Práctica 0: Introducción a GNU-Linux y a C

Sistemas Operativos

Objetivo: Familiarizarse con el entorno GNU/Linux, manejo de bibliotecas en GNU/Linux y escribir y ejecutar un programa muy simple en C.

Para ello vamos a ir probando los ejemplos de las transparencias de *Introducción a GNU/Linux y C for dummies* … Sigue las instrucciones y después tendrás que contestar a unas preguntas relacionadas con lo que has probado.

1. Editar, compilar, enlazar y ejecutar un programa muy básico:

a) Utilizando la aplicación *gedit* desde un terminal, edita un programa básico, llamado *basic1.c*, que escriba un mensaje en pantalla. Recuerda que necesitas la función *printf* y por lo tanto necesitas incluir la librería *stdio.h*.

$ gedit	#include <stdio.h>
	#include <stdlib.h>
	int main()
	{
	printf("Bienvenidos a SO\n")
	exit(0);
	}

b) Compila y enlaza tu programa usando el comando *gcc* con la opción *-o.* Recuerda que el nombre del ejecutable no tiene por qué coincidir con el del archivo fuente.

```
$ gcc –o nombre_del_ejecutable mi_fichero.c
```

Comprueba que se ha creado el ejecutable.

```
$ls –la
```

c) Ejecuta tu programa

```
$ ./nombre_del_ejecutable
Bienvenidos a SO
$
```

- ¿Es necesario incluir también la librería *stdlib.h* como aparece en las transparencias? ¿Por qué?.
- ¿Por qué no tienes un archivo basic1.o?.
- ¿Qué sucede si en lugar de usar la opción *-o* con *gcc* usas *-c*?.

2. Manejo de bibliotecas no estándar:

a) Editar dos programas, primero.c y segundo.c, que contienen cada uno una función.

```
#include <stdio.h>
void primero(int arg)
{
printf("Estudiantes matriculados en SO %d\n", arg);
}
//*----------------*//
#include <stdio.h>
void segundo(char *arg)
{
printf("Bienvenidos a SO %s\n", arg);
}
```

b) Compíla ambos programas con *gcc -c* en lugar de *-o* para que no los enlace ni genere un ejecutable (queremos que estas dos funciones formen parte de nuestra biblioteca).

```
$gcc –c primero.c segundo.c
```

Comprueba que se han generado los dos archivos objeto correspondientes.

```
$ls *.o
primero.o segundo.o
```

c) Crea un archivo cabecera, *lib.h*, donde esté la declaración de estas funciones que van a formar parte de tu biblioteca, para poder enlazarla desde los programas que hagan uso de estas funciones.

```
/*
Esta es la cabezera lib.h que declara las funciones
primero y segundo
*/
void segundo(char *);
void primero(int);
```

d) Crea la biblioteca, *mi_biblio.a*, que estará compuesta por los dos ficheros objeto de las funciones primero y segundo.

```
$ ar crv mi_biblio.a primero.o segundo.o
```

e) Ahora escribe un programa que utilice la función del archivo segundo.c. Tienes que incluir tu archivo cabecera, *lib.h*, con las declaraciones de las funciones de tu biblioteca.

```
#include <stdio.h>
#include “lib.h”
int main()
{
segundo(“estudiantes”);
exit (0);
}
```

f) Genera un ejecutable usando *gcc -o* y enlazando tu programa anterior con tu biblioteca y ejecútalo:.

```
$ gcc –o programa programa.c mi_biblio.a
$ ./programa
Bienvenidos a SO estudiantes
$
```

h) Ahora escribe un programa que utilice la función del archivo primero.c, inventándote el número de estudiantes matriculados/as en S0 (NO está en las transparencias).

Contesta a las siguientes preguntas:

- ¿Qué operaciones tienes que realizar si quieres cambiar algo en la función de primero?
- ¿Y si quisieras añadir una tercera función a tu librería?.

3. Manejo de punteros y archivos en C

a) Prueba las diferentes versiones de la función *intercambia* de las transparencias, hasta dar con la que sí funciona.

b) Añade la función *intercambia* a tu librería y utilízala para escribir el siguiente programa. Ejecútalo y comprueba su funcionamiento con dos pares de valores diferentes para *‘a’* y *‘b’*. Si quieres ahorrarte la compilación cada vez que les des nuevos valores, busca la función en C que te permite leer de teclado para dar valores a *‘a’* y *‘b’*.

```
int main()
{
int a = 5;
int b = 0;

printf(“Antes del cambio %d\n”, a);
intercambia (&a, &b)
printf(“Despues del cambio %d\n”, a);
exit(0);
}
```

c) Descarga desde el campus virtual el código del_archivo de manejo básico de ficheros y añádele comentarios explicando qué hace en cada línea.

d) Genera un ejecutable y comprueba su funcionamiento.

e) Modifícalo para que lea del teclado el nombre del archivo de entrada y lo copie junto al número de bytes que contiene en otro de salida llamado i*nfo.txt*.

Ayuda: scanf(“formato”, &archivo1)

f) Ejecútalo y comprueba su funcionamiento con el comando *cat*.

- ¿Es necesario que exista un fichero copia.txt antes de ejecutar el programa?
- ¿Qué sucede con el contenido de copia.txt si el fichero ya existía?

4. Uso del debugger gdb

a) Descarga del Campus Virtual el programa que compila pero no funciona correctamente y compílalo con la opción que permite usar el debugger:

```
$ gcc -g programa.c
$ gdb
```

Ejecuta el programa:

```
(gdb) file a.out
(gdb) run
```

b) Imprime los valores de las variables que están dando problemas:

```
(gdb) print a[0]
(gdb) print c[0]
(gdb) print i
(gdb) print j
```

c) Añade un breakpoint en el segundo *for* del código:

```
(gdb) break 28
(gdb) run
Breakpoint 1 at 0x0400620 file programa.c line 28
28
for(i=4; i>0; i++)
```

d) Ejecuta paso a paso:

```
(gdb) step
29  a[i-1] = c[j];
```

e) Imprime los valores de I y j de la línea 29, la que da el fallo (tienes que volver a ejecutar el bucle paso a paso hasta ésa línea).

¿Has encontrado el fallo del programa?