2

Paso de argumentos

Introducción

Cuando se desarrolla un programa bajo un sistema operativo basado en UNIX[®] como es el caso de LINUX, se debe tener en consideración que se estará en un ambiente multiproceso. Esto significa que, en un momento dado, se estará ejecutando nuestro programa de manera concurrente (o paralela en el caso de que el sistema de cómputo cuente con más de un procesador) con muchos otros programas. Inclusive podrían estarse ejecutando varias instancias del nuestro.

De modo que es importante para el programador que sus programas no interfieran con otros en el momento de estar haciendo uso simultáneo de un recurso. Dicho recurso puede ser memoria, algún archivo en disco, o inclusive el teclado.

Ejemplo práctico

Introduzca en un archivo con ayuda de un editor como vi, pico, o alguno otro de su preferencia, el programa 2.1 y guárdelo como programa 2-1.c

Programa 2.1

Imprime argumentos y opciones

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int tokens;

    for(tokens = 0; tokens < argc; tokens++) {
        if(argv[tokens][0] == '-')
            printf("opción: %s\n", argv[tokens]+1);
        else
            printf("argumento %d: %s\n", tokens, argv[tokens]);
    }
    exit(0);
}
```

Posteriormente construya el programa ejecutable con la siguiente instrucción en la línea de comandos:

```
gcc programa2-1.c -o programa2-1
```

gcc es el compilador más utilizado en los sistemas UNIX, la opción `-o` indica al compilador que el nombre del archivo ejecutable será el que se especifica a continuación (en este caso `programa2-1`). Se recomienda darle el mismo nombre al ejecutable que el código fuente para evitar confusiones. Es también común dejar al archivo ejecutable sin una extensión.

Pregunta 2.1

Quite la extensión `.c` al programa e intente compilar, posteriormente cambie la extensión por otra de su agrado. ¿Qué sucede?

Se puede ver el manual del compilador gcc mediante el comando de ayuda `man`. Sólo debemos teclear en la línea de comandos:

```
man gcc
```

Este comando es muy útil para ver el manual de cualquier llamada al sistema, función o comando.

Ejecute el programa anteponiendo `./` al nombre del programa ejecutable:

```
./programa2-1
```

Agregue otras palabras en la línea de comandos, éstas se conocen como argumentos del programa. Un argumento que inicia con el signo menos `-` se conoce como opción del programa.

Como podrá observar, nuestro programa detecta los argumentos y/o opciones que se le envían en la línea de comandos.

Descripción del funcionamiento

En su forma general un programa que se elabora en LINUX puede tener el siguiente prototipo para su función principal:

```
int main(int argc, char *argv[])
```

Esta función devuelve un entero y recibe dos argumentos, el número entero que devuelve es capturado por el sistema operativo. De modo que el sistema operativo puede saber cuál es el estado de terminación del programa.

Por convención, un programa que termina de manera normal debe devolver un cero, lo cual se logra con la instrucción:

```
exit(0);
```

en el caso de que exista un fallo se recomienda devolver un valor negativo.

El sistema operativo a través del intérprete de comandos (*shell*), puede enviarle información a nuestro programa cuando inicia. Esto se logra con ayuda de los parámetros en la función principal. El sistema operativo va a colocar en la variable `argc` el número de parámetros que introduzca el usuario en la línea de comandos. La variable `argv` almacena un arreglo de cadenas de caracteres, donde cada una de las cadenas contiene un argumento, comenzando por el nombre del programa, el cual se almacena en `argv[0]`, el siguiente argumento en `argv[1]` y así sucesivamente. La última cadena en el arreglo tendrá siempre un apuntador a NULL. Y como es normal en C, cada cadena va a terminar con el carácter fin de cadena `'\0'`.

Ejercicio 2.1

Suponga que no se dispone del argumento `argc` en la función principal, de modo que no se conoce el número de argumentos que el usuario introdujo en la línea de comandos. Modifique el programa 2-1 para que realice el mismo trabajo sin utilizar `argc`.

Ejercicio 2.2

Modifique el programa 2-1 para que imprima también, el número de caracteres que conforma cada uno de los argumentos.

Para que un programa pueda ejecutarse en segundo plano concurrentemente con otros, o para conectar la salida de un proceso con la entrada de otro, es necesario pasarle al programa toda la información que necesite, a través de la línea de comandos. Esta información se transmite en forma de opciones o con argumentos, ya que de otro modo tendrían que programarse menús interactivos, que evitarían a otros procesos compartir el teclado y la pantalla.