Programación I

Práctica #0 Curso 2024-2025 1º curso

Introducción al laboratorio

Introducción

Las prácticas de laboratorio de Programación I se realizarán en ordenadores dotados del sistema operativo Linux; concretamente, la distribución Ubuntu.

En esta práctica aprenderemos a utilizar algunos comandos básicos de la *shell* de Linux, que es la utilidad básica para interactuar con dicho sistema operativo.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Familiarizarse con el entorno de trabajo del laboratorio.
- Aprender a utilizar algunos de los comandos básicos del sistema operativo Linux.

Entorno

Los ordenadores del laboratorio ya están preparados para realizar las prácticas.
Si deseas realizar las prácticas en tu ordenador deberás[1]:

1. Tener instalada alguna distribución de LINUX.
2. Tener instalado un editor de texto y el compilador de C utilizado en el laboratorio.

[1] Se publicará en MOOVI un manual explicando la instalación del entorno de laboratorio en los ordenadores personales.

Edición , compilación y ejecución

La etapa de codificación dentro del proceso de desarrollo de un programa utilizando un lenguaje de alto nivel, como es C, se compone de tres pasos:

1. Introducir el **código fuente** de nuestro programa en un **fichero de texto**.
2. Compilar el código fuente para obtener un fichero que contenga el **código máquina** (ejecutable) de nuestro programa.
3. Ejecutar nuestro programa.

Para llevar a cabo estos pasos utilizaremos algunos programas disponibles en nuestro sistema. Así, para introducir el código fuente, utilizaremos el programa **editor de texto** denominado `emacs` (aunque el alumno, si lo prefiere puede utilizar otro), y para compilar el código fuente utilizaremos el programa **compilador** de lenguaje C denominado `gcc`. En esta práctica usaremos estos programas para crear nuestro primer programa en lenguaje C.

Planificación

Sesión 0 de laboratorio (2 horas):

- Asignación de cuentas de usuario
- Introducción del laboratorio.
- Realización de los ejercicios 1 a 4.

Algunos comandos y programas básicos de Linux

Introducción

Cuando utilizamos el término Linux normalmente nos estamos refiriendo a una **distribución de software** (Ubuntu, Debian, OpenSuse, …) que incluye un núcleo de sistema operativo y un conjunto de elementos software (programas, librerías, …) orientados a proporcionar funcionalidades diversas. Cuando instalamos una distribución de Linux en nuestro ordenador, siempre instalamos el **núcleo del sistema operativo** y algunas utilidades básicas y, además, seleccionamos qué software adicional queremos instalar.

En esta práctica vamos a utilizar fundamentalmente un programa que encontraremos en toda distribución de Linux: `bash`, ya que se instala por defecto como utilidad básica en cualquier distribución.

En las siguientes secciones introduciremos esta utilidad y empezaremos a aprender a trabajar con ella.

Shell o Intérprete de Comandos

Una *shell* en Linux es un programa cuya función principal, aunque no única, es permitir a los usuarios ejecutar comandos o programas directamente desde un terminal. Debido a esto, también se suele denominar "**Intérprete de Comandos**".

Su funcionamiento es sencillo: espera a que tecleemos un comando, una vez que lo tecleamos lo ejecuta y, a continuación espera a que tecleemos el siguiente comando, repitiendo continuamente esta secuencia.

En cualquier distribución podemos encontrar diferentes programas *shell*. En nuestro caso vamos a utilizar el denominado `bash`, que es el que ha sido adoptado como *shell* por defecto en la mayoría de distribuciones Linux.

Cuando en nuestro sistema Linux abrimos un terminal por defecto se ejecuta en él la *shell*, que presentará un *prompt*[1] y se quedará a la espera de que tecleemos algún comando para ejecutarlo. Su aspecto será similar al siguiente:

```
usuario@maquina:~$
```

Si ahora tecleamos un comando, la *shell* ejecutará dicho comando y nos presentará el resultado y, a continuación, de nuevo el *prompt* quedándose a la espera del siguiente comando.

```
usuario@maquina:~$ pwd
/home/eetlabs.local/usuario
usuario@maquina:~$
```

Si introducimos un comando que la shell no puede entender obtendremos un mensaje indicándolo:

```
usuario@maquina:~$ qwerty
qwerty: No se encontró la orden
usuario@maquina:~$
```

[1] Un *prompt* es una línea que presenta cierta información y se queda a la espera de que introduzcamos algún comando.

Si ahora presionamos la tecla de la flecha hacia arriba veremos que volvemos a tener nuestro comando anterior: la *shell* nos proporciona un histórico de comandos previos que podemos recorrer con las flechas. Esta funcionalidad nos facilitará mucho el trabajo.

La *shell* es un programa complejo que proporciona muchas otras funcionalidades útiles pero, por ahora, las aquí presentadas son suficientes para empezar a trabajar.

> Cuando tecleamos un comando no estamos interactuando directamente con el sistema operativo. Estamos interactuando con un programa denominado *shell*, que nos permite utilizar las funciones del núcleo del sistema operativo indirectamente procesando nuestros comandos.

Ejercicio 1:

Aprende a abrir un *terminal* en tu entorno Linux y comprueba las funcionalidades de la *shell* descritas en este apartado.

Algunos comandos básicos

Los primeros comandos que nos van a ser útiles están relacionados con el sistema de ficheros de Linux. El sistema de ficheros en *Linux* se basa en una jerarquía en árbol de directorios y ficheros, a partir de un único directorio raíz "`/`". Cada usuario dispone de un directorio *home* (por ejemplo `/home/eetlabs.local/pi1`) a partir del cual puede crear su propio árbol de directorios y ficheros. Los principales comandos para interactuar con el sistema de ficheros son: `pwd, cd, ls, cp, mv, rm, rmdir, mkdir, cat, more`. La siguiente tabla describe dichos comandos, incluyendo las opciones más comunes.

Comando	Descripción
`pwd`	*Print Working Directory*. Muestra el directorio actual.
`cd` *dir*	Cambia al directorio `dir` indicado.
`cd ..`	Cambia al directorio superior.
`cd`	*Change Directory*. Cambia al directorio *home* del usuario/a.
`ls`	Muestra una lista de los nombre de los ficheros y directorios del directorio actual.
`ls` *dir*	Igual que el anterior pero para el directorio `dir` especificado: `ls /home/eetlabs.local/pi`
`ls -l`	Muestra información detallada: propietario, permisos, tamaño, fecha.
`ls -a`	Muestra también los ficheros que empiezan por "`.`", como por ejemplo `.bashrc`
`cp` *origen destino*	Copia un fichero de una localización a otra
`mv` *origen destino*	Mueve un fichero de una localización a otra
`rm` *fichero*	Borra un fichero
`rmdir` *dir*	Borra un directorio
`mkdir` *dir*	Crea un directorio
`cat` *fichero*	Muestra el contenido de un fichero

Comando	Descripción
`more` *`fichero`*	Muestra el contenido de un fichero por páginas.
`man`	Muestra la página del manual de un comando.

La mayoría de comandos disponen de una opción `help` que muestra una breve descripción de la sintaxis del comando y sus parámetros. Dicha opción suele obtenerse invocando el propio comando con los parámetros `-h` o `--help`. Por ejemplo:

```
usuario@maquina:~$ ls --help
```

Sin embargo, la información de ayuda más útil suele encontrarse en las páginas de manual disponibles a través del comando `man`.

Ejercicio 2:

Realiza las siguientes acciones indicando para cada una de ellas: (i) el directorio de trabajo en el que estás, (ii) los comandos necesarios para realizarla, y (iii) el resultado de la ejecución:

1. Crea el fichero `prueba.txt` en tu directorio raíz:

```
usuario@maquina:~$ ls -la > prueba.txt
```

2. Crea el directorio `pr0`
3. Mueve el fichero `prueba.txt` al directorio `pr0`
4. Cambia tu directorio de trabajo a `pr0`
5. Muestra el listado de ficheros que contiene, incluyendo los ficheros ocultos. ¿En qué fecha fueron creados?
6. Vuelve a tu directorio raíz.
7. Borra el directorio `pr0` y su contenido.
8. Crea el directorio `pr1`
9. Crea el fichero `prueba.txt` en el directorio `pr1`
10. Muestra por pantalla el contenido del fichero `prueba.txt`
11. Borra todos los ficheros y directorios creados
12. Crea 11 directorios (`pr00`, `pr01`, … `pr10`), uno para cada una de las prácticas guiadas de laboratorio
13. Ejecuta el siguiente comando para conocer la jerarquía de directorios de Linux:

```
usuario@maquina:~$ man hier
```

Codificando un programa básico en C

El editor de texto *emacs*

El editor de texto es un programa que nos permitirá generar ficheros de texto que, en nuestro caso, contendrán el código fuente de los programas que desarrollemos.

Existen multitud de editores de texto para Linux, muchos de ellos disponibles por defecto en cualquier distribución. Nosotros vamos a usar un editor muy utilizado, denominado *emacs*[2]. Para ejecutarlo, simplemente debemos teclear el nombre del programa en una *shell*:

```
usuario@maquina:~/pr00$ emacs &
```

o

```
usuario@maquina:~/pr00$ emacs  nombre_fichero  &
```

El editor dispone de un conjunto de menús para realizar las operaciones de edición más habituales (y otras no tan habituales), por lo que el alumno no debería tener dificultades para empezar a trabajar con él.

Ejercicio 3:

Crea un directorio `pr00` en tu cuenta (si no lo has creado ya previamente).

Entra en ese directorio y utiliza el editor ***`emacs`*** para crear un fichero de texto `ej3.c` que contenga el código fuente de tu primer programa en C.

Teclea, dentro del ***`emacs`***, el código que aparece en el Anexo I del enunciado de esta práctica.

Guarda el código que has creado, y comprueba que el fichero se ha creado correctamente utilizando, en la *shell*, el comando `cat`:

```
usuario@maquina:~/pr00$ cat ej3.c
```

El compilador *gcc*

Como ya hemos indicado, el compilador es un programa que se encarga de transformar nuestro código fuente en código máquina, de tal forma que al final del proceso de compilación dispongamos de un programa ejecutable en nuestro ordenador. El compilador de C más utilizado en entornos Linux es el compilador `gcc`.

En su forma más sencilla, para compilar un fichero de código fuente simplemente tenemos que ejecutar el compilador utilizando como único parámetro el nombre del fichero que contiene el código fuente. Así, para compilar el programa que hemos editado en la práctica anterior simplemente tendremos que teclear, en el directorio correspondiente:

[2] Si no tenemos el editor `emacs` instalado, podemos consultar cómo instalarlo en el manual de instalación del entorno disponible en Moovi.

```
usuario@maquina:~/pr00$ gcc  ej3.c
```

Si todo ha ido bien el compilador no presentará ningún mensaje y generará un nuevo fichero en el directorio actual denominado `a.out`, que contendrá el código ejecutable de nuestro programa.

Por el contrario, si el compilador no ha sido capaz de generar el fichero ejecutable por algún motivo (normalmente algún error en el código fuente que queremos compilar), nos presentará un mensaje indicando el error detectado y no generará el fichero de código ejecutable.

El programa `gcc` admite muchas opciones que modifican su comportamiento (podemos consultarlas accediendo a su página de manual). En este punto vamos a introducir dos opciones que utilizaremos siempre a partir de ahora.

La primera de ellas es la opción "`-o`", que nos permite generar un fichero ejecutable con el nombre que indiquemos, en vez del nombre por defecto `a.out`.

La segunda es la opción "`-Wall`", que le indica al compilador que nos muestre todos los avisos (warnings). Los avisos nos alertan de posibles fallos en el código fuente que estamos compilando, pero que no impiden la generación del código ejecutable.

Ejercicio 4:

Compila el código fuente que generaste en el ejercicio anterior, utilizando las opciones del compilador indicadas, de tal manera que se genere un fichero ejecutable denominado `ej4:`

```
usuario@maquina:~/pr00$ gcc  -Wall ej3.c  -o ej4
```

Una vez que hemos generado correctamente un fichero ejecutable, para ejecutarlo simplemente tenemos que teclear su nombre en la *shell*.

Ejecuta el programa que acabas de crear:

```
usuario@maquina:~/pr00$ ./ej4
```

Debes teclear `./ej4` para ejecutar el programa `ej4`, ya que, por defecto, la shell no busca los programas a ejecutar en el directorio actual (.), por lo que debemos especificar la ruta del fichero ejecutable.

Resumen

Los principales resultados esperados de esta práctica son:

- Ser capaces de ejecutar los comandos básicos del sistema operativo Linux.
- Ser capaces de trabajar sobre la estructura de directorios del sistema de ficheros de Linux
- Ser capaces de editar, compilar y ejecutar programas en C.

Como posibles líneas de trabajo adicional (opcional) del alumno, se proponen las siguientes:

- Profundizar en el conocimiento de los comandos básicos de Linux.

Anexo I. Programa "Hola mundo" en C

Introducción

Para comenzar a programar en C debemos tener clara la estructura de un programa. Para conseguirlo vamos a escribir un programa básico, y ya clásico, denominado "Hola mundo":

```
// Hola Mundo en C

#include <stdio.h>

int main () {
  fprintf (stdout, "Hola mundo\n");
  return 0;
}
```

4.2. Estilo

Los programas deben utilizar sangría (indentación), llevar comentarios y utilizar nombres de identificadores significativos y no excesivamente largos.

4.3. Explicación

La primera línea, desde `//` es un comentario (no forma parte del código, pero sirve de ayuda para entenderlo a lectores humanos).

La línea `#include...` le indica al preprocesador de C que incluya en el programa el contenido del fichero de cabecera `stdio.h`, donde están declaradas las funciones de entrada/salida estándar de C (que vamos a utilizar, como `fprintf (stdout, ...)`).

La línea `int main (...)` define la cabecera de la función principal (*main()*). En este punto comenzará la ejecución de cualquier programa en C.

Las llaves `{…}` delimitan el bloque de instrucciones que forman parte de la función *main()*.

La llamada a la función `fprintf (stdout, "Hola mundo\n");` hace que se muestre por pantalla lo que se le pasa entre comillas.

La instrucción `return 0;` indica qué valor devuelve la función *main()* (en este caso, 0 indica que la operación fue correcta).