

Sistemas Operativos

[Práctica 1 – PROGRAMACIÓN C]

Inés Alonso Izquierdo y Marcos Ferrer Zalve

Tabla de contenido

[Autores 2](#_Toc115812632)

[Descripción del Código 3](#_Toc115812633)

[Diseño del Código 3](#_Toc115812634)

[Principales Funciones 3](#_Toc115812635)

[Casos de Prueba 3](#_Toc115812636)

[Comentarios Personales 4](#_Toc115812637)

# Autores

|  |  |  |
| --- | --- | --- |
| Inés Alonso Izquierdo | 54663934H | [i.alonso.2020@alumnos.urjc.es](mailto:i.alonso.2020@alumnos.urjc.es) |
| Marcos Ferrer Zalve | 20619602X | [m.ferrer.2020@alumnos.urjc.es](mailto:m.ferrer.2020@alumnos.urjc.es) |

# Descripción del Código

## Diseño del Código

Antes de adentrarnos en la explicación del código en sí, hemos de destacar que decidimos declarar una estructura **node\_t** que básicamente serán nuestros nodos en la lista doblemente enlazada que utilizamos a lo largo de toda la implementación. Estos nodos cuentan con el valor en la posición y con dos punteros, uno apuntando al nodo previo al actual y otro al siguiente. A continuación, procederemos a explicar el código de las funcionalidades requeridas en la práctica y las funciones que hemos añadido para facilitar la implementación.

### Función check\_errors(N)

En esta función se comprobará si el argumento N es un entero positivo. En caso de no serlo, la función devolverá el valor 1, lo que significa que el dato introducido es erróneo. En el caso contrario, se devolverá el valor 0. Se trata de una función *static* ya que solo la podrán usar los métodos de la *libreria.c*.

### Función isFeasible(check, list)

Esta función es utilizada en el método longlines(), que explicaremos más adelante, para saber si añadir un valor a la lista de elementos más largos o no, es decir, nos devuelve si el argumento es mayor que el más pequeño de la lista de elementos más largos. También hemos considerado que sea *static,* pues solo la utilizaremos dentro de la librería.

### Función insert (val, list, len, max)

En este caso, tomamos la lista dinámica, un valor, la longitud de la lista y el tamaño máximo(N) y añadimos el elemento en la posición adecuada para que esté ordenada en ascendentemente y no supere el tamaño máximo. Esto lo realizamos de la siguiente forma:

Primero, comprobamos si la lista está vacía. Si es así, simplemente se pone el valor en el primer nodo y se aumenta el tamaño en uno. En caso de no estar vacía, se itera la lista usando dos punteros. Moveremos **curr** hasta que el valor tenga una longitud mayor a la longitud de la linea leída. El puntero **prv** apuntara a la posición anterior a curr. Después, crearemos un nuevo nodo con el valor deseado y lo insertaremos entre prev y curr. En caso de que la longitud de la lista fuera menor a N antes de este proceso, se incrementará su valor en 1. Si por el contrario el valor ya era igual a N, moveremos la cabecera de la lista una posición a la izquierda, ya que la primera posición siempre será la más pequeña de la lista.

### Función head(N)

Primero llamamos a la función **check\_errors** y le pasamos N para ver si el argumento que nos pasan es válido. Creamos un nodo **head** como inicio de la lista, reservamos memoria para él y apuntamos **prev** y **next** a *null* para inicializar el nodo. Después, crearemos un nodo auxiliar **aux,** con el que iremos añadiendo los elementos en la lista doblemente enlazada, y el nodo **current,** el cual utilizaremos para reservar el último nodo añadido a la lista. Por ello, lo apuntamos al principio de la lista (**head**).

Entonces, empezamos a leer las líneas hasta llegar a la enésima, y si estas no son nulas, creamos un nodo **aux**, lo enlazamos a la lista y actualizamos el valor de **current** al nuevo nodo. Si nos quedamos sin líneas que leer antes de llegar a la enésima, se sale del bucle y se imprimirían todas las líneas del argumento. Finalmente, liberamos la memoria de **aux** y recorremos la lista para imprimir los nodos que hemos almacenado. Una vez impresos, devolvemos 0 para indicar que la función ha sido ejecutada con éxito.

### Función tail(N)

Primero llamamos a la función **check\_errors** y le pasamos N para ver si el argumento que nos pasan es válido. Creamos un nodo **first** para poder almacenar el primer nodo de la lista doblemente enlazada, reservamos memoria par él y lo inicializamos. También crearemos un nodo auxiliar **aux** y un nodo **current** como en la función **head**. En este caso apuntamos **current** a **first** y comenzaremos a leer la entrada.

Entonces, añadimos un nuevo nodo a la lista enlazada, pero, si nuestro contador **i** (el cual aumentamos mientras sea menor que **N**) ha sobrepasado **N**, eliminamos el primer nodo de la lista y actualizamos **first**. De este modo, al terminar de recorrer la entrada, tendremos **N** nodos con las ultimas líneas. Finalmente vamos imprimiendo igual que en **head.**

### Función longlines(N)

## Principales Funciones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **main** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **argc** | **int** | **Número de argumentos de la entrada** |
|  | Argumento 2 | **argv** | **Puntero a array de strings** | **Argumentos de la entrada estándar.** |
| **Variables Locales** | Variable 1 | **N** | **int** | **Número de líneas que se deberán mostrar por la salida estándar.** |
|  | Variable 2 | **h** | **string** | **Variable auxiliar** |
|  | Variable 3 | **t** | **string** | **Variable auxiliar** |
|  | Variable 4 | **ll** | **string** | **Variable auxiliar** |
| **Valor Devuelto** |  |  | **int** | **Devuelve 0 si se ha ejecutado con éxito y un valor entre 1 y 5 si ha habido algún error.** |
| **Descripción de la Función** | **Función gestora de errores y la responsable de ejecutar las funciones head, tail y longlines.** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **check\_errors** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas que se deberán mostrar por la salida estándar.** |
| **Variables Locales** | - | **-** | **-** | **-** |
| **Valor Devuelto** |  |  | **int** | **Si N < 0 devuelve 1, si no, devuelve 0.** |
| **Descripción de la Función** | **Comprueba errores en el argumento N.** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **isFeasible** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **check** | **char** | **Línea candidata a entrar en la lista enlazada.** |
|  | Argumento 2 | **list** | **node\_t** | **Lista enlazada.** |
| **Variables Locales** | - | **-** | **-** | **-** |
| **Valor Devuelto** |  |  | **bool** | **Devuelve si el valor es mayor que el más pequeño de la lista.** |
| **Descripción de la Función** | **Lo utilizamos en longlines para saber si añadir un nodo o no a la lista enlazada, es decir, si el nodo es más grande que algún nodo almacenado en esta.** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **insert** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **val** | **char** | **Elemento por añadir** |
|  | Argumento 2 | **list** | **node\_t** | **Lista enlazada** |
|  | Argumento 3 | **len** | **int** | **Longitud de la lista** |
|  | Argumento 4 | **max** | **int** | **Tamaño máximo** |
| **Variables Locales** | Variable 1 | **curr** | **node\_t** | **Nodo siguiente a la posición en la que debemos insertar el elemento** |
|  | Variable 2 | **prv** | **node\_t** | **Nodo previo a la posición en la que debemos insertar el elemento** |
|  | Variable 3 | **newNode** | **node\_t** | **Nodo nuevo creado con el valor a insertar** |
| **Valor Devuelto** |  |  | **node\_t** | **Si no ha habido errores, devuelve la lista con el nodo nuevo. En caso de error, devuelve NULL para detectar este error.** |
| **Descripción de la Función** | **Insert es una función usada en longlines(N) que se encarga de buscar la posición en la que debemos añadir un nodo a la lista para que contenga las N lineas más grandes, ordenadas por tamaño. También comprueba si ha de sacar elementos (en caso de que len == max) o aumentar el tamaño de la lista** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **head** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas que se deberán mostrar por la salida estándar.** |
| **Variables Locales** | Variable 1 | **head** | **node\_t** | **Inicio de la lista doblemente enlazada.** |
|  | Variable 2 | **aux** | **node\_t** | **Nodo auxiliar para añadir nodos.** |
|  | Variable 3 | **current** | **node\_t** | **Último nodo añadido a la lista.** |
| **Valor Devuelto** |  |  | **int** | **Devuelve 0 si todo ha salido bien, 1, si el argumento es erróneo y 2 si no se ha realizado correctamente un malloc.** |
| **Descripción de la Función** | **Se mostrarán las N primeras líneas de la entrada estándar.** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **tail** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas que se deberán mostrar por la salida estándar.** |
| **Variables Locales** | Variable 1 | **first** | **node\_t** | **Primer nodo de la lista doblemente enlazada.** |
|  | Variable 2 | **i** | **int** | **Contador.** |
|  | Variable 3 | **aux** | **node\_t** | **Nodo auxiliar para añadir nodos.** |
|  | Variable 4 | **current** | **node\_t** | **Último nodo añadido a la lista.** |
| **Valor Devuelto** |  |  | **int** | **Devuelve 0 si todo ha salido bien, 1, si el argumento es erróneo y 2 si no se ha realizado correctamente un malloc.** |
| **Descripción de la Función** | **Se mostrarán las N últimas líneas de la entrada estándar** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **longlines** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas que se deberán mostrar por la salida estándar.** |
| **Variables Locales** | Variable 1 | **head** | **node\_t** | **Inicio de la lista doblemente enlazada.** |
|  | Variable 2 | **aux** | **node\_t** | **Borrar la lista tras su uso.** |
|  | Variable 3 | **tail** | **node\_t** | **Final de la lista doblemente enlazada.** |
|  | Variable 4 | **str** | **char** | **String auxiliar.** |
|  | Variable 5 | **size** | **int** | **Tamaño de la lista.** |
| **Valor Devuelto** |  |  | **int** | **Devuelve 0 si todo ha salido bien, 1, si el argumento es erróneo y 2 si no se ha realizado correctamente un malloc.** |
| **Descripción de la Función** | **Se mostrarán las N líneas más largas de la entrada estándar, o todas ellas si hay menos de N líneas, ordenadas de mayor a menor.** | | | |

## Casos de Prueba

**<< LISTA Y BREVE EXPLICACIÓN DE CASOS DE PRUEBA UTILIZADOS PARA VALIDAR LA PRÁCTICA >>**

**//explicar un poco lo que hay en el test (mas extenso abajo)**

# Comentarios Personales

**<< MÁXIMO 2 PÁGINAS >>**

**<< PROBLEMAS ENCONTRADOS>>**

**<< CRÍTICAS CONSTRUCTIVAS >>**

**<< PROPUESTA DE MEJORAS >>**

**<< EVALUACIÓN DEL TIEMPO DEDICADO >>**

**En general, no hemos encontrado muchos problemas en la implementación, pero si que es verdad que nos ha costado organizar el espacio de la memoria, pues al tener 3 funciones extras hemos tenido que explicarlas y ponerlas en las tablas y nos hemos quedado sin espacio. Por ello, hay más páginas de las que se requerían y hemos tenido que descartar los diagramas de apoyo que habíamos hecho de las funciones head, tail y longlines. Cabe destacar que hemos comentado cada función en el propio código por si quedara alguna duda tras la explicación en el documento.**

**En cuestión al tiempo, hemos empleado unas 15 horas para elaborar el código de las librerías y el test y unas 3 horas para completar la memoria, lo cual es lo que más o menos habíamos estimado que íbamos a tardar en un principio.**

**//explicar lo del script de Shell y el test aquí un poco más extenso**