un determinado paquete en el entorno activo.

**list -n entorno** Muestra la lista de paquetes instalados en un determinado *entorno*. Si está activo no hace falta la opción -n.