OBTENCIÓN DE NÚMEROS ALEATORIOS

A veces queremos que nuestro programa obtenga números de forma aleatoria, por ejemplo, para simular una tirada de dados o el reparto de cartas en un juego. En C de linux tenemos varias funciones que nos permiten obtener estos valores aleatorios.

En esta explicación vamos a ver un uso básico de estas funciones. Algunas de las cosas que contamos 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, etc. Sin embargo, las explicaciones aquí presentadas servirán para la mayoría de nuestros programas.

LA FUNCIÓN rand()

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

El primer problema que se nos presenta es que no solemos querer un número aleatorio en ese rango, sería un dado muy curioso el que tenga tantas caras. Podemos querer, por ejemplo, un número aleatorio entre **0** y **10**. O de forma más general, entre **0** y **N**. La cuenta que debemos echar para eso es esta

```
numero = rand() % 11;
numero = rand() % (N+1);
```

La operación módulo (**%**) nos 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 queremos un rango que no empiece en **0**?. Por ejemplo, queremos un rango entre **20** y **30** (de forma más general, entre **M** y **N** con **N** mayor que **M**). Pues es fácil, obtenemos un número entre **0** y **10** y le sumamos **20** (un número entre **0** y **N-M** y le sumamos **M**)

```
numero = rand () % 11 + 20;  // Este está entreo 20 y 30
numero = rand () % (N-M+1) + M;   // Este está entre M y N
```

LA FUNCIÓN srand()

Se nos presenta un nuevo problema. Si ejecutamos varias veces nuestro programa, la secuencia de números aleatorios se repite. Imaginemos que tenemos un programa que escribe 3 número aleatorios entre **0** y **10**. Lo ejecutamos una vez y sale, por ejemplo 4, 7 y 9. Lo ejecutamos por segunda vez y vuelve a salir 4, 7 y 9. La tercera vez sale lo mismo y cuando ya nos hemos hartado de ejecutar, vuelve a salir lo mismo.

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

Si volvemos 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 tenemos 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 debemos llamarla una vez en nuestro programa.

¿Qué número le ponemos a este srand()?. No podemos ponerle un número fijo, porque entonces no hemos hecho nada. No podemos ponerle un número obtenido con rand(), porque la primera vez siempre nos dará el mismo y el resultado será igual que si le ponemos un número fijo. Debemos buscar la forma de obtener un número que sea distinto en la ejecución de cada programa.

Hay dos números que se utilizan habitualmente para ello:

- La fecha/hora del sistema. Este valor cambia si ejecutamos el programa en distinto instante de tiempo. Tendriamos 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));
```

- El número de proceso del programa. El primer programa que se arranca cuando se enciende el ordenador con el sistema operativo linux, tiene el número de proceso 1, el segundo el 2, el tercero el 3 y así sucesivamente. Cuando arrancamos nuestro programa, se le asignará el número que le toque, por ejemplo, el 215. Cuando lo volvamos a arrancar, se le asignará el que le toque (puede ser 216 si no hemos ejecutado nada entre medias o 345, si nos hemos entretenido con otras cosas). Después de ejecutar nuestro programa varios miles de veces, el número de proceso puede que se repita, pero ya no nos acordaremos de la secuencia que se sacó la primera vez. El número de proceso se obtiene con **getpid()**

```
srand (getpid());
```

A esta función **sólo hay que llamarla una vez al principio de nuestro programa**. Cada vez que la llamemos, estaremos reiniciando los calculos de números aleatoriosd desde el principio, con lo que se repetirá todo.

LAS FUNCIONES drand48() y srand48()

¿Y si queremos un número aleatorio con decimales?. Tenemos la función **drand48()** que nos devuelve un número aleatorio con decimales entre **0.0** (incluido, puede salir el **0,0**) y **1.0** (excluido, nunca saldrá **1.0**). A partir de ahí es fácil transformarlo al rango que queramos. Si queremos un rango entre **10.0** y **20.0** (o entre **M** y **N** siendo **N** mayor que **M** y ambos con decimales, aunque sean .0)

```
numero = drand48() * (20.0-10.0) + 10.0;
numero = drand48() * (N-M) + N;
```

De la misma forma que antes, la secuencia de números aleatorios se repetirá cada vez que ejecutemos nuestro programa y de igual manera, tenemos manera de cambiar la "semilla" de esa secuencia. Hay que llamar a la función **srand48()** pasándole un entero que sea disntinto en cada ejecución del programa. Nuevamente, tenemos las dos opciones anteriores

```
srand48(time(NULL));
srand48(getpid());
```