Práctica 1

Introducción al entorno de desarrollo

Objetivos

- Familiarizarse con el entorno de desarrollo de aplicaciones C en LINUX, y comprender los conceptos de proyecto, ejecutable y biblioteca en este contexto.
- Familiarizarse con el manejo básico del shell y aprender a desarrollar `shell scripts` sencillos

Requisitos

Para poder realizar con éxito la práctica el alumno debe haber leído y comprendido los siguientes documentos facilitados por el profesor:

- Presentación "Primeros pasos GNU/Linux", que nos introduce al entorno GNU/Linux que utilizaremos.
- Presentación "Revisión: Programación en C", que realiza un repaso de los conocimientos de C necesarios para realizar con éxito las prácticas, haciendo especial hincapié en los errores que cometen habitualmente los alumnos con poca experiencia con el lenguaje C.
- Manual "Entorno de desarrollo C para GNU/Linux", que describe las herramientas que componen el entorno de desarrollo que vamos a utilizar, así como las funciones básicas de la biblioteca estándar de C que los alumnos deben conocer.
- Presentación "Importación de proyectos con Makefile en Eclipse y perfiles de ejecución y depuración``, que nos introduce al uso del entorno de desarrollo integrado (IDE) Eclipse.
- Presentación "Introducción a Bash", que presenta una breve introducción al intérprete de órdenes (shell) Bash.

Desarrollo de la práctica

En esta práctica el alumno debe crear una herramienta de línea de órdenes llamada `mitar`, capaz de crear un archivo `tarball` que contenga varios ficheros, o de extraer los ficheros de un `tarball` previamente creado. Por ejemplo, la orden:

```
$ ./mitar -cf proyecto1.mtar mitar.c mitar.h rut_mitar.c
```

deberá crear un archivo `proyecto1.mtar` que recopile el contenido de los ficheros `mitar.c`, `mitar.h` y `rut_mitar.c`. Así mismo, la orden:

```
$ ./mitar -xf proyecto1.mtar
```

extraerá el contenido del `tarball proyecto1.mtar`, es decir, los ficheros `mitar.c`, `mitar.h` y `rut_mitar.c`.

Nuestra versión del `tarball` consistirá en una cabecera que describirá el contenido del archivo, seguida del contenido binario de cada uno de los ficheros incluidos en él. La cabecera estará formada por un entero de 4 bytes que indicará el número de ficheros almacenados (N), seguido de N descriptores de fichero. Cada uno de estos descriptores se compondrá de una cadena de caracteres (terminada en un byte 0) con el nombre del fichero (su ruta), y un un entero sin signo de cuatro bytes con el tamaño en bytes del fichero. Es decir, la estructura descriptor en memoria sería:

```
typedef struct {
```

```
  char* name;
  unsigned int size;
} stHeaderEntry;
```

Para almacenar el nombre del fichero habrá que reservar memoria del heap almacenando en el campo name la dirección de comienzo devuelta por `malloc`.

Volviendo sobre nuestro ejemplo, el fichero `proyecto1.mtar` comenzará con cuatro bytes que representan un entero con valor 3 (el número de ficheros almacenados), seguido de 3 descriptores de fichero, cada uno con el nombre de uno de los ficheros y su tamaño en bytes.

El proyecto `mitar` estará constituido por los siguientes ficheros que proporcionará el profesor:

- `makefile`: fichero de entrada a la herramienta make para la compilación del proyecto.
- `mitar.h`: contiene las declaraciones de tipos de datos que se usarán en el resto del código.
- `mitar.c`: es el fichero principal, que contiene la función **main**.
- `rut-mitar.c`: fichero en el que se implementan las funciones de tratamiento del *tarball*. Este fichero se da incompleto, y los alumnos deben completarlo siguiendo las indicaciones de los comentarios.

Script de comprobación

Con el fin de practicar el uso del shell, se pide al alumno que desarrolle su propio script bash para la comprobación de la práctica. El script creará un tarball con el contenido de unos ficheros utilizando nuestro programa `mitar`, y luego extraerá el contenido y comprobará que los ficheros extraídos son idénticos a los originales. Si todo es correcto terminará devolviendo 0. Si hay errores terminará devolviendo 1. El script deberá seguir el siguiente esquema:

1. Comprobará que el programa `mitar` está en el directorio actual y que es ejecutable. En caso contrario mostrará un mensaje informativo por pantalla y terminará.
2. Comprobará si existe un directorio tmp dentro del directorio actual. Si existe lo borrará, incluyendo todo lo que haya dentro de él (mirar la opción -r del comando rm).
3. Creará un nuevo directorio temporal tmp dentro del directorio actual y cambiará a este directorio.
4. Creará tres ficheros (dentro del directorio):
    - fich1.txt: con el contenido “Hola Mundo", utilizando la orden echo y redirigiendo la salida al fichero.
    - fich2.txt: con una copia de las 10 primeras líneas del fichero /etc/passwd. Se hace fácil utilizando el programa head y redirigiendo la salida al fichero.
    - fich3.dat: con un contenido binario aleatorio de 1024 bytes, tomado del dispositivo /dev/urandom. De nuevo conviene utilizar head con la opción -c.
5. Invocará el programa `mitar` que hemos desarrollado, para crear un fichero `fichtar.mtar` con el contenido de los tres ficheros anteriores.
6. Creará un directorio out (dentro del directorio actual, que debe ser tmp) y copiará el fichero `fichtar.mtar` al nuevo directorio.
7. Cambiará al directorio out y ejecutará el programa `mitar` para extraer el contenido del tarball.
8. Usará el programa diff para comparar los fichero extraídos con los originales, que estarán en el directorio anterior (..).
9. Si los tres ficheros extraídos son iguales que los originales, volverá al directorio original (../..), mostrará el mensaje “Correcto" por pantalla y devolverá 0. Si hay algún error, volverá al directorio original, mostrará un mensaje descriptivo por pantalla y devolverá 1.

Parte extra - Nueva función: listTar con opción -t

Crear un nuevo directorio que sea copia del desarrollo de la parte básica. Añadir una nueva función dentro del archivo `rut_mitar.c`:

```
int listTar(char tarName[]);
```

La función `listTar` recibe como argumento la ruta de un archivo de tipo `.mtar` y muestra por pantalla los ficheros contenidos en él (nombre y tamaño). La función, además, devuelve un entero que es 0 si la función se ha ejecutado correctamente.

Añadir una nueva opción `-t` al programa principal, que permita listar los ficheros contenidos en un archivo `.mtar`. Modificar para ello el tipo enumerado:

```
typedef enum{
  NONE,
  ERROR,
  CREATE,
  EXTRACT,
  LIST
} flags;
```

y las instrucciones de uso:

```
char uso[]="Uso: tar -c|x|t -f archivo_mtar [fich1 fich2 ...]\n";
```