Práctica 1

Todo lo que se realice en el Laboratorio se podrá subir al campus virtual. La entrega se realizará en un único archivo comprimido. Dicha entrega se realizará de forma individual, aunque el trabajo se realice en grupo.

Cada archivo de código fuente o script tendrá una pequeña sección indicando los nombres, apellidos y DNI de los componentes del grupo que han desarrollado dicho código.

A continuación se indican las actividades a realizar para la práctica, así como los ficheros a subir al campus virtual dentro del entregable final.

Ejercicio 1

Se realizará el Ejercicio 1 de la Práctica 1. Los ficheros a incorporar en la entrega serán ejercicios/`show_file.c` y posibles versiones.

Ejercicio 2

Se realizará el Ejercicio 2 de la Práctica 1. Los ficheros a incorporar en la tarea serán `ejercicios/badsort-ptr.c` y posibles versiones.

Desarrollo del código de la práctica

Esta sección incluye el desarrollo del código principal de la práctica. Una vez comprobado su correcto funcionamiento con diversas pruebas, se podrán añadir a la entrega todos los archivos .c y .h de la práctica. Opcionalmente, se podrá entregar un archivo Makefile si se utilizó para compilar el proyecto.

Script de comprobación

Esta tarea consiste en implementar el script indicado en el guión de la práctica. Se podrá añadir a la entrega el script bash implementado.

A continuación, se detallan algunas posibles extensiones que se podrán desarrollar si se considera oportuno. En todas ellas, se podrán añadir a la entrega los archivos .c y .h de la práctica. No es necesario añadir todas las versiones de la misma. Como son extensiones acumulables, bastará con entregar la última versión alcanzada.

Extensión 1: Listado del contenido del .mtar

Añadir una nueva función dentro del archivo `mytar_routines.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.

Para implementar esta parte, añadir una nueva opción `-l` 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 use[]="Usage: tar -c|x|l -f file_mytar [file1 file2 ...]\n";
```

Extensión 2: Empaquetar un fichero adicional en el .mtar existente

Modificar el programa para añadirle una nueva funcionalidad. El programa mytar presentará también la opción de añadir un nuevo fichero a un .mtar existente con la opción -a:

```
./mytar -af fichero.mtar archivo_nuevo.txt
```

Modificar adecuadamente la instrucciones de uso `use` y el tipo enumerado `flags`.

Extensión 3: Borrar un fichero del .mtar

Modificar el programa para añadirle una nueva funcionalidad. El programa mytar presentará también la opción de borrar un fichero ya empaquetado en un .mtar existente con la opción -r:

```
./mytar -rf fichero.mtar archivo_existente.txt
```

Modificar adecuadamente la instrucciones de uso `use` y el tipo enumerado `flags`.

Extensión 4: Almacenamiento en orden alternativo

Modificar el programa para añadirle una nueva funcionalidad. El programa mytar presentará también la opción de almacenamiento alternativo, con la opción -k:

```
./mytar -kf fichero.mtar archivo1.txt archivo2.txt archivo3.txt
```

de tal manera que el archivo .mtar se habrá formado de la siguiente manera:

- Número de archivos
- Tamaño total de los archivos
- Datos de los archivos
- Nombre de los archivos con su tamaño, en orden

El programa mytar presentará también la correspondiente opción para extraer el archivo empaquetado, -v:

```
./mytar -vf fichero.mtar
```

Modificar adecuadamente la instrucciones de uso `use` y el tipo enumerado `flags`.