Práctica Obligatoria 1
Programación en C

Sistemas Operativos 2017/18 – Grado en Ingeniería del Software

Iván Pérez Huete – Carlos Olmo Shauquillo
5-17-2017

Tabla de contenido

1. Introducción

La práctica consiste en la implementación de una librería en el lenguaje de programación C y un programa principal que sirva para probar las funciones que se van a programar en dicha librería.

Dentro de esta librería se pide implementar las siguientes funciones:

1. Fución Head
2. Función Tail
3. Función Longlines

La descripción de estas funciones será detalla posteriormente. Adicionalmente se ofrece una explicación de la implementación y de las dificultades encontradas en el transcurso de la práctica.

Junto a este documento, que corresponde a la memoria de la primera practica obligatoria de la asignatura de Sistemas Operativos, se adjuntará los siguientes archivos:

Librería.h, la cual es proporcionada de antemano y contiene las cabeceras de las funciones a implementar.
Librería.c , la propia implementación de las funciones.
Test.c , programa de prueba.

2. Implementación de librería.c

2.1 Función Head

Cabecera de la función: **int head (int N);**

2.1.1 Descripción

Esta función tiene como objetivo emular el funcionamiento del comando head, el cual imprime por salida estándar el número de líneas indicado. Por defecto imprimirá las primeras 10 líneas de la entrada estándar.

2.1.2 Implementación

Para implementar esta función hemos utilizado un bucle 'while', que nos permitirá leer las líneas de la entrada estándar (stdin).
El bucle estará activo mientras siga habiendo líneas que leer por la entrada o no se produzca un End of File. Las líneas que lee el programa son almacenadas en un buffer de 1024, que será una longitud fija. Al ser declarada como constante es fácilmente modificable.
Después de leer la cadena, se imprime por salida entandar.

2.1.3 Comentarios adicionales

Es un ejercicio sencillo y que ya habíamos realizado con anterioridad en clase. Sirve para entender cómo funciona la entrada y salida estándar, así como los arrays. Ayuda también a comprender y familiarizarse con los distintos bucles y la sintaxis del lenguaje C.
Tardamos poco tiempo en realizar esta función.

2.2 Función Tail

Cabecera de la función: **int tail (int N);**

2.2.1 Descripción

Esta función tiene como objetivo imprimir las ultimas N líneas de la entrada estándar, simulando así el comportamiento del comando tail. Por defecto imprimirá las ultimas 10 líneas.

2.2.2 Implementación

Para la implementación de esta función usaremos memoria dinámica ya que no es como el método head, necesitamos recorrer todo el archivo que se nos pasa hasta llegar al final e imprimir las N líneas finales.
Para ello y gracias a la función malloc, reservaremos memoria dinámica en un puntero a puntero de char, que llamaremos 'pila' ya que actuará como una de estas. Al reservar memoria, también hay que hacerlo para las distintas líneas que leeremos, esto es importante y no hay que olvidarlo, ya que cuando tengamos que liberar memoria debemos liberar toda aquella que reservamos, tanto la de la pila como la de las líneas de esta.

Tras reservar la memoria a usar, comprobaremos que nuestra función 'malloc' no ha fallado y si se ha reservado la memoria bien pasaremos a el bucle central de la función 'tail'.

Este bucle se compone de un while que irá leyendo de la entrada estándar hasta acabar en un EOF, lo leído por la entrada se almacenará en la variable buff.

Posteriormente tendremos un par de comprobaciones para verificar que la N que se nos pasa no es mayor que el número de posiciones que vamos a tomar, y en el caso de que tomemos una posición mayor a 0, ósea si queremos mostrar alguna línea, entonces moveremos dentro del array las posiciones de las filas para que se pueda imprimir las líneas que entraron en último lugar. Nos apoyaremos tanto en la variable 'lista' como en 'buff' para "dar la vuelta" al array y poder mostrar las líneas del final.

Una resumida forma de ver lo anteriormente comentado sería la que podemos ver a continuación en la ilustración 1.

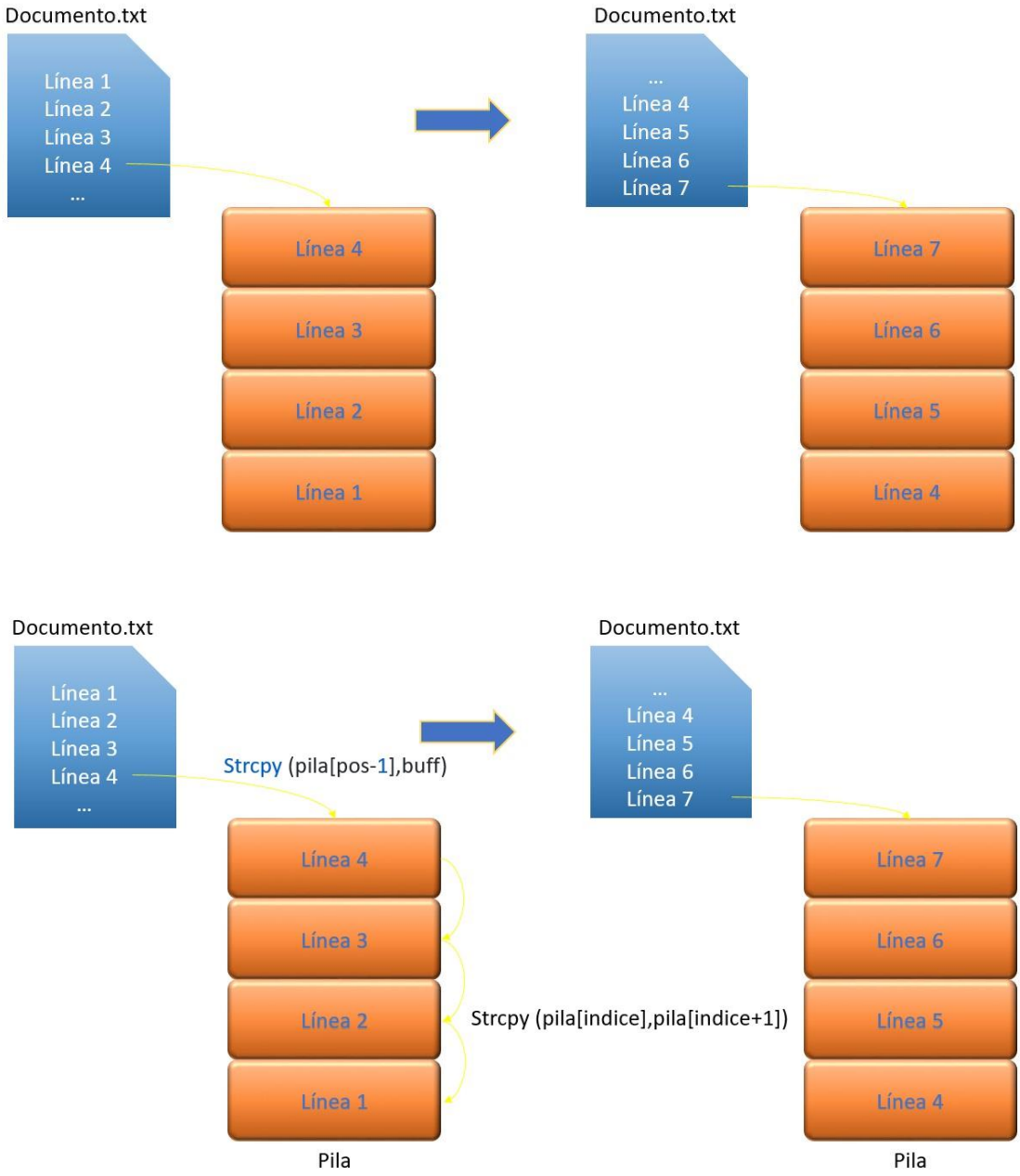

Ilustración 1

Para finalizar tendremos un bucle for, que recorrerá con un contador la pila resultante e imprimirá tantas líneas como N se le pasarán a la función.

Y, por último, como hemos comentado antes, debemos liberar la memoria reservada con anterioridad, haciendo uso de la función free. Liberaremos primero todas las reservas para las líneas del archivo ayudándonos de un bucle while y una variable auxiliar para recorrer y por último un free a la pila en cuestión.

2.2.3 Comentarios adicionales

Este ejercicio es un buen ejemplo a la hora de comprender el correcto uso de la memoria dinámica, ya que si lo entiendes y lo aplicas con esta clase de memoria es como más fácil vas a resolver el problema y además será de la forma más eficiente.

Hemos empleado bastante tiempo en este problema ya que la primera solución que tuvimos fue intentando realizar la función 'longlines' pero nos dimos cuenta a la mitad de esta que podíamos utilizar parte del código para la función 'tail' y así es como lo hicimos. Tardamos más porque hemos querido rehacer o reconstruir el código para que no se parezca a la otra función.

Por lo que la función 'Tail' resultante, por lo que hemos podido comprobar no es la más eficiente.

2.3 Función Longlines

Cabecera de la función: **int Longlines (Int N);**

2.3.1 Descripción

Esta función imprimirá por la salida estándar un numero N de líneas, las cuales serán las líneas de mayor longitud de la entrada estándar.
Imprimirá estas líneas de mayor a menor, simulando así el comando Longlines.

2.3.2 Implementación

A la hora de implementar esta función, hemos reservado memoria dinámicamente como hicimos en el método 'Tail'. así pues, reservamos memoria para la variable en la que tomaremos el array, que es este caso llamaremos 'lista'. También como hicimos en la función anterior debemos reservar memoria para cada una de las líneas de esa 'lista', esto lo haremos con un bucle for.

Posteriormente comprobaremos que el malloc de reserva de memoria no ha fallado y pasaremos al bucle principal en donde realizaremos el cambio de las líneas según su longitud.

Este cambio lo haremos ayudándonos de una variable auxiliar a la que llamaremos 'listaAux' de la que nos valdremos como lista que haga de puente entre la lista principal y el buffer. Así pues, según se va leyendo de entrada estándar comprobará si cada línea es mayor que la que ya había en la lista, entonces cambia la posición entre ellas.

Una resumida forma de verlo puede ser la que ofrecemos a continuación:

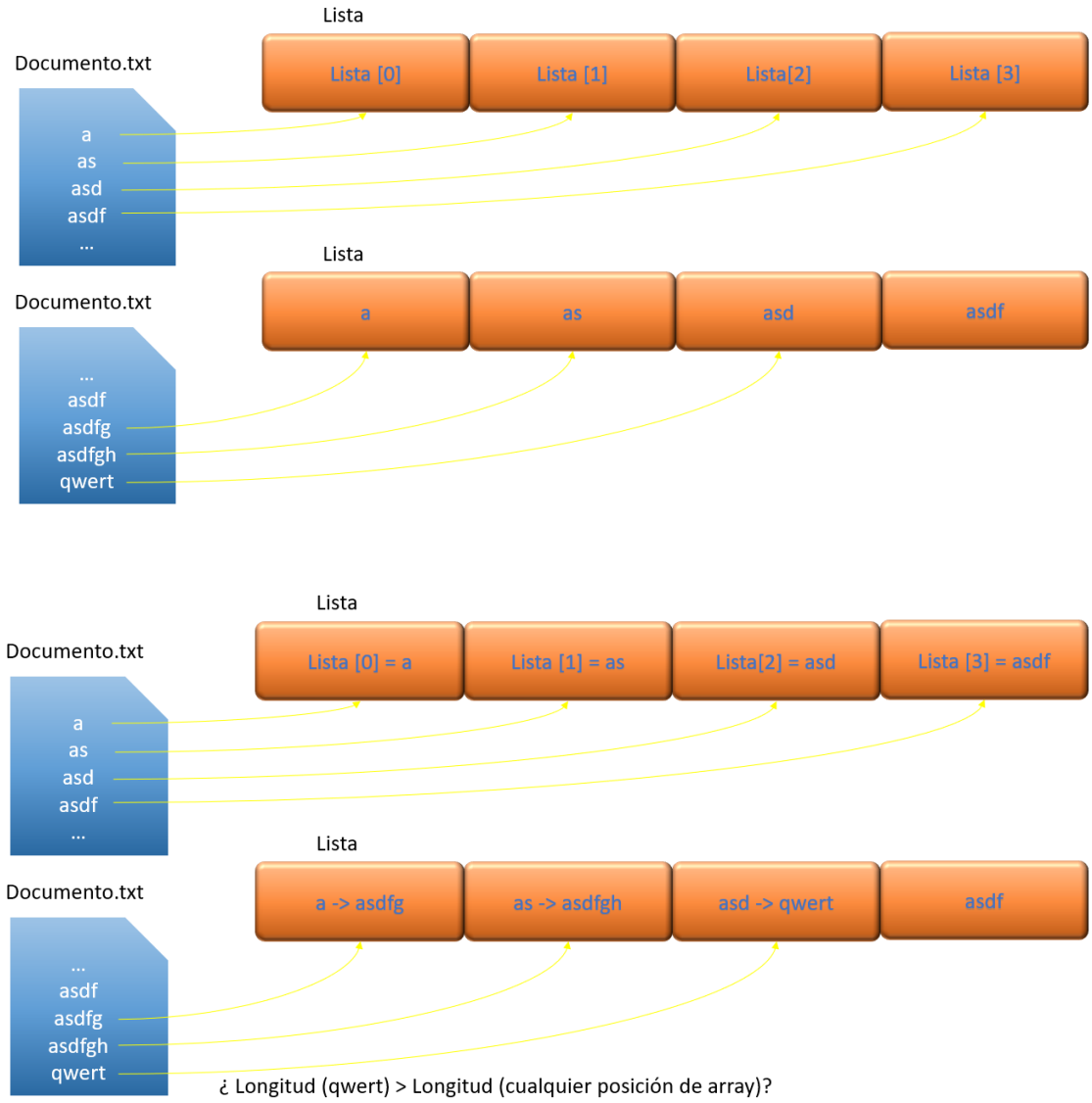

Ilustración 2

Dejando así al final la lista ordenada de mayor a menor por longitud de sus líneas.

Por último, tendríamos como las anteriores funciones, un for para imprimir por pantalla todo lo que se almacenó en la lista. Solo escribiendo la N líneas que se pidieron.

Y, para terminar, haremos uso de la función free para liberar memoria. Así como en la reserva de memoria lo hacemos tanto para la lista y sus líneas, a la hora de liberar memoria haremos lo mismo.

2.3.3 Comentarios adicionales

Esta función lleva más dificultad que las anteriores por lo que nos ha costado más tiempo realizarla que las anteriores.

El uso de la memoria dinámica es igual que en la función anterior. Pese a esto, hay distintas formas de implementar esta función, y hemos optado por la más sencilla.

3. Implementación de test.c

El documento test.c no es más que un documento de prueba. Se compone por un main en el que se nos mostrará un menú para saber los tres distintos tipos de funciones que puede realizar.
Posteriormente y según lo que le introduzcamos, realizará una u otra función.

A la hora de compilar, tendremos que compilar tanto la librería como el test a la vez.

gcc test.c librería.c -o test

En el test.c tendremos un subprograma al que llamaremos “metodoDecidir” el cual decidirá a que función debe de llamar en función de lo que le pasemos por la entrada estándar. Esto lo podremos hacer gracias a la función ‘strcmp’ que compara Strings.

Posteriormente decidiremos en función del número de argumentos que se pasan, si hay llamada a la función, si se llama a la función por defecto o si se la llama con un parámetro N.
Todo esto gracias al parámetro argc del main.

Si los parámetros no son correctos devolverá un error por pantalla.

4. Cambios de cara a junio

De cara a junio hemos tenido que hacer unas pequeñas modificaciones para cumplir con el estándar ANSI C, ya que el código si funcionaba y funcionaba bien, pero no cumplía los requisitos impuestos por dicho estándar.

En este apartado resumiremos de forma breve los cambios y si ha afectado a la realización de la práctica.

Primer cambio ha sido quitar un menú innecesario, ya que no se pedía en los requisitos de la practica añadir un menú para visualizar las distintas opciones.

Esto afectó a nuestro diseño, ya que lo que teníamos pensado era más genérico, en el sentido de que las 3 funciones cumplían los mismos requisitos, por lo que el programa principal era el mismo, solo se distinguía la función final que hacía.

Dado que no podíamos implementarlo así, hemos decidido tener un conjunto de “Ifs” que resuelvan ese problema, a la hora de distinguir las funciones. Es una solución menos flexible pero que se adapta al estándar perfectamente, ya que no tenemos una función dentro de otra función.

Por último, decidimos que una de nuestras variables, cuando el usuario no mete ningún numero a la hora de las líneas que quiere contar, siempre es 10. Por lo que lo pasamos a constante.