

Sesión 4: Módulos en C

Hoja de problemas


Programación para Sistemas


Ángel Herranz


aherranz@fi.upm.es

Universidad Politécnica de Madrid

Otoño 2018

 **Ejercicio 1.** Repasa las transparencias de clase. Presta especial atención a los detalles sintácticos del fichero Makefile. Intenta entender el *guión* que lleva a solucionar cada problema que va surgiendo.

 **Ejercicio 2.** Asegúrate de realizar todos los ejercicios de las transparencias antes de continuar. Deberías tener en un directorio los siguientes ficheros: `lcg1.c`, `sum1.c`, `lcg2.c`, `generador_lcg.c`, `generador_lcg.h` y un maravilloso Makefile con el que puedes *fabricar* los ejecutables `lcg1`, `sum1`, y `lcg2`.

 **Ejercicio 3.** Antes de continuar, vamos a enriquecer el Makefile para que con la simple ejecución de `make` se creen todos los ejecutables. Para ello basta con que añadas esta regla justo después de que establezcas la variable `CFLAGS`:

```
CFLAGS=-Wall -g -pedantic

todos: lcg1 lcg2 sum1


...
```

Prueba ahora a ejecutar

```
$ make todos
```

o simplemente

```
$ make todos
```

 **Ejercicio 4.** A veces, después de ejecutar `make`, el directorio en el que estás trabajando se llena de ficheros *feos* o inservibles. Vamos a añadir un par de reglas al final a nuestro el Makefile para realizar una limpieza:

```
...

limpio:
    rm -f *.o

muylimpio: limpio
    rm -f lcg1 lcg2 sum1
```

Ahora, la ejecución de

```
$ make limpio
```

borrará todos los ficheros .o y

```
$ make muylimpio
```

hará lo mismo que make limpio y además borrará todos los ejecutables.

```
El contenido final del Makefile debería ser al-
go parecido a esto:

CFLAGS=-Wall -g -pedantic

todos: lcg1 sum1 lcg2 test-lcg
    lcg1.o: lcg1.o $(CFLAGS) $(CC)
    sum1.o: sum1.o $(CFLAGS) $(CC)
    lcg2.o: lcg2.o lcg2.c generador-lcg.h
    lcg2.o: lcg2.o generador-lcg.o
    test-lcg: test-lcg.o generador-lcg.o
    test-lcg.o: test-lcg2.c generador-lcg.h
    test-lcg: test-lcg.o generador-lcg.o $(CFLAGS) $(CC)
    lcg1.o: lcg1.o $(CFLAGS) $(CC)
    sum1.o: sum1.o $(CFLAGS) $(CC)
    lcg1.o: lcg1.o $(CFLAGS) $(CC)
    sum1.o: sum1.o $(CFLAGS) $(CC)
    lcg2.o: lcg2.o lcg2.c generador-lcg.h
    lcg2.o: lcg2.o generador-lcg.o
    test-lcg: test-lcg.o generador-lcg.o $(CFLAGS) $(CC)
```

Ejercicio 5. Ejecución paso a paso: gdb lcg1

- Poner en marcha el depurador `gdb`.
- Colocar un *breakpoint* en `main`.
- Ejecutar el programa paso a paso y explorar las variables¹
- ¿Puedes ver el valor de `anterior` cuando está ejecutando `main`? ¿Y el valor de `x`?
- ¿Puedes ver el valor de `i`, variable de `main` cuando está ejecutando `generar_aleatorio`?

📄 **Ejercicio 6.** Ejecución paso a paso: `gdb sum1` (usa un número bajito ;))

- Poner en marcha el depurador `gdb`.
- Colocar un *breakpoint* en `main`.
- Ejecutar el programa paso a paso y explorar las variables²
- ¿Puedes ver el valor de `n` cuando está ejecutando `main`? ¿Y el valor de `i`?
- ¿Puedes ver el valor de `n`, variable de `main` cuando está ejecutando `sum`?

📄 **Ejercicio 7.** Tu labor en este ejercicio será modificar el módulo `generador_lcg` y enriquecerlo de la siguiente forma.

- La idea es tener un módulo desde el que poder cambiar en tiempo de ejecución los parámetros a , c y m del generador (ahora no es posible porque se han definido como macros).
- El módulo va a contener tres variables: `lcg_a`, `lcg_c`, y `lcg_m`.
- Además, se expondrá una operación para establecer la *semilla* del generador, es decir, el valor de X_0 .

Para facilitar la tarea dispones del *header* aquí:

`generador_lcg.h`

```
/* Parámetros a, c, y m del generador */
extern int lcg_a = 1;
extern int lcg_c = 1;
extern int lcg_m = 1;

/* Resetea el generador colocando el valor de  $X_0$  al valor de semilla */
extern void lcg_resetea(int semilla);

/* Devuelve el valor de  $X$  y genera el siguiente */
extern int lcg_generar();
```

Para que puedas ver si lo que has hecho funciona correctamente puedes usar este *tester*:

¹Ver transparencias de la sesión 2

²Ver transparencias de la sesión 2

test_lcg.h

```
#include <stdio.h>
#include <assert.h>
#include "generador_lcg.h"
int main() {

    lcg_a = 7;
    lcg_c = 1;
    lcg_m = 11;

    lcg_resetear(9);

    assert(lcg_generar() == 9);
    /* 7 * 9 + 1 % 11 == 9 */
    assert(lcg_generar() == 9);

    lcg_resetear(0);

    assert(lcg_generar() == 0);
    /* 7 * 0 + 1 % 11 == 1 */
    assert(lcg_generar() == 1);
    /* 7 * 1 + 1 % 11 == 8 */
    assert(lcg_generar() == 8);
    /* 7 * 8 + 1 % 11 == 2 */
    assert(lcg_generar() == 2);
}
```