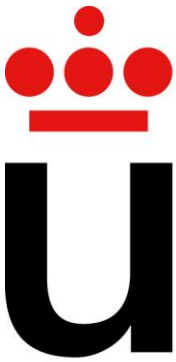

Universidad
Rey Juan Carlos

# Sistemas Operativos

## [PRÁCTICA 1 – PROGRAMACIÓN C]

GUILLERMO BLÁQUEZ BARBACID Y STEVEN ADRADOS KHUNLIANG

TABLA DE CONTENIDO

# Autores

**Guillermo Blázquez Barbacid**
– Ingeniería de Software
– g.blazquez.2019@alumnos.urjc.es
– DNI: 02753218A
**Steven Adrados Khuliang.**
– Ingeniería de Software
– s.adrados.2020@alumnos.urjc.es
– DNI: 50372801H

# Descripción del Código

**El código se compones de 4 archivos o librerías. Libreria.h que contiene las cabeceras de las funciones. Compila.sh que contiene la compilación de los programas. Test.c que tiene el main, control de errores y hace de ejecutable del programa. Libreria.c que contiene las principales funciones del sistema, Head, Tail y Longlines.**

## Diseño del Código

### Funciones de la librería

#### Función Head → Imprimir las N primeras líneas

Implementa un algoritmo de iteración para imprimir las primeras N líneas de entrada. Se trata de un bucle while que, con una variable contador, llevará la cuenta de cuántas líneas se van imprimiendo. En el momento en que se hayan impreso las N líneas, se dejará de leer y la función acabará.

#### Función Tail → Imprimir las N últimas líneas

Implementa un algoritmo de iteración para imprimir las ultimas N líneas de entrada. Hacemos una reserva de memoria para guardar las líneas que se pasan por entrada estándar solo guardando las líneas N, si se introducen más líneas de las que hay en N se borran las más antiguas y se introducen las nuevas de forma circular para evitar fugas en memoria.

Posteriormente se calcula el inicio de impresión, Si el total de líneas es mayor o igual a N, significa que el array de líneas está lleno y debe empezar a imprimir desde el índice correcto en el array circular. Si hay menos de N líneas, comenzará desde el principio.

Por último, se realiza la impresión de las líneas con un bucle y se libera memoria del array.

#### Función Longlines→ Imprimir las N últimas líneas

Esta función utiliza un algoritmo de ordenación y de búsqueda para encontrar las líneas más largas y ordenarlas de mayor a menor.

Tenemos un bucle while que redimensiona el array e introduce lo leído por entrada estándar, luego se ordenan por longitud con la función qsort y se calcula el límite a imprimir.

Se imprime cada una de las N líneas o todas si son menos que N y posteriormente se libera memoria del array.

### Programa Test

En este programa se hacen las comprobaciones necesarias para probar que se llama de forma correcta (. /test - [número de líneas]" o ". /test -").

La función main recibe dos argumentos; argc (número de argumentos) y argv (arreglo de strings que contiene los argumentos).

Realizamos una comprobación de argumentos para comprobar que se introduce entre uno y dos argumentos, el primero sería el nombre de la función que queremos realizar:

-head(N): Muestra las primeras N líneas de un archivo.

-tail(N): Muestra las últimas N líneas de un archivo.

-longlines(N): Muestra las N líneas más largas en el archivo.

Si no coincide con estas muestra un mensaje de error.

El segundo argumento es un N número positivo, si se le pasa otro tipo muestra un mensaje de error.

## Principales Funciones

| | **Head** | **Nombre** | **Tipo** | **Descripción** |
|---|---|---|---|---|
| **Argumentos** | Argumento 1 | **N** | **entero** | **Un entero que especifica a nuestro comando cuantas líneas de texto ha de imprimir teniendo en cuenta que las que imprima serán las primeras en orden descendente.** |
| **Variables Locales** | Variable 1 | **Buffer** | **carácter** | **Es un array de caracteres o más bien en nuestro caso líneas que ocupan cada una 1024 bit.** |
| | Variable 2 | **contador** | **entero** | **Es simplemente un contador para nuestro bucle. El cual es usado en la sentencia condicional del mismo para saber la línea por la que se va.** |
| **Valor Devuelto** | | | **entero** | **Se devuelve un 0 pues si se está ejecuta es que ha pasado el control de errores.** |
| **Descripción de la Función** | | | | **Esta función, como indicado en el texto de la práctica, recibe un número y este número será el número de líneas de texto que se mostrarán de un texto por la parte del principio en orden descendente.** |

| | **Tail** | **Nombre** | **Tipo** | **Descripción** |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| **Argumentos** | Argumento 1 | **N** | **entero** | **Un entero que especifica a nuestro comando cuantas líneas de texto ha de imprimir.** |
| **Variables Locales** | Variable 1 | **lines** | **carácter** | **Es un puntero a un puntero para reservar memoria para nuestro array** |
| | Variable 2 | **contador** | **entero** | **Es un contador usado en los bucles para saber en qué línea vamos y como condición en las sentencias condicionales de los mismos** |
| | Variable 3 | **Buffer** | **carácter** | **Es un array de caracteres o más bien en nuestro caso líneas que ocupan cada una 1024 bit.** |
| | Variable 4 | **start** | **entero** | **Es una variable que se usa para saber por dónde se empezara a imprimir las líneas.** |
| | Variable 5 | **numlins** | **entero** | **Es el número de líneas que se imprimirán por parte del comando** |
| **Valor Devuelto** | | | **Entero** | **Se devuelve un 0 pues si se está ejecuta es que ha pasado el control de errores.** |
| **Descripción de la Función** | | | | **Es, por así decirlo, la contraparte de la función head. Esta función recibe un número y con el texto sobre el que se actúa se imprimen el número de líneas previamente proporcionado empezando por el final y de manera ascendente** |

| | **longlines** | **Nombre** | **Tipo** | **Descripción** |
|---|---|---|---|---|
| **Argumentos** | Argumento 1 | **N** | **entero** | **Un entero que especifica a nuestro comando cuantas líneas de texto ha de imprimir.** |
| **Variables Locales** | Variable 1 | **lines** | **Carácter** | **Es un puntero a un puntero para reservar memoria para nuestro array, en este caso lo inicializamos a null.** |

| | | | | |
|---|---|---|---|---|
| | Variable 2 | **line_count** | **entero** | **Es un contador que registra en los bucles por qué línea se va.** |
| | Variable 3 | **Buffer** | **carácter** | **Es un array de caracteres o más bien en nuestro caso líneas que ocupan cada una 1024 bit.** |
| | Variable 4 | **limite** | **entero** | **Es una variable que nos ayuda a saber hasta donde se ha de imprimir en el caso de longlines.** |
| **Valor Devuelto** | | | **entero** | **Se devuelve un 0 pues si se está ejecuta es que ha pasado el control de errores.** |
| **Descripción de la Función** | | | | **La función imprime por pantalla un numero n de líneas que corresponde con el numero n de la variable proporcionada. Estas líneas son las que más longitud tienen y son mostradas en orden descendente de tamaño.** |

| | **Main** | **Nombre** | **Tipo** | **Descripción** |
|---|---|---|---|---|
| **Argumentos** | Argumento 1 | **argc** | **entero** | **Es una variable que contiene el número de argumentos que se le pasaran por la Shell a nuestros comandos** |
| | Argumento 2 | **argv** | **Array de caracter** | **Es un array que apunta a los diversos contenidos literales que se le proporcionan a nuestro comando como argumento.** |
| **Variables Locales** | Variable 1 | **N** | **entero** | **Es el número que se le pasará como entrada/argumento a nuestras funciones. Está inicializado a 10.** |
| **Valor Devuelto** | | | **entero** | **Depende del éxito o error de la ejecución. Si la ejecución es idónea se devuelve un 0 y en caso contrario un 1 por error estándar y 2 por error de tipos de argumento o errores más específicos.** |

| | | | | |
|---|---|---|---|---|
| **Descripción de la Función** | | | | **Esta función hace de organizante para redirigir al resto de funciones y como filtro para los argumentos. En el caso del filtrado, es para asegurarse de que cumplen los argumentos requeridos, ni más ni menos.** |

# Casos de Prueba

```
plantitas@plantitas-VirtualBox:~/Escritorio/Practica$ ./test
Uso: ./test -<función> [número de líneas]
-head -tail -longlines
```

**Introduccion de menos argumentos de los necesarios**

```
plantitas@plantitas-VirtualBox:~/Escritorio/Practica$ ./test -patata
Función desconocida: -patata
```

**Introducción de nombre de función errónea.**

```
plantitas@plantitas-VirtualBox:~/Escritorio/Practica$ ./test -tail -5
El número de líneas debe ser un entero positivo.
```

**Introducción de un número de líneas negativo.**

```
plantitas@plantitas-VirtualBox:~/Escritorio/Practica$ ./test -head 4 a a
Uso: ./test -<función> [número de líneas]
-head -tail -longlines
```

**Introducción de más argumentos de los necesarios.**

```
alumno@alumno:~/Desktop/practica1CGGSAK$ ./test -head 3 <ejemplo.txt
Ecos del Tiempo

En el susurro del viento, se oyen voces,
alumno@alumno:~/Desktop/practica1CGGSAK$ ./test -tail 3 <ejemplo.txt
pues en cada palabra hay un mundo querido,
y en cada poema, un eco de nuestros llenos.

alumno@alumno:~/Desktop/practica1CGGSAK$ ./test -longlines 3 <ejemplo.txt
Bajo el cielo estrellado, danzan los sueños,
sus cumbres son testigos de amores y guerras,
historias de antaño que el tiempo no borra,
```

**En el caso de nuestra práctica, hemos realizado un archivo de texto llamado ejemplo.txt , el cual es un poema. Este poema tiene varias partes a modo de párrafos y cuenta con una estructura perfecta para probar las distintas funcionalidades de los comandos que se han implementado como longlines o por ejemplo tail/head.**

# Comentarios Personales

## PROBLEMAS ENCONTRADOS

Uno de los problemas más grandes, como se explica en el apartado de tiempo dedicado, es que no usamos de manera correcta durante un tiempo la gestión de memoria. No es un error como tal, mas es un error de compresión, que nos impedía realizar correctamente la actividad. La solución, ha sido informarnos más sobre los procesos y el funcionamiento de c y la teoría.

## CRÍTICAS CONSTRUCTIVAS

Gracias a esta práctica he aprendido de los problemas de uso de memoria de C, de cómo reservar memoria, como aumentar memoria de un array ya guardado, y como liberar, permitiéndonos entender el funcionamiento y la diferencia entre un lenguaje de bajo nivel y alto nivel.

## PROPUESTA DE MEJORAS

Explicación inicial sobre bibliotecas y enlaces: Habría sido útil contar con una guía detallada al inicio sobre cómo enlazar y crear bibliotecas. Nos habría ayudado a entender mejor el flujo de trabajo y evitar problemas técnicos al principio.

Más explicaciones sobre el uso de memoria: Esta fue la parte más complicada de la práctica de lejos. El manejo de memoria en lenguajes del tipo de C es demasiado importante como para prescindir de entendimiento perfecto de este.

## EVALUACIÓN DEL TIEMPO DEDICADO

En general el tiempo dedicado a la práctica, en nuestra opinión, es entre un tiempo medio a uno medio-alto comparado con el resto de las actividades y diversas prácticas de carácter parecido que se nos ha encomendado en la universidad. Por lo general la única pega que más nos ha consumido el tiempo con lo que respecta a esta práctica son nuestros propios errores. Nuestros propios errores de compresión han causado un mayor tiempo de desarrollo, es decir, lo que más hemos encontrado problemático es el uso de la memoria en c puesto que es un tema que hay que entender en su completitud para usarlo correctamente.

Por lo que, en conclusión, la consumición de tiempo ha sido más dada por la dificultad del temario más que la laboriosidad de la tarea encomendada en sí.