Parámetros y argumentos

Valores aleatorios en C

Aquí se utilizarán valores aleatorios para representar datos, lo cual permitirá no tener que estar ingresando datos con el comando “SCANF”. En el lenguaje C sobre Linux existen varias funciones que permiten obtener estos valores aleatorios.

En este apunte se verá un uso básico de estas funciones. Algunas de las cosas que se mencionan aquí no son útiles para aplicaciones más serias, en las que se requiere que la secuencia de números aleatorios sea muy aleatoria, impredecible, que no se repita hasta pasado muchos números, etc. Sin embargo, las explicaciones aquí presentadas servirán para la mayoría de los programas que se usarán.

La función rand()

En C, para obtener números aleatorios, existe la función **rand()**. Esta función, cada vez que se la llama, devuelve un número entero aleatorio entre **0** y el **RAND_MAX** (un número enorme, como de 2 mil millones).

El primer problema que se presenta es que no se suele querer un número aleatorio en ese rango, sería un dado muy curioso el que tenga tantas caras. Se puede querer, por ejemplo, un número aleatorio entre **0** y **10**. O, de forma más general, entre **0** y **N**. La cuenta que se debe hacer para eso es:

```
número = rand() % 11;

número = rand() % (N+1);
```

La operación módulo (%) da el resto de dividir **rand()** entre **11**. Este resto puede ir de **0** a **10**. De la misma forma, el módulo de **rand()** entre **N+1** va de **0** a **N**.

¿Y si se quiere un rango que **no** empiece en **0**? Por ejemplo, un rango entre **20** y **30** (de forma más general, entre **M** y **N,** con **N** mayor que **M**). Pues es fácil, se obtiene un número entre **0** y **10** y le se suma **20** (un número entre **0** y **N-M** y se le suma **M**):

```
número = rand () % 11 + 20;  // Éste está entre 20 y 30

número = rand () % (N-M+1) + M;   // Éste está entre M y N
```

La función srand()

Se presenta un nuevo problema. Si se ejecuta varias veces el programa, la secuencia de números aleatorios se repite. Imaginar que se tiene un programa que escribe 3 números aleatorios entre **0** y **10**. Se ejecuta una vez y sale, por ejemplo: 4, 7 y 9. Se ejecuta por segunda vez y vuelve a salir 4, 7 y 9. La tercera vez sale lo mismo, y al cabo de ejecutar muchísimas veces más, vuelve a salir lo mismo.

El problema es que **rand()** "calcula" los números aleatorios. Parte de un número inicial (llamado semilla), hace unas cuentas y saca un número aleatorio. Para el segundo número, hace unas cuentas con el resultado anterior y saca un segundo número, y así sucesivamente.

Si se vuelve a ejecutar el programa desde el principio, el número inicial (la semilla) que usa **rand()** es el mismo, con lo que la secuencia de números aleatorios es la misma, ya que las cuentas son las mismas.

Para evitar este problema, existe la función **srand()**, a la que se le pasa de parámetro un número que se utilizará como número inicial para las cuentas. A esta función sólo se la debe llamar una vez en el programa.

¿Qué número se le asigna a este srand()? No se puede ingresar un número fijo, porque entonces no se ha hecho nada. No se puede poner un número obtenido con rand(), porque la primera vez siempre dará el mismo, y el resultado será igual que si se le pone un número fijo. Hay que buscar la forma de obtener un número que sea distinto en la ejecución de cada programa.

Existen dos números que se utilizan habitualmente para ello, el que se utilizará es:

- La fecha/hora del sistema. Este valor cambia si se ejecuta el programa en distintos instantes de tiempo. Se tendría que arrancar el programa dos veces en el mismo segundo para obtener la misma secuencia de números aleatorios. En C de Linux, esta fecha/hora se obtiene con la función **time()**

```
srand (time(NULL));
```

A esta función **sólo hay que llamarla una vez al principio del programa**. Cada vez que se la llame, se estarán reiniciando los cálculos de números aleatorios desde el principio, con lo que se repetirá todo.

Pasar valores por parámetro a un proceso desde la línea de comandos

Para hacer un programa tipo consola, de modo que pueda recibir parámetros pasados desde la línea de comandos, la función main() debe tener la siguiente forma:

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

- intargc: indica el número total de parámetros.
- char *argv[]: mediante este vector se puede acceder a los valores de los parámetros pasados al programa. Siempre el primer parámetro (es decir, argv[0]) contiene el nombre del programa.

Al escribir en línea de comandos en una terminal de Linux, quedaría así:

Valores del vector argv (según el ejemplo):

- argv[0] → nombre del programa.
- argv[1] → primer argumento (en este caso, el “10”).
- argv[2] → segundo argumento, el “20”.
- argv[3] → tercer argumento, el “30”.
- argv[n] → continua hasta la cantidad de argumentos que se le pase.

El vector argv almacena sus valores como cadenas de texto {“nombrePrograma”, “10”,”20”,”30”}. Por lo tanto, sus argumentos podrían ser números o textos, como algún nombre, etc.

Ejemplo

En el siguiente código de ejemplo, se van a obtener 10 (diez) números aleatorios con un rango preestablecido DESDE 0 HASTA 10, y se permitirá modificar dicho rango pasando por argumento el deseado:

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

#define CANTIDAD 10
#define DESDE 0
#define HASTA 10

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


    //se cargan valores por defecto en caso de no pasar por parametros
    int desde = DESDE;
    int hasta = HASTA;
    int n,num;

// se valida, si se reciben los 2 argumentos, cargarlos. Tener en cuenta que el primer argumento, el “0”, es el nombre del programa.
    if(argc==3)
    {
        // atoi() transforma un texto al valor numérico
        desde = atoi(argv[1]);
        hasta = atoi(argv[2]);
    }

    printf("cantidad = %d   desde = %d   hasta = %d  \n",CANTIDAD, desde, hasta);
    srand(time(NULL));

    //En C, para obtener números aleatorios, existe la función rand().
    //Esta función, cada vez que se la llama,
    //devuelve un número entero aleatorio entre 0 y el RAND_MAX (un número enorme, como de 2 mil millones).
    //La operación módulo (%) da el resto de dividir rand() entre 11. Este resto puede ir de 0 a 10.
    //De la misma forma, el módulo de rand(), entre N+1, va de 0 a N.
    //Si se desea un rango que no empiece en 0, por ejemplo, entre 20 y 30
    //(de forma más general, entre M y N con N mayor que M).
    //Para esto se obtiene un número entre 0 y 10 y se le suma 20 (un número entre 0 y N-M, y se le suma M)
    //numero = (rand () % 11) + 20;  // Éste está entre 20 y 30
    //         (rand () & (30-20+1)) + 20;
    //numero = (rand () % (N-M+1)) + M;   // Éste está entre M y N
    for (n=0; n<CANTIDAD;n++)
```

```
    {
        num = (rand()%(hasta-desde+1))+desde;
        printf("el numero es %d \n", num);
    }
    return 0;
}
```

Nota: la función **atoi()** convierte la cadena de texto (char[] o char*) a un valor int.