Práctica 1

Introducción a la programación con Picky

Grado en Ingeniería en Sistemas Audiovisuales y Multimedia
GSyC, Universidad Rey Juan Carlos

8 de septiembre de 2017

1. Introducción

En esta práctica vamos a aprender a manejar el *Terminal* de Linux, que nos servirá de ayuda a la hora de *compilar* nuestro programa escrito en lenguaje *Picky*, para lo que usaremos el editor de textos *gEdit*.

1.1. El Terminal de Linux

El *Terminal, Consola* o *Bash*, es una herramienta muy potente de los Sistemas Operativos basados en Unix que nos permite ejecutar infinidad de comandos para realizar distintas tareas.

Todo aquello que podemos hacer de forma gráfica, y muchísimo más, se puede hacer desde el Terminal. Nosotros la usaremos para casi todos los procedimientos. A continuación se muestran algunos comandos básicos, que nos serán de utilidad:

- Cat (de concatenar), es una maravillosa utilidad que nos permite visualizar el contenido de un archivo de texto sin la necesidad de un editor. Para utilizarlo solo debemos mencionarlo junto al archivo que deseamos visualizar: `$ cat prueba.txt`.
- Ls (de listar), permite listar el contenido de un directorio o fichero. La sintaxis es: `$ ls /home/directorio`. El comando ls tiene varias opciones que permiten organizar la salida, lo que resulta particularmente útil cuando es muy grande. Por ejemplo, puedes usar -a para mostrar los archivos ocultos y -l para mostrar los usuarios, permisos y la fecha de los archivos. Así como para todos los comandos Linux, estas opciones pueden combinarse, terminando en algo como: `$ ls -la /home/directorio`.
- Cd (de change directory o cambiar directorio), es como su nombre lo indica el comando que necesitarás para acceder a una ruta distinta de la que te encuentras. Por ejemplo, si estas en el directorio /home y deseas acceder a /home/ejercicios, seria: `$ cd /home/ejercicios`. Si estás en /home/ejercicios y deseas subir un nivel (es decir ir al directorio /home), ejecutas: `$ cd ..`
- Touch crea un archivo vacío, si el archivo existe actualiza la hora de modificación. Para crear el archivo prueba1.txt en /home, seria: `$ touch /home/prueba1.txt`.
- Mkdir (de make directory o crear directorio), crea un directorio nuevo tomando en cuenta la ubicación actual. Por ejemplo, si estas en /home y deseas crear el directorio ejercicios, sería: `$ mkdir /home/ejercicios`. Mkdir tiene una opción bastante útil que permite crear un árbol de directorios completo que no existe. Para eso usamos la opción -p: `$ mkdir -p /home/ejercicios/prueba/uno/dos/tres`
- Cp (de copy o copiar), copia un archivo o directorio origen a un archivo o directorio destino. Por ejemplo, para copiar el archivo prueba.txt ubicado en /home a un directorio de respaldo, podemos usar: `$ cp /home/prueba.txt /home/respaldo/prueba.txt`. En la sintaxis siempre se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, cp copiará el archivo o directorio con el nuevo nombre. El comando también cuenta con la opción -r que copia no sólo el directorio especificado sino todos sus directorios internos de forma recursiva. Suponiendo que deseamos hacer una copia del directorio /home/ejercicios que a su vez tiene las carpetas ejercicio1 y ejercicio2 en su interior, en lugar de ejecutar un comando para cada carpeta, ejecutamos: `$ cp -r /home/ejercicios /home/respaldos/`.

- Mv (de move o mover), mueve un archivo a una ruta específica, y a diferencia de cp, lo elimina del origen finalizada la operación. Por ejemplo: `$ mv /home/prueba.txt /home/respaldos/prueba2.txt`. Al igual que cp, en la sintaxis se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, mv moverá el archivo o directorio con el nuevo nombre.
- Rm (de remove o remover), es el comando necesario para borrar un archivo o directorio. Para borrar el archivo prueba.txt ubicado en /home, ejecutamos: `$ rm /home/prueba.txt`. Este comando también presenta varias opciones. La opción -r borra todos los archivos y directorios de forma recursiva. Por otra parte, -f borra todo sin pedir confirmación. Estas opciones pueden combinarse causando un borrado recursivo y sin confirmación del directorio que se especifique. Para realizar esto en el directorio respaldos ubicado en el /home, usamos: `$ rm -fr /home/respaldos`. Este comando es muy peligroso, por lo tanto es importante que nos documentemos bien acerca de los efectos de estas opciones en nuestro sistema para así evitar consecuencias nefastas.
- Pwd (de print working directory o imprimir directorio de trabajo), es un conveniente comando que imprime nuestra ruta o ubicación al momento de ejecutarlo, así evitamos perdernos si estamos trabajando con múltiples directorios y carpetas. Su sintaxis seria: `$ pwd`.
- Clear (de limpiar), es un sencillo comando que limpiara nuestra terminal por completo dejándola como recién abierta. Para ello ejecutamos: `$ clear`.
- Man, muestra una documentación completa de todos los comandos. Para clear, por ejemplo: `$ man clear`.

1.2. Instalación de Picky

En los ordenadores del Laboratorio tendremos ya disponible el entorno Picky. Si queremos también instalarlo en nuestro equipo personal, con Linux previamente instalado, deberemos hacer lo siguiente:

1. Descargar el archivo adjunto a esta práctica según sea nuestro ordenador.
   - Si es de 32 bits: Picky32bits.deb
   - Si es de 64 bits: Picky64bits.deb
2. Ir al Terminal y ejecutar el archivo descargado: `sudo dpkg -i PickyXXbits.deb`
3. Comprobar que se ha instalado correctamente: `pick -v`. Lo que nos debería mostrar algo así: `pick: version GRAPH1`

Si todo ha ido bien, ya tendremos instalado Picky en nuestro equipo y podremos empezar a programar.

1.3. Editor de textos gEdit

Este editor que nos ofrece Linux es una herramienta muy simple pero con mucho potencial. Nos permite escribir texto plano o sin formato, reconociendo y resaltando la sintaxis de muchísimos lenguajes de programación.

Nosotros lo usaremos para escribir nuestras líneas de código de programación en lenguaje Picky. Para ejecutar esta herramienta basta con escribir desde el Terminal: `gedit &`. O bien buscándolo desde el *Dash* de Ubuntu.

2. *Hola mundo* en Picky

A continuación se facilita el código íntegro de nuestro primer programa en Picky, lo que en el argot de la Programación se denomina el *Hola mundo*.

Todos los programas escritos en lenguaje Picky deben tener la extensión `.p` (de Picky). Así, éste se denomina `programa.p`, y también está disponible para descargarse como adjunto a esta práctica.

```
/*
 *  Fichero: programa.p
 * Descripción: programa de bienvenida en Picky
 * Autor: aturing
 */

/*
 * Nombre del programa.
 */
program programa;

/*
 * Tipos de datos
 */
types:
/* Dias de la semana */
TipoDiaSem = (Lun, Mar, Mier, Jue, Vie, Sab, Dom);

/*
 * Constantes
 */
consts:
Pi = 3.1415926;
RadioPrueba = 2.0;

/*
 * Funciones y procedimientos.
 */

function longitudcirculo(r: float): float
{
return 2.0 * Pi * r;
}

/*
 * Programa Principal.
 */
procedure main()
{
writeln("Hola que tal.");
writeln(longitudcirculo(RadioPrueba));
}
```

2.1. Editar, compilar y ejecutar

Tenemos que distinguir entre el proceso de edición o desarrollo de nuestro programa, el proceso de compilación y, por último, la ejecución:

1. Edición: escritura de las líneas de código del programa (.p) en lenguaje Picky. Para lo que usamos el editor gEdit.

2. Compilación: proceso de traducción de ese fichero a código binario ejecutable. Para ello, escribimos en el Terminal: `pick programa.p`. Recordemos dos cosas importantes: debemos estar en la misma ruta donde se encuentra el fichero, y éste debe haber sido guardado previamente. Al compilar, el sistema nos devolverá un nuevo fichero `out.pam`, que contiene el código binario o ejecutable.

3. Ejecución: como cualquier otro software que usamos, al ejecutar nuestro programa veremos el resultado o finalidad de éste. Para ejecutarlo, escribimos en el Terminal: `./out.pam`.

Al hacer este último paso, en este programa el resultado mostrado por pantalla debería ser: `Hola que tal. 12.566`. Si es así, es que todo ha ido bien. En otro caso, nos mostrará el error que hay; por ejemplo, si es un error de tipo sintáctico situado en la línea 21, se mostraría algo así: `programa.p:21: at 'consts': syntax error`. Lo que indica que, **por lo general antes de esa línea**, hay un error.

3. Entrega

En esta práctica nuestro objetivo es que compile y se ejecute mostrando correctamente el resultado ya comentado, a lo que habrá que hacer una pequeña modificación para que muestre, en lugar del `Hola que tal`, tu nombre y apellidos. Asimismo, en lugar de `12.566`, que muestre tu número de expediente.

La entrega de esta práctica se hará de forma no presencial, dejando todo el código fuente contenido en un fichero `apellido-nombre-P01.tar` que habrá que dejar adjunto en la tarea (Moodle).

Fecha de entrega: **14 de septiembre**

Se podrá entregar hasta las 23:59h de este día.