Compilación de Programas C en UNIX y LINUX

La forma de compilar programas C en el entorno UNIX varía considerablemente entre las diferentes plataformas UNIX. Las versiones de Linux y FreeBSD 3.4 de UNIX usan el potente compilador GNU. Para conocer la versión disponible se ejecuta la orden:

```
$ gcc --version
2.7.2.3
$
```

La orden **cc** que es la más usada en las plataformas de UNIX, para compilar programas C, como se muestra en la siguiente sesión de una versión de BSD:

```
$ type cc
cc is a tracked alias for /usr/bin/cc
$ ls -li /usr/bin/cc
7951 -r-xr-xr-x 2 root wheel 49680 Dec 20 00:46 /usr/bin/cc

$ type gcc
gcc is a tracked alias for /usr/bin/gcc
$ ls -li /usr/bin/gcc
7951 -r-xr-xr-x 2 root wheel 49680 Dec 20 00:46 /usr/bin/gcc
$
```

Otras plataformas UNIX proporcionan sus propios compiladores de C y C++, las cuales difieren substancialmente en las opciones que permiten del compilador de GNU, así como en los mensajes que se producen y su capacidad de optimización. A continuación se verán algunas de las diferencias.

A.1 Orden (comando) de compilación cc

La mayoría de las plataformas UNIX invocan sus compiladores de C con el nombre `cc`. Las plataformas Linux y FreeBSD tienen el nombre de comando `gcc`, además del nombre `cc`. Algunas veces el compilador de GNU es instalado como `gcc` en plataformas comerciales para distinguirlo del estándar. Por ejemplo, HP incluye un compilador no ANSI con su sistema operativo HPUX, que es denominado el compilador "envuelto" (este compilador es suficiente para reconstruir un nuevo kernel para HPUX). El compilador ANSI debe ser adquirido por separado y, cuando se instala, reemplaza al comando `cc`.

Sin embargo, dentro de la misma plataforma, hay también hay otras opciones. HPUX 10.2 soporta el compilador `cc` y el compilador conforme con POSIX (estándar) `c89` La plataforma IBM AIX 4.3 soporta un compilador “extendido” de C, `cc`, y un compilador de ANSI C, `xlc` o `c89`. La diferencia entre los compiladores `xlc` y `c89` en AIX son las opciones por defecto configuradas. Las opciones, relativamente estandarizadas, son:

Opción -c

Esta opción es probablemente la más estandarizada universalmente. La opción `-c` indica que el compilador debería producir un archivo (fichero) objeto (*`fichero`*`.o`) pero sin intentar enlazar para obtener un ejecutable. Esta opción se usa cuando se compilan varios módulos fuentes separados que serán enlazados juntos en una etapa posterior por

medio del enlazador. Por ejemplo, se ha editado el archivo fuente `ecuacion.c`, la compilación con el comando `cc` y la opción `-c`:

```
$ cc -c ecuacion.c
```

El resultado de la compilación es un listado con los errores sintácticos del programa. O bien, de no haber errores, el archivo con el código objeto `ecuacion.o`. Una vez generado el código objeto, se enlaza y se genera el archivo ejecutable:

```
$ cc ecuacion.o
```

El siguiente ejemplo muestra como se compila y enlaza en un solo paso:

```
$ cc hello.c
```

Esta orden, de paso único traduce el archivo fuente escrito en C `hello.c`; el resultado de la compilación, si no hay errores, es el archivo ejecutable `a.out`. El nombre de fichero `a.out` es el nombre por defecto de un ejecutable que se genera como salida del compilador y del enlazador (*link*). Esta práctica se remonta al menos a 1970 cuando UNIX estaba escrito en lenguaje ensamblador sobre el PDP-11. El nombre de los archivos de salida por defecto del enlazador de Digital Equipment (DEC) también es `a.out`.

El programa C se puede escribir en varios módulos y cada uno estar guardado en un archivo. La compilación puede hacerse archivo tras archivo y después enlazarse para formar el archivo ejecutable. Por ejemplo, la aplicación de cálculo de nóminas se escribe en los archivos independientes: `nomina1.c`, `nomina2.c` y `nomina3.c`. La compilación de cada archivo fuente:

```
$ cc -c nomina1.c
$ cc -c nomina2.c
$ cc -c nomina3.c
```

A continuación se enlazan los tres archivos objetos generados (una vez que no hay errores sintácticos) como sigue:

```
$ cc nomina1.o nomina2.o nomina3.o
```

el resultado es el archivo ejecutable `a.out`. La orden `cc` con la opción `-c`, ejecutado para cada archivo fuente, produce, respectivamente, los archivos `nomina1.o`, `nomina2.o` y `nomina3.o`. Después, la orden `cc` acepta cada archivo objeto como entrada y produce el archivo ejecutable final con el nombre `a.out`. A continuación, se puede ejecutar el programa generado.

Opción -o

Esta es también bastante estándar. La opción `-o` permite al usuario especificar el nombre del archivo de salida. Por ejemplo, para el archivo `ecuacion.c` podría hacerse:

```
$ cc -c ecuacion.c -o mat_ecuacion.o
```

La opción `-c` indica que se va a producir un archivo objeto y la opción `-o` nombrará el archivo objeto de salida como `mat_ecuacion.o`.

La opción `-o` puede usarse también para nombrar el archivo ejecutable. Por ejemplo, el archivo ejecutable que se genera, a continuación, se nombra `prog_ecuacion`:

```
$ cc mat_ecuacion.o -o prog_ecuacion
```

Opción -g (depuración)

Esta opción estándar indica al compilador que debe generarse información de depuración en la salida de la compilación. Esta información de depuración hace que sea posible que el depurador haga referencia al código fuente y a los nombres de las variables, así como el análisis de un archiv `core` tras abortar un programa. Incluya esta opción cuando se necesite depurar un programa interactivamente o realizar un análisis *post-mortem* de un archivo `core`. Hay que asegurarse de usar esta opción con todos los módulos objetos que vayan a ser inspeccionados por el depurador.

Opción -D (define)

Esta opción estándar del compilador de C permite definir un símbolo de macro desde la línea de comandos del compilador. Frecuentemente es utilizada sobre todo desde el archivo `makefile` pero no está limitada a esta práctica. Por ejemplo:

```
$ cc -c -D_POSIX_C_SOURCE=199309L hello.c
```

define la macro constante en C `_POSIX_C_SOURCE` con el valor `199309L`. Esta definición de macro tiene el efecto de elegir un estándar particular POSIX de entre los ficheros incluidos en la compilación. Se pueden definir macros adicionales en la misma línea de órdenes

```
$ cc -c -D_POSIX_C_SOURCE=199309L -DNDEBUG hello.c
```

En este ejemplo se han definido dos macros para el archivo `hello.c`, la primera `_POSIX_C_SOURCE`, y a continuación la macro `NDEBUG` (sin valor), con el fin de deshabilitar la generación de código en las innovaciones a la macro `assert(3)` dentro del programa.

Opción -I (inclusión)

La opción estándar `-I` permite especificar directorios adicionales para buscar *archivos de inclusión* `include`. Por ejemplo, si se tienen archivos adicionales `include` localizados en un directorio inusual tal como `/usr/local/include`, se podría añadir la opción `-I` como sigue:

```
$ cc -c -I/usr/local/include hello.c
```

Pueden añadirse más de una opción `-I` en la línea de comandos, y los directorios serán recorridos en el orden dado. Por ejemplo, si se ejecuta el comando:

```
$ cc -c -I/usr/local/include -I/opt/include gestion.c
```

Y el programa fuente (`gestion.c`) contiene la directiva `#include "file.h"`, entonces muchos compiladores (no-GNU) de UNIX procesarán la directiva buscando, primero, en el directorio actual, después en todos los directorios dados por la opción `-I`

y finalmente en el directorio `/usr/include`. Los mismos compiladores (no-GNU) de UNIX procesarán la directiva de C `#include <file.h>` de la misma forma, excepto que no buscan en el directorio actual. Sin embargo, el compilador de GNU extiende algo la opción `-I` como sigue:

- `-I-`, los directorios que preceden a una opción `-I-` son recorridos solamente para las directivas de la forma `#include "file.h"`.
- Los directorios proporcionados con las opciones `-I que` siguen a una opción `-I-` se recorren para las dos formas `#include "file.h"` y `#include <file.h>`.
- Si no aparece ninguna opción `-I-` en la línea de comandos, entonces el comportamiento es el mismo que para los compiladores no GNU de C.

Un ejemplo de todo esto es el comando de compilación siguiente:

```
$ gcc -c -I/usr/tipodato/include -I- -I/opt/oracle/include convo.c
```

La ejecución del comando del ejemplo permite a la directiva del preprocesador de C `#include "pila.h"` incluir el archivo `/usr/tipodato/include/pila.h`. Esta otra directiva `#include <sqlca.h>`, recorre los directorios que siguen a la opción `-I-`, `entonces` incluiría al fichero `/opt/oracle/include/sqlca.h`. Esto ocurre porque la forma `<file.h> no es buscada en los` directorios que preceden a la opcion `-I-`.

Opción -E (expandir)

Esta opción es relativamente estándar entre los compiladores de C de UNIX. Permite modificar la línea de comandos para hacer que el compilador envíe el código preprocesado en C a la salida estándar sin llegar a compilar el código. Esto es útil para controlar las directivas de preprocesamiento y las macros de C. La salida de lo que será compilado puede ser redirigida a otro archivo para que después se examine con un editor.

```
$ cc -c -E hello.c > cpp.out
```

En el ejemplo anterior, la opción `-E` hace que los archivos `include y el` programa sean preprocesados y redirigidos hacia el archivo `cpp.out`. Después, se puede examinar el archivo `cpp.out` con un editor para determinar como será el código final en C. Esto es útil especialmente cuando se trata de depurar el efecto de macros en C que en ocasiones provocan errores de compilación difíciles de diagnosticar.

Opción -O (optimizar)

Esta opción no es estándar entre los compiladores. Algunos compiladores requieren que un argumento siga a la `-O`, otros no y otros aceptarán opcionalmente un argumento. FreeBSD acepta lo siguiente:

- `-O y -O1` especifican optimización de nivel 1.
- `-O2` especifica optimización de nivel 2 (optimización mayor).
- `-O3` especifica optimización de nivel 3 (más que `-O2`).
- `-O0` especifica sin optimización.

Para el compilador de GNU, estas opciones pueden estar repetidas, y la última es la que establece el nivel final de optimización. Por ejemplo:

```
$ gcc -c -O3 -O0 helipse.c
```

compila sin optimizar porque al final aparece `-O0`.

En contraste con el compilador GNU, el compilador de HP soporta las siguientes opciones de para niveles crecientes de optimización:

- Optimización por defecto `+O0`
- Nivel 1 de optimización `+O1`
- Nivel 2 de optimización `+O2` (equivale a `-O`, sin argumentos, de FREEBSD)
- Nivel 3 de optimización `+O3`
- Nivel 4 de optimización `+O4`

El compilador de IBM AIX 4.3 soporta las opciones `-O`, `-O2` y `-O3` para niveles crecientes de optimización. Todo ello acentúa la necesidad de revisar para cada sistema las opciones del compilador en la página de cc del manual correspondiente.

La optimización analiza el código compilado, código objeto, para aumentar la eficiencia en la ejecución de las instrucciones. Cuanto mayor es el nivel de optimización mejor es el código ejecutable producido, por contra, mayor es el tiempo de compilación.

A tener en cuenta

La mayoría de los compiladores de C no aceptan las dos opciones -g (depuración) y -O (optimización) al mismo tiempo. El compilador de GNU tolera -g y optimización de primer nivel (-O), pero esto puede producir resultados sorprendentes en el depurador.