yo

Programación avanzada de UNIX

con Linux

1

Empezando

Ste capítulo se muestra cómo realizar los pasos básicos necesarios para crear una

C o C ++ programa de Linux. En particular, este capítulo le muestra cómo crear y modificar código fuente C y C ++, compilar ese código y depurar el resultado. Si eres ya acostumbrado a la programación bajo Linux, puede saltar adelante al Capítulo 2, "Escribir buen software GNU / Linux", preste mucha atención a la Sección 2.3, "Redacción y Uso de bibliotecas ", para obtener información sobre vínculos estáticos versus dinámicos que podría no saberlo.

A lo largo de este libro, vamos a suponer que usted está familiarizado con el C o C + + pro- los idiomas de programación y las funciones más comunes en la biblioteca C estándar. ejemplos de código fuente en este libro están en C, excepto cuando se demuestra una característica o complicación de la programación en C ++. También asumimos que usted sabe cómo realizar operaciones básicas en el shell de comandos de Linux, como crear directorios y Copiando documentos. Debido a que muchos programadores de Linux comenzaron a programar en el Entorno de Windows, ocasionalmente señalaremos similitudes y contrastes entre Windows y Linux.

Página 4

1.1 Edición con Emacs

Un *editor* es el programa que se utiliza para editar código fuente. Hay muchos editores diferentes disponible para Linux, pero el editor más popular y completo es probablemente GNU Emacs.

> **Acerca de Emacs**
> Emacs es mucho más que un editor. Es un programa increíblemente poderoso, tanto que al CodeSourcery, es cariñosamente conocido como el programa One True, o simplemente el OTP para abreviar. Puedes leer y enviar correo electrónico desde dentro de Emacs, y puede personalizar y extender Emacs de maneras demasiado numerosas para discutir aquí. ¡Incluso puedes navegar por la web desde Emacs!

Si está familiarizado con otro editor, puede usarlo en su lugar. Nada en el El resto de este libro depende de usar Emacs. Si aún no tienes un sistema Linux favorito editor, entonces usted debe seguir junto con el mini-tutorial dado aquí.

Si te gusta Emacs y quieres aprender acerca de sus características avanzadas, puedes considerar leyendo uno de los muchos libros de Emacs disponibles. Una excelente tutorial, *Aprender GNU Emacs,* está escrito por Debra Cameron, Bill Rosenblatt, y Eric S. Raymond (O'Reilly, 1996).

1.1.1 Apertura de un archivo de origen C o C ++

Puede iniciar Emacs escribiendo emacs en su ventana de terminal y presionar el
Cuando se ha iniciado Emacs, puede usar los menús en la parte superior para
un nuevo archivo de origen. Haga clic en el menú Archivos, elija Abrir archivos y, a continuación, escriba el nombre de
el archivo que desea abrir en el "minibúfer" en la parte inferior de la pantalla. [1] Si
que desee crear un archivo fuente de C, utilice un nombre de archivo que termina en .c o .h. Si quieres
crear un archivo fuente en C ++, utilice un nombre de archivo que termina en .cpp, .hpp, .cxx, .hxx, .C o .H.
Cuando el archivo está abierto, puede escribir como lo haría en cualquier procesador de texto ordinario
Para guardar el archivo, elija la entrada Save Buffer en el menú Files.
terminó de usar Emacs, puede elegir la opción Exit Emacs en la carpeta
menú.

Si no te gusta apuntar y hacer clic, puedes utilizar los atajos de teclado para
abrir archivos, guardar archivos y salida Emacs.To abrir un archivo, tipo Cx Cf. (El Cx significa
mantenga presionada la tecla Control y pulse la tecla x.) Para guardar un archivo, el tipo Cx Cs .Para
Emacs salida, simplemente escribe Cx Cc. Si quieres conocer mejor a Emacs,
seleccione la entrada Tutorial de Emacs en el menú Ayuda. El tutorial le proporciona
un montón de consejos sobre cómo utilizar Emacs eficazmente.

1. Si no se está ejecutando en un sistema X Window, tendrá que presionar F10 para
menús

1.1.2 Formato automático

Si estás acostumbrado a la programación en un *entorno de desarrollo integrado (IDE),*
también estarás acostumbrado a que el editor te ayude a formatear tu código. Emacs puede
proporcionan el mismo tipo de funcionalidad. Si abre un archivo de origen C o C ++, Emacs
automáticamente se da cuenta de que el archivo contiene código fuente, no sólo texto ordinario. Si
pulsa la tecla Tab en una línea en blanco, Emacs mueve el cursor a una
punto sangrado Si pulsa la tecla Tab en una línea que ya contiene algún texto, Emacs
marca el texto. Por ejemplo, supongamos que ha escrito lo siguiente:

```
int main ()
{
printf ("Hola, mundo \ n");
}
```

Si pulsa la tecla Tab en la línea con la llamada a printf, Emacs nuevo formato a su
código para parecerse a esto:

```
int main ()
{
  printf ("Hola, mundo \ n");
}
```

Observe cómo la línea se ha sangrado apropiadamente.

A medida que utilice Emacs más, verá cómo puede ayudarle a realizar todo tipo de
complicadas tareas de formateo. Si eres ambicioso, puedes programar Emacs para que realice
literalmente cualquier tipo de formato automático que te puedas imaginar. La gente ha usado este
facilidad para implementar los modos de Emacs para editar casi todo tipo de documentos,
para implementar juegos [2] e implementar interfaces de bases de datos.

1.1.3 Resaltado de sintaxis

Además de formatear su código, Emacs puede facilitar la lectura de C y C ++
código por color diferentes elementos de sintaxis. Por ejemplo, Emacs puede convertir palabras clave
un color, una función de tipos como int otro color, y los comentarios de otro color.
El uso del color hace que sea mucho más fácil detectar algunos errores de sintaxis comunes.

La forma más fácil de encender la coloración es editar el archivo ~ / .emacs e inserte el
siguiente cadena:

```
(global-font-lock-mode t)
```

Guarde el archivo, salga de Emacs y reinicie. Ahora abra un archivo fuente C o C ++ y disfrute!

Usted puede haber notado que la cadena ha insertado en su .emacs parece
código de la programación LISP de language.That porque *es* código LISP! Gran parte de
Emacs está escrito en LISP. Puede agregar funcionalidad a Emacs escribiendo más
Código LISP.

2.Try ejecutar el comando Mx Dunnet si desea reproducir un texto antiguo juego de aventura.

Página 6

1.2 Compilación con GCC

Un *compilador* convierte el código fuente legible por humanos en un código objeto legible por máquina que Los compiladores de elección en los sistemas Linux son todos parte de la GNU Colección del compilador, generalmente conocida como GCC. [3] GCC también incluye compiladores para C, C ++, Java, Objective-C, Fortran y Chill.Este libro se centra principalmente en C y C ++ programación.

Supongamos que tiene un proyecto como el del Listado 1.2 con una fuente C ++ presentar (reciprocal.cpp) y un archivo fuente de C (main.c) como en el Listado 1.1.These dos archivos se supone que se compilan y luego se unen para producir un programa llamado recíproco. [4] Este programa calcular el recíproco de un número entero.

Añadir 1,1 ***(main.c)*** **C fuente *main.c* file-**

```
#include <stdio.h>
#include "reciprocal.hpp"

int main (int argc, char ** argv)
{
  int i;

  i = atoi (argv [1]);
  printf ("El recíproco de% d es% g \ n", i, recíproco (i));
  return 0;
}
```

Añadir 1,2 ***(reciprocal.cpp)*** **C ++ fuente *reciprocal.cpp* file-**

```
#include <cassert>
#include "reciprocal.hpp"

doble recíproco (int i) {
  // Yo debería ser no-cero.
  afirmar (i = 0);
  return 1.0 / i;
}
```

3. Para obtener más información acerca de GCC, visite http://gcc.gnu.org.

4. En Windows, archivos ejecutables por lo general tienen nombres que terminan en .exe. Los programas de Linux, Por otra parte, por lo general no tienen extensión. Así, el equivalente de Windows de este programa sería probablemente ser llamado reciprocal.exe; la versión para Linux es sólo recíproca llanura.

Página 7

También hay un archivo de cabecera llamada reciprocal.hpp (ver Listado 1.3).

Añadir 1,3 *reciprocal.hpp* Archivo- *(reciprocal.hpp)* Cabecera

```
#ifdef __cplusplus
extern "C" {
#terminara si

externo doble recíproco (int i);

#ifdef __cplusplus
}
#terminara si
```

El primer paso es convertir el código fuente C y C ++ en código objeto.

1.2.1 Compilación de un único archivo de origen

El nombre del compilador C es gcc. Para compilar un archivo fuente C, se utiliza el -c opción. Así, por ejemplo, al introducir esto en el símbolo del sistema compila el main.c archivo fuente:

```
% gcc -c main.c
```

El archivo de objeto resultante se denomina main.o.

El compilador C ++ se llama g ++. Su funcionamiento es muy similar al de las CGC; compilar reciprocal.cpp se lleva a cabo introduciendo el siguiente:

```
% g ++ -c reciprocal.cpp
```

La opción -c le dice g ++ para compilar el programa en un archivo único objeto; sin él, g ++ intentará vincular el programa para producir un ejecutable. Después de haber escrito esto comando, tendrá un archivo de objeto llamado reciprocal.o.

Es probable que necesite un par de otras opciones para construir cualquier programa razonablemente grande. La opción -I se utiliza contar GCC dónde buscar los archivos de cabecera. De forma predeterminada, GCC busca en el directorio actual y en los directorios en los que los encabezados se instalan las bibliotecas. Si necesita incluir archivos de encabezado desde otro lugar, necesitará la opción -I. Por ejemplo, supongamos que su proyecto tiene un directorio llamado src, para archivos de código fuente, y otro llamado incluye .tiene compile los reciprocal.cpp como esto para indicar que g ++ debe utilizar el directorio ../include además de encontrar reciprocal.hpp:

```
% g ++ -c -I ../include reciprocal.cpp
```


A veces, deseará definir macros en la línea de comandos. Por ejemplo, en código de producción, no desea que la sobrecarga de la verificación de aserción reciprocal.cpp; que sólo está allí para ayudarle a depurar el programa. Usted apaga la comprobación mediante la definición de la macro .tiene NDEBUG podría agregar un #define explícita a reciprocal.cpp, pero eso requeriría cambiar la propia fuente. Es más fácil simplemente definir NDEBUG en la línea de comandos, así:

```
% g ++ -c -D NDEBUG reciprocal.cpp
```

Si hubiera querido definir NDEBUG a algún valor en particular, que podría haber hecho algo como esto:

```
% g ++ -c -D NDEBUG = 3 reciprocal.cpp
```

Si realmente está creando código de producción, probablemente desee que GCC optimice el código para que se ejecute lo más rápido possible.You puede hacer esto mediante el uso de la -O2 opción de línea de comandos. (GCC tiene varios niveles diferentes de optimización, el segundo nivel es apropiado para la mayoría de los programas.) Por ejemplo, las siguientes compilaciones reciprocal.cpp con la optimización de encendido:

```
% g ++ -c -O2 reciprocal.cpp
```

Tenga en cuenta que la compilación con la optimización puede dificultar su programa depurar con un depurador (consulte la Sección 1.4, "Depuración con GDB"). Además, en ciertas instancias, la compilación con optimización puede descubrir errores en su programa que no se manifiestan previamente.

Puede pasar un montón de otras opciones para gcc y g ++ .La mejor manera de conseguir una completa lista es ver la documentación en línea. Puede hacerlo escribiendo lo siguiente en el símbolo del sistema:

```
% info gcc
```

1.2.2 Enlazar archivos de objetos

Ahora que ha compilado main.c y utilities.cpp, tendrá que vincular them.You Siempre debe usar g ++ para vincular un programa que contiene el código de C ++, aunque también con- código C. Si su programa contiene código sólo C, se debe utilizar en lugar de gcc. Debido a que este programa contiene tanto C como C ++, debe utilizar g ++, así:

```
% g ++ -o recíproco principal.o recíproco.o
```

La opción -o le da el nombre del archivo a generar como salida de la etapa de enlace. Ahora se puede ejecutar recíproco de esta manera:

```
% ./reciprocal 7
El recíproco de 7 es 0,142857
```

Como se puede ver, g ++ ha vinculado automáticamente en la biblioteca estándar de C en tiempo de ejecución con- TaiNing la implementación de printf. Si hubieras tenido que enlazar en otra biblioteca (como un conjunto de herramientas de interfaz gráfica de usuario), habría especificado la biblioteca con

la opción -l. En Linux, los nombres de biblioteca casi siempre comienzan con lib. Por ejemplo, la biblioteca (PAM) módulo de autenticación conectable se llama libpam.a enlace en .Para libpam.a, se utiliza un comando como el siguiente:

```
% g ++ -o recíproco main.o reciprocal.o -lpam
```

El compilador agrega automáticamente el prefijo y el sufijo lib .a.

Al igual que con los archivos de encabezado, el vinculador busca bibliotecas en algunos lugares estándar, incluyendo el / lib y / usr / lib que contiene las bibliotecas del sistema estándar. Si tu quiere que el enlazador para buscar otros directorios, así, usted debe utilizar la opción -L, que es el paralelo de la opción -I discutió earlier.You puede usar esta línea para instruir el enlazador para buscar bibliotecas en el directorio / usr / local / lib / pam antes de buscar en los lugares habituales:

```
% g ++ -o recíproco main.o reciprocal.o -L / usr / local / lib / pam -lpam
```

A pesar de que no tiene que utilizar la opción -I para obtener el preprocesador para buscar la directorio actual, usted tiene que utilizar la opción -L para obtener el enlazador para buscar la directorio actual. En particular, podría utilizar lo siguiente para instruir al vinculador a encontrar la biblioteca de prueba en el directorio actual:

```
% gcc -o app app.o -L. la prueba
```

1.3 Automatización del proceso con GNU Make

Si está acostumbrado a programar para el sistema operativo Windows, está mente acostumbrados a trabajar con un entorno de desarrollo integrado (IDE). agregar archivos de fuentes a su proyecto y, a continuación, el IDE genera su proyecto automáticamente. Aunque los IDE están disponibles para Linux, este libro no los discute. En su lugar, este muestra cómo usar GNU Make para recompilar automáticamente su código, que es lo que la mayoría de los programadores de Linux realmente hacen.

La idea básica detrás de maquillaje es decir simple.You hacer lo que *apunta a* que desea construir y luego dar *reglas* que explican cómo construir them.You también especifican *dependencias* que indicar cuándo debe reconstruirse un objetivo en particular.

En nuestro proyecto recíproco muestra, hay tres objetivos obvios: reciprocal.o, main.o, y el número inverso itself.You ya tienen reglas en mente para la construcción de estos en la forma de las líneas de comando dadas anteriormente. Las dependencias requieren un poco de pensamiento. Claramente, recíproca depende de reciprocal.o y porque main.o no puede vincular el programa completo hasta que haya creado cada uno de los archivos de objeto. los archivos de objeto deben ser reconstruidos siempre que cambien los archivos de origen correspondientes.

una vuelta de tuerca más en que un cambio en reciprocal.hpp también debe causar tanto de la objeto a ser reconstruido porque ambos archivos de origen incluyen ese archivo de encabezado.

Además de los objetivos obvios, debe haber siempre un objetivo target.This limpia elimina todos los archivos y programas de objetos generados para que puedas empezar de nuevo. La regla para este destino utiliza el comando rm para eliminar los archivos.

Página 10

Puede transmitir toda esa información para hacer poniendo la información en un archivo llamado Makefile. Esto es lo que contiene Makefile:

```
recíproco
        g ++ $ (CFLAGS) -o recíproco principal.o recíproco.o

main.o: main.c reciprocal.hpp
        gcc $ (CFLAGS) -c main.c

reciprocal.o: reciprocal.cpp reciprocal.hpp
        g ++ $ (CFLAGS) -c reciprocal.cpp

limpiar:
        rm -f * .o recíproco
```

Puede ver que los objetivos se enumeran a la izquierda, seguidos de dos puntos y, a continuación, La regla para construir ese objetivo está en la siguiente línea. (No haga caso de los $ CFLAGS) bits (por el momento.) La línea con la regla de que debe comenzar con un carácter de tabulación, o hacer se confundirá. Si edita el Makefile en Emacs, Emacs le ayudará con la formato.

Si elimina los archivos de objeto que ya ha creado y simplemente escriba

```
% hacer
```

en la línea de comandos, verá lo siguiente:

```
% hacer
gcc -c main.c
g ++ -c reciprocal.cpp
g ++ -o recíproco principal.o recíproco.o
```

Se puede ver que hacen ha construido de forma automática los archivos de objetos y luego ellos vinculados. Si ahora cambia main.c de alguna manera trivial y ejecute make de nuevo, verá el siguiendo:

```
% hacer
gcc -c main.c
g ++ -o recíproco principal.o recíproco.o
```

Se puede ver que hacen sabía a reconstruir main.o y para volver a vincular el programa, pero no se molestó en volver a compilar reciprocal.cpp porque ninguna de las dependencias de reciprocal.o había cambiado.

El $ (CFLAGS) es una marca variable.You puede definir esta variable, ya sea en el Makefile en sí o en la línea de comandos. GNU make va a sustituir el valor de la cuando ejecuta la regla. Así, por ejemplo, para recompilar con optimización habilitado, usted haría esto:

```
% hacer limpia
rm -f * .o recíproco
% make CFLAGS = -O2
gcc -O2 -c main.c
g ++ -O2 -c reciprocal.cpp
g ++ -O2 -o recíproco principal.o recíproco.o
```

Página 11

Tenga en cuenta que el indicador de O2 se insertó en lugar de $ (CFLAGS) en las reglas.

En esta sección, usted ha visto sólo las capacidades más básicas de maquillaje Usted puede encontrar más escribiendo esto:

```
% info hacer
```

En ese manual, encontrará información sobre cómo hacer el mantenimiento de un Makefile más fácil, cómo reducir el número de reglas que usted necesita para escribir, y cómo automáticamente calcular dependencies.You también se puede encontrar más información en *GNU, Autoconf, Automake y Libtool* por Gary V.Vaughan, Ben Elliston, Tom Tromey, y Ian Lance Taylor (New Riders Publishing, 2000).

1.4 Depuración con depurador GNU (GDB)

El *depurador* es el programa que se utiliza para averiguar por qué su programa no es tamiento de la manera que crees que debería. Vas a estar haciendo esto mucho. [5] El depurador de GNU (GDB) es el depurador utilizado por la mayoría de los programadores de Linux. Puede utilizar GDB al paso a través de su código, establecer puntos de interrupción y examinar el valor de variables locales.

1.4.1 Compilación con información de depuración

Para usar GDB, tendrá que compilar con la información de depuración habilitada. Hacer esto por añadiendo el parámetro -g en la línea de comando de compilación. Si estás utilizando un Makefile se ha descrito anteriormente, sólo se puede establecer CFLAGS igual a -g cuando se ejecuta hacer, como se muestra aquí:

```
% make CFLAGS = -g
gcc -g -c main.c
g ++ -g -c reciprocal.cpp
g ++ -g -o recíproco principal.o recíproco.o
```

Cuando se compila con -g, el compilador incluye información adicional en los archivos de objetos y los ejecutables. El depurador utiliza esta información para averiguar qué direcciones corresponder a qué líneas en las que los archivos de origen, cómo imprimir las variables locales, y así adelante.

1.4.2 Ejecución de GDB

Puede iniciar el BGF escribiendo:

```
% gdb recíproco
```

Cuando se pone en marcha el BGF, debería ver el símbolo del BGF:

```
(gdb)
```

5. ... a menos que sus programas funcionen siempre la primera vez.

El primer paso es ejecutar su programa dentro del depurador. Solo tienes que introducir la instrucción ejecutada y cualquier argumento del programa. Intente ejecutar el programa sin argumentos, como esta:

```
(gdb) ejecutar
Programa de inicio: recíproco

Programa recibido señal SIGSEGV, Error de segmentación.
__strtol_internal (nptr = 0x0, endptr = 0x0, base = 10, grupo = 0)
en strtol.c: 287
287 strtol.c: No existe tal archivo o directorio.
(gdb)
```

El problema es que no existe un código de comprobación de errores en el programa principal .El espera un argumento, pero en este caso el programa se ha ejecutado sin arguments.The SIGSEGV mensaje indica un bloqueo del programa. GDB sabe que el accidente real ocurrió en un función llamada función .Es __strtol_internal está en la biblioteca estándar, y el

fuente no está instalado, lo que explica el mensaje "Sin dicho archivo o directorio". ver la pantalla usando el comando donde:

```
(gdb) donde
# 0 __strtol_internal (nptr = 0x0, endptr = 0x0, base = 10, grupo = 0)
    en strtol.c: 287
# 1 0x40096fb6 en atoi (nptr = 0x0) en ../stdlib/stdlib.h:251
# 2 0x804863e en main (argc = 1, argv = 0xbffff5e4) en main.c: 8
```

Se puede ver en esta pantalla principal que se llama la función atoi con un puntero NULL, que es la fuente del problema.

Se puede subir dos niveles en la pila hasta que llegue principal usando el hasta mando:

```
(gdb) hasta 2
# 2 0x804863e en main (argc = 1, argv = 0xbffff5e4) en main.c: 8
8           i = atoi (argv [1]);
```

Tenga en cuenta que el BGF es capaz de encontrar la fuente de main.c, y muestra la línea donde se ha producido la llamada de función errónea. Puede ver el valor de las variables comando de impresión:

```
(gdb) impresiones argv [1]
$ 2 = 0x0
```

Esto confirma que el problema es de hecho un puntero NULL pasado a atoi.

Se puede establecer un punto de interrupción utilizando el comando break:

```
(gdb) break main
Punto de interrupción 1 en 0x804862e: archivo main.c, línea 8.
```

Página 13

Este comando establece un punto de interrupción en la primera línea de la principal. ₒ Ahora intenta volver a ejecutar la programa con un argumento, como este:

```
(gdb) ejecutar 7
Programa de inicio: recíproco 7

Punto de ruptura 1, main (argc = 2, argv = 0xbffff5e4) en main.c: 8
8           i = atoi (argv [1]);
```

Puede ver que el depurador se ha detenido en el punto de interrupción.

Puede pasar por encima de la llamada a atoi usando el siguiente comando:

```
(gdb) siguiente
9           printf ("El recíproco de% d es% g \ n", i, recíproco (i));
```

Si quieres ver lo que está pasando en el interior de reciprocidad, utilice el comando de paso como esto:

```
(gdb) paso
recíproco (i = 7) en reciprocal.cpp: 6
6           afirmar (i = 0);
```

Ahora se encuentra en el cuerpo de la función recíproca.

Puede que le resulte más conveniente para ejecutar gdb desde Emacs en lugar de utilizar gdb directamente desde la línea de comandos. Utilice el comando Mx gdb para poner en marcha el BGF en una Ventana de Emacs. Si se detiene en un punto de interrupción, Emacs automáticamente archivo fuente apropiado. Es más fácil averiguar qué está pasando cuando estás mirando todo el archivo en lugar de sólo una línea de texto.

1.5 Encontrar más información

Casi todas las distribuciones de Linux vienen con mucha documentación útil. podría aprender la mayor parte de lo que hablaremos en este libro leyendo documentación en su distribución de Linux (aunque probablemente le llevaría mucho más tiempo). El doc- no siempre está bien organizada, aunque la parte difícil es encontrar lo que necesitar. La documentación también está a veces anticuada, así que tome todo lo que lee con un grano de sal. Si el sistema no se comporta de la manera que una *página del manual* (páginas del manual) dice que debería, por ejemplo, es posible que la página de manual esté obsoleta.

Para ayudarle a navegar, aquí están las fuentes de información más útiles sobre programación avanzada de Linux.

6. Algunas personas han comentado que romper diciendo principal es un poco gracioso porque por lo general usted quiere hacer esto sólo cuando principal ya está roto.

Página 14

1.5.1 Páginas de Man

Las distribuciones de Linux incluyen páginas de manual para la mayoría de los comandos estándar, llamadas al sistema y funciones de biblioteca estándar. Las páginas de manual se dividen en secciones numeradas; para programmers, los más importantes son estos:

(1) Comandos del usuario

(2) Llamadas del sistema

(3) Funciones estándar de la biblioteca

(8) Comandos de sistema / administrativos

Los números indican las secciones de la página de manual. Las páginas de manual de Linux vienen instaladas sistema; utilice el comando man para acceder them.To buscar una página de manual, basta con invocar *Nombre del* hombre, donde *nombre* es un nombre de comando o función. En algunos casos, el mismo nombre ocurre en más de una sección; puede especificar la sección explícitamente colocando la número de sección antes del nombre. Por ejemplo, si escribe lo siguiente, obtendrá la página del manual para el comando del sueño (en la sección 1 de las páginas de manual de Linux):

el hombre duerme

Para ver la página del manual para la función de la biblioteca del sueño, utilice este comando:

% hombre 3 sueño

Cada página de manual incluye un resumen de una línea del comando o la función.
whatis comando muestra *el nombre* todas las páginas del manual (en todas las secciones) para un comando o función de *nombre* coincidente. Si no está seguro de qué comando o función desea, se puede realizar una búsqueda por palabras clave en las líneas de resumen, el uso *de palabras clave* man -k.

Man páginas incluyen una gran cantidad de información muy útil y debe ser el primer lugar la página de manual de un comando describe las opciones de línea de comandos y argumentos, entrada y salida, códigos de error, configuración y similares. La página de manual para una llamada de sistema o una función de biblioteca describe parámetros y valores de retorno, lista el error códigos y efectos secundarios, y especifica qué archivo incluir para utilizar si llama a la función.

1.5.2 Información

El sistema de documentación de Info contiene documentación más detallada para muchos componentes del sistema GNU / Linux, además de varios otros programas. Las páginas de información son documentos de hipertexto, similares a las páginas web. Para iniciar el navegador de Tipo de información en una cáscara de window.You'll se presentará con un menú de documentos Info instalado en su sistema. (Pulse Control + H para mostrar las teclas para navegar por una Info documento.)

Entre los documentos de información más útiles se encuentran:

- -La gcc compilador gcc
- libc -La biblioteca de C de GNU, incluyendo muchas llamadas al sistema
- depurador GDB -La GNU

Página 15

- S editor de texto Emacs -La Emacs
- S -El sistema de información Info propio

Casi todas las herramientas de programación estándar de Linux (incluyendo ld, el enlazador; como, la ensamblador; y gprof, el perfilador) vienen con información útil pages.You puede saltar directamente a un determinado documento Info especificando el nombre de la página en la línea de comandos:

```
% info libc
```

Si realiza la mayor parte de su programación en Emacs, puede acceder a la información incorporada navegador tecleando Mx info o Ch i.

1.5.3 Archivos de encabezado

Usted puede aprender mucho acerca de las funciones del sistema que están disponibles y cómo usar ellos observando el sistema de cabecera files.These residen en / usr / include y / usr / include / sys. Si obtiene errores de compilación al utilizar una llamada de sistema, ejemplo, eche un vistazo en el archivo de encabezado correspondiente para verificar que la función La firma es la misma que la lista de la página de manual.

En sistemas Linux, muchos de los detalles básicos de cómo funciona el sistema de trabajo son reflejado en los archivos de cabecera en los directorios / usr / include / bits, / usr / include / asm, y / usr / include / Linux. Por ejemplo, los valores numéricos de las señales (descritos en la Sección 3.3, “Señales”, en el capítulo 3, “Procesos”) se definen en /usr/include/bits/signum.h. Estos archivos de cabecera hacen una buena lectura para mentes inquisitivas. No los incluya directamente en sus programas; utilice siempre los archivos de cabecera en / usr / include o como mencionado en la página de manual de la función que está utilizando.

1.5.4 Código fuente

Este es Open Source, ¿verdad? El árbitro final de cómo funciona el sistema es el sistema código fuente en sí, y por suerte para los programadores de Linux, ese código fuente está disponible libremente. poder. Lo más probable es que su distribución de Linux incluya código fuente completo para todo el sistema. y todos los programas incluidos en él; si no, usted tiene derecho bajo los términos de la GNU General Public License para solicitarlo al distribuidor. (El código fuente puede no estar instalado en su disco, sin embargo. Consulte la documentación de su distribución para instrucciones para instalarlo.)

El código fuente para el propio núcleo de Linux normalmente se almacena en / usr / src / linux. Si este libro deja sed de detalles sobre cómo los procesos, la memoria compartida y los sistemas los dispositivos de TI funcionan, siempre se puede aprender directamente del código fuente. La mayoría de las funciones del sistema descritas en este libro se implementan en la biblioteca C de GNU; consulte la documentación de su distribución para ver la ubicación del código fuente de la biblioteca C.