Práctica No. 3
Utileria make en Turbo C

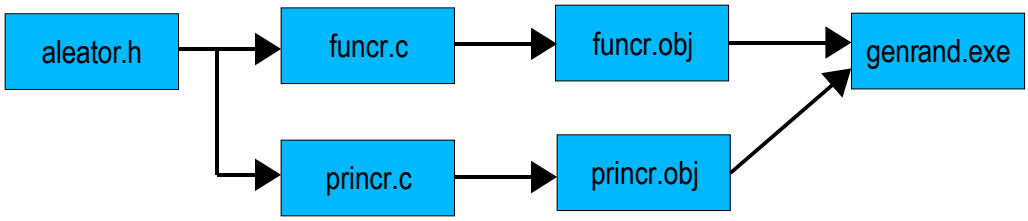
Objetivo:
Generar el archivo `makefile` para un programa generador de series de números aleatorios.

Introducción:
Se les ha pedido que escriban unprograma que genere una serie de números aleatorios. Este programa requiere como valores de entrada un número semilla y el tamaño de la serie de números a generar. El ejecutable debe llamarse **genrand**. A manera de ejemplo, para usar la semilla 3456 para generar una serie de 20 números, el programa debe invocarse así:

```
genrand 3456 20<enter>
```

Los archivos fuentes son: `princr.c`, `funcr.c` y `aleator.h`*

Las dependencias de estos archivos se ilustra en seguida:



*Nota: Los códigos de estos programas están al final de esta guía y en la página del curso.

Actividades

1.- Captura los archivos `aleator.h`, `funcr.c` y `princr.c`
Después de capturarlos, con el comando `dir` (tecleando *`dir al*.h`*, después *`dir fun*.c`* y finalmente *`dir pri*.c`*), registra tamaño y fecha y hora de cada uno de estos archivos.

`aleator.h`	Tamaño: _____	Fecha y hora: _____
`funcr.c`	Tamaño: _____	Fecha y hora: _____
`princr.c`	Tamaño: _____	Fecha y hora: _____

2.- Utilizando como ayuda la siguiente referencia la sección correspondiente al programa make en la siguiente referencia: http://www-etsi2.ugr.es/depar/ccia/mp2/practicas/Introgcc.pdf

Diseña y escribe el archivo `makefile` para generar este programa.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

2.1.- Utilizando el editor del IDE de Turbo C, captura tu diseño del archivo anterior y llámalo `makefile`.
Archivo `makefile` capturado: OK: _____ Tamaño: _____ Fecha/hora: _____

3.- En una ventana de MS-DOS y desde el directorio `c:tc\bin`, ejecuta la utilería `make` (tecleando *make<enter>*) para comprobar el funcionamiento de tu archivo `makefile`. Anota los resultados:

_____
_____
_____
_____

4.- Si no lo haz hecho, agrega el objetivo `clean` a tu archivo `makefile` para borrar todos los archivos objeto. (El comando para borrar en DOS es *del nombre_archivo.ext*). Escribe abajo las líneas que se requieren para agregar este nuevo objetivo a tu archivo `makefile`:

_____
_____

4.1.- Aplica las modificaciones anteriores, ejecuta `make clean` y reporta los resultados:

_____
_____

4.2.- Ejecuta nuevamente `make` , reporta y explica los resultados:

_____
_____

5.- Para comprobar el funcionamiento de `make`, vamos a alterar la fecha y hora de alguno de los archivos del programa. Para esto utilizaremos el programa `touch.com`.
Existe `touch.com` en `c:\tc\bin`? Sí: _____ No: _____
Si no existe, pide al profesor el diskette donde se encuentra este archivo y cópialo a tu direcotrio `c:\tc\bin` o bien bájalo de la página del curso.

5.1.- El comando `touch` actualiza la fecha de los archivos. Utilízalo para actualizar la fecha del archivo `aleator.h` tecleando *touch aleator.h* y, usando `dir ale*.h`, anota su nueva fecha y hora:
`aleator.h` Tamaño: _____ Fecha y hora: _____

Compáralo con los datos del inciso 1, y verifica que la nueva hora sea posterior a la primera.
Nueva hora de `aleator.h` es posterior: OK ____

5.2.- Dado que los dos archivos objeto y el ejecutable del programa dependen de `aleator.h`, al ejecutarse `make` deberán generarse nuevamente los 2 archivos objeto y el ejecutable. Ejecuta `make` y reporta los resultados:

_____
_____

5.3.- Utilizando `touch` actualiza la fecha del archivo `funcr.c`.
Archivo `funcr.c` Tamaño: _____ Fecha y hora: _____
Nueva hora de `funcr.c` es posterior: OK ____

Ejecuta nuevamente `make` y reporta cuáles archivos se recompilaron. Explica porqué.

_____
_____

5.4.- Un objetivo común de los archivos `makefile` es `install`. Agrega este objetivo para que copie tu ejecutable al directorio `c:\windows\system`, ejecuta `make install` y reporta los resultados. (el comando para copiar en DOS en *`copy nombre.ext dir_destino`*):

Objetivo:

_____

_____

Resultados de la ejecución de `make install`

_____

_____

_____

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

Código de los archivos de esta práctica

```
/* ---------------------------------------------------------
 * Archivo: aleator.h
 * Cabecera para generador de series de numeros aleatorios
 * --------------------------------------------------------- */
/* Librerias */
#include<stdlib.h>

/* Constantes */
#define MAX 100

/* Prototipos */
void inicializa_random(long);
double obten_sig_random(void);
```

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

```
/* ---------------------------------------------------------------------
 * Archivo: funcr.c
 * Código de las funciones con prototipos en aleator.h
 * --------------------------------------------------------------------- */
#include "aleator.h"

void inicializa_random(long semilla)
{
      srand(semilla);
}

double obten_sig_random(void)
{
      return rand();
}
```

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

```
/* ------------------------------------------------------------
 * Archivo: princr.c
 * Genera una serie de numeros aleatorios. Depende de func_r.c y aleator.h
 * ------------------------------------------------------------ */
#include "aleator.h"
#include<stdio.h>

int main(int argc, char *argv[])
{
      int iters,i;
      long misemilla;

      if(argc!=3) {
            fprintf(stderr,"Uso: %s semilla iteraciones\n",argv[0]);
            exit(1);
      }
```

```
      misemilla=(long)atoi(argv[1]);
      iters=atoi(argv[2]);

      if (iters > MAX) {
        fprintf(stderr,"El numero de iteraciones excede el maximo definido\n");
            exit(1);
      }

      system("cls");

      printf("Los %d numeros aleatorios generados son:\n",iters);
      inicializa_random(misemilla);

      for(i=0;i<iters;i++)
            printf("%d: %f\n",i,obten_sig_random());
      return 0;
}
```