

REDIRECCIONES

output (salida estándar)

<code>tee fichero</code>	# output a fichero y a pantalla
<code>> fichero</code>	# output a fichero
<code>>> fichero</code>	# output al final del fichero
<code>> /dev/null</code>	# descarta output

error

<code>2>&1</code>	# error a output
<code>2> fichero</code>	# error a fichero
<code>2>> fichero</code>	# error al final del fichero
<code>2> /dev/null</code>	# descarta error

output y error

<code>2>&1 tee fichero</code>	# ambos a fichero y a pantalla
<code>&> fichero</code>	# ambos a fichero
<code>&>> fichero</code>	# ambos al final del fichero

VARIABLES

variables de entorno

<code>\$PWD</code>	# directorio de trabajo actual
<code>\$OLDPWD</code>	# directorio de trabajo anterior
<code>\$PPID</code>	# identificador del proceso padre
<code>\$HOSTNAME</code>	# nombre del ordenador
<code>\$USER</code>	# nombre del usuario
<code>\$HOME</code>	# directorio del usuario
<code>\$PATH</code>	# rutas búsqueda de comandos
<code>\$LANG</code>	# idioma para los mensajes
<code>\$FUNCNAME</code>	# nombre función en ejecución
<code>\$LINENO</code>	# número de línea actual (del script)
<code>\$RANDOM</code>	# número aleatorio

variables especiales

<code>\$0</code>	# nombre del script
<code>\${N}</code>	# parámetro N
<code>\$\$</code>	# identificador del proceso actual
<code>\$!</code>	# identificador del último proceso
<code>\$@</code>	# todos los parámetros recibidos
<code>\$#</code>	# número de parámetros recibidos
<code>\$? # (0=normal, >0=error)</code>	# código de retorno del último comando
<code>shift</code>	# \$1=\$2, \$2=\$3, ... \${N-1}=\${N}

ARRAYS

<code>declare -a ARRAY</code>	# declaración array
<code>ARRAY=(valor1 ... valorN)</code>	# asignación compuesta
<code>ARRAY[N]=valorN</code>	# asignación simple
<code>ARRAY=([N]=valorN valorM [P]=valorP)</code>	# asigna celdas N, M y P
<code>\${ARRAY[N]}</code>	# valor celda N
<code>\${ARRAY[*]}</code>	# todos los valores

OPERADORES

operadores aritméticos

+	# suma
-	# resta
*	# multiplicación
/	# división
%	# resto
++	# incremento
--	# decremento

operadores comparaciones numéricas

numero1 -eq numero2	# numero1 igual que numero2
numero1 -ne numero2	# numero1 distinto que numero2
numero1 -lt numero2	# numero1 menor que numero2
numero1 -le numero2	# numero1 menor o igual que numero2
numero1 -gt numero2	# numero1 mayor que numero2
numero1 -ge numero2	# numero1 mayor o igual que numero2

operadores lógicos

!	# NOT
&& , -a	# AND
 , -o	# OR

operadores de ficheros

-e fichero	# existe
-s fichero	# no está vacío
-f fichero	# normal
-d fichero	# directorio
-h fichero	# enlace simbólico
-r fichero	# permiso de lectura
-w fichero	# permiso de escritura
-x fichero	# permiso de ejecución
-0 fichero	# propietario
-G fichero	# pertenece al grupo
f1 -ef f2	# f1 y f2 enlaces mismo archivo
f1 -nt f2	# f1 más nuevo que f2
f1 -ot f2	# f1 más antiguo que f2

operadores de cadenas

-n cadena	# no vacía
-z cadena	# vacía
cadena1 = cadena2	# cadena1 igual a cadena2
cadena1 == cadena2	# cadena1 igual a cadena2
cadena1 != cadena2	# cadena1 distinta a cadena2

ENTRECOMILLADO

<code>#! RUTA</code>	<code># ruta al interprete (/bin/bash)</code>
<code>\carácter</code>	<code># valor literal del carácter</code>
<code>linea1 \ linea2</code>	<code># para escribir en varias lineas</code>
<code>'cadena'</code>	<code># valor literal cadena</code>
<code>"cadena"</code>	<code># valor literal cadena, excepto \$ ' \</code>

EXPANSIÓN

<code>[prefijo]{cad1[,...],cadN}[sufijo]</code>	<code># = precad1suf ... precadNsuf</code>
<code>\${VARIABLE:-valor}</code>	<code># si VARIABLE nula, retorna valor</code>
<code>\${VARIABLE:=valor}</code>	<code># si VARIABLE nula, asigna valor</code>
<code>\${VARIABLE:?mensaje}</code>	<code># si VARIABLE nula, mensaje error y fin</code>
<code>\${VARIABLE:inicio}</code>	<code># recorta desde inicio hasta el final</code>
<code>\${VARIABLE:inicio:longitud}</code>	<code># recorta desde inicio hasta longitud</code>
<code>\${!prefijo*}</code>	<code># nombres de variables con prefijo</code>
<code>\${#VARIABLE}</code>	<code># número de caracteres de VARIABLE</code>
<code>\${#ARRAY[*]}</code>	<code># elementos de ARRAY</code>
<code>\${VARIABLE#patrón}</code>	<code># elimina mínimo patrón desde inicio</code>
<code>\${VARIABLE##patrón}</code>	<code># elimina máximo patrón desde inicio</code>
<code>\${VARIABLE%patrón}</code>	<code># elimina mínimo patrón desde fin</code>
<code>\${VARIABLE%%patrón}</code>	<code># elimina máximo patrón desde fin</code>
<code>\${VARIABLE/patrón/reemplazo}</code>	<code># reemplaza primera coincidencia</code>
<code>\${VARIABLE//patrón/reemplazo}</code>	<code># reemplaza todas las coincidencias</code>
<code>\$((expresión))</code>	<code># sustituye expresión por su valor</code>
<code>\$[expresión]</code>	<code># sustituye expresión por su valor</code>

EJECUCIÓN

<code>./comando</code>	<code># ejecuta desde directorio actual</code>
<code>\$RUTA/comando</code>	<code># ejecuta desde cualquier sitio</code>
<code>comando</code>	<code># ejecuta si está en el \$PATH</code>
<code>. script</code>	<code># ejecuta exportando variables</code>
<code>\$(comando param1 ... paramN)</code>	<code># ejecuta de forma literal</code>
<code>`comando param1 ... paramN`</code>	<code># ejecuta sustituyendo variables</code>
<code>comando &</code>	<code># ejecuta en segundo plano</code>
<code>c1 c2</code>	<code># redirige salida c1 a entrada c2</code>
<code>c1 ; c2</code>	<code># ejecuta c1 y luego c2</code>
<code>c1 && c2</code>	<code># ejecuta c2 si c1 termina sin errores</code>
<code>c1 c2</code>	<code># ejecuta c2 si c1 termina con errores</code>

ARGUMENTOS DE LÍNEA DE COMANDOS

<pre>while getopts "hs:" option ; do case "\$option" in h) DO_HELP=1 ;; s) argument=\$OPTARG ; DO_SEARCH=1 ;; *) echo "Invalid" ; return ;; esac done</pre>	<pre># getopts + "opciones disponibles" # mientras haya argumentos # seleccionamos # -h sin opciones # -s con opciones en \$OPTARG # * error</pre>
---	---

ESTRUCTURAS DE CONTROL

if expresión1; then bloque1 elif expresión2; then bloque2 else bloque3 fi	# condicional # si expresión1 entonces # bloque1 # sino y expresión2 entonces # bloque2 # si ninguna entonces # bloque2
case VARIABLE in patrón11 ... patrón1N) bloque1 ;; patrón21 ... patrón2N) bloque2 ;; *) bloqueDefecto ;; esac	# selectiva # si VARIABLE coincide con patrones1 # entonces bloque1 # si VARIABLE coincide con patrones2 # entonces bloque2 # si ninguna # entonces bloqueDefecto
for VARIABLE in LISTA; do bloque done	# iterativa con lista # ejecuta bloque sustituyendo # VARIABLE por cada elemento de LISTA
for ((expr1; expr2; expr3;)); do bloque done	# iterativa con contador # primero se evalúa exp1 # luego mientras exp2 sea cierta # se ejecutan el bloque y expr3
while expresión; do bloque done	# bucle "mientras" # se ejecuta bloque # mientras expresión sea cierta
until expresion; do expresion done	# bucle "hasta" # se ejecuta bloque # hasta que expresión sea cierta
[function] expresion () { ... [return [valor]] ... }	# función # se invoca con # nombreFunción [param1 ... paramN]

INTERACTIVIDAD

read [-p cadena] [variable1 ...]	# input # lee teclado y asigna a variables # puede mostrarse un mensaje antes # si ninguna variable, REPLY = todo
echo cadena -n no hace salto de linea -e interpreta caracteres con \	# output # manda el valor de la cadena # a la salida estándar
printf	# output formateado (igual que C)

CONTROL DE PROCESOS

comando &	# ejecuta en segundo plano
bg númeroProceso	# continúa ejecución en segundo plano
fg númeroProceso	# continúa ejecución en primer plano
jobs	# muestra procesos en ejecución
kill señal PID1 númeroProceso1	# mata proceso(s) indicado(s)
exit código	# salir con código de retorno # (0=normal, >0=error)
trap [comando] [código1 ...]	# ejecuta comando cuando señal(es)
wait [PID1 númeroProceso1]	# espera hasta fin proceso(s) hijo(s)
nice -n prioridad comando renice -n prioridad comando	# ejecuta comando con prioridad [-20/19] # modifica prioridad comando [-20/19] # -20 máxima prioridad y 19 mínima