Sistemas Operativos - Grado en Ingeniería del Software
27 de mayo de 2021, Curso 2020-21
Facultad de Informática, UCM

Entregable 1. Ampliar la funcionalidad del programa `mytar` de la Práctica 1 con la opción `-t`, que imprimirá por pantalla los pares *(nombre fichero, tamaño)* correspondientes a los ficheros almanenados en un mtar. Los pares se mostrarán en orden descendente por tamaño de fichero.

Uso: `./mytar -t -f <mtar_file>`

Para extender la funcionalidad del programa de esta forma, se ha de implementar la nueva función `listTar()` en `mytar_routines.c`. El prototipo de esta función es como sigue:

```
int listTar(char* tarName);
```

En este enlace se puede encontrar una nueva versión de los ficheros *mytar.c* y *mytar.h*. Para realizar este ejercicio se deben reemplazar las versiones antiguas de esos ficheros por los nuevos, que incluyen la versión actualizada del procesamiento de las opciones de línea de comando y el prototipo de la nueva función a implementar en *mytar_routines.c*.

Ejemplo de ejecución

```
# Compilar el programa
usuarioso@debian:~/Mytar$ make
gcc -g -Wall  -c  mytar.c -o mytar.o
gcc -g -Wall  -c  mytar_routines.c -o mytar_routines.o
gcc -g -Wall  -o mytar mytar.o mytar_routines.o

# Crear un conjunto de ficheros de texto de prueba
usuarioso@debian:~/Mytar$ echo Hello > a.txt
usuarioso@debian:~/Mytar$ echo Goodbye > b.txt
usuarioso@debian:~/Mytar$ echo Heeeeeey > c.txt
usuarioso@debian:~/Mytar$ echo This is a longer string > d.txt
usuarioso@debian:~/Mytar$ echo This is a much longer string > e.txt
usuarioso@debian:~/Mytar$ du -b *.txt
6   a.txt
8   b.txt
9   c.txt
24  d.txt
29  e.txt
# Crear un fichero mtar con los ficheros de texto creados
usuarioso@debian:~/Mytar$ ./mytar -c -f test.mtar a.txt b.txt c.txt d.txt e.txt
mtar file created successfully

# Listar ficheros (en orden descendente por tamaño)
usuarioso@debian:~/Mytar$ ./mytar -t -f test.mtar
[0] File name: e.txt, size: 29 bytes
[1] File name: d.txt, size: 24 bytes
[2] File name: c.txt, size: 9 bytes
[3] File name: b.txt, size: 8 bytes
[4] File name: a.txt, size: 6 bytes
```

Entregable 2. Escribe un programa en lenguaje C que permita generar un conjunto de hilos especificado por el usuario.

De forma **concurrente**, cada hilo deberá escribir en un fichero su TID (utilizando la función `gettid`), empleando para ello tantos bytes como ocupe una variable de tipo `pid_t`. La posición del fichero en la que escribirá dicha información vendrá determinada por el orden de ejecución de cada hilo; para ello, se utilizará una variable compartida por todos lo hilos llamada `posicion`.

Así, el primer hilo que escribirá su TID en la posición `0 * sizeof( pid_t )`, el segundo proceso lo hará en la posición `1 * sizeof( pid_t )`, y así sucesivamente.

El proceso principal esperará a la finalización de todos los hilos, y a continuación mostrará por salida estándar el contenido del fichero.

El programa a desarrollar recibirá como primer argumento el número de hilos a generar, y como segundo argumento el nombre del fichero a escribir (que, en caso de existir, será truncado). Si el número de argumentos es incorrecto, el programa finalizará con un mensaje de error.

Un posible ejemplo de ejecución podría ser:

```
$ ./ej_thread.x 4 fichero.txt

[TID 1395] Hilo escribiendo en posicion 2
[TID 1394] Hilo escribiendo en posicion 0
[TID 1393] Hilo escribiendo en posicion 3
[TID 1396] Hilo escribiendo en posicion 1

[PID 1392] Termina el hilo 0
[PID 1392] Termina el hilo 1
[PID 1392] Termina el hilo 2
[PID 1392] Termina el hilo 3

[PID 1392] Valor leido: 1394
[PID 1392] Valor leido: 1396
[PID 1392] Valor leido: 1395
[PID 1392] Valor leido: 1393
```

Puedes comprobar cuál es el contenido del fichero con la orden `hexdump`:

```
$ hexdump -d fichero.txt
0000000   01394   00000   01396   00000   01395   00000   01393   00000
0000010
```