503208 Programación I 2021–I

Laboratorio I — Introducción

Aníbal Tolosa

Temas

- Linux:
  - La terminal.
  - Comandos básicos.
  - Navegar en directorios.
- C:
  - Crear un código fuente en C.
  - Compilar.
- Variables:
  - Declaración y asignación.
- Entrada y salida:
  - `printf()`.
  - `scanf()`.

1. Linux

1.1. La terminal

La terminal es la interfaz de texto de Linux. Esta se usa mediante comandos y argumentos que se ingresan en ella para ser ejecutados.

1.2. Sistema de archivos

En Linux, los directorios se encuentran bajo una raíz `/`. Los directorios pueden representar *carpetas*, discos, etc...
Al abrir la terminal, esta inicia y abre la ubicación `/home/`*`[usuario]`*:

```
user@pc ~ $
```

La ubicación sobre la cual está trabajando la terminal aparece antes del signo $. En este caso, aparece $\sim$, que indica el directorio del usuario en el equipo, por lo que en este ejemplo $\sim$ y `/home/user/` son equivalentes.

Para conocer el contenido del directorio en el que nos encontramos, utilizamos el comando `ls`:

```
user@pc ~ $ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
user@pc ~ $
```

Para movernos entre directorio, utilizamos el comando `cd` seguido del nombre del directorio al que queremos ir:

```
user@pc ~ $ cd Desktop
user@pc ~/Desktop $
```

Para retonar al directorio superior, cambiamos a `..` :

```
user@pc ~/Desktop $ cd ..
user@pc ~ $
```

Para crear un directorio usamos `mkdir`:

```
user@pc ~ $ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
user@pc ~ $ mkdir 503208
user@pc ~ $ ls
```

```
503208   Documents  Music     Public     Videos
Desktop  Downloads  Pictures  Templates
user@pc ~ $ cd 503208
user@pc ~/503208 $
```

...y para eliminar los directorios utilizamos `rm -r`:

```
user@pc ~/503208 $ cd ..
user@pc ~ $ rm -r 503208
user@pc ~ $ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
user@pc ~ $
```

Para eliminar archivos usamos `rm`:

```
user@pc ~ $ cd Documents
user@pc ~/Documents $ ls
helloworld.c  a.out
user@pc ~/Documents $ rm a.out
user@pc ~/Documents $ ls
helloworld.c
user@pc ~/Documents $ rm helloworld.c
user@pc ~/Documents $ ls
user@pc ~/Documents $
```

2. C

2.1. El código fuente

El código fuente es un archivo de texto (o varios), que contiene las instrucciones del programa legibles para nosotros. Éste código fuente puede ser creado y editado en un editor de texto, como *Emacs*, *VI/VIM*, *nano*, *Gedit* o *Bloc de Notas*.

Para C, el archivo del código fuente es un archivo de texto cuya extensión será *.c*.

2.2. Creando un código

Ahora creemos un código fuente llamado "`hello.c`" con el siguiente contenido:

```
#include <stdio.h>

int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Este el el programa *Hello World*.

2.3. Compilando un programa

Compilar un programa significa transformar el código fuente — el código que nosotros podemos leer y escribir — a un código que la máquina entienda y pueda ejecutar. Para esto usamos un programa compilador.

Aquí, el compilador que utilizaremos es *GCC*. La forma más básica de usar GCC es desde la terminal, sólo pasandole como argumento el archivo del código fuente, esto da como resultado un archivo ejecutable *a.out*, que será el resultado de la compilación del código.

Para el ejemplo anterior, si guardamos el código como "`hello.c`" en nuestra carpeta de usuario, deberíamos escribir:

```
user@pc ~ $ GCC hello.c
```

Luego, si todo salió bien, podemos correr nuestro programa:

```
user@pc ~ $ ./a.out
Hello, world!
```

3. Variables

Las *variables* son espacios en memoria, donde se puede almacenar y modificar un valor.

En **C** las variables tienen un tipo de dato asociado a ellos, esto es, que tipo de valores pueden almacenar en ellas.

3.1. Declaración

La DECLARACIÓN de una variable es la creación de ésta en el programa, para poder ser utilizada.

Para declarar una variable en C, precisamos de dos cosas: el *tipo de dato* y el *nombre de variable*. El formato básico de declaración es este:

TIPO DE DATO NOMBRE DE VARIABLE;

Por ejemplo, para declarar una variable de tipo *entero* con nombre *number*:

```
int number;
```

3.2. Asignación

La ASIGNACIÓN de variables es darle un valor a la variable. En C, el formato es:

VARIABLE = VALOR;

Por ejemplo, para asignar el valor *42* a la variable *number*:

```
number = 42;
```

Si se quiere modificar el valor de la variable, basta con realizar una nueva asignación.

4. Entrada y salida

4.1. Función `printf()`.

La función `printf()` nos permite desplegar mensajes en la terminal.

Para usarla, se entregan como argumentos un *formato de salida* y las variables (si las hay) en orden respectivo.

Por ejemplo, para desplegar *`hello`* seguido del valor que contiene la variable *`number`* en la terminal:

```
printf("hello %d\n",number);
```

- *`"hello%d"`* indica que se debe desplegar *`hello`* seguido de un número entero, `\n` crea una nueva línea en la terminal después del mensaje.
- *`number`* es la variable cuyo valor se desplegará en lugar de *`%d`*.

4.2. Función `scanf()`.

La función `scanf()` nos permite recibir información de la terminal, y almacenarla en memoria.

Para usarla, se entregan como argumentos un *formato de entrada* y las variables en orden respectivo.

Por ejemplo, para recibir un número entero desde la terminal y almacenarlo en la variable llamada *number*:

```
scanf("%d",&number);
```

- *`%d`* indica que se va a leer un número entero.
- *`&number`* es la variable (`&` debe ir antes del nombre de la variable).

Ejercicios

Terminal y compilación

1. Crear una carpeta bajo el directorio de usuario.
2. Dentro de ese directorio, crear un programa que despliegue "`Hola, mundo!`" en la terminal.
3. Ejecutar el programa.
4. Eliminar el archivo ejecutable.
5. Volver al directorio anterior, y eliminar la carpeta.

Variables, entrada y salida

6. Crear un programa que declare y asigne un valor a una variable de tipo entero. Luego, desplegar esa variable en la terminal y salir.
7. Crear un programa que reciba un entero de la terminal y luego lo despliegue.
8. Crear un programa que cree e inicialize una variable en 0, reciba un entero de la terminal, lo almacene y lo despliegue.

Apéndice

Algunos tipos de datos básicos

Tipo de dato	**Uso común**	**Formato *printf()*/*scanf()***
`char`	Caracteres alfanuméricos y especiales	`%c`
`int`	Valores numéricos enteros	`%d`
`float`	Valores numéricos fraccionales (con decimales)	`%f`