3. Programación avanzada

1. **Evaluación de expresiones numéricas**
2. **Uso de ficheros inscritos *(here document)***
3. **Manejo de interrupciones dentro de un guión**
4. **Gestión de archivos E/S mediante exec**

3.1. Evaluación de expresiones numéricas

- Las variables del shell son cadenas de texto por defecto
- Incluso los valores numéricos se almacenan así
- Para realizar operaciones aritméticas y lógicas es necesario traducirlas a enteros
- **expr args**

  Evalua los argumentos y envía el resultado a la salida estándar

  Operadores:

  =, !=,\>, \>=, \<, \<= (relacionales)

  +, -, *, /, % (aritméticos)

  \| : proporciona la primera expresión si no es cero ni nula; proporciona la segunda en caso contrario (OR lógico, asociado a expresiones)

  \& : proporciona la primera expresión si ambas son no nulas; proporciona 0 en caso contrario (AND lógico, asociado a expresiones)

3.2. Uso de ficheros inscritos *(here document)*

Permite redirigir la entrada estándar para una orden o mandato dentro de un guión y proporcionársela de un fichero inscrito o interno:

mandato <<[-] *delimitador*

...datos...

delimitador

- Se realizan las substituciones de variables y mandatos dentro de *delimitador ... delimitador* antes de enviar los datos.
- Los *delimitadores* deben ir en líneas separadas
- Se pueden usar "" y ' ' para usar valores especiales de *delimitador* y no confundirlas con el texto interno
- **<<-** permite obviar los tabuladores en los *...datos...*
- Si se redireccionan además la salida estándar y la salida de errores estándar, debe hacerse antes del fichero inscrito.
- Su uso es principalmente mostrar menús.
- También agiliza la lectura de datos, frente a usar un fichero.

3.3. Manejo de interrupciones dentro de un guión

- Las señales/interrupciones que recibe un proceso pueden tener varios orígenes:
  - interrupción hardware (provocada por un dispositivo; habitualmente será el teclado: ^C, ^Z, ^D)
  - enviada desde un shell mediante la orden *kill*
- La respuesta de un proceso ante una señal o interrupción puede ser de 3 tipos:
  - ignorarla
  - responder con una acción por defecto (generalmente la de terminar la orden o mandato en primer plano)
  - llevar a cabo una acción prevista por el programador

trap ['lista_ordenes'] [lista_señales]

intercepta las señales de **lista_señales** y permite realizar una acción concreta (la **lista_ordenes** entre **'**')

Ejemplo:

- trap : es redundante
- trap ' ' 1 2 3 : ignora las señales de la lista

Señales habituales

SIGHUP	(1) Hangup - por defecto provoca: exit.
SIGINT	(2) Interrupt - por defecto provoca: exit (por teclado ^C)
SIGQUIT	(3) Quit - por defecto provoca: core.
SIGABRT	(6) Abort - por defecto provoca: core.
SIGKILL	(9) Killed - por defecto provoca: exit.
SIGSEGV	(11) Segmentation fault - provoca exit.
SIGTERM	(15) Terminated – cuando se utiliza kill sin señal, por defecto provoca: exit.
SIGCHLD	(20) Fin de ejecución del hijo, notifica que terminó uno de sus hijos.
SIGSTP	(23) Stopped – (por teclado ^Z)por defecto provoca: stop.

3.4. Gestión de archivos E/S mediante exec

exec es un mandato que permite cargar en la memoria la imagen de un nuevo proceso, que substituye a la actual

$> exec mandato

- A partir de ese momento no se puede recuperar la acción, que desaparece
- Cuando el nuevo mandato termina, se devuelve al control al proceso padre (getty si es el shell de conexión)

- Combinado con los operadores de redirección de E/S permite a los guiones leer o escribir en otros archivos que nos sean las E/S estándar

- Se pueden manejar simultáneamente hasta 10 descriptores de archivo en cada guión (descontando: 0 para E estándar, 1 para S estándar y 2 para errores)

exec < *entrada*

cada línea del archivo *entrada* se trata como un mandato a ejecutar por el shell → **hemos redirigido la entrada estándar**

Para devolver la E estándar al teclado es necesario:

exec < /dev/tty

exec > *salida*

redirigimos la salida estándar al archivo *salida*

Para devolver la S estándar a la pantalla es necesario:

exec > /dev/tty

exec > salida 2> salida_errores

redirigimos la salida estándar al archivo salida, y al mismo tiempo la salida de errores estándar a salida_errores

Para devolver la S estándar a la pantalla es necesario:

exec > /dev/tty

exec 2> /dev/tty

ambas nos permiten recuperar las redirecciones habituales

Sintaxis	**Significado**
exec < archivo	'archivo' se abre para leer, y se asocia como entrada estándar
exec > archivo	'archivo' se abre para escribir, y se le asocia la salida estándar
exec >> archivo	'archivo' se abre para escribir en modo añadir, y se le asocia la salida estándar
exec n< archivo	'archivo' se abre para leer, y se le asigna el descriptor 'n'
exec n> archivo	'archivo' se abre para escribir, y se le asigna el descriptor 'n'
exec n<< *marcador* ... *marcador*	abre un archivo inscrito para leer, con descriptor 'n'
exec n>> archivo	'archivo' se abre para escribir en modo añadir, se le asigna el descriptor 'n'
exec n>&m	Duplica 'm' en 'n'; todo lo que llegue al archivo con descriptor 'n' se copiará en el de descriptor 'm'
exec <&-	Cierra la entrada estándar
exec >&-	Cierra la salida estándar
exec n<&-	Cierra el archivo con descriptor 'n' asociado a la entrada estándar
exec n>&-	Cierra el archivo con descriptor 'n' asociado a la salida estándar