gcc arm

GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr. La sigla GCC significa "GNU Compiler Collection". Originalmente significaba "GNU C Compiler"; todavía se usa GCC para designar una compilación en C. G++ refiere a una compilación en C++.

```
Sintaxis.
gcc [ opción | archivo ] ...
g++ [ opción | archivo ] ...
```

Las opciones van precedidas de un guión, como es habitual en UNIX, pero las opciones en sí pueden tener varias letras; no pueden agruparse varias opciones tras un mismo guión. Algunas opciones requieren después un nombre de archivo o directorio, otras no. Finalmente, pueden darse varios nombres de archivo a incluir en el proceso de compilación.

Sufijos en nombres de archivo.

Son habituales las siguientes extensiones o sufijos de los nombres de archivo:

Extensión	Explicación
.c	Fuente en C
.C .cc .cpp .c++ .cp .cxx	Fuente en C++; se recomienda .cpp
.m	Fuente en Objective-C
.i	C preprocesado
.ii	C++ preprocesdo
.s	Fuente en lenguaje ensamblador
.o	Código objeto
.h	Archivo para preprocesador (encabezados), no suele figurar en la linea de comando de gcc

Opciones.

especificación	Acción a realizar
-c	Realiza preprocesamiento y compilación, obteniendo el archivo en código objeto; no realiza el enlazado.
-E	Realiza solamente el preprocesamiento, enviando el resultado a la salida estándar.
-o	Archivo indica el nombre del archivo de salida, cualesquiera sean las etapas cumplidas.
-Iruta	Especifica la ruta hacia el directorio donde se encuentran los archivos marcados para incluir en el programa fuente. No lleva espacio entre la I y la ruta, así: -I/usr/include -L especifica la ruta hacia el directorio donde se encuentran los archivos de biblioteca con el código objeto de las funciones referenciadas en el programa fuente. No lleva espacio entre la L y la ruta, así: -L/usr/lib
-Wall	Muestra todos los mensajes de error y advertencia del compilador, incluso algunos cuestionables pero en definitiva fáciles de evitar escribiendo el código con cuidado.
-g	Incluye en el ejecutable generado la información necesaria para poder rastrear los errores usando un depurador, tal como GDB (GNU Debugger).
-v	Muestra los comandos ejecutados en cada etapa de compilación y la versión del compilador. Es un informe muy detallado.

[1]

El gcc le da al programador un amplio control del proceso de compilación. Esto comprende 4 etapas: preprocesado, compilación, ensamblaje y vinculación. Uno puede detener el proceso despues de cualquera de dichas etapas para examinar el rendimiento del compilador en la misma. Cabe mencionar que incluya más de treinta advertencias individuales y tres niveles de advertnecia tipo "captura todo". También es importante mencionar que el gcc es un compilador cruzado, de modo que se pueda desarrollar el código en una arquitectura un de procesador y correrlo en otra, esto es imporante porque se pueden crear los codigos para x86s, PowerPC, Amiga, SCN Sparc, ARM, etc. Cada chip de procesador tiene una arquitectura física diferente de modo que la manera en que se debe construir el binario cambia para cada sistema.

Revisión de que se tenga instalado GCC
Normalmente gcc esta instalado por default en linux, sin embargo, no esta de mas revisarlo.
Desde la terminal de linux escribir (recuerda que $ significa el símbolo del sistema bash de linux)

$ whereis gcc

$ which gcc

$ gcc -v

Invocación de GCC

Para utilizar el gcc, suministre un nombre de archivo fuente de C y utilice la opción *-o* para especificar el nombre del archivo de salida. gcc *preprocesará, compilará ensamblara y vinculará (link)* el programa, generando un archivo ejecutable, a menudo denominado binario. La sintaxis más simple se escribe a continuación.

[1] Se puede consultar la guiá de referencia rápida *http://atc2.aut.uah.es/~jmruiz/Descarga_LE/GUIA_gcc.pdf*

```
gcc archivo_entrada.c [-o archivo_salida]
```

Archivo_salida.c es un archivo de código fuente en *c* y *-o* establece que el nombre del archivo de salida será archivo_salida.

Actividad

Objetivo: utilizar gcc para crear el programa hola a partir de la fuente hola.c

Procedimiento

Abrir la terminal de linux y escribir

```
$emacs -nw hola.c
```

Una vez abierto, escribir el código siguiente

```
/* Nombre del programa hola.c
*
* hola.c - programa canónico !hola, mundo¡
*/

#include <stdio.h>
#include <iostream>

int main(void)
{
puts("!Hola Alumno¡"); //Una forma de mostrar en terminal el mensaje
    printf("Hay que entregar las tareas"); //Una forma de mostrar en terminal el mensaje
std::cout << "y trabajos, y practicas y .... \n"; //Una forma de mostrar en terminal el mensaje
    return 0;
}
```

Una vez terminado, tecleamos *Ctrl-x s (es decir presionar la tecla Ctrl, x s)* con esto grabaremos el archivo. Compilamos el archivo fuente C++ con gcc, para ello, compilamos con *M-x (es decir la teclas Alt-x)* escribimos: compile y tecleamos enter, borramos *make -k* y colocamos la instrucción para compilar con gcc: gcc hola.c -o hola enter, si aparece algún error corregimos, guardamos y luego compilamos nuevamente.

Nota: Si te muestra fallas en la compilación puedes eliminar alguna de las lineas y el include iostream, esto es porque no viene por default en algunos linux y se debe agregar en forma posterior a la instalación.

El mensaje de que todo esta correcto es: *Compilation finished*

Ejecutar el archivo

Para ejecutar el programa desde el terminal Abrimos el terminal en la ruta donde esta el programa hola y escribimos ./hola enter

invocación de gcc paso a paso

En este ejercicio, hubo muchas cosas que tuvieron lugar de manera encubierta y ustedes no lo vieron, gcc primero corrió hola.c por el preprocesador, cpp, para expandir cualquier macro que pudiese haber e insertar los contenidos de los archivos incluidos mediante #include. Luego

compiló el código fuente preprocesado, conviertiendolo en código objeto, Finalmente el linker, Id, creo el archivo binario hola.

Uno puede recrear estas etapas manualmente, avanzando por le proceso de compilación un paso a la vez. Para indicarle a gcc que detenga la compilación luego de preprocesado, utilice la opción -E de gcc, como se indica a continuación.

```
$ gcc -E archivo_entrada.c -o archivo_salida.cpp
```

El paso siguiente consiste en compilar el archivo preprocesado para convertirlo en codigo objeto, utilizando la opcion -c de gcc. La opcion -x se utilizapara indicarle a gcc que comience la compilación a partir de cierta etapa. La sintaxis correcta para esta etapa es:

```
$ gcc -x cpp-output -c archivo_entrada.cpp -o archivo_salida.o
```

Linkeando el archivo objeto, finalmente, se obtiene su imagen binaria. El comando que llegaría a cabo la etapa de vinculación sería algo así como la siguiente

```
$ gcc archivo_etrada.o -o archivo salida
```

Veamos como se hace el ejercicio desarrollado

Para realizar el avance paso a paso a través del proceso de compilación tal como se describe a continuación. (en emacs recuerda que hay que escribir *M-x compile* en caso contrario, en la terminal escribir)

```
$ gcc -E hola.c -o hola.cpp
```

Si examinamos el archivo creado hola.cpp con el emacs (Ctrl-x f hola.cpp), se verá que el contenido del archivo de encabezamiento stdio.h ha sido efectivamente insertado en el código fuente, junto a otros símbolos de preprocesado.

El siguiente paso es compilar hola.cpp para converitrlo en código objeto:

```
$ gcc -x cpp-output -c hola.cpp -o hola.o
```

Se debe utilizar la opcion -x para indicarle a gcc que comience la compilación a cierta altura, en este caso, a partir del código fuente preprocesado. Cuando se efectúa el linkeo del código objeto, finalmente, se crea un archivo binario:

```
$ gcc hola.o -o hola
```

Espero que con esto, se pueda entender todo el proceso que corresponda a la compilación y creación de un archivo ejecutable.

Practica

Diseñar un programa que itere por un lazo de 10 veces, incrementando una variable de 2 en 2 por cada iteración. (mostrando en la terminal cada incremento). Realizar el preprocesado, compilado, ensamblado y vinculado (link) del programa y subir cada uno de ellos a la plataforma de git