Compilar programas para UNIX en C/C++

Guillermo Buriticá Tobón

13 de junio de 2013

Resumen

En este documento se mostrara el proceso básico que necesita saber para poder crear y compilar programas en C (o C++) . El proceso que se muestra describe formalmente el modelo de compilación C y como soportar librerias adicionales.

1. Crear, Compilar y Ejecutar su programa

Los pasos para crear y compilar un programa en C pueden ser los siguientes:

- Escribir y crear el codigo
- Compilar el codigo
- Ejecutar el programa

1.1. Crear el programa

Cree un archivo que contenga el programa completo como se muestra en el ejemplo para esto usted debe usar un editor de texto de los provistos en el sistema Unix como emacs, textedit, pico o vi. (El editor vi se encuentra en todas las distribuciones Unix).

La extencion del archivo por convencion debe ser ".c" (en letras minusculas), ejemplo programa.c o prueba.c. El contenido del archivo debe cumplir con las especificaciones sisntacticas del lenguaje C.

1.2. Compilar el programa

Existen muchos compiladores de C en el entorno. como por cc que es el compilador estandard de Sun, GNU C gcc es muy popular y se encuentra disponible para mucahs versiones de Unix o Linux. Los usuarios de Windows seguramante conoceran Borland compiler bcc.

Tambien se puede encontrar el compilador equivalente para C++ usualmente denotado como CC (observe las latras mayusculas CC). Por ejemplo Sun provee CC y GNU GCC. El compilador GNU tambien se escribe como g++

Otros compiladores de C/C++ (menos comunes). Todos los compiladores citados trabajan en esencia de la misma manera y comparten muchas opciones de la linea de parametros. Posteriormente en este documenteo se mostraran ejemplos de la linea de comandos. La mejor forma de obtener informacion acerca de cada compilador es mediante la documentacion en linea (manual pages) de sus sistema: Ejemplo man cc.

A fin de mantener este documento compacto nos referiremos y centrarnos en el tema de discucion solamente nos referiremos al compilador cc – otrso compladores solamente requieren cambiar cc por el comado apropiado.

Para compilar su programa simplemente teclee cc. El comando debe seguir con el nombre del programa que quiera compilar escrito en C. Y una serie de opciones que se quiera especifiacar. No nos concentraremos en muchas de estas opciones de compilacion que pueden ser consultadas en el manual en linea.

Los pasos para la compilacion basica serian:

```
cc programa.c
```

Donde programa.c es el nombre del archivo.

Si se encuentran errores en su programa (como errores tipograficos, falta de definiciones, o omiciones de punto y coma),el compilador los detectara y los reportara apropiadamente.

Claro que tambien pueden haber errores logicos que el compilador no esta en capacidad de detectar. Usted debe decirle al computador que hacer con las operaciones erroneas.

Cuando el compilador ha creado el ejecutable o version compilada de forma satisfactoria, este se encontrara en un archivo llamado a.out o si la opcion -o es usada en el nombre que continua a -o.

Es es mas apropiado utilizar -o con un nombre de archivo en la linea de comandos como se muestra a continuacion

```
cc -o programa programa.c
```

Esto pondra el programa compilado en el archivo programa (o en el archivo que usted designe a continuacion del argumento "-o") en cambio del archivo por defecto a.out.

1.3. Ejecutando el programa

El siguiente paso es ejecutar el programa. Para ejecutar un programa en UNIX simplemente escriba el nombre del programa en este caso programa o a.out segun si utilizo -o o no. (en algunos sistemas UNIX el administrador elimina el path de los directorios de usuario en esta caso el programa no se encuentra esto se soluciona mediante ./programa o ./a.out)

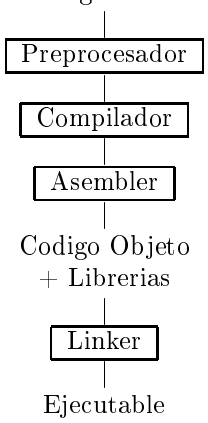
Este procedimiento ejecuta su programa, escribiendo los resultados en pantalla. En este momento se pueden tener errores de ejecucion conocidos como run-time errors, estos podrian ser una division por cero, o algunos mas evidentes como una salida erronea.

Se debe volver a editar el programa fuente, recompilarlo y ejecutarlo de nuevo.

2. El modelo de compilacion C

De forma rapida el modelo de compilacion seria como se muestra en la Figura 1.

Figura 1: El modelo de compilacion C



2.1. El Preprocesador

En terminos generales el preprocesador es el encargado recibir el codigo fuente y de eliminar los comentarios tambien es el encargado de interpretar las directivas especiales de compilacion identificadas con el caracter #

Ejemplo:

```
#include -- Incluye el contenido de un archivo.
#include <math.h> -- Libreria matematica.
#include <stdio.h> -- Libreria estandard de E/S
#define -- Define un simbolo, macro o una constante.
#define MAX_MAT 100
```

2.2. El compilador de C

El compilador C traduce el archivo fuente en codigo asembler. El codigo fuente es recibido desde el preprocesador.

Ensablador o Asembler

El asembler crea un codigo objeto . En el sistema UNIX se crea un archivo con el sufijo a .o (.OBJ on MSDOS) que indica un archivo en codigo objeto.

Encadenador o Linker

Si el archivo fuente hace referencia a funciones contenidas en librerias o funciones contenidas en otros archivos fuentes el linker convina estas funciones con (con main()) para crear el archivo ejecutable. Las referencias a variables externas tambien son resueltas en esta parte del proceso. y se crea el archivo ejecutable.

2.3. Algunas opciones de compilacion

Ahora que se tiene un conocimiento basico acerca del modelo de compilacion vamos a introducir algunas opciones de compilacion utiles. de nuevo mire el manual en linea para informacion futura.

-c Omite el proceso de linkeado y produce un archivo a .o para cada archivo fuente en la lista. Posteriormente pueden ser linkeados para producir un archivo ejecutable, por ejemplo:

```
cc file1.o file2.o ..... -o executable
```

-llibrary Introduce librarias objeto al linker. Esta opcion debe seguir del los argumentos del archivo. Las librerias objeto son archivos y pueden ser librerias estandard, fabricadas por terceras partes or librarias de usuario. Es probable que la libraria mas utilizada sea la libraria matematica (math.h). Usted debe introducirla esplicitamente si desea utilizar funciones matematicas (nota no olvide incluir el encabezado #include <math.h>), por ejemplo:

```
cc calc.c -o calc -lm
```

Muchas otras libreria pueden ser utilizadas en esta caracteristica.

-Ldirectorio Adiciona el directoria a la lista de directorios de busqueda de librerias. El linker siempre busca las librerias de forma predeterminada en /lib y /usr/lib. si quiere incluir liberias propias o creadas por terceros debe decirle al linker donde se encuentran, por ejemplo:

```
cc prog.c -L/home/myname/mylibs mylib.a
```

-Iruta Adiciona un camino de busqueda en los cuales se debe buscar los archivos a incluir mediante la directiva #include archivo espresado en forma relativa (no inician con slash /).

Por defecto, El preprocesador primero busca los archivos #include en el directorio que contiene el archivo fuente, luego en el directorio nombrado con el argumento de compilacion -I (si existe), y por ultimo en, /usr/include. en orden de incluir encabezados almacenados en /home/minombre/headers se debe hacer:

```
cc prog.c -I/home/myname/myheaders
```

Nota: Los emcabezados del sistema se encuentran en un directorio especial (/usr/include) y no se veran afectados por la opcion -I. Los encabezados del sistema y los encabezados de usuario son incluidos de una forma un poco diferente.

-g Invoca opciones de depuracion de errores. Esta directiva produce informacion adicional que es usada por una gran variedad de programas de depuración de errores como una tabla de simbolos.

-D Define simbolos como identificadores especiales (-Didentificador) o como valores (-Dsymbol=valor) es similar a la caracteristica #define del preprocesador.

2.4. Usando Librerias

C es un lenguaje muy pequeño. Muchas de las funciones que tienen los demas lenguajes de programacion no se encuentran incluidas en C. como por ejemplo no incluye operaciones de E/S, cadenas de caracteres o funciones matematicas.

Que usa entonces C?

C ofrece funcionalidad atraves de una gran cantidad de librerias.

Como resultado muchas implementaciones de C incluyen librerias de funciones que ofrecen funciones de E/S, manejo de cadenas de caracteres, etc. Por rasones practicas se dice que estas librerias son parte del lenguje C. Pero pueden cambiar de implementacion a implementacion o de maquina a maquina.

Un buen ingeniero debe poder desarrollar sus propias librerias de funciones e incluir librerias de terceras partes (e.j. NAG, PHIGS).

Todas las librerias (excepto la libreria estandard I/O <stdio.h>) requieren ser linkeadas explicitamente con la directiva -l y posiblemente con opciones -L.

2.5. UNIX Library Functions

El sistema UNIX provee una gran cantidad de librerias y funciones. Algunas implementas operaciones frecuentes, mientras que otras pueden ser muy especializadas para su aplicacion.

No reinvente la rueda: Es importante para los ingenieros tomarse el tiempo necesarios para revisar las caracteristicas de las librerias disponibles antes de escribir sus programas o sus propias versiones de funciones. Esto reduce

el tiempo de desarrollo y depuracion del software. Las funciones incluidas en las librerias han sido bien probadas y funcionan mejor que cualquier programa que alguien pueda escribir. Esto ahorra tiempo en todas las etapas del desarrollo de un programa.

2.6. Encontrando informacion acerca de las funciones de libreria

El manual UNIX tiene contenido todas las funciones disponibles. La documentacion esta almacenada en la seccion 3 del manual, y se encuentran muchas llamadas al sistema en la seccion 2. Si usted conoce de antemano el nombre de la funcion que busca puede leer la pagina particular nombrando la seccion (para leer acerca de la funcion sqrt):

```
man 3 sqrt
```

Si usted no conoce el nombre de la funcion, el listado completo se encuentra en la pagina introductoria de la seccion 3 del manual. para leerlo escriba lo siguiente:

```
man 3 intro
```

Se encuentran aproximadamente 700 funciones descritas ahi. Este numero tiende a incrementarse con cada actualizacion del sistema.

En algunos manuales, la seccion SYNOPSIS puede incluir informacion del uso de la funcion como por ejemplo :

```
#include <time.h>
char *ctime(time_t *clock)
```

Esto significa que debe incluir #include <time.h> en su archivo antes de utilizar la funcion ctime . y que la funcion ctime toma un apuntador del tipo time_t como argumento de llamada , y retorna un apuntador a una cadena de caracteres string (char *). la estructura time_t posiblemente este definida en la misma pagina del manual.

La sub-seccion DESCRIPCION le dara una breve descripcion de que es lo que la funcion hace por ejemplo:

```
ctime() converts a long integer, pointed to by clock, to a
26-character string of the form produced by asctime().
```

3. Lint – Un verificador de programas C

Usted descubrira (si no lo ha hecho aun) que el compilador de C en un poco simple en muchos aspectos como verificar la corectitud de los programas, especialmente en el chequeo de typos. Mediante el uso de prototipos de las funciones se puede ayudar a los compiladores modernos en esta tarea, no obstante esto

no garantiza que un programa compilado satisfactoriamente este se ejecute de forma adecuada.

La herramienta lint para UNIX puede ayudarlo en la revision de multiples errores de programacion. Revise el manual en linea (man lint) para obtener los detalles completos acerca de lint. Esto es una buena forma de ahorrar muchas horas de depuracion de programas escritos en C.

To run lint simply enter the command:

```
lint myprog.c
```

Lint es muy bueno en la revision de tipos de variables y asignacion de funciones, eficiencia, variables no utilizadas e identificadores de funciones, codigo no alcanzable y posibles fallos de memoria. Hay muchas opciones que permiten controlar el uso de lint (lea el manual).

4. Ejercicios

Ejercicios

1. Escriba, compile y ejecute el siguiente programa:

```
main()
{
int i;
printf("\t Number \t\t Square of Number\n\n");
for (i=0; i<=25;++i)
     printf("\t%d \t\t\t%d \n",i,i*i);
}
```

2. El siguiente programa utiliza la libreria matematica . Escriba, compile y ejecute el siguiente programa.

```
#include <math.h>
main()
{
int i;
printf("\t Number \t\t Square Root of Number\n\n");
for (i=0; i<=360; ++i)
    printf("\t%d \t\t\t%d \n",i, sqrt((double) i));
}
```

3. Mire en /lib y en /usr/lib buscando las librerias disponibles para su sistema.

   Use el programa man para obtener detalles de la libreria

   Esplore las librerias para ver cual de ellas contiene el comando ar t libfile.

4. Mire en /usr/include describa cuales encabezados se encuentran disponibles.

   Utilize el comado more o cat para mirar estas librerias

   Esplore los encabezados para mirar culaes contienen include, define, definiciones de tipo (type) and prototipos de funciones declaradas en ellas.

5. Suponga que tiene un programa en C con la funcion main en el archivo main.c y se tienen otras funciones en los archivos input.c and output.c:

   Que comandos debe usar para compilar y likear este programa?

   Como debe modificar el comando para incluir una libraria llamada proceso1 almacenado en el directorio que contiene la libreria estandard del sistema?

   Como debe modificar el comando para incluir una libraria llamada proceso2 almacenado en su directorio de trabajo?

   Algunos encabezados deben ser leidos y encontrados en un sub-directorio localizado en su directorio de trabajo. como se debe modificar el programa para incluir estos encabezados?

6. Suponga que tiene un programa escrito en C que incluye varios archivos separados, y estos incluyen algunos otros como se muestra a continuacion:

Archivo	#Include
principal.c	stdio.h; proceso1.h
entrada.c	stdio.h; listas.h
salida.c	stdio.h
proceso1.c	stdio.h; proceso1.h
proceso2.c	stdio.h; listas.h

   Cuales archivos deben ser recompilados se se hacen cambios a proceso1.c?

   Cuales archivos deben ser recompilados si se hacen cambios a proceso1.h?

   Cuales archivos deben ser recompilados despues de hacer cambios a lista.h?