Fundamentos de Software

Doble Grado en Ingeniería Informática y Matemáticas

Universidad de Granada

Ejercicios repaso Módulo II

Práctica 7

Considerando los ficheros proporcionados (main2.cpp, factorial.cpp, hello.cpp, sin.cpp, cos.cpp y tan.cpp), realiza las siguientes operaciones por pasos (no te saltes ninguno y ve mostrando cada uno de los resultados requeridos):

1. Escribe en un archivo *makefile* las órdenes necesarias para que dé la posibilidad de crear el archivo compilado **hello.o**.
    a. Mediante el *makefile*, haz que se cree dicho fichero.

2. Modifica el mismo archivo para se puedan compilar los archivos **tan.cpp, cos.cpp** y **sin.cpp** y obtener sus correspondientes archivos objeto.
    a. Mediante el makefile, haz que se creen cada uno de los ficheros independientemente.
    b. Mediante el makefile, haz que se creen cada uno de los ficheros de 1 sola vez.

3. Seguidamente, modifica el *makefile* para que genere una librería llamada **libmath**.
    a. Ejecuta el makefile y comprueba qué resultado ha proporcionado.

4. Añade las órdenes necesarias para obtener el programa principal compilado y un fichero ejecutable denominado Practica7.

5. Modifica el makefile añadiendo al código las siguientes variables:
    a. Una variable que almacene dónde están los ficheros **include**.
    b. Una variable que almacene el **compilador**.
    c. Utiliza la variable en la que se guarda el **nombre del objetivo** de la regla actual para no tener que repetir siempre dicha cadena.
    d. Modifica las reglas que puedas utilizando la variable donde se almacenan las **dependencias** de la regla actual.

Entrega el fichero makefile resultante en la tarea en Prado.

Práctica 8

Compila el código del siguiente programa, escribe un guion gdb y ejecútalo para mostrar el valor de la variable final2 justo antes de que se le asigne ningún valor dentro del bucle for. Describe todos los pasos que has seguido e incluye una captura de pantalla de los mismos.

```
#include <iostream>
/* Incrementa en 2 una variable */
int cuenta (int y) {
int tmp;
tmp = y + 2;
return tmp;
}
/* Calcula la multiplicación de dos números */
int multiplica (int x, int y) {
int final;
int i;
final = 0;
for (i = 0; i < x; i ++) {
final = final + y;
}
return final;
}
int main (void) {
int final1;
int final2;
int i;
final1 = multiplica(3, 2);
for (i = 0; i < 100; i ++)
final2 = cuenta(i);
std::cout << final1 << "\n";
return 0;
}
```

Siguiendo el ejercicio anterior, haz todas las configuraciones necesarias utilizando el depurador gdb para obtener el valor de la variable final2 cuando i vale 50. Muestra todos los comandos utilizados y el valor de las variables final2 e i.

Con el mismo código del ejercicio anterior no parece que la variable final2 pueda ser mayor de 101. Utilizando el depurador gdb haz, que sin tocar la variable final2, esta variable tenga un valor por encima de 1000 al final del programa. Describe todos los pasos que has seguido para conseguirlo e incluye la captura de pantalla correspondiente.

EJERCICIO 7

```
# Nombre archivo: makefileEjercicio7


# Variable que indica el compilador que se va a utilizar
CC=g++

# Variable que indica el directorio en donde se encuentran los archivos de cabecera
INC_DIR=.

# Variable que indica el directorio en donde se encuentran los archivos objeto
LIB_DIR=.

Practica7: main2.o factorial.o hello.o $(LIB_DIR)libmath.a
	$(CC) -L$(LIB_DIR) -o $@ $^

main2.o: main2.cpp $(INC_DIR)/mates.a
	$(CC) -I$(INC_DIR) -c $<

factorial.o: factorial.cpp $(INC_DIR)/functions.h
	$(CC) -I$(INC_DIR) -c $<

hello.o: hello.cpp $(INC_DIR)/functions.h
	$(CC) -I$(INC_DIR) -c $<

libmath.a: tan.o cos.o sin.o
	ar -rvs libmath.a $^

tan.o: tan.cpp
	$(CC) -I$(INC_DIR) -c $<

cos.o: cos.cpp
	$(CC) -I$(INC_DIR) -c $<

sin.o: sin.cpp
	$(CC) -I$(INC_DIR) -c $<
```

EJERCICIO 8

Aclaración: En este ejercicio voy a hacer uso del fichero cpp mainsesion09a.cpp de la sesión de prácticas anterior ya que su código coincide con el del enunciado, por lo que las líneas en las que pondré los breakpoints serán fijándome en ese fichero cpp.

Primero de todo, necesitamos crear el archivo ejecutable para poder depurarlo con gdb:

joshoc7@joshoc7-Aspire-A315-56:~$ g++ -g mainsesion09.cpp -o ficheroejec

Apartado a) Como queremos mostrar el valor de final2 antes de que se incialice, pondremos un breakpoint en la línea 47, justo antes de que se modifique final2. Por lo tanto, el código de nuestro primer guión gdb (guionA.gdb) será el siguiente:

```
break 47
run
print final2
```

Lo ejecutamos con **gdb -x guionA.gdb ficheroejec** y el resultado es:

```
Reading symbols from ficheroejec...
Punto de interrupción 1 at 0x1222: file mainsesion09a.cpp, line 48.

Breakpoint 1, main () at mainsesion09a.cpp:48
48          final2 = cuenta(i);
$1 = 0
```

Donde $1 es la variable final2 que, lógicamente, tiene el valor 0.

Apartado b) Ahora se quiere mostrar final2 e i justo cuando i vale 50, es decir, antes de que se ejecute el bucle cuando i vale 50. Por ello, necesitamos poner un breakpoint condicional, y el código de nuestro segundo guión gdb (guionB.gdb) será el siguiente:

```
break 47 if i == 50
run
info locals
```

Lo ejecutamos con **gdb -x guionB.gdb ficheroejec** y el resultado es:

```
Reading symbols from ficheroejec...
Punto de interrupción 1 at 0x1222: file mainsesion09a.cpp, line 48.

Breakpoint 1, main () at mainsesion09a.cpp:48
48          final2 = cuenta(i);
final1 = 6
```

final2 = 51
i = 50

Y así ya podemos ver los valores de i y final50, que son correctos ya que i es 50 y final2 tiene el valor de la anterior iteración porque 49+2 = 51

Apartado c) En este último apartado pondremos un breakpoint justo antes de que finalice el programa, al acabar el bucle for. Entonces, usaremos la orden set variable para asignarle a final2 un valor por encima de 1000, por ejemplo, 1001. El código de nuestro tercer guión gdb (guionC.gdb) será el siguiente:

```
break 49
run
set variable final2=1001
print final2
```

Lo ejecutamos con **gdb -x guionC.gdb ficheroejec** y el resultado es:

```
Reading symbols from ficheroejec...
Punto de interrupción 1 at 0x1235: file mainsesion09a.cpp, line 50.

Breakpoint 1, main () at mainsesion09a.cpp:50
50        std::cout << final1 << "\n";
$1 = 1001
```

Siendo $1 la variable final2, que tiene justo el valor que se le ha asignado con la orden set variable.