Programando con el intérprete de órdenes (shell) de UNIX

Fundamentos de Informática II.
Práctica 2

21 de abril de 2006

Resumen

Con el fin de facilitar la lectura y comprensión de los guiones de prácticas seguiremos el siguiente convenio que se presenta a continuación.

- Las palabras reservadas de UNIX se presentarán en **negrita** y deben escribirse tal y como se presentan.
- La escritura en letra *cursiva* está reservada para variables y constantes. Es decir, las palabras escritas de esta manera deben sustituirse por las variables o constantes que se consideren convenientes.
- Las palabras escritas en letras MAYÚSCULAS están reservadas para las variable de entorno que el intérprete predefine por defecto u omisión (*default* en inglés). Estas variables se analizarán brevemente en este documento.
- Siempre que aparezca el símbolo **$** al principio de la línea se trata del **indicador** (*prompt* en inglés) de la línea de órdenes del intérprete y no debe escribirse como parte de las órdenes. De aquí en delante se tomarán las siguientes convenciones para la descripción de las órdenes del intérprete de UNIX.

1. Ejecución condicional usando la sentencia *if*

La ejecución condicional de un grupo de órdenes es una herramienta importante de programación que también está disponible en el intérprete de órdenes. La sintaxis general de esta sentencia es

if *condición*
then
  Grupo de órdenes
else
  Grupo de órdenes
fi

donde el primer grupo de órdenes se ejecuta únicamente cuando se cumple la condición especificada, mientras que el segundo grupo de órdenes se ejecuta en caso contrario. La orden **else** y el grupo de órdenes que le sigue pueden omitirse de la estructura de la sentencia.

Si bien no tiene mucho sentido, también es posible ejecutar esta sentencia desde la línea de órdenes de UNIX, aparte de los guiones; esta característica se utilizará más adelante para ilustrar algunos ejemplos. Para usar esta característica de la sentencia considérese lo siguiente: una vez tecleada la orden **if** con la condición, aparece un indicador secundario [>] de petición de orden. Se siguen escribiendo las líneas que componen la sentencia y en el momento que se introduzca la palabra clave de fin de sentencia **fi**, el intérprete la ejecutará y al finalizar devolverá el control al proceso desde el que se ejecutó.

Se puede utilizar la orden **set** para dar previamente argumentos al guión ejecutado de esta manera; la sintaxis es en este caso **set** *arg1 arg2 arg3 ...*

1.1. Uso de *test* para especificar la condición

Para especificar la condición se puede utilizar la orden **test**. Básicamente, los argumentos de **test** forman una expresión; si la expresión es verdadera **test** devuelve el valor 0, mientras que si la expresión es falsa **test** devuelve un valor distinto de cero. Hay tres clases principales de comprobaciones

- **test** sobre valores numéricos
- **test** sobre ficheros
- **test** sobre cadenas de caracteres

y las comprobaciones se pueden realizar usando variables. Si se van a utilizar variables en la comprobación, se recomienda encerrarlas entre comillas dobles para evitar que el guión produzca errores al realizar comprobaciones sobre variables no definidas. Si bien no es siempre necesario, es preferible utilizar siempre comillas para evitar errores.

1.1.1. *test* sobre valores numéricos

La comprobación sobre valores numéricos examina la relación que existe entre ellos y tiene como forma general **test** *N primitiva M* donde N y M son los valores numéricos a comprobar.
Las primitivas que se pueden utilizar son:

-eq Comprueba si N es igual a M
-ne Comprueba si N es diferente de M
-gt Comprueba si N es mayor que M
-lt Comprueba si N es menor M
-ge Comprueba si N es mayor o igual que M
-le Comprueba si N es menor o igual que M

Ejemplo: Ejecutar las siguientes órdenes de la sentencia **if** desde la línea de órdenes del sistema:

```
usuarios=`who | wc -l`
if test "$usuarios" -lt 8
then
echo Hay menos de 8 usuarios conectados
fi
```

NOTA: El intérprete solo trata números enteros. Los números 1.0 y 1.3, por ejemplo, son iguales.

1.1.2. *test* sobre ficheros

También se puede comprobar mediante la sentencia **if** la existencia o ausencia de ficheros, así como sus propiedades. La forma general de la expresión es **test** *primitiva* nombre_del_fichero
Las primitivas más comunes son en este tipo de comprobaciones son:

-s Verifica que el fichero existe y no está vacío.
-f Verifica que el fichero es ordinario (no es un directorio).
-d Verifica que el fichero es un directorio.
-w Verifica que el fichero puede ser escrito.
-r Verifica que el fichero puede ser leído.

EJEMPLO: Supóngase que el primer argumento introducido al ejecutar un guión ha de ser un directorio. Si se desea comprobar que ese primer argumento es un directorio válido, se puede utilizar la sentencia **if** de esta manera:

```
if test -d "$1"
then
echo El directorio es válido
ls -l $1
fi
```

Para negar el sentido de las verificaciones anteriores se puede emplear el signo de admiración, que es el operador unario de negación. La sintaxis en este caso es **test** ! -d nombre_del_fichero

EJEMPLO: Para el caso anterior, se puede verificar la validez del directorio de la siguiente manera:

if test ! -d "$1"
then
echo El directorio no es válido
fi

1.1.3. *test* sobre cadenas de caracteres

La orden **test** se puede utilizar también para comprobar y comparar cadenas de caracteres. La tabla siguiente resume los operadores y la sintaxis a utilizar en estos casos.

Operador	Sintaxis	Significado
=	**test** *cadena1* = *cadena2*	¿Es la cadena1 igual a la cadena2?
!=	**test** *cadena1* != *cadena2*	¿Es la cadena1 diferente de la cadena2?
-n	**test** -n *cadena*	¿Contiene la cadena caracteres (no es nula)?
-z	**test** -z *cadena*	¿La cadena está vacía (es nula)?

EJEMPLO: Se desea que un determinado guión solo se ejecute cuando el primer argumento tiene un valor igual a "permiso"(sin comillas). Esto se puede hacer de las siguiente manera, comprobando inicialmente la existencia del primer argumento:

if test "$1" = ""
then
echo Hay que introducir un argumento para ejecutar el guión
exit
fi
if test "$1" != permiso
echo La palabra clave no es correcta. Ejecución abortada.
exit
fi

EJEMPLO: La primera comprobación del ejemplo anterior se podría haber realizado también de la siguiente manera:

if test -z "$1"
then
echo Hay que introducir un argumento para ejecutar el guión
exit
fi
if test "$1" != permiso
echo La palabra clave no es correcta. Ejecución abortada.
exit
fi

1.2. Usando los operadores -a y -o para combinar expresiones

Los operadores -a y -o permiten combinar expresiones de comprobación en una sola orden **test**. El operador -a representa la operación "*Y*" lógica mientras que que -o representa la operación "*O*" lógica.

EJEMPLO: Las dos comprobaciones de los ejemplos anteriores se pueden combinar en una sola comprobación de la siguiente manera:

```
if test -z "$1" -o "$1" != permiso
then
echo Argumento nulo o no válido
exit
fi
```

PRÁCTICA: Crear los siguientes guiones:

- Un programa llamado `inicializa.sh` que verifique si existe el directorio `$HOME/guiones`; en caso de que exista poner un mensaje en pantalla. Si no existe el directorio deberá crearlo y emitir en pantalla un mensaje adecuado.
- Un programa llamado `contar.sh` que verifique que recibe solo un argumento. En caso de que no sea así debe emitir un mensaje de error.
- Un programa llamado `igual.sh` que verifique que solo recibe un argumento. Deberá comprobar que ese argumento único es el signo " = " y emitir por pantalla un mensaje adecuado. En caso contrario deberá emitir un mensaje de error.