Programando con el intérprete de órdenes (shell) de UNIX

Fundamentos de Informática II.
Práctica 4

9 de mayo de 2005

Resumen

Cuando programamos es lícito elegir cualquier sentencia de control; sin embargo, es conveniente elegir la más adecuada en cada caso. En esta práctica veremos las sentencias de control **case**, **while** y **until**. Su sintaxis y un ejemplo para cada una. Además se presentan algunos consejos y ejercicios para ejercitar dichos conceptos.

1. Secuencia de selección múltiple *case*

La sintaxis de esta sentencia es:

case *expresión* **in**
Patrón1) bloque de instrucciones;;
Patrón2) bloque de instrucciones;;
Patrón3) bloque de instrucciones;;
⋮
...) bloque de instrucciones;;
esac

donde:
esac indica el fin del cuerpo del bucle

Si "expresión"coincide con alguno de los patrones reseñados se ejecuta el bloque de instrucciones que figura a continuación.
El asterisco) representa el patrón por defecto. Su bloque de instrucciones asociado se ejecuta en el caso de que "expresión"no coincida con ninguno de los patrones precedentes. Expresión puede ser una constante o una variable.
Los patrones pueden contener metacaracteres (, ?, []) y valores alternativos separados por |, que actúa como un operador lógico OR.
La última instrucción de cada bloque debe ir seguida de doble punto y coma (;;)

EJEMPLO:

```
case carnet in
    E)          echo Remolques;;
    D)          echo Autobuses;;
    C1 |C2)     echo camiones;;
    B[12)       echo Automoviles;;
    A*)         echo Motocicletas;;
    *)          echo categoría no reconocida;;
esac
```

Según el valor de la variable "carnet"se mostrará el mensaje correspondiente.

Nótese que el tercer patrón admite indistintamente C1 o C2, el cuarto admite B seguido de 1 ó 2, y el quinto admite cualquier valor que empiece por A.

2. Bucle o grupo de instrucciones *while*

Bucle o grupo de instrucciones que se ejecutan reiteradamente mientras se cumpla una condición. La sintaxis de esta sentencia es:

while *condición*
do
bloque de instrucciones
done

EJEMPLO:

```
while test $n -lt 9
do
    read coste
    acum= `expr $acum + $coste`
    n=`expr $n + 1`
done
```

En la variable "acum"se van acumulando los datos introducidos a través de la variable "coste"y se incrementa el contador `n`. El bucle se repite mientras el valor de `n` sea menor de 9.

3. Bucle o grupo de instrucciones *until*

Bucle o grupo de instrucciones que se ejecutan reiteradamente hasta que se cumpla una condición. La sintaxis de esta sentencia es:

until *condición*
do
bloque de instrucciones
done

EJEMPLO:

```
until test $n -eq 10
do
    read coste
    acum= `expr $acum + $coste`
    n=`expr $n + 1`
done
```

El bucle se repite hasta que `n` alcanza el valor 10.

4. Eligiendo sentencias de control

A lo largo de este curso hemos visto las sentencias de control típicas de la programación estructurada (**if, for, case, while, until**). A continuación se presentan algunas indicaciones para un uso eficiente de las mismas.

Si se trata de dos alternativas, la construcción **if** es la adecuada. En el caso de más de dos opciones debemos utilizar una secuencia **case**.

Los **if** anidados deben evitarse en lo posible. Se usarán en el caso de sucesivas opciones binarias realmente anidadas, cuando cada elección dependa de los resultados de opciones anteriores.

En el caso de bucles o repeticiones, cuando los valores son conocidos antes del inicio de dicho bucle y su valor no va a ser modificado en éste, se debe utilizar una construcción **for**. Es una construcción simple pero rígida. Los contadores nunca deben modificarse dentro del bucle salvo en la sentencia de incremento.

La construcción **while** es adecuada a los casos en los que hay que evaluar "cuándo actuar". Puede no verificarse de entrada y el bucle no realizaría ningún ciclo. Las variables implicadas en la condición tampoco deben ser manipuladas en el interior del bucle.

La construcción **until** es adecuada a los casos en los que hay que evaluar "cuándo parar". Las variables implicadas en la condición tampoco deben ser manipuladas en el interior del bucle.

No es preciso que la "condición"que se vaya a utilizar en una estructura **if**, **while** o **until** sea un **test**. Puede ser cualquier orden o acción ejecutada por el shell. Lo que se considera es el código de retorno $? generado.

Por ejemplo, **if ls** hará ejecutar el bloque de instrucciones que sigue a **then** si la ejecución de la orden **ls** tiene éxito.

Problemas

1. Crea un programa PARIDAD.SH que solicite un número y diga si es par o impar.
2. Crea un programa DIVISIBLE.SH que compruebe si dos números son o no divisibles. (Recuerda que el resto de la división entera entre `a` y `b` se calcula `` `expr $a% $b` ``)
3. Construye un programa COMPROB.SH que solicite un nombre y compruebe si figura en un archivo llamado LISTA.TXT.
4. Construye un programa TABLAS.SH que muestre por pantalla las tablas de multiplicar de los números 1 a 10. Muestra cada tabla tras limpiar la pantalla y con un intervalo de 3 segundos entre tabla y tabla. Utiliza *sólo* sentencias **for**.
5. Crea un nuevo programa TABLA2.SH que muestre las tablas de multiplicar del 1 al 10 en una sola pantalla. Utiliza *sólo* sentencias **for**.
6. Construye TABLAS.SH utilizando dos construcciones **until** anidadas.
7. Construye TABLAS.SH utilizando construcciones **while**.