NOMBRE

bash - GNU Bourne-Again Shell (el Shell de Bourne otra vez, de GNU)

SINOPSIS

bash [opciones] [fichero]

DERECHOS DE COPIA

DESCRIPCIÓN

Bash es un intérprete de un lenguaje de órdenes compatible con sh que ejecuta órdenes leídas desde la entrada estándar o desde un fichero.

Bash también incorpora características útiles tomadas de los shells de Korn y C (ksh y csh).

Bash está pensado con la intención de ser una implementación conforme con la especificación POSIX de Shell y Herramientas, de la IEEE (Grupo de Trabajo 1003.2 de la IEEE).

OPCIONES

Además de las opciones de un solo carácter documentadas en la descripción de la orden interna set, bash interpreta las siguientes opciones cuando es llamado:

-c cadena Si la opción -c está presente, entonces las órdenes se leen de cadena. Si hay argumentos tras la cadena, se asignan a los parámetros posicionales, empezando por $0.

-r Si la opción -r está presente, entonces el shell se vuelve restringido (vea SHELL RESTRINGIDO más abajo).

-i Si la opción -i está presente, el shell es interactivo.

-s Si la opción -s está presente, o si no quedan argumentos tras el procesado de las opciones, entonces las órdenes se leen desde la entrada estándar. Esta opción permite definir los parámetros posicionales cuando se llama a un shell interac-

tivo.

-D Se muestra en la salida estándar una lista de cadenas de car-
acteres entrecomilladas precedidas por $. Estas cadenas son
las que están sujetas a traducción cuando la localización en curso no es C ni POSIX. Esta opción implica también la -n;
no se ejecuta ninguna orden.

-- Un – señala el fin de las opciones e inhabilita cualquier
posterior procesado de opciones. Cualesquier argumentos tras
-- se tratan como nombres de fichero y argumentos. Un argu-
mento – es equivalente a --.

Bash también interpreta una variedad de opciones multi-carácter. Estas opciones deben aparecer en la línea de órdenes antes de las opciones de un solo carácter para que puedan ser reconocidas.

--dump-po-strings
Equivalente a -D, pero la salida es en el formato de un fichero po (objeto portable), del gettext de GNU.
--dump-strings
Equivalente a -D.
--help Muestra un mensaje de modo de empleo en la salida estándar y acaba con éxito.
--login
Hace que bash actúe como si se le hubiera llamado como un shell de entrada (vea LLAMADA más abajo).
--noediting
No utiliza la biblioteca de GNU readline para leer líneas de órdenes en interactivo.
--noprofile
No lee ni el fichero de arranque de sistema /etc/profile ni ninguno de los ficheros de inicio personales ~/.bash_profile, ~/.bash_login, ni ~/.profile. Por omisión, bash lee estos ficheros cuando se le llama como un shell de entrada (vea LLA-MADA más adelante).
--norc No lee ni ejecuta el fichero de inicio personal ~/.bashrc si el shell es interactivo. Esta opción está activa de forma prede-

terminada si el shell se llama como sh.

--posix

Cambia el comportamiento de bash donde la operación normal difiera del estándar POSIX 1003.2, de forma que concuerde con éste.

--rcfile fichero

Ejecuta órdenes desde fichero en vez de desde el fichero de inicio personal estándar ~/.bashrc si el shell es interactivo (vea LLAMADA más abajo).

--restricted

El shell se vuelve restringido (vea SHELL RESTRINGIDO más abajo).

--verbose

Equivale a -v.

--version

Muestra información en la salida estándar sobre la versión de esta instanciación de bash y acaba con éxito.

ARGUMENTOS

Si quedan argumentos tras el procesado de las opciones, y no se han dado ni la opción -c ni la -s, se supone que el primer argumento es el nombre de un fichero que contiene órdenes del shell. Si bash se llama de esta manera, $0 se define con el nombre del fichero, y los parámetros posicionales se definen con los restantes argumentos. Bash lee y ejecuta órdenes de este fichero, luego acaba. El estado de salida de bash es el de la última orden ejecutada en el guión. Si no se ejecuta ninguna orden, el estado de salida es 0.

LLAMADA

Un shell de entrada es aquél cuyo primer carácter del argumento cero es un -, o uno que ha sido llamado con la opción --login.

Un shell interactivo es uno cuya entrada y salida estándares están conectadas a terminales (según determina isatty(3)), o uno que ha sido llamado con la opción -i. Se define PS1 y $- incluye i si bash es interactivo, permitiendo así a un guión del shell o a un fichero de

arranque el comprobar este estado.

Los siguientes párrafos describen cómo bash ejecuta sus ficheros de arranque. Si cualquiera de los ficheros existe pero no puede leerse, bash informa de un error. Las tildes de la ñ se expanden en nombres de ficheros como se describe más abajo en Expansión de la tilde en la sección EXPANSIÓN.

Cuando bash se llama como un shell de entrada interactivo, primero lee y ejecuta órdenes desde el fichero /etc/profile, si es que existe. Tras leer ese fichero, busca ~/.bash_profile, ~/.bash_login, y ~/.profile, en ese orden, y lee y ejecuta órdenes del primero de ellos que exista y se pueda leer. La opción --noprofile puede emplearse cuando se llame al shell para inhibir este comportamiento.
Cuando un shell de entrada termina, bash lee y ejecuta órdenes desde el fichero ~/.bash_logout, si existe.

Cuando se arranca un shell interactivo que no es de entrada, bash lee y ejecuta órdenes desde ~/.bashrc, si es que existe. Esto puede evitarse mediante la opción --norc. La opción --rcfile fichero forzará a bash a leer y ejecutar órdenes desde fichero en vez de ~/.bashrc.

Cuando bash se arranque de forma no interactiva, por ejemplo para ejecutar un guión del shell, busca la variable BASH_ENV en el entorno, expande su valor si está definida, y utiliza el valor expandido como el nombre de un fichero a leer y ejecutar. Bash se comporta como si se ejecutaran las siguientes órdenes:
if [-n "$BASH_ENV"]; then . "$BASH_ENV"; fi
pero el valor de la variable PATH no se emplea para buscar el nombre del fichero.

Si bash se llama con el nombre sh, intenta imitar el comportamiento de arranque de versiones históricas de sh tanto como sea posible, pero sin salirse del estándar POSIX. Cuando se llama como un shell interactivo de entrada, primero intenta leer y ejecutar órdenes desde /etc/profile y ~/.profile, en ese orden. La opción --noprofile puede emplearse para

inhibir este comportamiento. Cuando se llama como un shell interactivo con el nombre sh, bash busca la variable ENV, expande su valor si está definida, y emplea el valor expandido como el nombre de un fichero a leer y ejecutar. Como un shell llamado como sh no intenta leer y ejecutar órdenes desde cualquier otro fichero de arranque, la opción --rcfile no tiene efecto. Un shell no interactivo llamado con el nombre sh no intenta leer ningún fichero de arranque. Cuando se llama como sh, bash entra en modo posix después de leer los ficheros de arranque.

Cuando bash se arranca en modo posix, como ocurre cuando se da la opción de línea de órdenes --posix, sigue el estándar POSIX para los ficheros de arranque. En este modo, los shells interactivos expanden la variable ENV y se leen y ejecutan órdenes desde el fichero cuyo nombre es el valor expandido de dicha variable. No se lee ningún otro fichero de arranque.

Bash intenta determinar cuándo está siendo llamado por el demonio de shell remoto, usualmente rshd. Si bash determina que está siendo ejecutado por rshd, lee y ejecuta órdenes desde ~/.bashrc, si tal fichero existe y se puede leer. Esto no lo hará si se le llama con el nombre sh. La opción --norc puede emplearse para inhibir este comportamiento, y la opción --rcfile puede utilizarse para forzar la lectura de otro fichero, pero rshd no llama generalmente al shell con estas opciones ni permite que se den.

DEFINICIONES

Las siguientes definiciones se usan a través del resto de este documento.

Blanco Un espacio en blanco o tabulación (tab).

Palabra

Una secuencia de caracteres considerados por el shell como una sola unidad. También se conoce como un lexema (token).

Nombre Una palabra que consiste solamente en caracteres alfanuméricos y subrayados, y comienza con un carácter alfabético o un subrayado. También se llama identificador.

Metacarácter
Un carácter que, cuando no está entrecomillado, separa palabras.
Uno de los siguientes:
| & ; ( ) < > espacio tab
operador de control
Un lexema que realiza una función de control. Es uno de los siguientes símbolos:
|| & && ; ;; ( ) | <nueva-línea>

PALABRAS RESERVADAS
Palabras reservadas son palabras que tienen un significado especial para el shell. Las siguientes palabras se reconocen como reservadas cuando no van entrecomilladas y, o son la primera palabra de una orden simple (vea GRAMÁTICA DEL SHELL más abajo) o la tercera palabra de una orden case o for:

! case do done elif else esac fi for function if in select then until while { } time [[ ]]

GRAMÁTICA DEL SHELL
Órdenes simples
Una orden simple es una secuencia de asignaciones opcionales de variables seguida por palabras separadas por blancos y redirecciones, y terminadas por un operador de control. La primera palabra especifica la orden a ser ejecutada. Las palabras restantes se pasan como argumentos a la orden pedida.

El valor devuelto de una orden simple es su estado de salida, ó 128+n si la orden ha terminado debido a la señal n.

Tuberías
Una tubería es una secuencia de una o más órdenes separadas por el carácter |. El formato de una tubería es:

[time [-p]] [!] orden [| orden2 ...]

La salida estándar de orden se conecta a la entrada estándar de orden2.

Esta conexión se realiza antes que cualquier redirección especificada por la orden (vea REDIRECCIÓN más abajo).

Si la palabra reservada ! precede una tubería, el estado de salida de ésta es el NO lógico del de la última orden. De otro modo, el estado de la tubería es el de salida de la última orden. El shell espera que todas las órdenes de la tubería terminen antes de devolver un valor.

Si la palabra reservada time precede una tubería, se informa del tiempo transcurrido, así como del de usuario y sistema, consumido en la ejecución de la tubería, cuando ésta termina. La opción -p cambia el formato de salida al especificado por POSIX. La variable TIMEFORMAT puede definirse como una cadena de caracteres de formato que especifique cómo la información de tiempos debería mostrarse; vea la descripción de TIMEFORMAT bajo Variables del Shell más abajo.

Cada orden en una tubería se ejecuta como un proceso separado (esto es, en un subshell).

Listas

Una lista es una secuencia de una o más tuberías separadas por uno de los operadores ;, &, &&, o ||, y terminada opcionalmente por uno de ;, &, o <nueva-línea>.

De estos operadores de listas, && y || tienen igual precedencia, seguidos por ; y &, que tienen igual precedencia.

Si una orden se termina mediante el operador de control &, el shell ejecuta la orden en segundo plano en un subshell. El shell no espera que la orden acabe, y el estado devuelto es 0. Las órdenes separadas por un ; se ejecutan secuencialmente; el shell espera que cada orden termine, por orden. El estado devuelto es el estado de salida de la última orden ejecutada.

Los operadores de control && y || denotan listas Y (AND) y O (OR) respectivamente. Una lista Y tiene la forma

orden && orden2

orden2 se ejecuta si y sólo si orden devuelve un estado de salida 0.

Una lista O tiene la forma

orden || orden2

orden2 se ejecuta si y sólo si orden devuelve un estado de salida distinto de cero. El estado de salida de las listas Y y O es el de la última orden ejecutada en la lista.

Órdenes compuestas
Una orden compuesta es una de las siguientes:

(lista)
lista se ejecuta en un subshell. Después de que la orden se completa, las asignaciones a variables y órdenes internas que afectaran al entorno del shell no permanecen en efecto. El estado de retorno es el de salida de lista.

{ lista; }
lista se ejecuta simplemente en el entorno del shell en curso. Lista debe terminarse con un salto de línea o un punto y coma. Esto se conoce como una orden de grupo. El estado de retorno es el de salida de lista.

((expresión))
La expresión se evalúa de acuerdo a las reglas descritas abajo bajo la sección EVALUACIÓN ARITMÉTICA. Si el valor de la expresión es distinto de cero, el estado de retorno es 0; de otro modo el estado de retorno es 1. Esto es equivalente exactamente a let "expresión".

[[expresión]]
Devuelve un estado de 0 ó 1 dependiendo de la evaluación de la expresión condicional expresión. Las expresiones se componen de las primarias descritas adelante bajo EXPRESIONES CONDICIONALES.

No tienen lugar división de palabras ni expansión de nombres de camino en las palabras entre el [[y el]]; sí se realizan la expansión de tilde, expansión de parámetros y variables, expansión aritmética, sustitución de orden, sustitución de proceso y eliminación de comillas.

Cuando se emplean los operadores == y !=, la cadena a la derecha del operador se considera un patrón y se hace concordar de acuerdo a las reglas descritas más adelante bajo el epígrafe Concordancia de patrones. El valor devuelto es 0 si la cadena concuerda o no concuerda con el patrón, respectivamente, y 1 en otro caso. Cualquier parte del patrón puede entrecomillarse para forzar la concordancia como una cadena de caracteres pura.

Las expresiones pueden combinarse mediante los siguientes operadores, listados en orden decreciente de precedencia:

(expresión)
Devuelve el valor de expresión. Esto puede emplearse para cambiar la precedencia normal de los operadores.
! expresión
Verdad si expresión es falsa.
Expresión1 && expresión2
Verdad si ambas expresión1 y expresión2 son verdaderas.
Expresión1 || expresión2
Verdad si una al menos de expresión1 o expresión2 es verdad.

Los operadores && y || no ejecutan expresión2 si el valor de expresión1 es suficiente para determinar el valor de retorno de la expresión condicional entera.

For nombre [in palabra;] do lista ; done
La lista de palabras que va detrás de in se expande, generando una lista de elementos. La variable nombre se define como cada elemento de la lista en cada iteración, y lista se ejecuta cada

vez. Si la palabra de in se omite, la orden for ejecuta lista una vez para cada parámetro posicional que esté definido (vea PARÁMETROS más abajo). El estado de retorno es el de salida de la última orden que se ejecuta. Si la expansión de los elementos después del in resulta en una lista vacía, no se ejecuta ninguna orden y el estado de salida es 0.

select nombre [in palabra;] do lista ; done
La lista de palabras que sigue a in se expande, generando una lista de elementos. El conjunto de palabras expandidas se muestra en la salida estándar de errores, cada una precedida por un número. Si la palabra del in se omite, se muestran los parámetros posicionales (vea PARÁMETROS más abajo). Entonces se muestra el indicador PS3 y se lee una línea desde la entrada estándar. Si la línea consiste en un número correspondiente a una de las palabras mostradas, entonces el valor de nombre se pone a esa palabra. Si la línea está vacía, las palabras y el indicador se muestran de nuevo. Si se lee EOF, la orden se completa. Cualquier otro valor leída hace que nombre se ponga a un valor vacío. La línea leída se guarda en la variable REPLY. La lista se ejecuta tras cada selección hasta que se ejecute una orden break o return. El estado de salida de select es el de la última orden ejecutada en lista, o cero si no se ha ejecutado ninguna orden.

Case palabra in [(patrón [| patrón] ...) lista ;;] ... esac
Una orden case expande primero palabra, e intenta hacerla concordar contra cada patrón por turnos, empleando las mismas reglas de concordancia que para la expansión de nombres de caminos (vea Expansión de nombre de camino más abajo). Cuando se encuentre una concordancia, se ejecuta la lista correspondiente. Tras la primera concordancia, no se intentan más. El estado de salida es cero si no concuerda ningún patrón. De otro modo, es el estado de salida de la última orden ejecutada en lista.

If lista; then lista; [elif lista; then lista;] ... [else lista;]
fi
La lista if se ejecuta. Si su estado de salida es cero, se ejecuta la lista then. De otro modo, se ejecuta por turno cada lista elif, y si su estado de salida es cero, se ejecuta la lista then correspondiente y la orden se completa. Si no, se ejecuta la lista then si está presente. El estado de salida es el de la última orden ejecutada, o cero si ninguna condición fue verdadera.

While lista; do lista; done
until lista; do lista; done
La orden while ejecuta continuamente la lista do siempre que la última orden de lista devuelva un estado de salida cero. La orden until es idéntica a la while, excepto en que la comprobación es al revés; la lista do se ejecuta mientras que la última orden en lista devuelva un estado de salida distinto de cero. El estado de salida de las órdenes while y until es el de la última orden de la lista do ejecutada, o cero si no se ejecutó ninguna orden.

[function] nombre () { lista; }
Esto define una función llamada nombre. El cuerpo de la función es la lista de órdenes entre { y } . Esta lista se ejecuta cada vez que se especifica nombre como el nombre de una orden simple. El estado de salida de una función es el de la última orden ejecutada en el cuerpo. (Vea FUNCIONES más abajo.)

COMENTARIOS
En un shell no interactivo, o en uno interactivo en el que la opción interactive_comments de la orden interna shopt está activa (vea ÓRDENES INCORPORADAS DEL SHELL más abajo), una palabra que empiece por # hace que esa palabra y todos los caracteres que queden en esa línea no sean tenidos en cuenta. Un shell interactivo sin la opción interactive_comments habilitada, no admite comentarios. La opción interactive_comments está activa de forma predeterminada en shells interactivos.

ENTRECOMILLADO

El entrecomillado se emplea para quitar el significado especial para el shell de ciertos metacaracteres o palabras. Puede emplearse para que no se traten caracteres especiales de forma especial, para que palabras reservadas no sean reconocidas como tales, y para evitar la expansión de parámetros.

Cada uno de los metacaracteres listados a continuación bajo el epígrafe DEFINICIONES tiene un significado especial para el shell y deben ser protegidos o entrecomillados si quieren representarse a sí mismos. Hay 3 mecanismos de protección: el carácter de escape, comillas simples, y comillas dobles.

Una barra inclinada inversa no entrecomillada (\) es el carácter de escape. Preserva el valor literal del siguiente carácter que lo acompaña, con la excepción de <nueva-línea>. Si aparece un par \<nueva-línea> y la barra invertida no está ella misma entre comillas, el \<nueva-línea> se trata como una continuación de linea (esto es, se quita del flujo de entrada y no se tiene efectivamente en cuenta).

Encerrar caracteres entre apóstrofos preserva el valor literal de cada carácter entre las comillas. Una comilla simple no puede estar entre comillas simples, ni siquiera precedida de una barra invertida.

Encerrar caracteres entre comillas dobles preserva el valor literal de todos los caracteres de dentro de las comillas, con la excepción de $, `, y \. Los caracteres $ y ` mantienen sus significados especiales dentro de comillas dobles. La barra invertida mantiene su significado especial solamente cuando está seguida por uno de los siguientes caracteres: $, `, “”, \, o <nueva-línea>. Una comilla doble puede ser entrecomillada entre otras comillas dobles precediéndola de una barra invertida.

Los parámetros especiales * y @ tienen un significado especial cuando están entre comillas dobles (vea PARÁMETROS más abajo).

Las palabras de la forma $'cadena' se tratan de forma especial. La palabra se expanda a cadena, con los caracteres protegidos par barra invertida reemplazados según especifica el estándar ANSI/ISO de C. Las secuencias de escape con barra invertida, si están presentes, se descodifican como sigue:

\a alerta (campana)
\b espacio-atrás
\e un carácter de escape (ESC)
\f nueva página
\n nueva linea
\r retorno de carro
\t tabulación horizontal
\v tabulación vertical
\\ barra invertida
\nnn el carácter cuyo código es el valor octal nnn (de uno a tres dígitos)
\xnnn el carácter cuyo código es el valor hexadecimal nnn

El resultado traducido es entrecomillado con comillas simples, como si el signo de dólar no hubiera estado presente.

Una cadena entre comillas dobles precedida por un signo de dólar ($) hará que la cadena se traduzca según la localización en curso. Si ésta es C o POSIX, el signo de dólar no se tiene en cuenta. Si la cadena se traduce y reemplaza, el reemplazo se entrecomilla con comillas dobles.

PARÁMETROS

Un parámetro es una entidad que almacena valores. Puede ser un nombre, un número, o uno de los caracteres especiales listados a continuación bajo el epígrafe Parámetros especiales. En lo que se refiere al shell, una variable es un parámetro identificado por un nombre.

Un parámetro está definido si se le ha asignado un valor. La cadena vacía es un valor válido. Una vez que una variable está definida, sólo puede quitarse de la lista de variables mediante la orden interna unset (vea ÓRDENES INTERNAS DEL SHELL más adelante).

A una variable se le puede asignar un valor mediante una sentencia de la forma

nombre=[valor]

Si no se da el valor, a la variable se asigna la cadena vacía. Todos los valores están sujetos a expansión de tilde, de parámetros y variables, de cadena, de orden, aritmética, y eliminación de comillas (vea EXPANSIÓN más abajo). Si la variable tiene activado su atributo integer (vea declare más abajo en ÓRDENES INTERNAS DEL SHELL) entonces valor está sujeto a expansión aritmética incluso si no se emplea la expansión $((...)) (vea Expansión aritmética más adelante). No se realiza la división de palabras, con la excepción de "$@" como se explica más adelante bajo el epígrafe Parámetros especiales. La expansión de nombres de camino no se efectúa.

Parámetros posicionales

Un parámetro posicional es un parámetro denotado por uno o más dígitos, distintos del simple 0. Los parámetros posicionales se asignan a partir de los argumentos del shell cuando éste es llamado, y pueden ser reasignados mediante la orden interna set. Los parámetros posicionales no pueden ser asignados con sentencias de asignación. Los parámetros posicionales se reemplazan temporalmente cuando se ejecuta una función del shell (vea FUNCIONES abajo).

Cuando un parámetro posicional consistente en más de un solo dígito se expande, debe rodearse por llaves (vea EXPANSIÓN abajo).

Parámetros especiales

El shell trata de forma especial a ciertos parámetros. Éstos sólo pueden referenciarse; no se permite asignarles nada.

- **Se expande a los parámetros posicionales, empezando por 1.**

Cuando la expansión ocurre entre comillas dobles, se expande a una sola palabra con el valor de cada parámetro separado por el primer carácter de la variable especial IFS. Esto es, "$*" es equivalente a "$1c$2c...", donde c es el primer carácter del

valor de la variable IFS. Si IFS no está definida, los parámetros se separan por espacios. Si IFS es la cadena vacía, los parámetros se juntan sin ningún separador.

@ Se expande a los parámetros posicionales, empezando desde 1. Cuando la expansión ocurre dentro de comillas dobles, cada parámetro se expande a una palabra separada. Esto es, "$@" es equivalente a "$1" "$2" ... Cuando no hay parámetros posicionales, "$@" y $@ se expanden a nada (esto es, se borran).

 # Se expande al número en base 10 de parámetros posicionales.

? Se expande al estado de la tubería más recientemente ejecutada en primer plano.

- **Se expande a las opciones del shell activas actualmente según se hayan especificado en la llamada, mediante la orden interna set, o las que haya puesto el mismo shell (como la opción -i).**

$ Se expande al PID del shell. En un subshell (), se expande al PID del shell actual, no al del subshell.

! Se expande al PID de la orden más recientemente ejecutada en segundo plano (asíncronamente).

0 Se expande al nombre del shell o guión del shell. Este parámetro se pone en el inicio del shell. Si bash se llama con un fichero de órdenes, $0 se pone al nombre de ese fichero. Si bash se arranca con la opción -c, entonces $0 se pone al primer argumento tras la cadena que se va a ejecutar, si hay alguno presente. Si no, se pone al nombre de fichero empleado para llamar a bash, como se da en el argumento cero.

_ En el arranque del shell, se pone al nombre absoluto de fichero del shell o guión del shell que se está ejecutando, tal como se ha pasado en la lista de argumentos. Subsecuentemente, se expande al último argumento de la orden anterior, tras la expansión. También se pone al nombre completo del fichero de cada orden ejecutada, y se pone en el entorno exportado a esa orden. Cuando se está comprobando si hay correo nuevo, este parámetro contiene el nombre del fichero de correo o buzón que se está comprobando actualmente.

Variables del shell

El shell pone automáticamente las siguientes variables:

PPID El PID del proceso padre del shell. Esta variable es de lectura exclusiva.

PWD El directorio de trabajo actual como lo pone la orden cd.

OLDPWD El directorio de trabajo anterior como lo puso la orden cd.

REPLY La línea de entrada leída por la orden interna read cuando no se le dan argumentos.

UID Se expande al UID del usuario en curso, puesta en el arranque del shell. Esta variable es de lectura exclusiva.

EUID Se expande al UID efectivo del usuario en curso, puesta en el arranque del shell. Esta variable es de lectura exclusiva.

GROUPS Una variable vector conteniendo la lista de grupos de los que el usuario actual es miembro. Esta variable es de lectura exclusiva.

BASH Se expande al nombre completo del fichero empleado para llamar a esta instancia de bash.

BASH_VERSION

Se expande a una cadena que describe la versión de esta instancia de bash.

BASH_VERSINFO

Una variable vector de lectura exclusiva cuyos miembros contienen información de versión para esta instancia de bash. Los valores asignados a los miembros del vector son como sigue:

BASH_VERSINFO[0] El número mayor de versión (la distribución).

BASH_VERSINFO[1] El número menor de versión (la versión).

BASH_VERSINFO[2] El nivel de parcheo.

BASH_VERSINFO[3] La versión de construcción.

BASH_VERSINFO[4] El estado de la distribución (por ejemplo, beta1).

BASH_VERSINFO[5] El valor de MACHTYPE.

SHLVL Se incrementa en uno cada vez que se arranca una nueva instancia de bash.

RANDOM Cada vez que este parámetro sea referenciado, se genera un entero aleatorio entre 0 y 32767. La secuencia de números aleatorios puede iniciarse asignando un valor a RANDOM. Si RANDOM no está definido, pierde sus propiedades especiales, incluso si posteriormente es redefinido.

SECONDS
Cada vez que este parámetro es referenciado, se devuelve en él el número de segundos transcurridos desde la llamada al shell. Si se asigna un valor a SECONDS, el valor devuelto en posteriores referencias es el número de segundos desde la asignación más el valor asignado. Si SECONDS no está definido, pierde sus propiedades especiales, incluso si posteriormente es redefinido.

LINENO Cada vez que este parámetro es referenciado, el shell sustituye un número en base 10 representando el número de línea secuencial actual (empezando por 1) dentro de un guión o función. Si no estamos en un guión o función, no se garantiza que el valor sustituido tenga significado. Si LINENO no está definido, pierde sus propiedades especiales, incluso si posteriormente es redefinido.

HISTCMD
El número de "historia", o índice en la lista "histórica", de la orden actual. Si HISTCMD no está definido, pierde sus propiedades especiales, incluso si posteriormente es redefinido.

DIRSTACK
Una variable vector (vea Vectores más abajo) que aloja los contenidos actuales de la pila de directorios. Los directorios aparecen en la pila en el orden en el que se muestran con la orden interna dirs. La asignación a miembros de este vector puede emplearse para modificar directorios que ya estén en la pila, pero entonces deben utilizarse las órdenes internas pushd y popd para añadir y quitar directorios. La asignación a esta

variable no cambiará el directorio de trabajo. Si DIRSTACK no está definido, pierde sus propiedades especiales, incluso si posteriormente es redefinido.

PIPESTATUS
Una variable vector (vea Vectores más abajo) que contiene una lista de valores de estado de salida de los procesos en la tubería en primer plano ejecutada más recientemente (que puede contener una sola orden).

OPTARG El valor del último argumento que es una opción procesado por la orden interna getopts (vea ÓRDENES INTERNAS DEL SHELL más abajo).

OPTIND El índice del siguiente argumento a ser procesado por la orden interna getopts (vea ÓRDENES INTERNAS DEL SHELL más abajo).

HOSTNAME
Puesto automáticamente al nombre del anfitrión (computador) actual.

HOSTTYPE
Puesto automáticamente a una cadena que describe de forma unívoca el tipo de máquina en la que bash se está ejecutando. El valor predefinido depende del sistema.

OSTYPE Puesto automáticamente a una cadena que describe el sistema operativo en el que bash se está ejecutando. El valor predefinido depende del sistema. En Linux es “linux”.

MACHTYPE
Puesto automáticamente a una cadena que describe completamente el tipo de sistema en el que bash se está ejecutando, en el formato estándar de GNU cpu-compañía-sistema. El valor predefinido depende del sistema.

SHELLOPTS

Una lista, de elementos separados por dos puntos, de opciones activas del shell. Cada palabra en la lista es un argumento válido para la opción -o de la orden interna set (vea ÓRDENES INTERNAS DEL SHELL abajo). Las opciones que aparecen en SHELLOPTS son aquéllas que aparecen como on en set -o. Si esta variable está en el ambiente cuando bash empieza, cada opción del shell en la lista se activará antes de leer cualquier fichero de inicio. Esta variable es de lectura exclusiva.

El shell hace uso de las siguientes variables. En algunos casos, bash asigna un valor predeterminado a una variable; estos casos se dicen abajo.

IFS El Separador Interno de Campo que se emplea para la división de palabras tras la expansión y para dividir líneas en palabras con la orden interna read. El valor predeterminado es `<espacio><tab><nueva-línea>".

PATH El camino de búsqueda para órdenes, programas ejecutables. Es una lista de directorios separados por dos puntos en los cuales el shell busca órdenes (vea EJECUCIÓN DE ÓRDENES más abajo). El camino predeterminado depende del sistema, y lo pone el administrador que instala bash. Un valor común es `/usr/local/bin:/bin:/usr/bin:.".

HOME El directorio inicial de trabajo del usuario en curso; el argumento predeterminado para la orden interna cd. El valor de esta variable se usa también cuando se realiza la expansión de tilde.

CDPATH El camino de búsqueda para la orden cd. Es una lista de directorios separados por dos puntos en los cuales el shell busca directorios destino especificados por la orden cd. Un valor de muestra es `.::/usr".

BASH_ENV

Si este parámetro está definido cuando bash está ejecutando un guión del shell, su valor se interpreta como un nombre de fichero que contiene órdenes para iniciar el shell, como en ~/.bashrc. El valor de BASH_ENV está sujeto a expansión de parámetros, sustitución de órdenes y expansión aritmética, antes

de ser interpretado como un nombre de fichero. PATH no se usa para buscar el nombre de fichero resultante.

MAIL Si este parámetro está puesto a un nombre de fichero y la variable MAILPATH no está definida, bash informa al usuario de la llegada de correo en el fichero especificado.

MAILCHECK

Especifica cuán a menudo (en segundos) bash comprueba si hay correo nuevo. El valor predeterminado es 60 s. Cuando es tiempo de comprobar si hay correo, el shell lo hace antes de mostrar el indicador primario. Si esta variable no está definida, el shell no comprueba si hay correo nuevo.

MAILPATH

Una lista de nombres de fichero separados por dos puntos donde hay que comprobar si hay correo nuevo. El mensaje que haya que mostrar cuando llegue correo a un fichero particular puede especificarse separando el nombre de fichero del mensaje con un '?'. Cuando se use en el texto del mensaje, $_ se expande al nombre del fichero de correo en curso. Ejemplo:

MAILPATH='/var/spool/mail/bfox?"Tienes carta":~/shell-mail?"¡$_ tiene carta!"'

Bash proporciona un valor predeterminado para esta variable, pero la localización de los ficheros de correo del usuario que emplea es dependiente del sistema (e.g., /var/spool/mail/$USER).

PS1 El valor de este parámetro se expande (vea INDICADORES abajo) y se emplea como la cadena del indicador primario. El valor predeterminado es '\s-\v\$ '.

PS2 El valor de este parámetro se expande como con PS1 y se emplea como la cadena del indicador secundario. El valor predeterminado es '> '.

PS3 El valor de este parámetro se emplea como el indicador para la orden select (vea GRAMÁTICA DEL SHELL más arriba).

PS4 El valor de este parámetro se expande como con PS1 y el valor se imprime antes de cada orden que bash muestra durante una traza de ejecución. El primer carácter de PS4 se replica múltiples veces, tantas como sean necesarias, para indicar múltiples nive-

les de indirección. El valor predeterminado es `+ ''.

TIMEFORMAT

El valor de este parámetro se emplea como una cadena de formato para especificar cómo debe mostrarse la información de tiempos para tuberías precedidas por la palabra reservada time. El carácter % introduce una secuencia de escape que se expande a un valor de tiempo o a otra información. Las secuencias de escape y sus significados son como sigue; los corchetes denotan partes opcionales.

%% Un % literal.

%[p][l]R El tiempo total transcurrido en segundos.

%[p][l]U El número de segundos de CPU gastados en modo usuario.

%[p][l]S El número de segundos de CPU gastados en modo sistema.

%P El porcentaje de CPU, computado como (%U + %S) ÷ %R.

La p opcional es un dígito que especifica la precisión, el número de decimales. Un valor de 0 hace que no se muestre ningún decimal, ni el punto o coma decimal. Como mucho se pueden especificar tres decimales; valores de p mayores de 3 se cambian a 3. Si p no se especifica, se usa precisamente el valor 3.

La l opcional especifica un formato más largo, incluyendo minutos, en la forma MmmSS.Ffs. El valor de p determina si se incluye o no la fracción.

Si esta variable no está definida, bash actúa como si tuviera el valor $'\nreal\t%3lR\nuser\t%3lU\nsys%3lS'. Si el valor es nulo, no se muestra ninguna información de tiempos. Se añade un salto de línea al final cuando se muestra la cadena de formato.

HISTSIZE

El número de órdenes a recordar en la historia de órdenes (vea HISTORIA abajo). El valor predeterminado es 500.

HISTFILE

El nombre del fichero en el que se guarda la historia de órdenes (vea HISTORIA abajo). El valor predeterminado es ~/.bash_history. Si no está definido, no se guarda la historia de órdenes cuando se acaba un shell interactivo.

HISTFILESIZE
El número máximo de líneas contenidas en el fichero de historia. Cuando se asigna un valor a esta variable, el fichero de historia se trunca, si es menester, para contener no más de ese número de líneas. El valor predeterminado es 500. El fichero de historia se trunca también a este tamaño tras escribir en él cuando un shell interactivo termina.

OPTERR Si se pone al valor 1, bash muestra mensajes de error generados por la orden interna getopts (vea ÓRDENES INTERNAS DEL SHELL abajo). OPTERR se inicia a 1 cada vez que se llama al shell o cuando se ejecuta un guión del shell.
LANG Empleado para determinar la categoría de localización ("escenario") para cualquier categoría no seleccionada específicamente con una variable de las que empiezan por LC_.

LC_ALL Esta variable tiene preferencia sobre el valor de LANG y de cualquier otra variable de las que empiecen por LC_ especificando una categoría de localización.

LC_COLLATE
Esta variable determina el orden de clasificación empleado cuando se ordene el resultado de una expansión de nombres de caminos, y determina el comportamiento de expresiones de rango, clases de equivalencia, y secuencias de clasificación dentro de expansiones de nombres de caminos y concordancia de patrones.

LC_CTYPE
Esta variable determina la interpretación de caracteres y el comportamiento de clases de caracteres dentro de expansiones de nombres de caminos y concordancia de patrones.

LC_MESSAGES
Esta variable determina la localización empleada para traducir cadenas entrecomilladas con comillas dobles precedidas por un $.

PROMPT_COMMAND
Si está definido, el valor se ejecuta como una orden antes de mostrarse cada indicador primario.

IGNOREEOF
Controla la acción de un shell interactivo al recibir un carácter EOF como sola entrada. Si está definido, el valor es el número de caracteres EOF consecutivos que deben teclearse como los primeros caracteres de una línea de entrada antes de que bash acabe. Si la variable existe pero no tiene un valor numérico, o ninguno, el valor predeterminado es 10. Si no existe, EOF significa el final de la entrada para el shell.

TMOUT Si se define a un valor mayor que cero, el valor se interpreta como el número de segundos que hay que esperar una entrada después de mostrarse el indicador primario. Bash termina después de esperar ese número de segundos si no recibe ninguna entrada.

FCEDIT El editor predeterminado para la orden interna fc.

FIGNORE
Una lista de sufijos separados por dos puntos que no hay que tener en cuenta cuando se realice una terminación de nombres de ficheros (vea READLINE abajo). Un nombre de fichero cuyo sufijo concuerde con una de las entradas en FIGNORE se excluye de la lista de nombres de ficheros a completar. Un valor de muestra es `.o:~".

GLOBIGNORE
Una lista de patrones separados por dos puntos que definen en conjunto de nombres de ficheros que no hay que tener en cuenta

en la expansión de nombres de caminos. Si un nombre de fichero que concordaba en un patrón de expansión de nombres de caminos también concuerda con uno de los patrones en GLOBIGNORE, se quita de la lista de concordancias.

INPUTRC

El nombre de fichero para el de inicio de readline, en vez del predeterminado ~/.inputrc (vea READLINE abajo).

HISTCONTROL

Si se define a un valor de ignorespace, las líneas que comiencen con un carácter espacio no se meten en la lista de historia. Si se pone a un valor de ignoredups, las líneas que concuerden con la última línea de la historia, no se meten. Un valor de ignoreboth combina las dos opciones. Si no está definido, o si lo está a otro valor distinto de los de antes, todas las líneas leídas por el analizador léxico se guardan en la lista de la historia, sujeto esto al valor de HISTIGNORE. La función de esta variable ha sido tomada por HISTIGNORE. La segunda línea y siguientes de una orden compuesta multi-línea no se comprueban, y se añaden a la historia sin importar el valor de HISTCONTROL.

HISTIGNORE

Una lista de patrones separados por dos puntos empleados para decidir qué líneas de órdenes deben guardarse en la lista de historia. Cada patrón se ancla al principio de la línea y debe especificar la línea completamente (no se añade ningún `*'). Cada patrón se comprueba con la línea tras aplicarse las comprobaciones especificadas por HISTCONTROL. Además de los caracteres normales de concordancia de patrones del shell, `&' concuerda con la línea de historia anterior. `&' puede protegerse empleando una barra inversa. Ésta se quita antes de intentarse una concordancia. La segunda línea y siguientes de una orden compuesta multi-línea no se comprueban, y se añaden a la historia sin importar el valor de HISTIGNORE.

Histchars
Los dos o tres caracteres que controlan la expansión y separación en lexemas de la historia (vea EXPANSIÓN DE HISTORIA abajo). El primer carácter es el carácter de expansión de historia, el carácter que señala el comienzo de una expansión de historia, normalmente '!'. El segundo carácter es el carácter de sustitución rápida, que se usa como una abreviatura para reejecutar la orden anterior, sustituyendo una cadena por otra en la orden. El valor predeterminado es '^'. El tercer carácter, opcional, es el carácter que indica que el resto de la línea es un comentario cuando se encuentra como el primer carácter de una palabra, normalmente es '#'. El carácter de comentario de historia hace que la sustitución de historia se salte en el resto de palabras de la línea. No hace que necesariamente el analizador léxico del shell trate al resto de la línea como un a un comentario.

HOSTFILE
Contiene el nombre de un fichero con el mismo formato que /etc/hosts que debería leerse cuando el shell necesite completar un nombre de anfitrión (computador). El fichero puede ser cambiado interactivamente; la siguiente vez que se intente completar un nombre de computador, bash añade el contenido del nuevo fichero a la base de datos ya existente.

auto_resume
Esta variable controla cómo el shell interactúa con el usuario para el control de trabajos. Si esta variable está definida, las órdenes simples de una palabra sin redirección se tratan como candidatas para reanudar un trabajo existente parado. No se permite ninguna ambigüedad; si hay más de un trabajo que empiece con la cadena tecleada, se selecciona el trabajo al que se ha accedido más recientemente. El nombre de un trabajo parado, en este contexto, es la línea de órdenes empleada para arrancarlo. Si se define al valor exact, la cadena suministrada debe concordar exactamente con el nombre de un trabajo parado; si se define

como substring, la cadena suministrada necesita concordar con una subcadena del nombre de un trabajo parado. El valor substring proporciona una funcionalidad análoga a la del identificador de trabajo %? (vea CONTROL DE TRABAJOS adelante). Si se pone a cualquier otro valor, la cadena suministrada debe ser un prefijo del nombre de un trabajo parado; esto proporciona una funcionalidad análoga a la del identificador de trabajo %.

Vectores

Bash proporciona variables vectores, monodimensionales. Cualquier variable puede usarse como un vector; la orden interna declare declarará un vector explícitamente. No hay un límite máximo en el tamaño de un vector, ni ningún requisito para que los miembros se indexen o asignen de forma contigua. Los vectores se indexan empleando enteros y su primer elemento es el de índice cero, como en C.

Un vector se crea automáticamente si se asigna algo a una variable con la sintaxis nombre[índice]=valor. El índice se trata como una expresión aritmética que debe evaluarse a un número mayor o igual a cero. Para declarar un vector explícitamente, emplee declare -a nombre (vea ÓRDENES INTERNAS DEL SHELL abajo). También se acepta declare -a nombre[índice], donde el índice no se tiene en cuenta. Se pueden especificar atributos para una variable vector mediante las órdenes internas declare y readonly. Cada atributo se aplica a cada uno de los miembros del vector.

Se asignan valores a los vectores mediante asignaciones compuestas de la forma nombre=(valor1 ... valorn), donde cada valor es de la forma [índice]=cadena. Sólo cadena es necesario. Si se suministra el índice entre corchetes y la asignación, se asigna a ese índice; si no, el índice del elemento asignado es el último índice al que la sentencia le asignó algo, más uno. Los índices empiezan en cero. Esta sintaxis también la acepta la orden interna declare. Se puede asignar a los elementos individuales del vector empleando la sintaxis nombre[índice]=valor, ya presentada antes.

Cualquier elemento de un vector puede referenciarse mediante ${nom-

bre[índice]}. Las llaves son necesarias para evitar conflictos con la expansión de nombres de caminos. Si índice es @ o *, la palabra se expande a todos los miembros de nombre. Estos índices difieren solamente cuando la palabra aparece entre comillas dobles. Si la palabra está entre comillas dobles, ${nombre[*]} se expande a una sola palabra con el valor de cada miembro del vector separados por el primer carácter de la variable especial IFS, y ${nombre[@]} expande cada elemento de nombre a una palabra separada. Cuando no hay miembros del vector, ${nombre[@]} se expande a nada. Esto es análogo a la expansión de los parámetros especiales * y @ (vea Parámetros especiales arriba). ${#nombre[índice]} se expande a la longitud de ${nombre[índice]}. Si índice es * o @, la expansión es el número de elementos del vector. Referenciar una variable vector sin índice es equivalente a referenciar el elemento cero.

La orden interna unset se emplea para destruir vectores. Unset nombre[índice] destruye el elemento del vector con el índice especificado. Unset nombre, donde nombre es un vector, o unset nombre[índice], donde índice es * o @, borra el vector entero.

Las órdenes internas declare, local, y readonly aceptan cada una una opción -a para especificar un vector (array). La orden interna read acepta una -a para asignar una lista de palabras leídas desde la entrada estándar a un vector. Las órdenes internas set y declare muestran valores de un vector en una manera tal que les permite ser reutilizadas como asignaciones.

EXPANSIÓN

La expansión se realiza en la línea de órdenes una vez que la orden ha sido dividida en palabras. Hay siete clases de expansión: expansión de llaves, expansión de tilde, expansión de parámetro y variable, sustitución de orden, expansión aritmética, división de palabras, expansión de nombre de camino.

La orden de las expansiones es: expansión de llaves, de tilde, de parámetro, variable y aritmética, y sustitución de orden (hechas de

izquierda a derecha), división de palabras, y expansión de nombre de camino.

En sistemas que puedan admitirla, hay una expansión adicional disponible: sustitución de proceso.

Sólo la expansión de llaves, división de palabras, y expansión de nombre de camino, pueden cambiar el número de palabras de la expansión; las otras expanden una palabra simple a otra palabra simple. Las únicas excepciones a esto son las expansiones de "$@" y "${nombre[@]}" como se ha explicado más arriba (vea PARÁMETROS).

Expansión de llaves

La expansión de llaves es un mecanismo por el cual pueden generarse cadenas arbitrarias. Este mecanismo es similar a la expansión de nombre de camino, pero los nombres de ficheros no necesitan existir. Los patrones a ser expandidos con la expansión de llaves toman la forma de un preámbulo opcional seguido por una serie de cadenas separadas por comas entre un par de llaves, seguido por un post scriptum opcional. El preámbulo sirve de prefijo a cada cadena de entre las llaves, y el post scriptum se añade luego a cada cadena resultante, expandiendo de izquierda a derecha.

Las expansiones de llaves pueden anidarse. Los resultados de cada cadena expandida no se ordenan; se preserva el orden de izquierda a derecha. Por ejemplo, a{d,c,b}e se expande a`ade ace abe'.

La expansión de llaves se realiza antes que cualquier otra, y cualquier carácter especial para otras expansiones se preserva en el resultado. Es estrictamente textual. Bash no aplica ninguna interpretación sintáctica al contexto de la expansión ni al texto entre las llaves.

Una expansión de llaves correctamente formada debe contener llaves de apertura y cierre sin entrecomillar, y por lo menos una coma sin entrecomillar. Cualquier expansión de llaves formada incorrectamente se deja tal cual. Una { o , puede protegerse con una barra invertida para evitar que sea considerada como parte de una expansión de llaves.

Esta construcción se emplea normalmente como una abreviatura cuando el prefijo común de las cadenas a generar es mayor que en el ejemplo de antes:

```
mkdir /usr/local/src/bash/{old,new,dist,bugs}
o
chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}}
```

La expansión de llaves introduce una pequeña incompatibilidad con versiones históricas de sh. Sh no trata especialmente a las llaves de apertura o cierre cuando aparecen como parte de una palabra, y las deja en la entrada. Bash quita las llaves de palabras, como una consecuencia de la expansión de llaves. Por ejemplo, una palabra introducida a sh como fichero{1,2} aparece así mismo en la entrada. La misma palabra se muestra como fichero1 fichero2 tras la expansión hecha por bash. Si se desea una compatibilidad estricta con sh, arranque bash con la opción +B o inhabilite la expansión de llaves con la opción +B de la orden interna set (vea ÓRDENES INTERNAS DEL SHELL abajo).

Expansión de tilde

Si una palabra comienza con un carácter tilde de la ñ sin entrecomillar (~), todos los caracteres que precedan a la primera barra inclinada sin entrecomillar (o todos los caracteres, si no hay ninguna barra inclinada sin proteger), se consideran un prefijo-tilde. Si ninguno de los caracteres en el prefijo-tilde están protegidos, los caracteres en el prefijo-tilde que siguen a la tilde se tratan como un posible nombre de entrada de usuario (login). Si este nombre de entrada es la cadena vacía, la tilde se reemplaza con el valor del parámetro del shell HOME. Si HOME no está definida, se sustituye en su lugar el directorio de trabajo inicial del usuario que está ejecutando el shell. De otra forma, el prefijo-tilde se reemplaza con el directorio de trabajo inicial (de casa) asociado con el nombre de entrada especificado.

Si el prefijo-tilde es un '~+', el valor de la variable del shell PWD reemplaza al prefijo-tilde. Si el prefijo-tilde es un '~-', el valor de la variable del shell OLDPWD, si está definido, se sustituye. Si

los caracteres que siguen a la tilde en el prefijo-tilde consisten en un número N, prefijado opcionalmente por un `+' o un `-', el prefijo-tilde se reemplaza con el elemento correspondiente de la pila de directorios, como lo mostraría la orden interna dirs llamada con el prefijo-tilde como argumento. Si los caracteres tras la tilde en el prefijo-tilde consisten en un número sin un `+' ni `-' iniciales, se supone `+'.

Si el nombre de entrada es inválido, o si la expansión de tilde falla, la palabra se deja tal cual.

A cada asignación de variable se comprueba si hay prefijos-tilde sin entrecomillar inmediatamente tras un : o un =. En estos casos, la expansión de tilde también tiene lugar. Consecuentemente, uno puede usar nombres de ficheros con tildes en asignaciones a PATH, MAILPATH, y CDPATH, y el shell asigna el valor expandido.

Expansión de parámetro

El carácter `$' introduce la expansión de parámetro, sustitución de orden, o expansión aritmética. El nombre de parámetro o símbolo a ser expandido puede estar encerrado entre llaves, que son opcionales pero sirven para proteger a la variable en la expansión de caracteres que la sigan y puedan interpretarse como parte de su nombre.

Cuando se empleen llaves, la de cierre es la primera `}' no protegida con una barra invertida o en una cadena entrecomillada, y no dentro de una expansión aritmética empotrada, sustitución de orden, o expansión de parámetro.

${parámetro}

Se sustituye el valor de parámetro. Se requieren llaves cuando parámetro es un parámetro posicional de más de un dígito, o cuando parámetro viene seguido por un carácter que no debe interpretarse como parte de su nombre.

Si el primer carácter de parámetro es un signo de cierre de exclamación, se introduce un nivel de indirección de variable. Bash emplea

el valor de la variable formada a partir del resto de parámetro como el nombre de la variable; luego, esta variable se expande y ese valor se emplea en el resto de la sustitución, en vez del propio valor de parámetro. Esto se conoce como expansión indirecta.

En cada uno de los casos anteriores, palabra está sujeta a expansión de tilde, expansión de parámetro, sustitución de orden, y expansión aritmética. Cuando no se realiza la expansión de subcadena, bash comprueba si un parámetro está sin definir o es nulo; si omitimos los dos puntos la comprobación sólo es para la no definición del parámetro.

${parámetro:-palabra}
Emplear valores predeterminados. Si parámetro no está definido o está vacío, se sustituye la expansión de palabra. De otro modo, se sustituye el valor de parámetro.
${parámetro:=palabra}
Asignar valores predeterminados. Si parámetro no está definido o es nulo, la expansión de palabra se asigna a parámetro. Luego, el valor de parámetro se sustituye. No se puede asignar nada de esta manera a los parámetros posicionales ni a los especiales.
${parámetro:?palabra}
Muestra un error si no está definido o está vacío. Si parámetro es nulo o no está definido, la expansión de palabra (o un mensaje a tal efecto si palabra no está presente) se escribe en la salida estándar de errores y el shell, si no es interactivo, acaba. De otra manera, se sustituye el valor de parámetro.
${parámetro:+palabra}
Emplear un valor alternativo. Si parámetro está vacío o no está definido, no se sustituye nada; de otro modo, se sustituye la expansión de palabra.
${parámetro:desplazamiento}
${parámetro:desplazamiento:longitud}
Expansión de subcadena. Expande hasta longitud caracteres de parámetro, empezando en los caracteres especificados por desplazamiento. Si se omite la longitud, se expande a la subca-

dena de parámetro, empezando en el carácter especificado por desplazamiento. Longitud y desplazamiento son expresiones aritméticas (vea EVALUACIÓN ARITMÉTICA más abajo). longitud debe evaluarse a un número mayor o igual a cero. Si desplazamiento se evalúa a un número menor que cero, el valor se emplea como un desplazamiento desde el final del valor de parámetro. Si parámetro es @, el resultado es longitud parámetros posicionales empezando en desplazamiento. Si parámetro es un nombre de vector indexado por @ o *, el resultado es longitud miembros del vector empezando con ${parámetro[desplazamiento]}. La indexación de la subcadena empieza por cero a menos que se empleen los parámetros posicionales, en cuyo caso empieza por 1.

${#parámetro}
Se sustituye la longitud en caracteres del valor de parámetro. Si parámetro es * o @, el valor sustituido es el número de parámetros posicionales. Si parámetro es un nombre de vector indexado por * o @, el valor sustituido es el número de elementos en el vector.

${parámetro#palabra}
${parámetro##palabra}
La palabra se expande para producir un patrón lo mismo que en una expansión de nombre de camino. Si el patrón concuerda con el principio del valor de parámetro, entonces el resultado de la expansión es el valor expandido de parámetro con el patrón más corto que concuerde (el caso `#) o con el patrón más largo que concuerde (el caso `##") eliminado. Si parámetro es @ o *, la operación de borrado del patrón se aplica a cada parámetro posicional por turnos, y la expansión es la lista resultante. Si parámetro es una variable vector indexada con @ o *, la operación de borrado del patrón se aplica a cada miembro del vector por orden, y la expansión es la lista resultante.

${parámetro%palabra}
${parámetro%%palabra}

La palabra se expande para producir un patrón justo como en la expansión de nombre de camino. Si el patrón concuerda con una porción del final del valor expandido de parámetro, entonces el resultado de la expansión es el valor expandido de parámetro con el patrón más corto que concuerde (el caso "%") o el más largo (el caso "%%") borrado. Si parámetro es @ o *, la operación de borrado del patrón se aplica a cada parámetro posicional por orden, y la expansión es la lista resultante. Si parámetro es una variable vector indexada por @ o *, la operación de borrado del patrón se aplica a cada miembro del vector por orden, y la expansión es la lista resultante.

${parámetro/patrón/cadena}
${parámetro//patrón/cadena}
El patrón se expande para producir un patrón justo como en la expansión de nombre de camino. Parámetro se expande y la concordancia más larga de patrón contra este valor se reemplaza por cadena. En la primera forma, sólo se reemplaza la primera concordancia. La segunda forma hace que todas las concordancias de patrón se reemplacen con cadena. Si patrón empieza por #, debe concordar con el principio de la cadena. Si patrón empieza por %, debe concordar con el final de la cadena. Si cadena está vacía, las concordancias de patrón se borran y el / que sigue al patrón puede omitirse. Si parámetro es @ o *, la operación de sustitución se aplica a cada parámetro posicional por orden, y la expansión es la lista resultante. Si parámetro es una variable vector indexada por @ o *, la operación de sustitución se aplica a cada miembro del vector por orden, y la expansión es la lista resultante.

Sustitución de orden
La sustitución de orden permite a la salida de una orden reemplazar al nombre de la orden. Hay dos formas:

$(orden)
u

`orden`

Bash realiza la expansión ejecutando orden y reemplazando la sustitución de orden con la salida estándar de la orden, quitando los saltos de línea finales. Los saltos de línea empotrados no se borran, pero pueden ser eliminados durante la división de palabras. La sustitución de orden $(cat fichero) puede reemplazarse por lo equivalente pero más rápido $(< fichero).

Cuando se emplea la forma de sustitución al viejo estilo con las comillas inversas, la barra invertida mantiene su significado literal excepto cuando es seguida por $, `, o \. La primera comilla inversa no precedida por una barra invertida termina la sustitución de orden. Cuando se emplea la forma $(orden), todos los caracteres entre los paréntesis forman parte de la orden; ninguno se trata especialmente.

Las sustituciones de orden pueden anidarse. Para anidar cuando se emplee la forma de comillas inversas, proteja las comillas inversas internas con barras inversas.

Si la sustitución aparece dentro de las dobles comillas, la división de palabras y la expansión de nombres no se realiza en los resultados.

Expansión aritmética

La expansión aritmética permite la evaluación de una expresión aritmética y la sustitución del resultado. El formato para la expansión aritmética es:

$((expresión))

La expresión se trata como si estuviera entre comillas dobles, pero un signo de doble comilla dentro de los paréntesis no se trata especialmente. Todos los lexemas en la expresión están sujetos a expansión de parámetro, expansión de cadena, sustitución de orden, y eliminación de comillas. Las sustituciones aritméticas pueden anidarse.

La evaluación se realiza de acuerdo a las reglas listadas más abajo en

EVALUACIÓN ARITMÉTICA. Si expresión no es válida, bash imprime un mensaje indicando el fallo y no ocurre ninguna sustitución.

Sustitución de proceso

La sustitución de proceso se admite en sistemas que acepten tuberías con nombre (FIFOs) o el método /dev/fd de nombrar ficheros abiertos. Toma una de las formas siguientes: <(lista) o >(lista). El proceso lista se ejecuta con su entrada o salida conectada a un FIFO o a algún fichero en /dev/fd. El nombre de este fichero se pasa como argumento a la orden en curso como el resultado de la expansión. Si se emplea la forma >(lista), escribir en el fichero proporciona la entrada para lista. Si se emplea la forma <(lista), el fichero pasado como argumento deberá leerse para obtener la salida de lista.

Cuando esté disponible, se realiza la sustitución de proceso simultáneamente con la expansión de parámetro y variable, sustitución de orden y expansión aritmética.

División de palabras

El shell examina los resultados de la expansión de parámetro, sustitución de orden y expansión aritmética que no ocurrieron dentro de comillas dobles para realizar la división de palabras.

El shell trata cada carácter de IFS como un delimitador, y divide los resultados de las otras expansiones en palabras separadas por estos caracteres. Si IFS no está definido, o su valor es exactamente <espacio><tab><nueva-línea>, el valor predeterminado, entonces cualquier secuencia de caracteres de IFS sirve para delimitar palabras. Si IFS tiene algún otro valor, entonces las secuencias de los caracteres blancos espacio y tabulador no se tienen en cuenta al principio y al final de la palabra, siempre que el carácter de espacio en blanco esté en el valor de IFS (un carácter de espacio en blanco de IFS). Cualquier carácter en IFS que no sea espacio en blanco de IFS, junto con cualquier carácter de espacio en blanco adyacente de IFS, delimita un campo. Una secuencia de caracteres de espacio en blanco de IFS también se trata como un delimitador. Si el valor de IFS es nulo, no se real-

iza la división de palabras.

Los argumentos nulos explícitos (”” o ‘’) se mantienen. Los argumentos nulos no protegidos implícitos, resultantes de la expansión de parámetros que no tienen valores, se eliminan. Si un parámetro sin ningún valor se expande dentro de comillas dobles, el resultado es un argumento nulo, y es mantenido.

Observe que si no hay expansión, tampoco se realiza la división de palabras.

Expansión de nombre de camino
Tras la división de palabras, a menos que la opción -f esté puesta, bash examina cada palabra buscando los caracteres *, ?, (, y [. Si uno de estos caracteres aparece, entonces la palabra se considera como un patrón, y se reemplaza por una lista ordenada alfabéticamente de nombres de ficheros que concuerden con el patrón. Si no se encuentran nombres de ficheros concordantes, y la opción del shell nullglob está deshabilitada, la palabra se deja tal cual. Si la opción nullglob está puesta, y no ha habido ninguna concordancia, la palabra se elimina. Si la opción del shell nocaseglob está puesta, la búsqueda de la concordancia se realiza sin importar si los caracteres alfabéticos son letras mayúsculas o minúsculas. Cuando un patrón se usa para expansión de nombre de camino, el carácter `.” al principio de un nombre o inmediatamente después de una barra inclinada debe concordar explícitamente, a menos que esté puesta la opción del shell dotglob. En la concordancia de un nombre de camino, el carácter de barra inclinada debe siempre coincidir explícitamente. En otros casos, el carácter `.” no se trata de forma especial. Vea la descripción de shopt más abajo en ÓRDENES INTERNAS DEL SHELL para una descripción de las opciones del shell nocaseglob, nullglob, y dotglob.

La variable del shell GLOBIGNORE puede utilizarse para restringir el conjunto de nombres de ficheros que concuerden con un patrón. Si GLOBIGNORE está definido, cada nombre de fichero concordante que también coincida con uno de los patrones en GLOBIGNORE se quita de la lista de

concordancias. Los nombres de fichero `.” y`..” nunca son tenidos en cuenta, incluso cuando GLOBIGNORE esté puesto. Sin embargo, definir GLOBIGNORE tiene el efecto de activar la opción del shell dotglob, de modo que todos los otros nombres de fichero que comiencen con un `.” concordarán. Para obtener el comportamiento antiguo de no hacer caso de nombres de ficheros que comienzan con un `.”, haga que `.*” sea uno de los patrones de GLOBIGNORE. La opción dotglob está deshabilitada cuando GLOBIGNORE no está definido.

Patrones

Cualquier carácter que aparezca en un patrón, aparte de los especiales descritos más adelante, concuerda consigo mismo. El carácter NUL no puede estar en un patrón. Los caracteres de patrón especiales deben protegerse si han de concordar literalmente consigo mismos.

Los caracteres de patrón especiales tienen los siguientes significados:

- **Concuerda con cualquier cadena de caracteres, incluida la cadena vacía.**

? Concuerda con un solo carácter cualquiera.
[...] Concuerda con uno de los caracteres entre corchetes. Un par de caracteres separados por un signo menos denota un rango; cualquier carácter léxicamente entre esos dos, incluidos, concuerda. Si el primer carácter tras el [es un ! o un ^, entonces la concordancia es con cualquier carácter de los que no estén entre los corchetes. Un - puede representarse para la concordancia incluyéndolo como el primer o último carácter del conjunto. Un] puede hacerse concordar incluyéndolo como el primer carácter del conjunto.

Dentro de [y], se pueden especificar clases de caracteres mediante la sintaxis [:clase:], donde clase es una de las siguientes clases definidas en el estándar POSIX.2:
alnum alpha ascii blank cntrl digit graph lower print punct space upper xdigit

Una clase de caracteres concuerda con cualquier carácter que pertenezca a esa clase.

Dentro de [y], una clase de equivalencia se puede especificar empleando la sintaxis [=c=], que concuerda con todos los caracteres con el mismo peso de clasificación (tal como lo defina la localización en curso) que el carácter c.

Dentro de [y], la sintaxis [.símbolo.] concuerda con el símbolo de clasificación símbolo.
Si la opción del shell extglob se activa usando la orden interna shopt, se reconocen algunos operadores de patrones extendidos. En la siguiente descripción, una lista-patrón es una lista de uno o más patrones separados por un |. Se pueden formar patrones compuestos usando uno o más de los siguientes sub-patrones:

?(lista-patrón)
Concuerda con ninguna o una ocurrencia de los patrones dados
***(lista-patrón)**
Concuerda con ninguna o más ocurrencias de los patrones dados
+(lista-patrón)
Concuerda con una o más ocurrencias de los patrones dados
@(lista-patrón)
Concuerda exactamente con uno de los patrones dados
!(lista-patrón)
Concuerda con cualquier cosa excepto con uno de los patrones dados

Eliminación de comillas
Tras las expansiones precedentes, todas las ocurrencias no entrecomilladas de los caracteres \, ‘, y “ que no resulten de una de las expansiones anteriores, se eliminan.

REDIRECCIÓN

Antes de que se ejecute una orden, su entrada y salida pueden ser redirigidas usando una notación especial interpretada por el shell. La redirección también se puede emplear para abrir y cerrar ficheros en el entorno de ejecución del shell en curso. Los operadores de redirección siguientes pueden preceder o aparecer en cualquier sitio de una orden simple o pueden ir detrás de una orden. Las redirecciones se procesan en el orden en el que aparecen, de izquierda a derecha.

En las descripciones siguientes, si se omite el número del descriptor de fichero, y el primer carácter del operador de redirección es <, la redirección se refiere a la entrada estándar (descriptor de fichero 0). Si el primer carácter del operador de redirección es >, la redirección se refiere a la salida estándar (descriptor de fichero 1).

La palabra tras el operador de redirección en las descripciones siguientes, a menos que se diga otra cosa, está sujeta a la expansión de llaves, expansión de tilde, expansión de parámetro, sustitución de orden, expansión aritmética, eliminación de comillas, y expansión de nombre de camino. Si se expande a más de una palabra, bash informa de un error.

Observe que el orden de las redirecciones es significativo. Por ejemplo, la orden

ls > listadir 2>&1

dirige la salida estándar normal y la de errores, ambas, al fichero listadir, mientras que la orden

ls 2>&1 > listadir

dirige solamente la salida estándar al fichero listadir, porque la salida de errores estándar se ha duplicado como salida estándar antes de que ésta se redirigiera a listadir.

Un fallo en la apertura o creación de un fichero hace que la redirección fracase.

Redirección de la entrada

La redirección de la entrada hace que el fichero cuyo nombre resulte de la expansión de palabra se abra para lectura en el descriptor de fichero n, o la entrada estándar (descriptor de fichero 0) si no se especificó n.

El formato general para la redirección de la entrada es:

[n]<palabra

Redirección de la salida

la redirección de la salida hace que el fichero cuyo nombre resulte de la expansión de palabra se abra para escritura en el descriptor de fichero n, o la salida estándar (descriptor de fichero 1) si n no se especificó. Si el fichero no existe se crea; si existe se trunca a longitud cero, se vacía.

El formato general para la redirección de la salida es:
ientes, a menos que se diga otra cosa, está sujeta a la expansión de llaves, expansión de tilde, expansión de parámetro, sustitución de orden, expansión aritmética, eliminación de comillas, y expansión de nombre de camino. Si se expande a más de una palabra, bash informa de un error.

Observe que el orden de las redirecciones es significativo. Por ejemplo, la orden

ls > listadir 2>&1

dirige la salida estándar normal y la de errores, ambas, al fichero listadir, mientras que la orden

ls 2>&1 > listadir

dirige solamente la salida estándar al fichero listadir, porque la salida de errores estándar se ha duplicado como salida estándar antes de que ésta se redirigiera a listadir.

Un fallo en la apertura o creación de un fichero hace que la redirección fracase.

Redirección de la entrada
La redirección de la entrada hace que el fichero cuyo nombre resulte de la expansión de palabra se abra para lectura en el descriptor de fichero n, o la entrada estándar (descriptor de fichero 0) si no se especificó n.

El formato general para la redirección de la entrada es:

[n]<palabra

Redirección de la salida
la redirección de la salida hace que el fichero cuyo nombre resulte de la expansión de palabra se abra para escritura en el descriptor de fichero n, o la salida estándar (descriptor de fichero 1) si n no se especificó. Si el fichero no existe se crea; si existe se trunca a longitud cero, se vacía.

El formato general para la redirección de la salida es:
[n]>palabra

Si el operador de redirección es >, y la opción noclobber de la orden interna set ha sido activada, la redirección fallará si el nombre de fichero resultante de la expansión de palabra existiera y fuera un fichero regular. Si el operador de redirección es >|, o es > y la opción noclobber de la orden interna set no está activada, la redirección se intenta incluso si el fichero nombrado por palabra existe.

Añadir a la salida redirigida
La redirección de la salida en esta forma hace que el fichero cuyo nombre resulte de la expansión de palabra se abra para añadir en el descriptor de fichero n, o la salida estándar (descriptor de fichero 1) si n no se especificó. Si el fichero no existe, se crea.

El formato general para añadir a la salida es:

[n]>>palabra

Redirección de la salida estándar normal y de errores
Bash permite que ambas salidas estándares, la normal (descriptor de fichero 1) y la de errores (descriptor de fichero 2) se redirijan hacia el fichero cuyo nombre sea la expansión de palabra con esta construcción.

Hay dos formatos para la redirección de la salida estándar y la salida de errores:

&>palabra
y
>&palabra

De las dos formas, es preferible la primera. Ésta es semánticamente equivalente a

>palabra 2>&1

Documentos internos
Este tipo de redirección instruye al shell a leer la entrada desde la fuente en curso hasta que vea una línea que contenga solamente palabra (sin blancos detrás). Todas las líneas leídas hasta ese punto se emplean como la entrada estándar de una orden.

El formato de los documentos internos es como sigue:

<<[-]palabra
documento-interno
delimitador

No se realiza la expansión de parámetro, sustitución de orden, expansión de nombre de camino ni expansión aritmética en palabra. Si cualquier carácter de palabra está entrecomillado, el delimitador es el resultado de la eliminación de comillas en palabra, y las líneas en el documento interno no se expanden. Si palabra no está entrecomillada,

todas las líneas del documento interno están sujetas a expansión de parámetro, sustitución de orden y expansión aritmética. En el último caso, el par \<nueva-línea> no se tiene en cuenta, y debe emplearse \ para proteger los caracteres \, $, y `.

Si el operador de redirección es <<-, entonces se quitan de las líneas de la entrada todos los caracteres de tabulación iniciales, así como de la línea que contiene delimitador. Esto permite que los documentos internos dentro de guiones del shell se sangren de manera natural.

Duplicación de descriptores de fichero
El operador de redirección

[n]<&palabra

se emplea para duplicar descriptores de ficheros de entrada. Si palabra se expande a uno o más dígitos, el descriptor de fichero denotado por n se hace ser una copia de este descriptor de fichero. Si los dígitos en palabra no especifican un descriptor de fichero abierto para entrada, se produce un error de redirección. Si palabra se evalúa a -, el descriptor de fichero n se cierra. Si n no se especifica, se emplea la entrada estándar (descriptor de fichero 0).

El operador

[n]>&palabra

se emplea similarmente para duplicar descriptores de ficheros de salida. Si n no se especificó, se emplea la salida estándar (descriptor de fichero 1). Si los dígitos en palabra no especifican un descriptor de fichero abierto para salida, se produce un error de redirección. Como un caso especial, si n se omite, y palabra no se expande a uno o más dígitos, se redirigen la salida estándar y la salida estándar de errores como se describió con anterioridad.

Apertura de descriptores de ficheros para lectura y escritura
El operador de redirección

[n]<>palabra

hace que el fichero cuyo nombres sea la expansión de palabra se abra para lectura y para escritura en el descriptor de fichero n, o en el descriptor de fichero 0 si no se especifica n. Si el fichero no existe, se crea.

ALIAS

Los alias permiten que una cadena se sustituya por una palabra cuando se emplee como la primera palabra de una orden simple. El shell mantiene una lista de alias que pueden ponerse y quitarse con las órdenes internas alias y unalias (vea ÓRDENES INTERNAS DEL SHELL abajo). Se mira a ver si la primera palabra de cada orden, si no está entrecomillada, tiene un alias. Si es así, cada palabra se reemplaza con el texto del alias. El nombre del alias y el texto de reemplazo pueden contener cualquier entrada válida para el shell, incluyendo los metacaracteres listados arriba, con la excepción de que el nombre del alias no puede contener un =. La primera palabra del texto de reemplazo se comprueba si es un alias, pero si es un alias idéntico al que se está expandiendo, no se expande una segunda vez. Esto significa que uno puede poner un alias ls a ls -F, por ejemplo, y bash no intenta expandir recursivamente el texto de reemplazo. Si el último carácter del valor del alias es un blanco, entonces la siguiente palabra de la orden que sigue al alias también se mira para la expansión de alias.

Los alias se crean y listan con la orden alias, y se quitan con la orden unalias.

No hay ningún mecanismo para poder usar argumentos en el texto de reemplazo. Si se necesitan, debería emplearse mejor una función del shell. Los alias no se expanden cuando el shell no es interactivo, a menos que se haya puesto la opción expand_aliases mediante shopt (vea la descripción de shopt bajo ÓRDENES INTERNAS DEL SHELL abajo).

Las reglas que conciernen a la definición y uso de los alias son algo confusas. Bash siempre lee por lo menos una línea completa de entrada

antes de ejecutar cualquiera de las órdenes de esa línea. Los alias se expanden cuando se lee una orden, no cuando se ejecuta. Por lo tanto, una definición de alias que aparezca en la misma línea que otra orden no tiene efecto hasta que se lea la siguiente línea de entrada. Las órdenes que sigan a la definición de alias en esa línea no se ven afectadas por el nuevo alias. Este comportamiento también hay que tenerlo en cuenta cuando se ejecutan funciones. Los alias se expanden cuando se lee una definición de función, no cuando la función se ejecuta, porque una definición de función es en sí misma una orden compuesta. Como consecuencia, los alias definidos en una función no están disponibles hasta después de que esa función se ejecute. Para asegurarse, ponga siempre las definiciones de alias en una línea separada, y no emplee la orden alias en órdenes compuestas.

Para casi cualquier propósito, los alias pueden sustituirse por funciones del shell.

FUNCIONES

Una función del shell, definida como se describió anteriormente bajo GRAMÁTICA DEL SHELL, guarda una serie de órdenes para una ejecución posterior. Las funciones se ejecutan en el contexto del shell en curso; no se crea ningún nuevo proceso para interpretarlas (en contraste con la ejecución de un guión del shell). Cuando una función se ejecuta, los argumentos de la función se convierten en los parámetros posicionales durante su ejecución. El parámetro especial # se actualiza para reflejar el cambio. El parámetro posicional 0 permanece intacto. Todos los demás aspectos del entorno de ejecución del shell son idénticos entre una función y quien la llama con la excepción de que la trampa DEBUG (vea la descripción de la orden interna trap bajo ÓRDENES INTERNAS DEL SHELL más adelante) no se hereda.

Variables locales a la función se pueden declarar con la orden interna local. Normalmente, las variables y sus valores se comparten entre la función y quien la llama, como variables globales.
Si se ejecuta la orden interna return en una función, éste se acaba y la ejecución se reanuda con la siguiente orden tras la llamada a la

función. Cuando una función se completa, los valores de los parámetros posicionales y el parámetro especial # se restauran a los valores que tenían antes de la ejecución de la función.

Los nombres de función y sus definiciones pueden listarse con la opción -f de las órdenes internas declare o typeset. La opción -F de declare o typeset listará solamente los nombres de las funciones. Las funciones pueden exportarse de modo que los subshells las tengan definidas automáticamente con la opción -f de la orden interna export.

Las funciones pueden ser recursivas. No se impone ningún límite en el número de llamadas recursivas.

EVALUACIÓN ARITMÉTICA

El shell permite que se evalúen expresiones aritméticas, bajo ciertas circunstancias (vea la orden interna let y Expansión aritmética). La evaluación se hace con enteros largos sin comprobación de desbordamiento, aunque la división por 0 se atrapa y se señala como un error. La lista siguiente de operadores se agrupa en niveles de operadores de igual precedencia. Los niveles se listan en orden de precedencia decreciente.

- **+ menos y más unarios**

!~ negación lógica y de bits

**** exponenciación**

- **/ % multiplicación, división, resto**
- **- adición, sustracción**

<< >> desplazamientos de bits a izquierda y derecha

<= >= < >

comparación

== != igualdad y desigualdad

& Y de bits (AND)

^ O exclusivo de bits (XOR)

| O inclusivo de bits (OR)

&& Y lógico (AND)

|| O lógico (OR)

expr?expr:expr
evaluación condicional

= *= /= %= += -= <<= >>= &= ^= |=
asignación

Se permite que las variables del shell actúen como operandos; se realiza la expansión de parámetro antes de la evaluación de la expresión. El valor de un parámetro se fuerza a un entero largo dentro de una expresión. Una variable del shell no necesita tener activado su atributo de entero para emplearse en una expresión.

Las constantes con un 0 inicial se interpretan como números octales. Un 0x ó 0X inicial denota un número en hexadecimal. De otro modo, los números toman la forma [base#]n, donde base es un número en base 10 entre 2 y 64 que representa la base aritmética, y n es un número en esa base. Si base se omite, entonces se emplea la base 10. Los dígitos mayores que 9 se representan por las letras minúsculas, las metras mayúsculas, _, y @, en este orden. Si base es menor o igual que 36, las letras minúsculas y mayúsculas pueden emplearse indistintamente para representar números entre 10 y 35.

Los operadores se evalúan en orden de precedencia. Las sub-expresiones entre paréntesis se evalúan primero y pueden sustituir a las reglas de precedencia anteriores.

EXPRESIONES CONDICIONALES
Las expresiones condicionales son empleadas por la orden compuesta [[y por las órdenes internas test y [para comprobar los atributos de ficheros y realizar comparaciones de cadenas y aritméticas. Las expresiones se forman a partir de las primarias monarias o binarias siguientes. Si cualquier argumento fichero de una de estas primarias es de la forma /dev/fd/n, entonces se comprueba el descriptor de fichero n.

-a fichero
Verdad si fichero existe.
-b fichero
Verdad si fichero existe y es un fichero especial de bloques.

-c fichero

Verdad si fichero existe y es un fichero especial de caracteres.

-d fichero

Verdad si fichero existe y es un directorio.

-e fichero

Verdad si fichero existe.

-f fichero

Verdad si fichero existe y es un fichero regular.

-g fichero

Verdad si fichero existe y tiene el bit SGID.

-k fichero

Verdad si fichero existe y tiene el bit "pegajoso" (STIcky).

-p fichero

Verdad si fichero existe y es una tubería con nombre (FIFO).

-r fichero

Verdad si fichero existe y se puede leer.

-s fichero

Verdad si fichero existe y tiene un tamaño mayor que cero.

-t fd Verdad si el descriptor de fichero fd está abierto y se refiere a una terminal.

-u fichero

Verdad si fichero existe y tiene el bit SUID.

-w fichero

Verdad si fichero existe y se puede modificar.

-x fichero

Verdad si fichero existe y es ejecutable.

-O fichero

Verdad si fichero existe y su propietario es el UID efectivo.

-G fichero

Verdad si fichero existe y su grupo es el GID efectivo.

-L fichero

Verdad si fichero existe y es un enlace simbólico o blando.

-S fichero

Verdad si fichero existe y es un zócalo (socket).

-N fichero

Verdad si fichero existe y ha sido modificado desde que se leyó la última vez.

Fichero1 -nt fichero2

Verdad si fichero1 es más reciente (según la fecha de modificación) que fichero2.

Fichero1 -ot fichero2

Verdad si fichero1 es más antiguo que fichero2.

Fichero1 -ef fichero2

Verdad si fichero1 y fichero2 tienen los mismos números de nodo-í y de dispositivo.

-o nombre-opción

Verdad si la opción del shell nombre-opción está activada. Vea la lista de opciones bajo la descripción de la opción -o de la orden interna set más abajo.

-z cadena

Verdad si la longitud de cadena es cero.

-n cadena

cadena Verdad si la longitud de cadena no es cero.

Cadena1 == cadena2

Verdad si las cadenas son iguales. También se puede emplear = en vez de ==.

cadena1 != cadena2

Verdad si las cadenas no son iguales.

Cadena1 < cadena2

Verdad si cadena1 se ordena lexicográficamente antes de cadena2 en la localización en curso.

Cadena1 > cadena2

Verdad si cadena1 se clasifica lexicográficamente tras cadena2 en la localización en curso.

Arg1 OP arg2

OP es uno de -eq, -en, -lt, -le, -gt, o -ge. Estos operadores aritméticos binarios devuelven verdadero si arg1 es igual a, distinto de, menor que, menor o igual a, mayor que, o mayor o igual a arg2, respectivamente. Arg1 y arg2 pueden ser enteros positivos o negativos.

EXPANSIÓN DE ORDEN SIMPLE

Cuando se ejecuta una orden simple, el shell realiza las siguientes expansiones, asignaciones y redirecciones, de izquierda a derecha.

1. Las palabras que el analizador ha marcado como asignaciones de variables (aquéllas que preceden al nombre de la orden) y redirecciones se guardan para un procesado posterior.

2. Las palabras que no sean asignaciones de variables ni redirecciones se expanden. Si tras la expansión quedan aún palabras, la primera palabra se toma como el nombre de la orden y las palabras restantes son los argumentos.

3. Se ejecutan las redirecciones como se describió más arriba bajo REDIRECCIÓN.

4. El texto tras el = en cada asignación de variable está sujeto a expansión de tilde, expansión de parámetro, sustitución de orden, expansión aritmética y eliminado de comillas antes de ser asignado a la variable.

Si no resulta ningún nombre de orden, las asignaciones de variables afectan al entorno actual del shell. De otro modo, las variables se añaden al entorno de la orden ejecutada y no afectan al entorno del shell en curso. Si cualquiera de las asignaciones intenta dar un valor a una variable de lectura exclusiva, se produce un error, y la orden acaba con un estado distinto de cero.

Si no resulta ningún nombre de orden, se hacen las redirecciones, pero no afectan al entorno del shell en curso. Un error de redirección hace que el shell acabe con un estado distinto de cero.

Si hay un nombre de orden tras la expansión, la ejecución procede como se describió antes. De otro modo, la orden sale. Si una de las expansiones contenía una sustitución de orden, el estado de salida de la orden es el de la última sustitución de orden realizada. Si no había

sustituciones de órdenes, la orden acaba con un estado de cero.

EJECUCIÓN DE ÓRDENES

Después de que una orden ha sido dividida en palabras, si el resultado es una orden simple y una lista opcional de argumentos, tienen lugar las siguientes acciones.

Si el nombre de la orden no contiene barras inclinadas, el shell intenta localizarla. Si existe una función del shell con ese nombre, esa función se llama como se describió arriba en FUNCIONES. Si el nombre no coincide con el de ninguna función, el shell lo busca en la lista de órdenes internas. Si se encuentra, se llama a la orden interna correspondiente.

Si el nombre no es ni una función del shell ni una orden interna, y no contiene barras inclinadas, bash busca en cada elemento de PATH un directorio que contenga un fichero ejecutable con ese nombre. Bash emplea una tabla de dispersión (hash) para recordar los nombres de camino completos de los ficheros ejecutables (vea hash en ÓRDENES INTERNAS DEL SHELL abajo). Sólo se realiza una búsqueda completa de los directorios de PATH si la orden no se encuentra en la tabla de dispersión. Si la búsqueda no es satisfactoria, el shell muestra un mensaje de error y devuelve un estado de salida de 127.

Si la búsqueda fue exitosa, o si el nombre de la orden contiene una o más barras inclinadas, el shell ejecuta el programa con ese nombre en un entorno de ejecución separado. El argumento 0 se pone al nombre dado, y el resto de argumentos de la orden se ponen a los argumentos dados, si los hay.

Si esta ejecución fallara porque el fichero no tuviera un formato ejecutable, y el fichero no fuera un directorio, se supone que es un guión del shell, un fichero que contiene órdenes del shell. Se crea un subshell para ejecutarlo. Este subshell se reinicia a sí mismo, así que el efecto es el mismo que si se hubiera llamado a un nuevo shell para manejar el guión, con la excepción de que el hijo retiene las localizaciones de órdenes recordadas por el padre (vea hash abajo en ÓRDENES

INTERNAS DEL SHELL).

Si el programa es un fichero que empieza con los dos caracteres #!, el resto de la primera línea especifica un intérprete para el programa. El shell ejecuta el intérprete especificado en sistemas operativos que no manejen por sí mismos este formato de ejecutable. Los argumentos del intérprete consisten en un solo argumento opcional tras el nombre del intérprete en la primera linea del programa, seguido del nombre del programa, seguido por los argumentos de la orden, si los hubiera.

ENTORNO DE EJECUCIÓN DE ÓRDENES

El shell tiene un entorno de ejecución, que consiste en lo siguiente:

- **ficheros abiertos heredados por el shell en la llamada, quizás modificada por redirecciones suministradas a la orden interna exec**
- **el directorio de trabajo en curso, establecido por cd, pushd o popd, o heredado por el shell en la llamada**
- **la máscara de modo de creación de ficheros, establecida por umask o heredada del padre del shell**
- **las trampas en curso establecidas por trap**
- **parámetros del shell que han sido establecidos por asignaciones de variables o con set, o heredados del padre del shell en el entorno**
- **funciones del shell definidas durante la ejecución o heredadas del padre del shell en el entorno**
- **opciones activadas en la llamada (bien por omisión o mediante argumentos en la línea de órdenes) o por set**
- **opciones activadas mediante shopt**
- **alias del shell definidos con alias**

o varios identificadores de proceso, incluyendo los de trabajos en segundo plano, el valor de $$, y el valor de $PPID

Cuando una orden simple distinta de una interna o una función del shell se va a ejecutar, se llama en un entorno de ejecución separado que consiste en lo siguiente. A menos que se diga otra cosa, los valores se heredan del shell.

O los ficheros abiertos del shell, más las modificaciones y adiciones especificadas en la orden por redirecciones

o el directorio de trabajo en curso

o la máscara de modo de creación de ficheros

o variables del shell marcadas para la exportación, junto con variables exportadas para la orden, pasadas en el entorno

o las trampas capturadas por el shell se restauran a los valores del padre del shell, y las trampas que no son tenidas en cuenta por el shell tampoco lo son

Una orden llamada en este entorno separado no puede afectar al entorno de ejecución del shell.

La sustitución de órdenes y las órdenes asíncronas se llaman en un entorno de subshell que es un duplicado del entorno del shell, excepto que las trampas capturadas por el shell se restauran a los valores que el shell heredó de su padre en la llamada. Las órdenes internas que se llaman como parte de una tubería se ejecutan también en un entorno de subshell. Los cambios hechos al entorno del subshell no pueden afectar al entorno de ejecución del shell.

ENTORNO

Cuando se llama a un programa, se le da un vector de cadenas de caracteres llamado el entorno. Esto es una lista de parejas nombre-valor, de la forma nombre=valor.

El shell le permite manipular el entorno de varias maneras. En la hora de la llamada, el shell escudriña el entorno y crea un parámetro para cada nombre encontrado, marcándolo automáticamente para la exportación a procesos hijos. Las órdenes ejecutadas heredan el entorno. Las órdenes export y declare -x permiten añadir y quitar parámetros y funciones del entorno. Si el valor de un parámetro en el entorno se modifica, el nuevo valor pasa a formar parte del entorno, reemplazando al antiguo. El entorno heredado por cualquier orden ejecutada consiste en el entorno inicial del shell, cuyos valores pueden ser modificados en el shell, menos las parejas quitadas mediante la orden unset más las adiciones con las órdenes export y declare -x.

El entorno para cualquier orden simple o función puede aumentarse temporalmente prefijándola con asignaciones de parámetros, como se describió arriba en PARÁMETROS. Estas sentencias de asignación afectan solamente al entorno visto por esa orden.

Si la opción -k está puesta (vea la orden interna set más adelante), entonces todas las asignaciones de parámetros se ponen en el entorno para una orden, no sólo las que preceden a su nombre.

Cuando bash llama a una orden externa, la variable _ se pone con el nombre completo del fichero de la orden y se pasa a esa orden en su entorno.

ESTADO DE SALIDA

Para los propósitos del shell, una orden que acabe con un estado de salida cero, ha tenido éxito. Un estado de salida de cero indica éxito. Un estado de salida distinto de cero indica fallo. Cuando una orden termina por una señal fatal, bash emplea el valor de 128+señal como el estado de salida.

Si una orden no se encuentra, el proceso hijo creado para ejecutarla devuelve un estado de 127. Si una orden se encuentra pero no es ejecutable, el estado de salida es 126.

Si una orden falla debido a un error durante la expansión o redi-

rección, el estado de salida es mayor que cero.

Las órdenes incorporadas en el shell devuelven un estado de 0 (verdad) si acaban con éxito, y distinto de cero (falso) si ocurre un error mientras se ejecutan. Todas las órdenes internas devuelven un estado de salida de 2 para indicar un modo de empleo incorrecto.

El propio bash devuelve el estado de salida de la última orden ejecutada, a menos que ocurra un error de sintaxis, en cuyo caso acaba con un estado distinto de cero. Vea también la orden interna exit abajo.

SEÑALES

Cuando bash es interactivo, en la ausencia de trampas, no hace caso de SIGTERM (de modo que kill 0 no mata un shell interactivo), y se captura y maneja SIGINT (de manera que la orden interna wait es interrumpible). En todos los casos, bash no hace caso de SIGQUIT. Si el control de trabajos tiene efecto, bash no hace caso de SIGTTIN, SIGTTOU, y SIGTSTP.

Los trabajos síncronos empezados por bash tienen manejadores de señal puestos a los valores heredados por el shell de su padre. Cuando el control de trabajo no está en efecto, las órdenes internas no hacen caso de SIGINT ni SIGQUIT tampoco. Las órdenes ejecutadas como resultado de sustitución de orden no hacen caso de las señales de control de trabajo generadas mediante el teclado SIGTTIN, SIGTTOU, ni SIGTSTP.

El shell, de forma predeterminada, acaba cuando recibe una señal SIGHUP. Antes de salir, reenvía la señal SIGHUP a todos los trabajos, en ejecución o parados. A los trabajos parados se les envía SIGCONT para asegurarse de que reciben la señal SIGHUP. Para prevenir que el shell envíe la señal a un trabajo particular, debería quitarse de la lista de trabajos con la orden interna disown (vea ÓRDENES INTERNAS DEL SHELL más abajo) o marcarlo para no recibir SIGHUP empleando disown -h.

Si la opción del shell huponexit se ha puesto mediante shopt, bash envía una señal SIGHUP a todos los trabajos cuando un shell de entrada interactivo se acaba.

Cuando bash recibe una señal para la que se ha puesto una trampa mientras se se está esperando que una orden se complete, la trampa no se ejecutará hasta que la orden se complete. Cuando bash está esperando una orden asíncrona mediante la orden interna wait, la recepción de una señal para la que se ha definido una trampa, hará que la orden interna wait regrese inmediatamente con un estado de salida mayor que 128, inmediatamente tras que se ejecute la trampa.

CONTROL DE TRABAJOS

El control de trabajos se refiere a la capacidad de parar selectivamente (suspender) la ejecución de procesos y continuar (reanudar) su ejecución posteriormente. Un usuario emplea esta facilidad típicamente a través de una interfaz interactiva suministrada conjuntamente por el controlador de terminal del sistema y bash.

El shell asocia un trabajo con cada tubería. Mantiene una tabla de trabajos ejecutándose actualmente, que pueden listarse con la orden jobs. Cuando bash arranca un trabajo asíncronamente (en segundo plano), imprime una línea con un aspecto como ésta:

[1] 25647

indicando que este trabajo es el número 1 y que el PID del último proceso en la tubería asociada con él es 25647. Todos los procesos en una misma tubería son miembros del mismo trabajo. Bash emplea la abstracción del trabajo como la base para el control de trabajos.

Para facilitar la implementación de la interfaz del usuario al control de trabajos, el sistema mantiene la noción de un identificador (ID) de grupo de procesos de la terminal en curso. Los miembros de este grupo de procesos (procesos cuyo PGID es igual al PGID de la terminal en curso) reciben señales generadas por el teclado como SIGINT. Se dice que estos procesos están en primer plano. Los procesos en segundo plano son aquéllos cuyo PGID difiere del de la terminal; tales procesos son inmunes a señales generadas desde el teclado. Sólo los procesos en primer plano tienen permitido leer o escribir en la terminal. A los

procesos en segundo plano que intenten leer de (o escribir en) la terminal, el controlador de terminal les manda una señal SIGTTIN (SIGTTOU) que, a menos que sea capturada, suspende el proceso.

Si el sistema operativo en el que bash se está ejecutando, admite el control de trabajos (Linux lo admite, por supuesto), bash le permite usarlo. Teclear el carácter suspender (típicamente ^Z, Control-Z) mientras que un proceso se está ejecutando, hace que ese proceso se pare y le devuelve a Ud. al bash. Teclear el carácter suspensión diferida (típicamente ^Y, Control-Y) hace que el proceso se pare cuando intente leer entrada desde la terminal, y el control se devuelve a bash. El usuario puede entonces manipular el estado de este trabajo, empleando la orden bg para continuar con él en segundo plano, la orden fg para continuar con él en primer plano, o la orden kill para matarlo. Un ^Z tiene efecto inmediatamente, y tiene el efecto adicional colateral de que la salida pendiente y lo que haya en el búfer de entrada del teclado se descartan.

Hay varias formas de referirse a un trabajo en el shell. El carácter % introduce un nombre de trabajo. El trabajo número n puede ser referenciado como %n. Un trabajo puede ser referenciado utilizando un prefijo del nombre empleado para arrancarlo, o usando una subcadena que aparezca en su línea de órdenes. Por ejemplo. %ce se refiere a un trabajo ce parado. Si un prefijo concuerda con más de un trabajo, bash informa de un error. Usar %?ce, por otra parte, se refiere a cualquier trabajo que contenga la cadena ce en su línea de órdenes. Si la subcadena concuerda con más de un trabajo, bash informa de un error. Los símbolos %% y %+ se refieren a la noción que tiene el shell del trabajo en curso, que es el último trabajo parado mientras estaba en primer plano o se arrancó en segundo plano. El trabajo anterior puede referenciarse usando %-. En la salida relativa a trabajos

(e.g., la salida de la orden jobs), el trabajo actual se marca siempre con un +, y el anterior con un -.

Si simplemente damos el nombre de un trabajo, esto puede traerlo a primer plano: %1 es un sinónimo de ‘fg %1”, que trae el trabajo número 1 desde el segundo plano al primero. Similarmente, ‘%1 &” reanuda el trabajo 1 en el segundo plano, equivalente a ‘bg %1”.

El shell se entera inmediatamente de cuando un trabajo cambia de estado. Normalmente, bash espera hasta que está a punto de mostrar un indicador antes de informar de cambios en el estado de un trabajo, para no interrumpir cualquier otra salida. Si la opción -b de la orden interna set está activada, bash informa de tales cambios inmediatamente.

Si se intenta salir de bash mientras hay trabajos parados, el shell muestra un mensaje de aviso. La orden jobs se puede usar entonces para inspeccionar sus estados. Si se hace un segundo intento de salir sin ninguna otra orden intermedia, el shell no muestra ningún otro aviso, y los trabajos parados se hacen terminar.

INDICADORES

Cuando se ejecuta interactivamente, bash muestra el indicador primario PS1 cuando está dispuesto para leer una orden, y el secundario PS2 cuando necesita más entrada para completar una orden. Bash permite que estas cadenas indicadoras se personalicen insertando un número de caracteres especiales protegidos con la barra inversa, que se interpretan como sigue:

\a un carácter de alerta ASCII (07)

\d la fecha en el formato “Día-Semana Mes Día” (ejemplo, “Tue May 26”) en inglés

\e un carácter de escape (ESC) ASCII (033)

\h el nombre del computador anfitrión hasta el primer ‘.’

\H el nombre del computador anfitrión completo

\n salto de línea

\r retorno de carro

\s el nombre del shell, el nombre base de $0 (la porción que sigue a la última barra

inclinada)

\t la hora actual en el formato de 24 horas HH:MM:SS

\T la hora actual en el formato de 12 horas HH:MM:SS

\@ la hora actual en el formato de 12 horas con indicador AM/PM

\u el nombre de usuario del usuario en curso

\v la versión de bash (e.g., 2.00)

\V la distribución de bash, versión + nivel de parcheo (e.g., 2.00.0)

\w el directorio de trabajo en curso

\W el nombre base del directorio de trabajo

\! el número de historia de esta orden

\# el número de orden de esta orden

\$ si el UID efectivo es 0 (el super-usuario), un #, si no un $

\nnn el carácter correspondiente al número octal nnn

\\ una barra inclinada invertida

\[empieza una secuencia de caracteres no imprimibles, que pueden emplearse para empotrar una secuencia de control del terminal en el indicador

\] termina una secuencia de caracteres no imprimibles

El número de orden y el número de historia son usualmente diferentes: el número de historia de una orden es su posición en la lista de historia, que puede incluir órdenes restauradas desde el fichero de historia (vea HISTORIA más abajo), mientras que el número de orden es la posición en la secuencia de órdenes ejecutadas durante la sesión de shell actual. Después de que la cadena es descodificada, se expande mediante la expansión de parámetros, sustitución de órdenes, expansión aritmética, expansión de cadena, y eliminación de comillas, sujeta al valor de la opción del shell promptvars (vea la descripción de la orden shopt bajo ÓRDENES INTERNAS DEL SHELL más adelante).

READLINE

Readline (leer línea) es la biblioteca que maneja la lectura de la entrada cuando se usa un shell interactivo, a menos que se haya dado la opción --noediting cuando se llamó. De forma predetermi-nada, las órdenes de edición de la línea son similares a las de emacs. También se dispone de

una

interfaz de edición de líneas al estilo de vi. Para desactivar la edición de líneas una vez que el

shell está en ejecución, use las opciones +o emacs o +o vi de la orden interna set (vea ÓRDENES

INTERNAS DEL SHELL abajo).

Notación de Readline

En esta sección, se emplea la notación al estilo de emacs para denotar las teclas. Las teclas de

control se representan por medio de C-tecla; así, C-n significa Control-N. De forma similar, las

teclas meta (Alt) se representan con M-tecla, de forma que M-x significa Meta-X. (En teclados sin

tecla meta, M-x significa ESC x, i.e., pulsar la tecla de escape (Esc) y luego la tecla x. Esto

hace que ESC sea el prefijo meta. La combinación M-C-x quiere decir ESC-Control-x, o pulsar la

tecla Escape y luego mantener presionada la tecla Control mientras se pulsa la tecla x.)

Las órdenes de readline pueden recibir argumentos numéricos que actúan normalmente como un número de

repetición. Algunas veces, empero, lo que tiene significado es el signo del argumento. Pasar un

argumento negativo a una orden que actúa en la dirección adelante (p. ej., kill-line) hace que esa

orden actúe en la dirección contraria, hacia atrás. Las órdenes cuyo comportamiento con argumentos

se desvíe de esto se señalan más adelante.

Cuando una orden se describe como que corta texto, el texto borrado se guarda para una posible

futura recuperación (pegado). El texto cortado se guarda en un anillo de corte. Cortes consecutivos

hacen que el texto se acumule en una unidad, que puede pegarse toda de golpe, de una vez. Las

órdenes que no cortan texto separan los trozos de texto en el anillo de corte.

Inicio de Readline

Readline se personaliza poniendo órdenes en un fichero de inicio (el fichero inputrc). El nombre de

este fichero se toma del valor de la variable INPUTRC. Si esta variable no está definida, el

valor

predeterminado es ~/.inputrc. Cuando un programa que hace uso de la biblioteca readline arranca, el

fichero de inicio se lee, y se establecen las definiciones de teclas y variables. Sólo se permiten

unas pocas construcciones básicas en el fichero de inicio de readline. Las líneas en blanco no se

tienen en cuenta. Las líneas que comiencen con un # son comentarios. Las líneas que comiencen con

un $ indican construcciones condicionales. Otras líneas representan definiciones de teclas y

definiciones de variables.

Las definiciones predeterminadas de teclas pueden cambiarse con un fichero inputrc. Otros programas

que usen esta biblioteca pueden añadir sus propias órdenes y definiciones.

Por ejemplo, el poner

M-Control-u: universal-argument

o

C-Meta-u: universal-argument

dentro del fichero inputrc haría que M-C-u ejecutara la orden de readline universal-argument.

Se reconocen los siguientes nombres simbólicos de caracteres: RUBOUT, DEL, ESC, LFD, NEWLINE, RET,

RETURN, SPC, SPACE, y TAB. Además de los nombres de órdenes, readline permite que se enlace una

tecla cualquiera a una cadena de caracteres que se inserta cuando la tecla se pulse (una macro).

Definiciones de teclas de Readline

La sintaxis para controlar las definiciones de teclas en el fichero inputrc es simple. Todo lo que

se requiere es el nombre de la orden o el texto de una macro y una secuencia de teclas con la cual

debe enlazarse. El nombre se puede especificar en una de dos formas: como un nombre simbólico de

tecla, posiblemente con prefijos Meta- o Control-, o como una secuencia de teclas. Cuando se use la

forma nombre-tecla:nombre-función o macro, nombre-tecla es el nombre de una tecla en inglés. Por

ejemplo:

Control-u: universal-argument

Meta-Rubout: backward-kill-word

Control-o: "> salida"

En el ejemplo de arriba, C-u se enlaza a la función universal-argument, M-DEL se vincula a la

función backward-kill-word, y C-o se define como que se ejecute la macro expresada en la parte

derecha (esto es, insertar el texto > salida en la línea).

En la segunda forma, "sectecla":nombre-función o macro, sectecla difiere de la nombre-tecla de antes

en que las cadenas que representan una secuencia entera de teclas pueden especificarse poniendo la

secuencia entre comillas dobles. Así se pueden utilizar algunas teclas de escape al estilo de GNU

Emacs, como en el siguiente ejemplo.

"\C-u": universal-argument

"\C-x\C-r": re-read-init-file

"\e[11~": "Tecla de Función 1"

En este ejemplo, C-u se enlaza de nuevo a la función universal-argument. C-x C-r se vincula a la

función re-read-init-file, y ESC [1 1 ~ se define como que se inserte el texto Tecla de Función 1.

El conjunto completo de las secuencias de escape al estilo de GNU Emacs es

\C- prefijo de control

\M- prefijo meta

\e un carácter de Escape

\\ barra inclinada inversa

\" una " literal

\' un ' literal

Además de las secuencias de escape al estilo de GNU Emacs, se dispone de un segundo conjunto de

escapes con la barra invertida:

\a alerta (campana)

\b espacio atrás

\d borrado

\f salto de página

\n salto de línea

\r retorno de carro

\t tabulador horizontal

\v tabulador vertical

\nnn el carácter cuyo código es el valor octal nnn (de 1 a 3 dígitos)

\xnnn el carácter cuyo código es el valor hexadecimal nnn (de 1 a 3 dígitos)

Cuando se mete el texto de una macro, se deben emplear comillas simples o dobles para indicar una

definición de macro. El texto no entrecomillado se supone un nombre de función. En el cuerpo de la

macro, los escapes con barra inversa de arriba se expanden. La barra inversa protegerá a cualquier

otro carácter en el texto de la macro, incluyendo a " y a '.

Bash permite mostrar o modificar las definiciones de teclas en curso mediante la orden interna bind.

El modo de edición puede cambiarse durante una sesión interactiva empleando la opción -o de la orden

interna set (vea ÓRDENES INTERNAS DEL SHELL abajo).

Variables de Readline

Readline tiene variables que se pueden usar para personalizar más aún su comportamiento. Una vari-

able se puede definir en el fichero inputrc con una sentencia de la forma

set nombre-variable valor

Excepto cuando se diga, las variables de readline pueden tomar los valores On u Off. Las variables

y sus valores predeterminados son:

bell-style (audible)

Controla qué pasa cuando readline quiere tocar el pito de la terminal. Si se define como none, readline nunca toca el pito. Si se pone a visible, readline usa una alerta visible si está disponible. Si se pone como audible, readline intenta hacer sonar el pito de la termi-

nal.

Comment-begin (#")

La cadena que se inserta cuando se ejecuta la orden de readline insert-comment. Esta orden está enlazada a M-# en modo emacs y a # en modo vi.

Completion-ignore-case (Off)

Si se pone a On, readline realiza la concordancia y terminación de nombres de ficheros sin importar si las letras son mayúsculas o minúsculas.

Completion-query-items (100)

Esto determina cuándo se pregunta al usuario si quiere ver el número de terminaciones posibles generadas por la orden possible-completions. Puede ponerse a cualquier valor entero mayor o igual a cero. Si el número de posibles terminaciones es mayor o igual que el valor de esta variable, al usuario se le pregunta si desea o no verlas; si no, simplemente se muestran en la terminal.

Convert-meta (On)

Si se pone a On, readline convertirá caracteres con el octavo bit a uno a una secuencia de teclas ASCII poniendo el octavo bit a cero y prefijando un carácter Esc (en efecto, usando Esc como el prefijo meta).

Disable-completion (Off)

Si se pone a On, readline inhibirá la terminación de palabras. Los caracteres para la terminación se insertarán en la línea como si se hubieran hecho corresponder con self-insert.

Editing-mode (emacs)

Controla si readline empieza con un conjunto de definiciones de teclas similar a las de emacs o vi. Editing-mode puede ponerse a los valores emacs o vi.

Enable-keypad (Off)

Cuando se pone a On, readline intentará activar el teclado auxiliar cuando se llame. Algunos sistemas necesitan esto para activar las teclas de flechas de cursor.

Expand-tilde (Off)

Si se pone a on, la expansión de tilde se realiza cuando readline intenta la terminación de palabras.

Horizontal-scroll-mode (Off)

Cuando se pone a On, esto hace que readline use una sola línea para pantalla, haciendo rodar la entrada horizontalmente en una sola línea de la pantalla cuando ésta es más larga que la anchura de la pantalla; en lugar de seguir en la línea siguiente.

Input-meta (Off)

Si se pone a On, readline habilitará la entrada de 8 bits (esto es, no pondrá a cero el

octavo bit de los caracteres que lea), sin importar lo que la terminal diga que admite. El nombre meta-flag es un sinónimo para esta variable.

Keymap (emacs)

Establece el mapa de teclado actual para readline. El conjunto de nombres de mapas de teclado válidos es emacs, emacs-standard, emacs-meta, emacs-ctlx, vi, vi-command, y vi-insert. Vi es equivalente a vi-command; emacs es equivalente a emacs-standard. El valor predeterminado es

emacs; el valor de editing-mode también afecta al mapa de teclado predeterminado.

Mark-directories (On)

Si se pone a On, los nombres de directorios completados tendrán una barra inclinada añadida al final.

Mark-modified-lines (Off)

Si se pone a On, las líneas de historia que hayan sido modificadas se muestran con un asterisco precediéndolas (*).

output-meta (Off)

Si se pone a On, readline mostrará directamente los caracteres con el octavo bit a uno, en vez de como una secuencia de escape prefijada con meta.

Print-completions-horizontally (Off)

Si se pone a On, readline mostrará terminaciones con concordancias clasificadas horizontalmente en orden alfabético, en vez de verticalmente.

Show-all-if-ambiguous (Off)

Esto altera el comportamiento predeterminado de las funciones de terminación. Si se pone a on, las palabras que tienen más de una posible terminación hacen que las concordancias se muestren inmediatamente en vez de tocarse el pito de la terminal.

Visible-stats (Off)

Si se pone a On, se añade un carácter que represente un tipo de fichero según lo que devuelve stat(2) cuando se listan las terminaciones posibles.

Construcciones condicionales de Readline

Readline implementa una facilidad similar en espíritu a las características de compilación condi-

cional del preprocesador de C que permite que las definiciones de teclas y variables se realicen en

función de pruebas o condiciones. Hay cuatro directivas del analizador que se usan.

$if La construcción $if permite que las definiciones se hagan según el modo de edición, la termi-

nal en uso, o la aplicación que haga uso de readline. El texto de la condición se extiende hasta el final de la línea; no se requieren caracteres para aislarla.

Mode La forma mode= de la directiva $if se usa para ver si readline está en modo emacs o vi.Esto se puede emplear en conjunción con la orden set keymap, por ejemplo, para poner el teclado en los mapas emacs-standard y emacs-ctlx sólo si readline está arrancando en modo emacs.

Term La forma term= puede emplearse para incluir definiciones de teclas específicas de una terminal determinada, quizás para enlazar las secuencias de teclas generadas por las teclas de función de la terminal. La palabra en la parte derecha del = se prueba contra el nombre completo de la terminal y contra la porción del nombre de la terminal antes del primer -. Esto permite que sun concuerde con sun y con sun-cmd, por ejemplo.

Application
La construcción application se emplea para incluir definiciones específicas de la aplicación. Cada programa que usa la biblioteca readline define el nombre de la aplicación, y en un fichero de arranque se puede comprobar si existe un valor en particular. Esto se podría emplear para enlazar secuencias de teclas a funciones útiles para un programa específico. Por ejemplo, la orden siguiente añade una secuencia de teclas que entrecomilla la palabra en curso o la anterior en bash:

```
$if Bash
# Entrecomilla la palabra actual o previa
“\C-xq”: “\eb\”\ef\””
$endif
```

$endif Esta orden, como se ha visto en el ejemplo precedente, termina una orden $if.

$else Las órdenes en esta rama de la directiva $if se ejecutan si la comprobación falla.

$include
Esta directiva toma como argumento un solo nombre de fichero y lee órdenes y definiciones de ese fichero. Por ejemplo, la siguiente directiva leería de /etc/inputrc:

```
$include /etc/inputrc
```

Búsquedas

Readline proporciona órdenes para buscar a través de la historia de órdenes (vea HISTORIA abajo)

líneas que contengan una cadena especificada. Hay dos modos de búsqueda: incremental y no incremen-

tal.

La búsqueda incremental comienza antes de que el usuario haya acabado de teclear la cadena a buscar.

Tan pronto como se teclea cada carácter de la cadena de búsqueda, readline muestra la siguiente

entrada de la historia que concuerde con la cadena que se esté tecleando hasta este punto. Una

búsqueda incremental requiere solamente tantos caracteres como se necesiten para encontrar la

entrada deseada en la lista de historia. El carácter Escape se emplea para terminar una búsqueda

incremental. Control-J también dará por terminada la búsqueda. Control-G parará abruptamente una

búsqueda incremental y restaurará la línea original. Cuando la búsqueda se termine, la entrada de

la historia que contuviera la cadena de búsqueda se convierte en la línea en curso. Para encontrar

otras entradas en la lista de historia, teclee Control-S o Control-R, según sea apropiado. Esto

buscará hacia atrás o adelante en la historia la siguiente entrada concordante con la cadena de

búsqueda tecleada hasta este punto. Cualquier otra secuencia de teclas asociada a una orden de

readline terminará la búsqueda y ejecutará esa orden. Por ejemplo, una nueva-línea terminará la

búsqueda y aceptará la línea, ejecutando de ese modo la orden de la lista de historia.

Las búsquedas no incrementales leen la cadena de búsqueda entera antes de empezar a buscar en las

líneas de la lista de historia. La cadena de búsqueda puede ser tecleada por el usuario o ser parte

de los contenidos de la línea en curso.

Nombres de órdenes de Readline

Lo siguiente es una lista de los nombres de las órdenes y las secuencias de teclas predeterminadas a

las que están asociadas. Los nombres de órdenes sin una secuencia de tecla acompañante pertenecen a

órdenes que no están asociadas a ninguna secuencia de teclas de forma predeterminada.

Órdenes para el movimiento

beginning-of-line (C-a)

Mover al principio de la línea en curso.

End-of-line (C-e)

Mover al final de la línea.

Forward-char (C-f)

Mover un carácter hacia adelante.

Backward-char (C-b)

Mover un carácter hacia atrás.

Forward-word (M-f)

Mover adelante hasta el final de la siguiente palabra. Las palabras se componen de caracteres alfanuméricos (letras y dígitos).

Backward-word (M-b)

Mover atrás hasta el principio de esta palabra o la anterior. Las palabras se componen de caracteres alfanuméricos (letras y dígitos).

Clear-screen (C-l)

Limpiar la pantalla dejando la línea en curso al principio de la pantalla. Con un argumento, refresca la línea en curso sin borrar la pantalla.

Redraw-current-line

Refrescar la línea en curso.

Órdenes para manipular la lista de historia

accept-line (Nueva-línea, Intro)

Aceptar la línea sin importar dónde esté el cursor. Si esta línea no está vacía, añadirla a la lista de historia de acuerdo con el estado de la variable HISTCONTROL. Si la línea es una de las de la historia, modificada, entonces restaurar la línea de la historia a su estado original.

Previous-history (C-p)

Obtener la orden anterior de la lista de historia, moviéndose hacia atrás en ella.

Next-history (C-n)

Obtener la orden siguiente de la lista de historia, moviéndose hacia delante en ella.

Beginning-of-history (M-<)

Mover a la primera línea de la lista de historia.

End-of-history (M->)

Mover al final de la historia de entrada; esto es, la línea que está siendo introducida en la actualidad.

Reverse-search-history (C-r)

Buscar hacia atrás empezando en la línea en curso y moviéndose `arriba' a través de la lista de historia si es necesario. Esta búsqueda es incremental.forward-search-history (C-s)

Buscar hacia adelante empezando en la linea en curso y moviéndose `abajo' a través de la lista de historia si es necesario. Esta búsqueda es incremental.

Non-incremental-reverse-search-history (M-p)

Buscar hacia atrás a través de la lista de historia empezando en la línea en curso empleando una búsqueda no incremental de una cadena suministrada por el usuario.

Non-incremental-forward-search-history (M-n)

Buscar hacia delante a través de la lista de historia empezando en la línea en curso empleando una búsqueda no incremental de una cadena suministrada por el usuario.

History-search-forward

Buscar hacia delante a través de la lista de historia una cadena de caracteres entre el comienzo de la línea en curso y la posición actual del cursor (el punto). Esta búsqueda no es incremental.

History-search-backward

Buscar hacia atrás a través de la lista de historia una cadena de caracteres entre el comienzo de la línea en curso y la posición actual del cursor (el punto). Esta búsqueda no es incremental.

Yank-nth-arg (M-C-y)

Insertar el primer argumento de la orden anterior (normalmente la segunda palabra de la línea previa) en el punto (la posición actual del cursor). Con un argumento n, insertar la n-sima palabra de la orden anterior (las palabras en la orden anterior empiezan con la palabra 0). Un argumento negativo inserta la n-sima palabra desde el final de la orden anterior.

Yank-last-arg (M-., M-_)

Insertar el último argumento de la orden anterior (la última palabra de la entrada anterior de la lista de historia). Con un argumento, se comporta exactamente como yank-nth-arg. Llamadas sucesivas a yank-last-arg mueven hacia atrás en la lista de historia, insertando cada vez el último argumento de cada línea.

Shell-expand-line (M-C-e)

Expandir la línea como hace el shell. Esto realiza la expansión de alias y de historia así

como todas las expansiones de palabra del shell. Vea EXPANSIÓN DE HISTORIA abajo para una

descripción de la expansión de historia.

History-expand-line (M-^)

Realizar la expansión de historia en la línea en curso. Vea EXPANSIÓN DE HISTORIA abajo para

una descripción de la expansión de historia.

Magic-space

Efectuar la expansión de historia en la línea en curso e insertar un espacio. Vea EXPANSIÓN DE HISTORIA abajo para una descripción de la expansión de historia.

Alias-expand-line

Realizar la expansión de alias en la línea en curso. Vea ALIAS arriba para una descripción de la expansión de alias.

History-and-alias-expand-line

Realizar la expansión de historia y alias en la línea en curso.

Insert-last-argument (M-., M-_)

Un sinónimo para yank-last-arg.

Operate-and-get-next (C-o)

Aceptar la línea en curso para la ejecución y obtener la línea siguiente relativa a la actual desde la lista de historia, para la edición. Cualquier argumento no se tiene en cuenta.

Órdenes para cambiar el texto

delete-char (C-d)

Borrar el carácter bajo el cursor. Si el punto está al principio de la línea, no hay caracteres en la línea, y el último carácter tecleado no estaba asociado a delete-char, entonces devolver EOF.

Backward-delete-char (<X], DEL)

Borrar el carácter tras el cursor. Cuando se da un argumento numérico, guardar el texto borrado en el anillo de corte.

Quoted-insert (C-q, C-v)

Añadir el siguiente carácter tecleado a la línea tal cual. Así es como se pueden insertar caracteres como C-q, por ejemplo.

Tab-insert (C-v TAB)

Insertar un carácter de tabulación.

Self-insert (a, b, A, 1, !, ...)

Insertar el carácter tecleado.

Transpose-chars (C-t)
Arrastrar el carácter antes del punto hacia adelante sobre el carácter en el punto. El punto se mueve adelante también. Si el punto está al final de la línea, entonces transpone los dos caracteres antes del punto. Los argumentos negativos no funcionan.
Transpose-words (M-t)
Arrastrar la palabra tras el cursor al lugar pasado la palabra enfrente del cursor, moviendo el cursor también sobre esa palabra.
Upcase-word (M-u)
Poner en mayúsculas la palabra en curso (o la siguiente). Con un argumento negativo, pone la anterior, pero no mueve el punto.
Downcase-word (M-l)
Poner en minúsculas la palabra en curso (o la siguiente). Con un argumento negativo, pone la anterior, pero no mueve el punto.
Capitalize-word (M-c)
Poner en mayúscula la inicial de la palabra en curso (o la siguiente). Con un argumento negativo, pone la anterior, pero no mueve el punto.

Cortar y pegar
kill-line (C-k)
Cortar el texto desde la posición actual del cursor hasta el final de la línea.
Backward-kill-line (C-x DEL)
Cortar hacia atrás hasta el principio de la línea.
Unix-line-discard (C-u)
Cortar hacia atrás desde el punto hasta el principio de la línea. El texto cortado se guarda en el anillo de corte.

Kill-whole-line
Cortar todos los caracteres de la línea en curso, sin importar dónde esté el cursor.
Kill-word (M-d)
Cortar desde el cursor hasta el final de la palabra en curso, o si entre palabras, hasta el final de la siguiente. Los extremos de las palabras son los mismos que los empleados por forward-word.
Backward-kill-word (M-DEL)
Cortar la palabra tras el cursor. Los extremos de las palabras son los mismos que los empleados por backward-word.
Unix-word-rubout (C-w)

Cortar la palabra tras el cursor, empleando el espacio en blanco como un límite de palabra. Los extremos de las palabras son diferentes de los de backward-kill-word.

Delete-horizontal-space (M-\)

Borrar todos los espacios y tabuladores alrededor del punto.

Kill-region

Cortar el texto entre el punto y la marca (posición registrada del cursor). Este texto se conoce como la región.

Copy-region-as-kill

Copiar el texto en la región al anillo de corte.

Copy-backward-word

Copiar la palabra antes del punto al búfer de corte. Los extremos de palabras son los mismos que con backward-word.

Copy-forward-word

Copiar la palabra que sigue al punto al búfer de corte. Los extremos de palabra son los mismos que con forward-word.

Yank (C-y)

Pegar la cima del anillo de corte en el búfer en donde esté el cursor.

Yank-pop (M-y)

Rotar en el anillo de corte, y pegar la nueva cima. Sólo funciona tras un yank o yank-pop.

Argumentos numéricos

digit-argument (M-0, M-1, ..., M--)

Añadir este dígito al argumento, acumulándolo, o comenzar con un nuevo argumento. M-- empieza un argumento negativo.

Universal-argument

Ésta es otra forma de especificar un argumento. Si esta orden se hace seguir de uno o más dígitos, opcionalmente con un signo menos inicial, estos dígitos definen el argumento. Si a la orden siguen dígitos, ejecutar de nuevo universal-argument finaliza el argumento numérico, pero si no, no se tiene en cuenta. Como un caso especial, si a esta orden sigue inmediatamente un carácter que no es ni un dígito ni un signo menos, el número del argumento para la siguiente orden se multiplica por cuatro. El número del argumento es inicialmente uno, así que ejecutar esta función por primera vez hace que el número del argumento sea cuatro, una segunda vez lo hace dieciséis, y así sucesivamente.

Terminación

complete (TAB)

Intentar realizar una terminación del texto antes del punto. Bash intenta la terminación tratando al texto como una variable (si el texto comienza con $), como un nombre de usuario (si el texto empieza con ~), como un nombre de computador anfitrión (si el texto comienza con @), o como una orden (incluyendo alias y funciones), por este orden. Si nada de esto concuerda, se intenta la terminación de un nombre de fichero.

Possible-completions (M-?)

Listar las terminaciones posibles del texto antes del punto.

Insert-completions (M-*)

Insertar todas las terminaciones del texto antes del punto que habrían sido generadas por possible-completions.

Menu-complete

Similar a complete, pero reemplaza la palabra a ser completada con una sola concordancia de la lista de terminaciones posibles. La ejecución repetida de menu-complete camina por la lista de terminaciones posibles, insertando cada concordancia por turnos. Al final de la lista de terminaciones, se hace sonar el pito de la terminal y el texto original se restaura.

Un argumento n mueve n posiciones hacia delante en la lista de concordancias; un argumento negativo se puede emplear para moverse hacia atrás en la lista. Esta orden está pensada para ser asociada a TAB, pero no está asociada a ninguna tecla de forma predeterminada.

Complete-filename (M-/)

Intentar la terminación de un nombre de fichero en el texto antes del punto.

Possible-filename-completions (C-x /)

Listar las posibles terminaciones del texto antes del punto, tratándolo como un nombre de fichero.

Complete-username (M-~)

Intentar la terminación del texto antes del punto, tratándolo como un nombre de usuario.

Possible-username-completions (C-x ~)

Listar las posibles terminaciones del texto antes del punto, tratándolo como un nombre de usuario.

Complete-variable (M-$)

Intentar la terminación del texto antes del punto, tratándolo como una variable del shell.

Possible-variable-completions (C-x $)

Listar las posibles terminaciones del texto antes del punto, tratándolo como una variable del shell.

Complete-hostname (M-@)

Intentar la terminación del texto antes del punto, tratándolo como un nombre de computador

anfitrión.

Possible-hostname-completions (C-x @)

Listar las posibles terminaciones del texto antes del punto, tratándolo como un nombre de computador anfitrión.

Complete-command (M-!)

Intentar la terminación del texto antes del punto, tratándolo como un nombre de orden. La terminación de orden intenta hacer concordar el texto con alias, palabras reservadas, funciones del shell, órdenes internas del shell, y finalmente nombres de ficheros ejecutables, en ese orden.

Possible-command-completions (C-x !)

Listar las posibles terminaciones del texto antes del punto, tratándolo como un nombre de orden.

Dynamic-complete-history (M-TAB)

Intentar la terminación del texto antes del punto, comparando el texto con líneas de la lista de historia buscando concordancias para la terminación.

Complete-into-braces (M-{)

Efectuar la terminación de nombres de ficheros y devolver la lista de terminaciones posibles encerrada entre llaves de forma que la lista esté disponible al shell (vea Expansión de llaves arriba).

Macros de teclado

start-kbd-macro (C-x ()

Empezar a grabar los caracteres tecleados, en la macro de teclado en curso.

End-kbd-macro (C-x))

Parar de grabar los caracteres tecleados en la macro de teclado en curso, y almacenar la definición.

Call-last-kbd-macro (C-x e)

Reejecutar la última macro de teclado definida, haciendo que los caracteres en la macro aparezcan como si se hubieran pulsado en el teclado.

Miscelánea

re-read-init-file (C-x C-r)

Leer los contenidos del fichero inputrc, e incorporar cualesquiera definiciones de teclas o asignaciones de variables que se hubieran encontrado en él.

Abort (C-g)

Terminar abruptamente la orden de edición en curso y tocar el pito de la terminal (según el

establecimiento de bell-style).

Do-uppercase-version (M-a, M-b, M-x, ...)

Si el carácter meta x está en minúscula, ejecutar la orden que esté asociada al carácter correspondiente en mayúscula.

Prefix-meta (ESC)

Convertir en meta el siguiente carácter tecleado. ESC f es equivalente a Meta-f.

Undo (C-_, C-x C-u)

Deshacer de forma incremental, recordado separadamente para cada línea.

Revert-line (M-r)

Deshacer todos los cambios hechos a esta línea. Esto es como ejecutar la orden undo las veces suficientes como para devolver la línea a su estado inicial.

Tilde-expand (M-~)

Efectuar la expansión de tilde en la palabra en curso.

Set-mark (C-@, M-<espacio>)

Establecer la marca en el punto actual. Si se da un argumento numérico, la marca se establece en esa posición.

Exchange-point-and-mark (C-x C-x)

Cambia el punto con la marca. La posición actual del cursor se pone en la posición guardada, y la vieja posición del cursor se guarda como la marca.

Character-search (C-])

Se lee un carácter y el punto se mueve a la siguiente ocurrencia de ese carácter. Un argumento numérico negativo hace que la búsqueda sea de las ocurrencias anteriores.

Character-search-backward (M-C-])

Se lee un carácter y el punto se mueve a la anterior ocurrencia de ese carácter. Un argumento numérico negativo hace que la búsqueda sea de las ocurrencias siguientes.

Insert-comment (M-#)

El valor de la variable de readline comment-begin se inserta al principio de la línea en curso, y la línea se acepta como si se hubiera tecleado Intro (nueva-línea). Esto convierte la línea en curso en un comentario del shell.

Glob-expand-word (C-x *)

La palabra antes del punto se trata como un patrón para la expansión de nombres de caminos y la lista de nombres de fichero concordantes se inserta, reemplazando a la palabra.

Glob-list-expansions (C-x g)

Mostrar la lista de expansiones que habrían sido generadas por glob-expand-word y redibujar la línea.

Dump-functions

Mostrar todas las funciones y sus asociaciones de teclas en el flujo de salida de readline. Si se da un argumento numérico, la salida se formatea de tal modo que pueda formar parte de un fichero inputrc.

Dump-variables

Mostrar todas las variables de readline a las que se puedan asignar valores, y éstos, en el flujo de salida de readline. Si se da un argumento numérico, la salida se formatea de tal manera que pueda formar parte de un fichero inputrc.

Dump-macros

Mostrar todas las secuencias de teclas de readline asociadas a macros y las cadenas de caracteres asociadas correspondientes. Si se da un argumento numérico, la salida se formatea de tal manera que pueda formar parte de un fichero inputrc.

Display-shell-version (C-x C-v)

Mostrar información de versión acerca de la instancia actual de bash.

HISTORIA

Cuando se habilita la opción -o history de la orden interna set, el shell da acceso a la historia de órdenes, la lista de órdenes tecleadas con anterioridad. El texto de los últimos HISTSIZE mandatos (por omisión, 500) se guarda en una lista de historia. El shell almacena cada orden en la lista de historia antes de la expansión de parámetros y variables (vea EXPANSIÓN arriba) pero tras efectuar la expansión de historia, sujeta a los valores de las variables del shell HISTIGNORE e HISTCONTROL.

En el arranque, la historia se inicia a partir del fichero nombrado en la variable HISTFILE (por omisión ~/.bash_history). HISTFILE se trunca, si es necesario, para contener no más de HISTFILESIZE líneas. Cuando un shell interactivo termina, las últimas HISTSIZE líneas se copian de la lista de historia a HISTFILE. Si la opción del shell histappend está activa (vea la descripción de shopt bajo ÓRDENES INTERNAS DEL SHELL más adelante), las líneas se añaden al fichero de historia; si no, el fichero de historia se sobreescribe. Si HISTFILE no está definido, o si no se puede escribir en el fichero de historia, la historia no se guarda. Tras guardar la historia, el fichero de historia

se trunca para contener no más de HISTFILESIZE líneas. Si HISTFILESIZE no está definido, no se

trunca.

La orden interna fc (vea ÓRDENES INTERNAS DEL SHELL abajo) puede emplearse para listar o editar y

re-ejecutar una porción de la lista de historia. La orden interna history se puede utilizar para

mostrar o modificar la lista de historia y manipular el fichero de historia. Cuando se emplea la

edición de líneas de órdenes, están disponibles las órdenes de búsqueda en cada modo de edición que

proporcionan acceso a la lista de historia.

El shell permite el control sobre qué órdenes se guarden en la lista de historia. Las variables

HISTCONTROL y HISTIGNORE se pueden definir de forma que el shell guarde solamente un subconjunto de

las órdenes introducidas. La opción del shell cmdhist, si está habilitada, hace que el shell

intente guardar cada línea de una orden multi-línea en la misma entrada de la historia, añadiendo

punto y comas donde sea necesario para preservar la corrección sintáctica. La opción del shell

lithist hace que el shell guarde la orden con saltos de línea empotrados en vez de punto y comas.

Vea la descripción de la orden interna shopt abajo en ÓRDENES INTERNAS DEL SHELL para información

sobre cómo establecer y anular opciones del shell.

EXPANSIÓN DE HISTORIA

El shell admite una característica de expansión de historia que es parecida a la expansión de histo-

ria en csh. Esta sección describe qué características sintácticas están disponibles. Esta carac-

terística está habilitada de forma predeterminada en shells interactivos, y puede ser desactivada

mediante la opción +H de la orden interna set (vea ÓRDENES INTERNAS DEL SHELL abajo). Los shells no

interactivos no realizan la expansión de la historia de forma predeterminada.

Las expansiones de historia introducen palabras desde la lista de historia en el flujo de entrada,

facilitando así la repetición de órdenes, la inserción de argumentos de una orden anterior en la

línea de entrada en curso, o la corrección rápida de errores en una orden anterior.

La expansión de historia se realiza inmediatamente tras la lectura de una línea completa, antes de que el shell la divida en palabras. Tiene lugar en dos fases. En la primera se determina qué línea de la lista de historia hay que emplear durante la sustitución. En la segunda se seleccionan porciones de esa línea para su inclusión en la actual. La línea seleccionada desde la historia es el evento, y la porción de esa línea sobre la que se actúa son palabras. Se dispone de varios modificadores para manipular las palabras seleccionadas. La línea se divide en palabras de la misma manera que cuando se lee la entrada, de forma que ciertas palabras separadas por meta-caracteres rodeadas por comillas se consideran una sola palabra. Las expansiones de historia se introducen por la aparición del carácter de expansión de historia, que es por omisión !. Sólo las barras inversas (\) y las comillas simples pueden proteger al carácter de expansión de historia.

Se pueden emplear ciertas opciones que se ponen con la orden interna shopt para cambiar el comportamiento de la expansión de historia. Si la opción del shell histverify está activa (vea la descripción de la orden interna shopt), y se está usando readline, las sustituciones de historia no se pasan inmediatamente al analizador del shell. En vez de eso, la línea expandida se vuelve a cargar en el búfer de edición de readline para una modificación posterior. Si se está usando readline y la opción del shell histreedit está activada, una sustitución de historia fallida se volverá a cargar en el búfer de edición de readline para su corrección. La opción -p de la orden interna history se puede emplear para ver qué hará una expansión de historia antes de usarse. La opción -s de la orden interna history se puede emplear para añadir órdenes al final de la lista de historia sin ejecutarlas realmente, de modo que estén disponibles para rellamadas posteriores.

El shell permite el control de los diversos caracteres empleados por el mecanismo de

expansión de

historia (vea la descripción de histchars arriba en Variables del shell).

Designadores de eventos

Un designador de evento es una referencia a una entrada de línea de orden en la lista de historia.

! Comenzar una sustitución de historia, excepto cuando le siga un blanco, salto de línea, = o

(.

!n Referirse a la línea de órdenes número n.

!-n Referirse a la línea de orden en curso menos n.

!! Referirse a la orden anterior. Esto es lo mismo que `!-1'.

!cadena

Referirse a la orden más reciente que comience con cadena.

!?cadena[?]

Referirse a la orden más reciente que contenga cadena. El ? Del final puede omitirse si a cadena le sigue inmediatamente un salto de línea.

^cadena1^cadena2^

Sustitución rápida. Repetir la última orden, reemplazando cadena1 con cadena2. Equivalente a `!!:s/cadena1/cadena2/" (vea Modificadores abajo).

!# La línea de orden entera tecleada hasta ahora.

Designadores de palabras

Los designadores de palabras se emplean para seleccionar las palabras que se deseen del evento. Un

: separa la especificación de evento del designador de palabra. Puede omitirse si el designador de

palabra comienza con un ^, $, *, -, o %. Las palabras se numeran desde el principio de la línea,

con la primera palabra denotada por 0 (cero). Las palabras se insertan en la línea en curso separadas por espacios simples.

0 (cero)

La palabra número 0. Para el shell, ésta es la palabra de la orden.

N La n-sima palabra.

^ El primer argumento. Esto es, la palabra número 1.

$ El último argumento.

% La palabra que concordaba con la más reciente búsqueda con `?cadena?'.

x-y Un rango de palabras; `-y' abrevia `0-y'.

- **Todas las palabras menos la número cero. Esto es un sinónimo de `1-$'. No es un error**

emplear * si sólo hay una palabra en el evento; en este caso se devuelve la cadena vacía.

X* Abreviatura de x-$.

x- Abrevia x-$ como x*, pero omite la última palabra.

Si se suministra un designador de palabra sin una especificación de evento, se usa la orden anterior

como el evento.

Modificadores

Tras el designador opcional de palabra, puede haber una secuencia de uno o más de los siguientes

modificadores, precedido cada uno por un `:'.

h Quitar un componente final de nombre de fichero, dejando sólo la parte izquierda.

T Quitar todos los primeros componentes de un nombre de fichero, dejando la última parte.

R Quitar un sufijo final de la forma .xxx, dejando el nombre base.

E Quitar todo salvo el sufijo final.

P Mostrar la nueva orden pero no ejecutarla.

Q Entrecomillar las palabras sustituidas, escapando de posteriores sustituciones.

X Entrecomillar las palabras sustituidas como con q, pero romper entre palabras en los blancos

y saltos de línea.

s/viejo/nuevo/

Substituir nuevo por la primera ocurrencia de viejo en la línea de evento. Se puede emplear cualquier delimitador en vez de /. El delimitador final es opcional si es el último carácter de la línea de evento. El delimitador puede entrecomillarse en viejo y nuevo con una sola barra inclinada inversa. Si & aparece en nuevo, se reemplaza por viejo. Una sola barra inversa protegerá el &. Si viejo está vacío, se pone al último viejo substituido, o, si no tuvo lugar ninguna sustitución de historia previa, a la última cadena en una búsqueda del tipo !?cadena[?].

& Repetir la sustitución anterior.

G Hace que los cambios se apliquen sobre la línea entera de evento. Esto se emplea en

con-
junción con `:s' (p.ej., `:gs/viejo/nuevo/') o `:&'. Si se usa con `:s', cualquier delimitador se puede utilizar en lugar de /, y el delimitador final es opcional si es el último carácter de la linea de evento.

ÓRDENES INTERNAS DEL SHELL

A menos que se diga otra cosa, cada orden interna documentada en esta sección que acepte opciones
precedidas por – también acepta – para significar el final de las opciones.

: [argumentos]
Sin efecto; la orden no hace nada más que expandir argumentos y realizar cualquier redirección que se haya especificado. Se devuelve un código de salida cero.

. nombre-fichero [argumentos]
source nombre-fichero [argumentos]
Lee y ejecuta órdenes desde nombre-fichero en el entorno actual del shell y devuelve el estado de salida de la última orden ejecutada desde nombre-fichero. Si nombre-fichero no contiene una barra inclinada, se usan los nombres de fichero en PATH para encontrar el directorio que contenga a nombre-fichero. El fichero que se busca en PATH no necesita ser ejecutable. Se busca en el directorio de trabajo si no se encontró el fichero en PATH. Si la opción sourcepath de la orden interna shopt está desactivada, la búsqueda en PATH no se realiza. Si se suministran argumentos, se convierten en los parámetros posicionales cuando se ejecuta nombre-fichero. Si no, los parámetros posicionales permanecen inalterados. El estado de retorno es el de la última orden de dentro del guión (0 si no se ejecutó ninguna orden), y `falso' si nombre-fichero no se encontró o no se pudo leer.

Alias [-p] [nombre[=valor] ...]
Alias sin argumentos o con la opción -p muestra la lista de alias en la forma alias nombre=valor en la salida estándar. Cuando se dan argumentos, se define un alias para cada nombre cuyo valor se da. Un espacio extra tras valor hace que en la siguiente palabra se realice la sustitución de alias cuando el alias se expande. Para cada nombre en la lista de argumentos para el que no se suministre un valor, se muestran el nombre y el valor del alias. Alias devuelve `verdad' a menos que se dé un nombre para el que no se haya definido un alias.

Bg [espectrab]
Reanuda el trabajo suspendido espectrab en segundo plano, como si se hubiera arrancado con &.

Si espectrab no está presente, se emplea la noción que tiene el shell del trabajo en curso. Bg espectrab devuelve 0 a menos que se ejecute cuando el control de trabajos esté deshabilitado, o cuando se ejecute con el control de trabajos habilitado si espectrab no se encontró, o cuando se hubo arrancado sin control de trabajos.

Bind [-m mapatecl] [-lpsvPSV]
bind [-m mapatecl] [-q función] [-u función] [-r sectecl] ***bind [-m mapatecl] -f nombre-fichero***
bind [-m mapatecl] sectecl:nombre-función
Muestra las asociaciones actuales de readline de teclas y funciones, o asocia una secuencia de teclas a una función o macro de readline. La sintaxis aceptada es idéntica a la de .inputrc, pero cada asociación debe pasarse como un argumento separado; p.ej., '"\C-x\C-r": re-read-init-file'. Las opciones, si se dan, tienen los siguientes significados:
-m mapatecl
Usar mapatecl como el mapa de teclado que va a verse afectado por subsiguientes asociaciones. Los nombres aceptables de mapatecl son emacs, emacs-standard, emacs-meta, emacs-ctlx, vi, vi-command y vi-insert. Vi equivale a vi-command; emacs es equivalente a emacs-standard.
-l Lista los nombres de todas las funciones de readline.
-p Muestra los nombres de funciones de readline de tal forma que puedan volver a ser leídas.
-P Lista los nombres de funciones de readline actuales y las asociaciones.
-v Muestra los nombres de variables de readline y los valores de tal manera que puedan volver a ser leídas.
-V Lista los nombres de variables de readline actuales y los valores.
-s Muestra las secuencias de teclas de readline asociadas a macros y las cadenas correspondientes de tal manera que puedan ser leídas de nuevo.
-S Muestra las secuencias de teclas de readline asociadas a macros y las cadenas de caracteres correspondientes.
-f nombre-fichero
Lee las asociaciones de teclas desde nombre-fichero.
-q función
Pregunta qué teclas llaman a la función especificada.
-u función
Desenlaza todas las teclas asociadas a la función nombrada.
-r sectecl

Borra cualquier asociación actual de sectecl.

El valor devuelto es 0 a menos que se dé una opción no reconocida o que ocurra un error.

Break [n]
Sale de un bucle for, while, until, o select. Si se especifica n, sale de n niveles. N debe ser >= 1. Si n es mayor que el número de bucles, se sale de todos. El valor devuelto es 0 a menos que el shell no esté ejecutando un bucle cuando se ejecute break.

Builtin orden-interna [argumentos]
Ejecuta la orden interna del shell especificada, pasándole los argumentos, y devuelve su estado de salida. Esto es útil cuando se define una función cuyo nombre es el mismo que una orden interna del shell, reteniendo la funcionalidad de esa orden interna dentro de la función. Por ejemplo, la orden interna cd se puede redefinir normalmente de esta manera. El estado de salida es `falso' si orden-interna no es una orden incorporada del shell.

Cd [-LP] [dir]
Cambia el directorio de trabajo en curso a dir. La variable HOME contiene el nombre del directorio predeterminado dir. La variable CDPATH define el camino de búsqueda del directorio que contenga dir. Los nombres de directorios alternativos en CDPATH se separan por dos puntos (:). Un nombre de directorio vacío en CDPATH es lo mismo que el directorio de trabajo en curso, o sea, `.". Si dir comienza con una barra inclinada (/), entonces CDPATH no se usa. La opción -P dice que se va a emplear la estructura física de directorios en vez de seguir enlaces simbólicos (vea también la opción -P de la orden interna set); la opción -L hace que se sigan siempre los enlaces simbólicos. El argumento - es equivalente a $OLDPWD.
El valor de retorno es `verdad' si el directorio de trabajo se cambió con éxito; `falso' en otro caso.

Command [-pVv] orden [arg ...]
Ejecuta orden con args suprimiendo la búsqueda normal de funciones del shell. Sólo se ejecutarán órdenes incorporadas en el shell o programas encontrados en la variable PATH. Si se da la opción -p, se busca la orden empleándose un valor predeterminado para PATH que garantiza
encontrar todas las utilidades estándar del sistema. Si se da la opción -V o la -v, se muestra una descripción de orden. La opción -v muestra una sola palabra que indica la orden o nombre de fichero ejecutable empleado para la llamada a orden; la opción -V produce una

descripción algo más prolija. Si se dan las opciones -V o -v, el estado de salida es 0 si orden se encontró, y 1 si no. Si no se da ninguna de las dos opciones y ocurrió un error u orden no se encontró, el estado de salida es 127. Si no, el estado de salida de command es el estado de salida de orden.

Continue [n]

Reanuda la siguiente iteración del bucle for, while, until, o select donde estamos. Si se especifica n, la reanudación es en el n-simo bucle que nos rodea. N debe ser >= 1. Si n es mayor que el número de bucles que nos rodean, se reanuda el bucle más exterior (el de `más alto nivel"). El valor de retorno es 0 a menos que el shell no esté ejecutando un bucle cuando se ejecute continue.

Declare [-afFirx] [-p] [nombre[=valor]]

typeset [-afFirx] [-p] [nombre[=valor]]

Declaran variables o les dan atributos. Si no se dan nombres, entonces muestran los valores de las variables. La opción -p mostrará los atributos y valores de cada nombre. Cuando se emplee -p, otras opciones que se hayan dado no se tienen en cuenta. La opción -F inhibe la presentación de las definiciones de funciones; sólo se muestran sus nombres y atributos. La opción -F implica -f. Se pueden emplear las siguientes opciones para restringir la salida a variables con el atributo especificado o para dar atributos a variables:

-a Cada nombre es una variable vector (vea Vectores arriba).

-f Usar solamente nombres de funciones.

-i La variable se trata como un entero; se realiza la evaluación aritmética (vea EVALUACIÓN ARITMÉTICA) cuando a la variable se le asigne un valor.

-r Hace que nombres sean de lectura exclusiva. A estos nombres no se les pueden asignar valores por medio de subsiguientes asignaciones, ni se puede anular su definición con unset.

-x Marca nombres para la exportación a órdenes subsecuentes a través del entorno.

Usar `+' en vez de `-' desactiva el atributo en vez de activarlo, con la excepción de que no puede emplearse +a para destruir una variable vector. Cuando se usa en una función, hace local cada nombre, como con la orden interna local. El valor de retorno es 0 a menos que se encuentre una opción inválida, se intente definir una función utilizando "-f fuu=bar", se intente asignar un valor a una variable de lectura exclusiva, se intente asignar un valor a una variable vector sin emplear la sintaxis de asignación compuesta (vea Vectores arriba), uno de los nombres no sea un nombre válido de variable del shell, se intente desactivar el

estado de lectura exclusiva para una variable de sólo lectura, se intente desactivar el estado de vector para una variable vector, o se intente mostrar una función no existente con -f.

dirs [-clpv] [+n] [-n]

Sin opciones, muestra la lista de directorios actualmente recordados. La forma predeterminada de mostrarlos es en una sola línea con los nombres de directorios separados por espacios. Los directorios se añaden a la lista (en realidad, una pila) con la orden pushd; la orden popd los quita de la pila.

+n Muestra la n-sima entrada contando desde la izquierda de la lista mostrada por dirs cuando se llama sin opciones, empezando por cero.

-n Muestra la n-sima entrada contando desde la derecha de la lista mostrada por dirs cuando se llama sin opciones, empezando por cero.

-c Limpia la pila de directorios borrando todas las entradas.

-l Produce un listado más largo; el formato predeterminado de listado emplea una tilde de la eñe para señalar el directorio inicial de trabajo (el "hogar").

-p Muestra la pila de directorios con una entrada por cada línea.

-v Muestra la pila de directorios con una entrada por línea, prefijando cada entrada con su índice en la pila.

El valor de retorno es 0 a menos que se dé una opción inválida o que n indexe más allá del final de la pila de directorios.

Disown [-ar] [-h] [espectrab ...]

Sin opciones, cada espectrab se quita de la tabla de trabajos activos. Si se da la opción -h, cada espectrab no se quita de la tabla, sino que se marca de manera que no se le enviará la señal SIGHUP si el shell recibe una señal SIGHUP. Si no hay ningún espectrab presente, y ni se dan las opciones -a ni -r, se utiliza el trabajo en curso. Si no se suministra espectrab, la opción -a significa quitar o marcar todos los trabajos; la opción -r sin un argumento espectrab restringe la operación a los trabajos en ejecución. El valor de retorno es 0 a menos que espectrab no se refiera a un trabajo válido.

Echo [-neE] [arg ...]

Repite los args, separados por espacios, seguidos por un salto de línea. El estado de retorno es siempre 0. Si se especifica -n, se suprime el salto de línea final. Si se da la opción -e, se activa la interpretación de los siguientes caracteres de escape (con barra

inversa). La opción -E desactiva la interpretación de estos caracteres de escape, incluso en sistemas donde se interpreten de forma predeterminada. Echo no interpreta – como el fin de las opciones. Echo interpreta las siguientes secuencias de escape:

\a alerta (pito)

\b espacio atrás

\c suprime el salto de línea final

\e un carácter Escape

\f salto de página

\n nueva línea

\r retorno de carro

\t tabulador horizontal

\v tabulador vertical

\\ barra inclinada invertida *nnn el carácter cuyo código es el valor octal nnn (de uno a tres dígitos) \xnnn el carácter cuyo código es el valor hexadecimal nnn (de uno a tres dígitos)*

enable [-adnps] [-f nombre-fichero] [nombre ...]

Activa y desactiva órdenes internas incorporadas en el shell. Desactivar una orden incorporada en el shell permite que un programa cuya imagen ejecutable esté en el disco con el mismo nombre, se ejecute sin tener que especificar un camino completo, aun cuando el shell normalmente busca las órdenes internas antes que las órdenes externas en disco. Si se da -n, cada nombre se desactiva; si no, nombres se activan. Por ejemplo, para emplear el programa binario test encontrado en el PATH en vez de la versión incorporada en el shell, ejecute enable -n test. La opción -f significa cargar la nueva orden incorporada nombre desde el objeto compartido nombre-fichero, en sistemas que admiten la carga dinámica. La opción -d borrará una orden interna cargada previamente con -f. Si no se dan argumentos nombre, o si se da la opción -p, se muestra una lista de órdenes internas del shell. Sin ninguna opción más, la lista consiste en todas las órdenes internas activas. Si se da -n, sólo se muestran las órdenes desactivadas. Si se da -a, la lista impresa incluye todas las órdenes internas, con una indicación en cada una de si está activada o no. Si se da -s, la salida se restringe a las órdenes internas especiales de POSIX. El valor de retorno es 0 a menos que un nombre no sea una orden interna del shell o que haya un problema cargando una nueva orden incorporada desde un objeto compartido.

Eval [arg ...]

Los args se leen y concatenan juntos para formar una sola orden. Entonces el shell lee y ejecuta esta orden, y su estado de salida se devuelve como el valor de eval. Si no hay args, o

solamente argumentos vacíos, eval devuelve 0.

exec [-cl] [-a nombre] [orden [argumentos]]

Si se especifica orden, reemplaza al shell. No se crea ningún proceso nuevo. Los argumentos se convierten en los argumentos de orden. Si se da la opción -l, el shell pone un guión en el argumento número cero pasado a orden. Esto es lo que hace login(1). La opción -c hace que orden se ejecute con un entorno vacío. Si se da -a , el shell pasa nombre como el argumento número cero de la orden ejecutada. Si orden no puede ejecutarse por alguna razón, un shell no interactivo se acaba, a menos que se haya activado la opción del shell execfail, en cuyo caso devuelve 'fallo'. Un shell interactivo devuelve 'fallo' si el fichero no puede ejecutarse. Si orden no se especifica, cualquier redirección tiene efecto en el shell en curso, y el estado devuelto es 0. Si hay un error de redirección, el estado de retorno es 1.

exit [n]

Hace que el shell acabe con un estado de valor n. Si n se omite, el valor de salida es el de la última orden ejecutada. Se ejecuta una trampa puesta en EXIT antes de que el shell termine.

Export [-fn] [nombre[=palabra]] ...

export -p

Los nombres suministrados se marcan para exportación automática al entorno de órdenes que se

ejecuten de ahí en adelante. Si la opción -f se da, los nombres se referirán a funciones. Si no se dan nombres, o si se da la opción -p, se muestra una lista de todos los nombres que están exportados en este shell. La opción -n hace que la propiedad de exportación se quite de las variables nombradas; o sea, que se quiten del entorno. Export devuelve un estado de salida de 0 a menos que se encuentre una opción inválida, uno de los nombres no sea un nombre

válido de variable del shell, o que se dé -f no siendo nombre una función.

Fc [-e editor] [-nlr] [primero] [último]

fc -s [pat=rep] [ord]

Corrige orden (Fix Command). En la primera forma, se selecciona un rango de órdenes desde primero hasta último de la lista de historia. Primero y último pueden especificarse como una cadena de caracteres (para localizar la última orden que empezara con esa cadena) o como un número (un índice en la lista de historia, donde un número negativo se emplea como un desplazamiento a partir del número de orden actual). Si no se da último, se pone a la orden

en curso para un listado (de forma que fc -l -10 muestra las 10 últimas órdenes) o a primero en otro caso. Si no se especifica primero, se pone a la orden previa para una edición y a -16 para un listado.

La opción -n suprime los números de orden en un listado. La opción -r invierte el orden de los mandatos. Si se da la opción -l , las órdenes se listan en la salida estándar. Si no, se llama al editor dado por editor en un fichero conteniendo esas órdenes. Si no se da editor, se emplea el valor de la variable FCEDIT, o el valor de EDITOR si FCEDIT no está definido. Si no está definida ni una variable ni otra, vi es el editor que se empleará. Cuando la edición se haya completado, las órdenes editadas se muestran y ejecutan.

En la segunda forma, orden se re-ejecuta tras que cada instancia de pat se reemplace por rep. Un alias de utilidad para emplear con esto es `r=fc -s", de tal modo que al teclear `r cc" se ejecute la última orden que empezara con `cc" y que al teclear `r" se re-ejecute la última orden.

Si se usa la primera forma, el valor de retorno es 0 a menos que se encuentre una opción inválida o que primero o último especifiquen líneas de historia fuera de rango. Si se suministra la opción -e, el valor de retorno es el de la última orden ejecutada, o `fallo' si ocurre un error con el fichero temporal de órdenes. Si se emplea la segunda forma, el estado de retorno es de la orden reejecutada, a menos que ord no especifique una línea válida de historia, en cuyo caso fc devuelve `fallo'.

Fg [espectrab] ***Reanuda espectrab en primer plano, y lo hace ser el trabajo en curso. Si espectrab no está presente, se usa la noción que tiene el shell del trabajo en curso. El valor de retorno es el de la orden puesta en primer plano, o `fallo' si se ejecuta fg cuando el control de trabajos está desactivado o, cuando se ejecuta con el control de trabajos activado, si espectrab no especifica un trabajo válido o espectrab se refiere a una trabajo que se arrancó sin control de trabajos.***

Getopts cadenaopcs nombre [args]

getopts se usa en procedimientos del shell para analizar los parámetros posicionales. Cadenaopcs contiene las letras de opción que se reconocerán; si a una letra le sigue un signo de dos puntos, se espera que la opción tenga un argumento, que debería estar separado de ella por espacio en blanco. Cada vez que se llama, getopts pone la siguiente opción en la variable del shell nombre, inicializando nombre si no existe, y el índice del siguiente argumento a ser procesado en la variable OPTIND. OPTIND se inicializa a 1 cada vez que se llama al shell o al guión del shell. Cuando una opción requiera un argumento, getopts pone ese argu-

mento en la variable OPTARG. El shell no pone a cero OPTIND automáticamente; debe ser puesto a cero manualmente entre llamadas múltiples a getopts dentro de la misma llamada al shell si hubiera que usar un nuevo conjunto de parámetros.

Cuando se encuentra el fin de las opciones, getopts sale con un valor de retorno mayor que cero. OPTIND se pone al índice del primer argumento que no es una opción, y nombre se pone a "?".

getopts normalmente analiza los parámetros posicionales, pero si se dan más argumentos en args, getopts analiza aquéllos en su lugar.

Getopts puede informar de errores de dos maneras. Si el primer carácter de optstring es dos puntos, se emplea un informe de error silencioso. En un modo normal de operación se muestran mensajes de diagnósticos cuando se encuentran opciones inválidas o faltan argumentos de opciones que los requieran. Si la variable OPTERR se pone a 0, no se mostrará ningún mensaje de error, incluso si el primer carácter de optstring no es dos puntos.

Si se ve una opción inválida, getopts pone ? en nombre y, si no estamos en el modo silencioso, se muestra un mensaje de error y se anula OPTARG. Si getopts es silencioso, el carácter de opción encontrado se pone en OPTARG y no se muestra ningún mensaje de diagnóstico.

Si no se encuentra un argumento necesario, y getopts no es silencioso, se pone un signo de cierre de interrogación (?) en nombre, OPTARG se anula, y se muestra un mensaje de diagnóstico. Si getopts es silencioso, entonces se pone un signo de dos puntos (:) en nombrey OPTARG toma el valor del carácter de opción encontrado.

Getopts devuelve 'verdad' si una opción, especificada o no, se encuentra. Devuelve 'falso' si se encuentra el final de las opciones o si ocurre un error.

Hash [-r] [-p nombre-fichero] [nombre]

Para cada nombre, se determina el nombre completo del fichero correspondiente buscando los directorios en $PATH y dicho nombre completo se registra. Si se da la opción -p no se busca en el PATH, y se emplea en su lugar nombre-fichero como el nombre completo de fichero para la

orden. La opción -r hace que el shell se olvide de todas las localizaciones registradas. Si no se dan argumentos, se muestra información acerca de las órdenes recordadas. El estado de salida es ‘verdad’ a menos que un nombre no se encuentre o se dé una opción inválida.

Help [patrón]
Muestra información de ayuda sobre las órdenes internas. Si se especifica patrón, help da ayuda detallada sobre todas las órdenes que concuerden con patrón; si no, se muestra ayuda para todas las órdenes internas y estructuras de control del shell. El estado de retorno es 0 a menos que ninguna orden concuerde con patrón.

History [-c] [n]
history -anrw [nombre-fichero]
history -p arg [arg ...]
history -s arg [arg ...]
Sin opciones, muestra la lista de historia de órdenes con números de línea. Las líneas marcadas con un * han sido modificadas. Un argumento de n lista solamente las últimas n líneas. Si nombre-fichero se da, se emplea como el nombre del fichero de historia; si no, se usa el valor de HISTFILE. Las opciones, si se dan, tienen los siguientes significados:
-a Añade las líneas de historia ‘nuevas” (las introducidas desde el inicio de la sesión de bash en curso) al fichero de historia.
-n Lee las líneas de historia que aún no han sido leídas del fichero de historia y las mete en la lista de historia actual. Éstas son las líneas añadidas al fichero de historia desde el comienzo de la sesión de bash actual.
-r Lee los contenidos del fichero de historia y los usa como la historia en curso.
-w Escribe la historia en curso en el fichero de historia, sobreescribiendo sus contenidos.
-c Limpia la lista de historia borrando todas las entradas.
-p Realiza la sustitución de historia de los siguientes args y muestra el resultado en la salida estándar. No almacena los resultados en el fichero de historia. Cada arg debe protegerse para deshabilitar la expansión de historia normal.
-s Almacena los args en la lista de historia como una sola entrada. La última orden en la lista de historia se elimina antes de que se añadan los args.
El valor de retorno es 0 a menos que se encuentre una opción inválida o que ocurra un error mientras se lee o se escribe el fichero de historia.

Jobs [-lnprs] [espectrab ...]

jobs -x orden [args ...]

La primera forma lista los trabajos activos. Las opciones tienen los significados siguientes:

-l Lista PIDs además de la información normal.

-p Lista solamente el PID del líder del grupo de proceso del trabajo.

-n Muestra información solamente sobre trabajos que han cambiado de estado desde que se notificó al usuario por última vez de su estado.

-r Restringe la salida a los trabajos en ejecución.

-s Restringe la salida a los trabajos parados.

Si se da espectrab, la salida se restringe a la información sobre ese trabajo. El estado de retorno es cero a menos que se encuentre una opción inválida o se suministre un espectrab inválido.

Si se da la opción -x, jobs reemplaza cualquier espectrab encontrado en orden o args con el ID de grupo de proceso correspondiente, y ejecuta orden pasándole args, devolviendo su estado de salida.

Kill [-s nombre-señal | -n numseñal | -nombre-señal] [pid | espectrab] ...

kill -l [nombre-señal | status_salida]

Envía la señal especificada por nombre-señal o numseñal a los procesos nombrados por pid o espectrab. Nombre-señal es o bien un nombre de señal como SIGKILL o un número de señal;

numseñal es un número de señal. Si nombre-señal es un nombre de señal, éste puede darse con o

sin el prefijo SIG. Si nombre-señal no está presente, entonces se supone SIGTERM. Un argu-

mento de -l lista los nombres de señales. Si se suministran más argumentos cuando se da -l, se listan los nombres de las señales correspondientes a los argumentos, y el estado de retorno es 0. El argumento status_salida de -l es un número que especifica o bien un número de señal o el estado de salida de un proceso terminado por una señal. Kill devuelve `verdad" si por lo menos se envió con éxito una señal, o `falso" si ocurrió un error o se encontró una opción inválida.

Let arg [arg ...]

Cada arg es una expresión aritmética a ser evaluada (vea EVALUACIÓN ARITMÉTICA). Si el

último arg se evalúa a 0, let devuelve 1; si no, devuelve 0.

local [nombre[=valor] ...]

Para cada argumento, se crea una variable local llamada nombre, y se le asigna el valor valor. Cuando local se emplea dentro de una función, hace que la variable nombre tenga una *visibilidad restringida a esa función y sus hijas. Sin operandos*

local escribe en la salida

estándar una lista de las variables locales. Es un error emplear local fuera de una función. El estado de retorno es 0 a menos que local se use fuera de una función, o se dé un nombre inválido.

Logout Hace terminar un shell de entrada.

Popd [-n] [+n] [-n]

Quita entradas de la pila de directorios. Sin argumentos, desapila el directorio de la cima de la pila, y realiza un cd al nuevo directorio de la pila. Los argumentos, si se suministran, tienen los siguientes significados:

+n Quita la n-sima entrada contando desde la izquierda de la lista mostrada por dirs, empezando por cero. Por ejemplo: `popd +0" quita el primer directorio, `popd +1" el segundo.

-n Quita la n-sima entrada contando desde la derecha de la lista mostrada por dirs, empezando por cero. Por ejemplo: `popd -0" quita el último directorio, `popd -1" el penúltimo.

-n Suprime el normal cambio de directorio cuando se desapilan directorios, de forma que solamente la pila se manipula.

Si la orden popd es exitosa, también se realiza un dirs, y el estado de retorno es 0. popd devuelve `falso' si se encuentra una opción inválida, la pila de directorios está vacía, se especifica una entrada de la pila de directorios no existente, o falla el cambio de directorio.

Printf formato [argumentos]

Escribe los argumentos formateados en la salida estándar bajo el control del formato. El formato es una cadena de caracteres que contiene tres tipos de objetos: caracteres normales, que simplemente se copian en la salida estándar, caracteres de secuencias de escape, que se convierten y copian en la salida estándar, y especificaciones de formato, cada una de las cuales causa la impresión del siguiente argumento sucesivo de una determinada manera. Además

de los formatos estándares normales de printf(1), %b hace que printf expanda las secuencias de escape de barras inversas en el correspondiente argumento, y %q causa que printf muestre en la salida el argumento correspondiente en un formato tal que pueda ser reutilizado como entrada para el shell.

El formato se reutiliza tanto como sea necesario para consumir todos los argumentos. Si el formato requiere más argumentos que los que se suministran, las especificaciones de formato extra se comportan como si se hubiera dado un valor cero o una cadena vacía, según lo apropiado. Pushd [-n] [dir]
pushd [-n] [+n] [-n]
Añade un directorio a la cima de la pila de directorios, o rota la pila, haciendo que el directorio de trabajo en curso sea la nueva cima de la pila. Sin argumentos, intercambia los dos directorios de más arriba y devuelve 0, a menos que la pila de directorios esté vacía.
Los argumentos, si se suministran, tienen los siguientes significados:
+n Rota la pila de forma que el n-simo directorio (contando desde la izquierda de la lista mostrada por dirs, empezando por cero) esté en la cima.
-n Rota la pila de forma que el n-simo directorio (contando desde la derecha de la lista mostrada por dirs, empezando por cero) esté en la cima.
-n Suprime el cambio normal de directorio cuando se añaden directorios a la pila, de forma que solamente se manipula la pila.
Dir Añade dir a la pila de directorios en la cima, haciéndolo el nuevo directorio de trabajo.

Si la orden pushd acaba con éxito, se realiza un dirs también. Si se emplea la primera forma, pushd devuelve 0 a menos que falle el cambio a dir. Con la segunda forma, pushd devuelve 0 a menos que la pila de directorios esté vacía, que se especifique un elemento de la pila de directorios inexistente, o que el cambio de directorio falle.

Pwd [-LP]
Muestra el nombre de fichero absoluto del directorio de trabajo en curso. El nombre de fichero mostrado no contiene enlaces simbólicos si se da la opción -P o está puesta la opción -o physical de la orden interna set. Si se usa la opción -L, los enlaces simbólicos se siguen. El estado de retorno es 0 a menos que ocurra un error mientras se lea el nombre del directorio actual o se dé una opción inválida.

Read [-er] [-a array] [-p prompt] [nombre ...]

Se lee una línea desde la entrada estándar, y la primera palabra se asigna al primer nombre, la segunda palabra al segundo nombre, y así sucesivamente, con las palabras que sobren y sus separadores intervinientes asignadas al último nombre. Si hay menos palabras leídas de la entrada estándar que nombres, a los sobrantes se les asignan valores vacíos. Se emplean los caracteres en IFS para dividir la línea en palabras. Las opciones, si se dan, tienen los siguientes significados:

-r Un par formado por una barra inclinada inversa y un salto de línea a continuación sí es tenido en cuenta, y la barra inversa se considera parte de la línea.

-p Muestra prompt, sin un salto de línea al final, antes de intentar leer nada de la entrada. El indicador se muestra solamente si la entrada viene de una terminal.

-a Las palabras se asignan secuencialmente a los elementos de la variable vector array, empezando por 0. Se anula la posible previa definición de array antes de que se asignen nuevos valores. Otros argumentos nombre no se tienen en consideración.

-e Si la entrada estándar viene de una terminal, se emplea readline (vea READLINE arriba) para obtener la línea.

Si no se suministran nombres, la línea leída se asigna a la variable REPLY. El código de retorno es cero, a menos que se llegue al fin de la entrada.

Readonly [-apf] [nombre ...]

Los nombres dados se marcan como de lectura exclusiva; los valores de estos nombres no pueden

cambiarse por posteriores asignaciones. Si se da la opción -f, se marcan así las funciones correspondientes a los nombres. La opción -a restringe las variables a vectores. Si no se da ningún argumento nombre, o si se da la opción -p, se muestra una lista de todos los nombres de lectura exclusiva. La opción -p hace que la salida se muestre en un formato que puede ser reutilizado como entrada. El estado de retorno es 0 a menos que se encuentre una opción inválida, uno de los nombres no sea un nombre válido de variable del shell, o se dé -f con un nombre que no es una función.

Return [n]

Hace que una función acabe y devuelva el valor especificado por n. Si n se omite, el estado devuelto es el de la última orden ejecutada en el cuerpo de la función. Si se emplea fuera de una función, pero durante la ejecución de un guión por la orden . (source), hace que el shell pare la ejecución de ese guión y devuelva o n o el estado de salida de la última orden ejecutada dentro del guión como el estado de salida del guión. Si se emplea fuera de una

función y no durante la ejecución de un guión por ., el estado de salida es 'falso'.

Set [--abefhkmnptuvxBCHP] [-o opción] [arg ...]

Sin opciones, se muestra el nombre y valor de cada variable del shell en un formato que puede ser reutilizado como entrada. La salida se clasifica según la localización en curso. Cuando se dan opciones, se establecen o quitan atributos del shell. Cualesquier argumentos que queden tras que se procesen las opciones se tratan como valores para los parámetros posicionales y se asignan, en orden, a $1, $2, ... $n. Las opciones, si se especifican, tienen los siguientes significados:

-a Automáticamente marca variables, que se modifiquen o creen, para exportación al entorno de las órdenes subsiguientes.

-b Informa del estado de los trabajos en segundo plano terminados inmediatamente, en vez de esperar a justo antes de mostrar el siguiente indicador primario. Esto sólo es efectivo cuando el control de trabajos está habilitado.

-e Sale inmediatamente si una orden simple (vea GRAMÁTICA DEL SHELL arriba) acaba con un

estado distinto de cero. El shell no acaba si la orden que falle es parte de un bucle until o while, parte de una sentencia if, parte de una lista && o ||, o si el valor devuelto por la orden se invierte mediante !.

-f Deshabilita la expansión de nombres de caminos.

-h Recuerda la localización de órdenes una vez que se buscan para la ¨ejecución. Esto está habilitado de forma predeterminada.

-k Todos los argumentos en forma de sentencias de asignación se ponen en el entorno para una orden, no solamente aquéllos que precedan al nombre de la orden.

-m Modo de monitor. Se habilita el control de trabajos. Esta opción está puesta de forma predeterminada para shells interactivos en sistemas que lo admitan (vea CONTROL DE TRABAJOS arriba). Los procesos en segundo plano se ejecutan en un grupo de proceso separado, y se imprime cuando se completan una línea conteniendo su estado de salida.

-n Lee órdenes pero no las ejecuta. Esto puede emplearse para comprobar si un guión del shell tiene errores de sintaxis. Para shells interactivos esta opción no tiene efecto.

-o nombre-opción

El nombre-opción puede ser uno de los siguientes:

allexport

Lo miso que -a.

braceexpand

Lo miso que -B.

emacs Emplea una interfaz de edición de líneas de órdenes al estilo de emacs. Esto está activo de forma predeterminada cuando el shell es interactivo, a menos que se haya arrancado con la opción --noediting.

Errexit Lo mismo que -e.

hashall Lo mismo que -h.

histexpand

Lo mismo que -H.

history Habilita la historia de órdenes, según se describió arriba bajo HISTORIA. Esta opción está habilitada por omisión en shells interactivos.

Ignoreeof

El efecto es como si se hubiera ejecutado la orden del shell IGNOREEOF=10 (vea Variables del shell arriba).

Keyword Lo mismo que -k.

Monitor Lo mismo que -m.

noclobber

Lo mismo que -C.

noexec Lo mismo que -n.

noglob Lo mismo que -f.

notify Lo mismo que -b.

nounset Lo mismo que -u.

onecmd Lo mismo que -t.

physical

Lo mismo que -P.

posix Cambia el comportamiento de bash donde la operación predeterminada difiera del estándar POSIX 1003.2, de forma que se siga éste.

Privileged

Lo mismo que -p.

verbose Lo mismo que -v.

vi Emplea una interfaz de edición de la línea de órdenes al estilo de vi.

Xtrace Lo mismo que -x.

Si se da -o sin ningún nombre-opción, se muestran los valores de las opciones activas. Si se da +o sin ningún nombre-opción, se muestra en la salida estándar una serie de órdenes set para recrear las opciones según están puestas o no actualmente.

-p Activa el modo privilegiado. En este modo, el fichero correspondiente a $ENV no es

procesado, las funciones del shell no se heredan desde el entorno,y la variable SHELLOPTS, si aparece en el entorno, no se tiene en consideración. Esta opción se activa automáticamente en el arranque si el identificador efectivo del usuario (o grupo) no es igual al identificador real del usuario (o grupo). Desactivar esta opción hace que los identificadores efectivos de usuario y grupo se pongan con los valores de los identificadores reales de usuario y grupo respectivamente.

-t Sale tras leer y ejecutar una sola orden.

-u Trata las variables no definidas como un error cuando realiza la expansión de parámetros. Si la expansión se intenta hacer sobre una variable no definida, el shell muestra un mensaje de error y, si no es interactivo, sale con un estado distinto de cero.

-v Repite en la salida las líneas de entrada del shell tras leerlas.

-x Tras expandir cada orden simple, muestra el valor expandido de PS4, seguido por la orden y sus argumentos expandidos.

-B El shell realiza la expansión de llaves (vea Expansión de llaves arriba). Esto está activado de forma predeterminada.

-C Si está activo, bash no sobreescribe un fichero existente con los operadores de redirección >, >&, ni <>. Esto puede cambiarse cuando se crean ficheros de salida mediante el empleo del operador >| en vez de >.

-H Permite la sustitución de historia mediante !. Esta opción está activada por omisión cuando el shell es interactivo.

-P Si está activada, el shell no sigue enlaces simbólicos cuando ejecuta órdenes como cd que cambian el directorio de trabajo. En su lugar emplea la estructura de directorio física. De forma predeterminada, bash sigue la cadena lógica de directorios cuando ejecuta órdenes que cambian el directorio de trabajo.

-- Si a esta opción no sigue ningún argumento más, entonces los parámetros posicionales se anulan. Si no, los parámetros posicionales se ponen con los valores dados por los args, incluso si alguno de ellos comienza con un -.

- **Señala el final de las opciones, haciendo que el resto de args se asignen a los parámetros posicionales. Las opciones -x y -v se desactivan. Si no hay más args, los parámetros posicionales permanecen sin cambios.**

Las opciones están desactivadas de forma predeterminada a menos que se diga otra cosa. Usar + en vez de − hace que estas opciones se desactiven. Las opciones también pueden darse como argumentos al llamar al shell. El conjunto de opciones activadas puede examinarse en $-. El estado de retorno es siempre `verdad' a menos que se encuentre una opción inválida.

Shift [n]

Los parámetros posicionales desde n+1 ... se renombran a $1 ... Los parámetros representados por los números desde $# hasta $#-n+1 se anulan. N debe ser un número no negativo menor o

igual a $#. Si n es 0, no se cambia ningún parámetro. Si n no se da, se supone 1. Si n es mayor que $#, los parámetros posicionales no se cambian. El estado de retorno es mayor que cero si n es mayor que $# o menor que cero; en otro caso es cero.

Shopt [-pqsu] [-o] [nombreopc ...]

Cambia los valores de variables que controlan un comportamiento determinado opcional del shell. Sin opciones, o con la opción -p, se muestra una lista de todas las opciones disponibles, con una indicación en cada una de si está activa o no. La opción -p hace que la salida se muestre de una forma tal que pueda reutilizarse como entrada. Otras opciones tienen los significados siguientes:

-s Activa (set) cada nombreopc.

-u Desactiva (unset) cada nombreopc.

-q Suprime la salida normal (modo silencioso); el estado de salida indica si el nombreopc está activado o no. Si se dan varios argumentos nombreopc con -q, el estado de salida es cero si todos los nombreopcs están activados; distinto de cero en otro caso.

-o Restringe los valores de nombreopc a aquéllos definidos para la opción -o de la orden interna set.

Si se emplean -s o -u sin argumentos nombreopc, la lista mostrada se limita a aquellas opciones que están activadas o desactivadas, respectivamente. A menos que se diga otra cosa, las opciones de shopt están inactivas (unset) de forma predeterminada.

El estado de retorno cuando se listan opciones es cero si todos los nombreopcs están activos, distinto de cero en otro caso. Cuando se activan o desactivan opciones, el estado de salida es cero a menos que un nombreopc no sea una opción del shell válida.

La lista de las opciones de shopt es:

cdable_vars

Si está activa, un argumento de la orden interna cd que no sea un directorio, se supone el nombre de una variable cuyo valor es el directorio al que hay que cambiarse.

Cdspell Si está activa, se corregirán pequeños errores que hubiera en la escritura de un componente directorio de una orden cd. Los errores que se buscan se refieren a caracteres transpuestos, un carácter que falte, y un carácter que sobre. Si se encuentra una corrección, se muestra el nombre de fichero corregido y la orden procede. Esta opción sólo se emplea en shells interactivos.

Checkhash

Si está activa, bash comprueba que una orden encontrada en la tabla de dispersión existe antes de intentar ejecutarlo. Si una orden en la tabla de dispersión ya no existe, se realiza una búsqueda normal en el PATH.

Checkwinsize

Si está activa, bash comprueba el tamaño de ventana tras cada orden y, si es necesario, actualiza los valores de LINES y COLUMNS.

Cmdhist Si está activa, bash intenta guardar todas las líneas de una orden de varias líneas en la misma entrada de historia. Esto permite una re-edición fácil de dichas órdenes.

Dotglob Si está activa, bash incluye los nombres de ficheros que comiencen con un `.' en los resultados de la expansión de nombres de caminos.

Execfail

Si está activa, un shell no interactivo no acabará si no puede ejecutar el fichero especificado como un argumento de la orden interna exec. Un shell interactivo no acaba si exec falla.

expand_aliases

Si está activa, los alias se expanden como se describió arriba bajo ALIAS. Esta opción está habilitada de forma predeterminada para shells interactivos.

Extglob Si está activa, se habilitan las características de concordancia de patrones extendidas descritas más arriba en Expansión de nombres de camino.

Histappend

Si está activa, la lista de historia se añade al fichero nombrado según el valor de la variable HISTFILE cuando el shell acaba, en vez de sobreescribir el fichero.

Histreedit

Si está activa, y readline se está utilizando, se le da al usuario la oportunidad de re-editar la sustitución de historia fallida.

Histverify

Si está activa, y readline se está utilizando, los resultados de la sustitución de historia no se pasan inmediatamente al analizador del shell. En vez de eso, la línea resultante se carga en el búfer de edición de readline, permitiendo así una modifi-

cación posterior.

Hostcomplete

Si está activa, y readline se está utilizando, bash intentará terminar de escribir un nombre de computador anfitrión cuando se esté completando una palabra que contenga una @ (vea Terminación bajo READLINE arriba). Esto está activado de forma predeterminada.

Huponexit

Si está activa, bash enviará una señal SIGHUP a todos los trabajos cuando un shell de entrada interactivo finalice.

interactive_comments

Si está activa, permite a una palabra que empiece por # hacer que esa palabra y todos los caracteres restantes de esa línea no sean tenidos en cuenta en un shell interactivo (vea COMENTARIOS arriba). Esta opción está habilitada por omisión.

Lithist Si está activa, y la opción cmdhist también lo está, las órdenes multi-línea se guardan en la historia con saltos de línea empotrados en vez de emplear como separador el punto y coma, donde sea posible.

Mailwarn

Si está activa, y a un fichero donde bash está buscando correo nuevo se ha accedido desde la última vez que se buscó, se muestra el mensaje ˜The mail in mailfile has been read", o su equivalente en el idioma local, que en español sería ˜El correo en buzón ha sido leído".

Nocaseglob

Si está activa, bash busca concordancias de nombres de ficheros sin importar mayúsculas o minúsculas cuando realice la expansión de nombres de caminos (vea Expansión de nombres de caminos arriba).

Nullglob

Si está activa, bash permite que los patrones que no concuerden con ningún fichero (vea Expansión de nombres de caminos arriba) se expandan a una cadena vacía, en vez de a sí mismos.

Promptvars

Si está activa, las cadenas de caracteres que sirven de indicadores están sujetas a expansión de variable y parámetro tras ser expandidas como se describió en INDICADORES arriba. Esta opción está activa de forma predeterminada.

shift_verbose

Si está activa, la orden interna shift muestra un mensaje de error cuando el número

de shift excede al de parámetros posicionales.

Sourcepath

Si está activa, la orden interna source (.) emplea el valor de PATH para buscar el directorio que contenga al fichero suministrado como argumento. Esta opción está activa por omisión.

Suspend [-f]

Suspende la ejecución de este shell hasta que reciba una señal SIGCONT. La opción -f dice que no hay que protestar si éste es un shell de entrada; simplemente suspenderlo de todas formas. El estado de retorno es 0 a menos que el shell sea de entrada y la opción -f no se haya dado, o si el control de trabajos no está habilitado.

Test expr

[expr]

Devuelve un estado de 0 ó 1 dependiendo de la evaluación de la expresión condicional expr. Cada operador y operando debe ser un argumento separado. Las expresiones se componen de las

primarias descritas más arriba bajo EXPRESIONES CONDICIONALES.

Las expresiones se pueden combinar mediante los operadores siguientes, listados en orden decreciente de precedencia.

! expr Verdad si expr es falsa.

(expr)

Devuelve el valor de expr. Esto puede emplearse para cambiar la precedencia normal de los operadores.

Expr1 -a expr2

Verdad si tanto expr1 como expr2 son verdad.

Expr1 -o expr2

Verdad si uno al menos de expr1 o expr2 es verdad.

Test y [evalúan expresiones condicionales según un conjunto de reglas basadas en el número de argumentos.

0 argumentos

La expresión es falsa.

1 argumento

La expresión es verdad si y sólo si el argumento no está vacío.

2 argumentos

Si el primer argumento es !, la expresión es verdad si y sólo si el segundo argumento es nulo. Si el primer argumento es uno de los operadores condicionales monarios listados arriba en EXPRESIONES CONDICIONALES, la expresión es verdad si el test monario es verdad. Si el primer argumento no es un operador condicional monario válido, la expresión es falsa.

3 argumentos

Si el segundo argumento es uno de los operadores condicionales binarios listados arriba en EXPRESIONES CONDICIONALES, el resultado de la expresión es el resultado del test binario empleando el primer y tercer argumentos como operandos. Si el primer argumento es !, el valor es la negación del test de dos argumentos empleando el segundo y tercer argumentos. Si el primer argumento es exactamente (y el tercer argumento es exactamente), el resultado es el test de un argumento del segundo argumento. De otro modo, la expresión es falsa. Los operadores -a y -o se consideran como operadores binarios en este caso.

4 argumentos

Si el primer argumento es !, el resultado es la negación de la expresión de tres argumentos compuesta por los argumentos restantes. De otra forma, la expresión se analiza y evalúa de acuerdo con la precedencia utilizando las reglas listadas arriba.

5 ó más argumentos

La expresión se analiza y evalúa de acuerdo con la precedencia usando las reglas mencionadas arriba.

Times Muestra los tiempos acumulados de usuario y sistema para el shell y para procesos ejecutados
desde él. El estado de retorno es 0.

trap [-lp] [arg] [nombre-señal ...]

La orden arg va a leerse y ejecutarse cuando el shell reciba la(s) señal(es) especificada(s) por nombre-señal. Si arg está ausente o es -, todas las señales especificadas se reestablecen a sus valores originales (los que tenían cuando se entró en el shell). Si arg es la cadena vacía, la señal especificada por cada nombre-señal no se tiene en cuenta por parte del shell y de las órdenes que se llamen desde él. Si arg es -p entonces se muestran las órdenes de las trampas asociadas con cada nombre-señal. Si no se pasan argumentos o si sólo se da -p, trap muestra la lista de órdenes asociadas con cada número de señal. Cada nombre-señal es o bien un nombre de señal de los definidos en <signal.h>, o un número de señal. Si un nombre-señal es EXIT (0), la orden arg se ejecuta cuando se sale del shell. Si un nombre-

señal es DEBUG, la orden arg se ejecuta tras cada orden simple (vea GRAMÁTICA DEL SHELL

arriba). La opción -l hace que el shell muestre una lista de nombres de señal y sus números correspondientes. Las señales que no se tienen en cuenta ya cuando se entra en el shell no pueden ser atrapadas ni restablecidas. Las señales atrapadas se reestablecen a sus valores originales en un proceso hijo cuando se crea. El estado de retorno es 'falso' si cualquier nombre-señal no es válido; de otro modo, trap devuelve 'verdad'.

Type [-atp] nombre [nombre ...]

Sin opciones, indica cómo será interpretado cada nombre si se usa como un nombre de orden. Si se emplea la opción -t, type muestra una de las siguientes cadenas de caracteres: alias, keyword, function, builtin, o file si nombre es respectivamente un alias, una palabra reservada del shell, una función, una orden interna incorporada en el shell, o un fichero ejecutable de disco. Si el nombre no se encuentra, no se muestra nada, y se devuelve un estado de salida de 'falso'. Si se emplea la opción -p, type devuelve o bien el nombre del fichero de disco que se ejecutaría si se especificara nombre como un nombre de orden, o bien nada si type -t nombre no devolviera file. Si una orden está en la tabla de dispersión, -p muestra el valor de dicha tabla, no necesariamente el fichero que aparezca primero en PATH. Si se emplea la opción -a, type muestra todos los sitios que contengan un ejecutable llamado nombre. Esto incluye alias y funciones, si y sólo si la opción -p no se ha usado también. La tabla de dispersión de las órdenes no se consulta cuando se emplea -a. type devuelve 'verdad' si cualquiera de los argumentos se encuentra, 'falso' si no se encuentra ninguno.

Ulimit [-Shacdflmnpstuv [límite]]

Proporciona control sobre los recursos disponibles para el shell y para los procesos arrancados por él, en sistemas que permitan tal control (Linux por ejemplo, y por supuesto). El valor de límite puede ser un número en la unidad especificada para el recurso, o el valor unlimited, o sea, ilimitado. Las opciones -H y -S especifican que el límite para el recurso dado va a ser duro o blando. Un límite duro es aquél que no puede ser aumentado una vez puesto; un límite blando puede incrementarse hasta el valor dado por el límite duro. Si no se especifican ni -H ni -S, se establecen ambos límites. Si límite se omite, se muestran los valores del límite blando del recurso, a menos que se dé la opción -H. Cuando se especifica más de un recurso, se imprime el nombre del límite y la unidad antes del valor. Otras opciones se interpretan como sigue:

-a Se informa de todos los límites actuales

-c El tamaño máximo de ficheros de volcados de memoria (core)

-d El tamaño máximo del segmento de datos de un proceso

-f El tamaño máximo de ficheros creados por el shell

-l El tamaño máximo que puede ser bloqueado en memoria

-m El tamaño del máximo conjunto residente (memoria)

-n EL número máximo de descriptores de ficheros abiertos (la mayoría de sistemas no permiten establecer este valor)

-p El tamaño de una tubería en bloques de 512 B (esto puede no estar establecido)

-s El tamaño máximo de pila

-t La máxima cantidad de tiempo de CPU en segundos

-u El número máximo de procesos disponibles para un solo usuario

-v La máxima cantidad de memoria virtual disponible para el shell

Si se da límite, es el nuevo valor del recurso especificado (la opción -a es sólo para mostrar los valores). Si no se da ninguna opción, entonces se supone -f. Los valores están en incrementos de 1024 B, excepto para -t, que está en segundos, -p, que está en unidades de bloques de 512 B, y -n y -u, que son valores adimensionales. El estado de retorno es 0 a menos que se encuentre una opción inválida, se dé como límite un argumento no numérico distinto de unlimited, o bien ocurra un error mientras se establece un nuevo límite.

Umask [-p] [-S] [modo]

La máscara de creación de ficheros del usuario se establece a modo. Si modo empieza con un dígito, se interpreta como un número octal; si no, se interpreta como un modo simbólico similar al aceptado por chmod(1). Si modo se omite, o si se da la opción -S, se muestra el valor actual de la máscara. La opción -S hace que la máscara se imprima en forma simbólica; la salida predeterminada es como un número octal. Si se da la opción -p y modo se omite, la salida es de tal forma que puede reutilizarse como entrada. El estado de retorno es 0 si el modo se cambió exitosamente o si no se dio el argumento modo, y ̃falso' en otra circunstancia.

Unalias [-a] [nombre ...]

Quita nombres de la lista de alias definidos. Si se da la opción -a, se quitan todas las definiciones de alias. El estado de salida es ̃verdad' a menos que un nombre dado no sea un alias definido.

Unset [-fv] [nombre ...]

Para cada nombre, borra la variable o función correspondiente. Si no se dan opciones, o se da la opción -v, cada nombre se refiere a una variable del shell. Las variables de lectura exclusiva no pueden borrarse. Si se especifica -f, cada nombre se refiere a una función del shell, y se borra la definición de la función. Cada variable o función se quita del entorno pasado a órdenes subsiguientes. Si se quita cualquiera de RANDOM, SECONDS, LINENO, HISTCMD,
o DIRSTACK, pierde su propiedad especial, incluso aunque más adelante se vuelva a definir. El estado de salida es 'verdad' a menos que nombre no exista o sea de lectura exclusiva.

Wait [n]
Espera al proceso especificado y devuelve su estado de terminación. N puede ser un identificador de proceso (PID) o una especificación de trabajo; si se da una espec. De trabajo, se espera a todos los procesos en la tubería de ese trabajo. Si n no se da, se espera a todos los procesos hijos activos actualmente, y el estado de retorno es cero. Si n especifica un proceso o trabajo no existente, el estado de retorno es 127. De otro modo, el estado de retorno es el estado de salida del último proceso o trabajo al que se esperó.

SHELL RESTRINGIDA
Si bash se arranca con el nombre rbash, o se da la opción -r en la llamada, el shell se convierte en
restringido. Un shell restringido se emplea para establecer un ambiente más controlado que el que
el shell estándar proporciona. Se comporta de forma idéntica a bash con la excepción de que lo
siguiente no está permitido o no se realiza:

- **cambiar de directorio con cd**
- **establecer o anular los valores de SHELL o de PATH**
- **especificar nombres de órdenes que contengan /**
- **especificar un nombre de fichero que contenga al menos una / como un argumento de la orden**
interna . (source)
- **importar definiciones de funciones desde el entorno del shell en el arranque**
- **analizar el valor de SHELLOPTS desde el entorno del shell en el arranque**

o redirigir la salida usando los operadores de redirección >, >|, <>, >&, &>, y >>

o utilizar la orden interna exec para reemplazar el shell por otro programa

o añadir o eliminar órdenes incorporadas con las opciones -f o -d de la orden interna enable.

O dar la opción -p a la orden interna command.

O desactivar el modo restringido con set +r o set +o restricted.

Estas restricciones entran en vigor después de que se lean los ficheros de arranque que hubiera.

Cuando se ejecuta una orden que resulta ser un guión del shell (vea EJECUCIÓN DE ÓRDENES arriba),
rbash desactiva todas las restricciones en el shell lanzado para ejecutar el guión.

VÉASE TAMBIÉN

Bash Features (Características de Bash), Brian Fox & Chet Ramey

The Gnu Readline Library (La Biblioteca Readline de GNU), Brian Fox & Chet Ramey

The Gnu History Library (La Biblioteca de Historia de GNU), Brian Fox & Chet Ramey

Portable Operating System Interface (POSIX) Part 2: Shell and Utilities (Interfaz para Sistemas

Operativos Transportables (POSIX) Parte 2: Shell y Utilidades), IEEE

zsh(1), ash(1), sh(1), ksh(1), csh(1), tcsh(1).

Emacs(1), vi(1)

readline(3)

FICHEROS

/bin/bash

La imagen ejecutable de bash

/etc/profile

El fichero de inicio general, leído en shells de entrada

~/.bash_profile

El fichero de inicio personal, leído para shells de entrada

~/.bashrc

El fichero individual de arranque para shells interactivos

~/.inputrc

El fichero de inicio individual de readline

AUTORES

Brian Fox, Free Software Foundation

bfox@gnu.ai.MIT.Edu

Chet Ramey, Case Western Reserve University

chet@ins.CWRU.Edu