Sistemas Operativos

Curso
2020-2021

Módulo 2: Gestión de Ficheros
Práctica 1:Introducción al entorno de desarrollo

Agenda

Objetivo

- Familiarizarse con el entorno de desarrollo utilizado en el laboratorio.
- Familiarizarse con el manejo básico del shell.
- Revisión sobre la programación en C y las funciones de la librería estándar de C para gestión de ficheros

Agenda

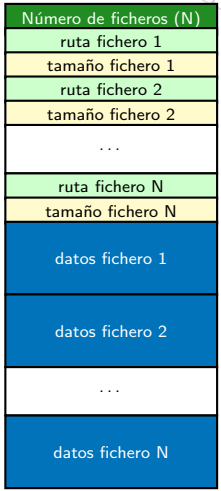
El formato TAR

- El formato de ficheros `TAR` es un formato estandarizado por POSIX muy popular en los sistemas Unix para "archivar" ficheros.
  - Archivar: almacenar ficheros y directorios en un único *tarball*, el fichero con extensión `.tar`).
  - `TAR` hace referencia a `Tape ARchiver`. Se diseño para almacenar ficheros en cintas magnéticas (acceso a bloques secuencial).
- La utilidad `tar` permite manipular ficheros con el formato `tar`:
  - Crear archivo `tar`
  - Extraer ficheros de un archivo `tar`
  - Añadir ficheros a un archivo `tar`
  - ...

Nuestro formato `mtar`

En la práctica utilizaremos una simplificación del formato `tar` que llamamos `mtar`. Los archivos `mtar` constan de:

- Cabecera: Describe qué ficheros se almacenan en el archivo
  - Número de ficheros: entero de 4 bytes
  - Pares: <ruta, tamaño >
    - Ruta: cadena con terminador \0
    - Tamaño: entero de 4 bytes
- Ficheros: concatenados uno a continuación de otro



Programa `mitar`

Modo de uso

```
mitar -c|x -f archivo_mtar [fich1 fich2 ...]
```

- `-c` : Crear archivo mtar
  - Ejemplo:

```
$ mitar -c -f ejemplo.mtar a.txt b.txt c.txt
```

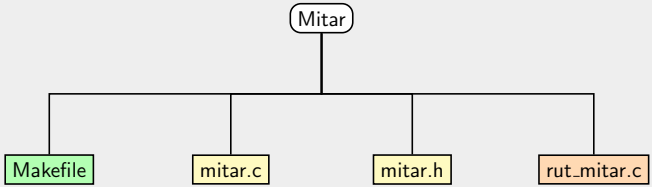
- `-x` : Extraer archivo mtar
  - Ejemplo:

```
$ mitar -x -f ejemplo.mtar
```

Implementación (I)

Proyecto

- El proyecto consta de los siguientes ficheros:
  - `makefile`
  - `mitar.c`: función `main()` del programa
  - `mitar.h`: declaraciones de tipos de datos y funciones
  - `rut_mitar.c`: funciones de creación y extracción de archivos mtar (Único fichero a modificar)



Implementación (II)

mitar.h

```
#ifndef _MITAR_H
#define _MITAR_H

#include <limits.h>

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

typedef struct {
  char name[PATH_MAX];
  unsigned int size;
} stHeaderEntry;

int createTar(int nFiles, char *fileNames[], char tarName[]);
int extractTar(char tarName[]);

#endif /* _MITAR_H */
```

Implementación (III)

Funciones a implementar (`rut_mitar.c`)

- `int createTar(int nFiles, char *fileNames[], char *tarName);`
  - Crea un fichero mtar con nombre 'tarName' incluyendo en él los ficheros cuya rutas están especificadas en el array `fileNames`
- `int extractTar(char* tarName);`
  - Extrae el fichero mtar cuya ruta se pasa como parámetro
- `int copynFile(FILE *origen, FILE *destimo, int nBytes);`
  - Transfiere `nBytes` del fichero origen al fichero destino
    - La copia de datos finalizará cuando se transfieran `nBytes` o se llegue al fin del fichero `origen`
  - Devuelve el número de bytes que se han transferido realmente
- `int readHeader(FILE *tarFile, stHeaderEntry **header, int *nFiles);`
  - Lee la cabecera del fichero mtar `tarFile` y copia la metainformación en el array `header`
    - La función ha de reservar memoria para el array `header` (de ahí el doble puntero → puntero por referencia)
  - Devuelve en `nFiles` (entero por referencia) el número de ficheros contenidos en el mtar

Implementación (IV)

Uso del doble puntero en `readHeader()`

```
int readHeader(FILE *tarFile, stHeaderEntry **header, int *nFiles)
{
    stHeaderEntry* array=NULL;
    int nr_files=0;

    ... Leemos el número de ficheros (N) del tarFile y lo volcamos en nr_files ...

    /* Reservamos memoria para el array */
    array=malloc(sizeof(stHeaderEntry)*nr_files);

    ... Leemos la metainformacion del tarFile y la volcamos en el array ...

    /* Devolvemos los valores leídos a la función invocadora */
    (*nFiles)=nr_files;
    (*header)=array;

    return (EXIT_SUCCESS);
}
```

Creación de un fichero mtar (I)

- La creación de un fichero mtar exige realizar escrituras en el fichero en desorden
  - No sabemos de antemano cuál es el tamaño en bytes de cada uno de los ficheros que hay que introducir en el mtar
  - Solo sabremos el tamaño de cada archivo una vez lo hayamos leído por completo y transferido al fichero mtar vía `copynFile()`

Creación de un fichero mtar (II)

Pasos a llevar a cabo en `createTar()`

1. Abrimos el fichero mtar para escritura (fichero destino)
2. Reservamos memoria (con `malloc()`) para un array de `stHeaderEntry`
   - El array tendrá tantas posiciones como ficheros haya que introducir en el mtar
3. Saltamos al byte del fichero mtar donde comienza la region de datos
   - Dejamos hueco para la cabecera
4. Por cada (`inputFile`) que haya que copiar en el mtar:
   - Abrimos `inputFile`;
   - Copiamos: `copynFile(inputFile,tarFile,INT_MAX);`
   - Cerramos `inputFile`
   - Rellenamos el `stHeaderEntry` correspondiente con la ruta y tamaño de `inputFile`;
5. Nos posicionamos para escribir en el byte 0 del fichero mtar para:
   - Escribir el número de ficheros (int de 4 bytes)
   - Copiar el array de `stHeaderEntry`
6. Liberamos memoria y cerramos el fichero mtar

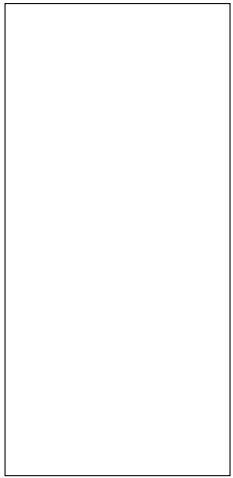
Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Archivo `test.mtar` **(en disco)**

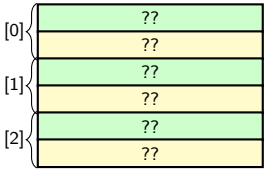


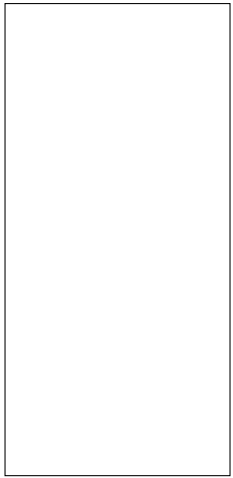
Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Archivo `test.mtar` **(en disco)**

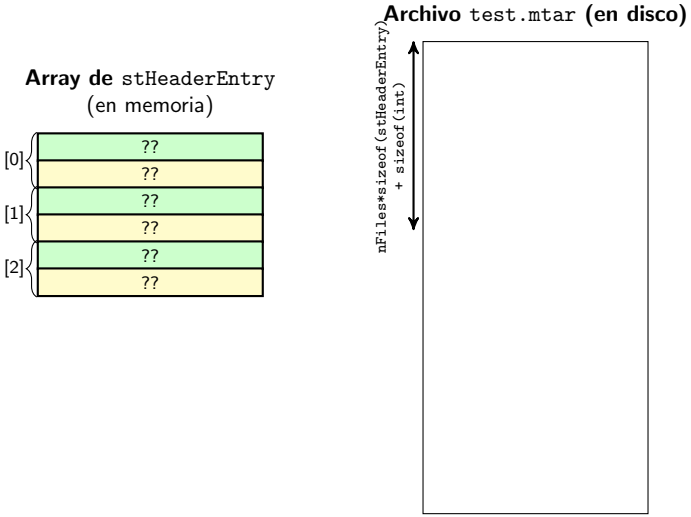
Array de `stHeaderEntry`
(en memoria)





Ejemplo: Creación de un fichero mtar

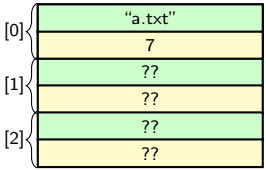
```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```



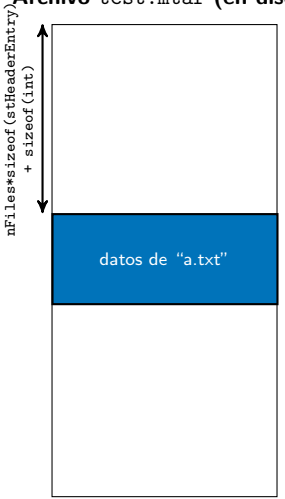
Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Array de `stHeaderEntry`
(en memoria)

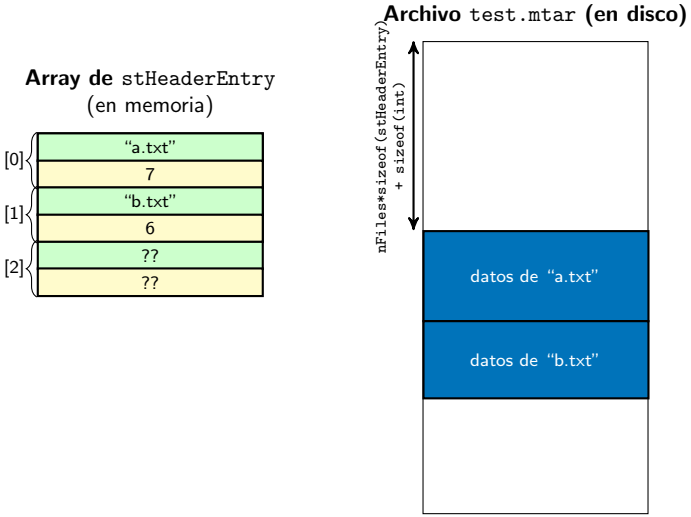


Archivo `test.mtar` **(en disco)**



Ejemplo: Creación de un fichero mtar

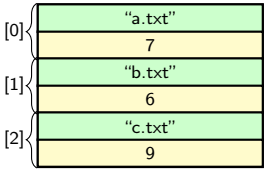
```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```



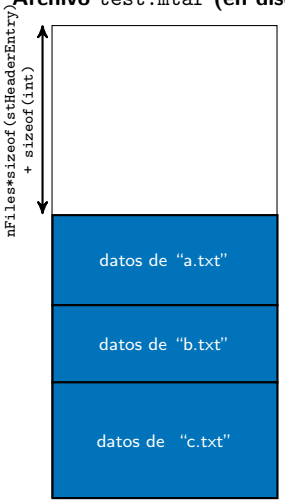
Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Array de `stHeaderEntry`
(en memoria)



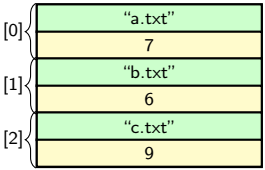
Archivo `test.mtar` **(en disco)**



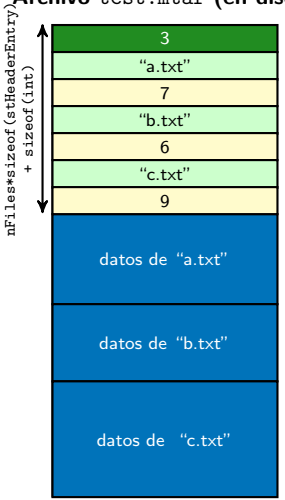
Ejemplo: Creación de un fichero mtar

```
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
```

Array de `stHeaderEntry`
(en memoria)



Archivo `test.mtar` **(en disco)**



Ejemplo de ejecución

terminal

```
$ ls
a.txt  b.txt  c.txt  makefile  mitar.c  mitar.h  rut_mitar.c
$ du -b *.txt
7   a.txt
6   b.txt
9   c.txt
$ make
gcc -g -Wall -c mitar.c -o mitar.o
gcc -g -Wall -c rut_mitar.c -o rut_mitar.o
gcc -g -Wall -o mitar mitar.o rut_mitar.o
$ ./mitar -c -f test.mtar a.txt b.txt c.txt
Fichero mitar creado con exito
$ ls
a.txt  c.txt     mitar    mitar.h  rut_mitar.c  test.mtar
b.txt  makefile  mitar.c  mitar.o  rut_mitar.o
```

Ejemplo de ejecución (cont.)

terminal

```
$ mkdir tmp
$ cd tmp/
$ ../mitar -x -f ../test.mtar
[0]: Creando fichero a.txt, tamano 7 Bytes...Ok
[1]: Creando fichero b.txt, tamano 6 Bytes...Ok
[2]: Creando fichero c.txt, tamano 9 Bytes...Ok
$ ls
a.txt  b.txt  c.txt
$ diff a.txt ../a.txt
$ diff b.txt ../b.txt
$ diff c.txt ../c.txt
$
```

Agenda

Entrega de la práctica

- Para realizar la entrega de cada práctica de la asignatura debe subirse un único fichero ".zip" o ".tar.gz" al Campus Virtual
  - Ha de contener todos los ficheros necesarios para compilar y probar la práctica (fuentes + Makefile + script de comprobación).
  - Debe ejecutarse "make clean" antes de generar el fichero comprimido
  - Nombre del fichero comprimido:

    `<apellido1>_<apellido2>_<nombre>_Pr<num_práctica>.tar.gz`