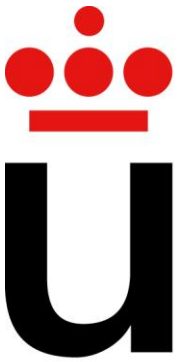


Universidad
Rey Juan Carlos

Sistemas Operativos

[PRÁCTICA 1 – PROGRAMACIÓN C]

ALBERTO GARCÍA SRODA & IVAN SÁNCHEZ LABRADOR

TABLA DE CONTENIDO

Autores

Iván Sánchez Labrador

Alberto García Sroda

Descripción del Código

Diseño del Código

El código busca realizar 3 funciones principales y proteger a las mismas de posibles errores en caso de que el usuario introduzca demasiados valores o estos sean incorrectos. Nuestro programa consta de un Main desde el cual se leen los parámetros introducimos y tenemos unos primeros controles de errores en caso de meter demasiados argumentos. Tras esto comprobamos que se haya elegido una opción correcta, y si es el caso se selecciona una de las 3 funciones, siendo la primera la que imprime el valor de las n primeras líneas en caso de que introduzcamos un número de líneas positivo, la segunda que nos imprime las n ultimas líneas, o, por último, la tercera, que nos muestra por orden de número de caracteres por línea de mayor a menor las n líneas que hayamos introducido.

Partiendo de esta base tenemos primero nuestro fichero test.c, que consta de nuestro Main mencionado previamente. Este tiene una estructura simple con varios if para los casos de tener más o menos argumentos de los que esperábamos, tras ello asignamos el parámetro de la opción introducida y la valoramos en un switch-case, donde si coincide con 1,2 ó 3, se llamará a la función head, tail o longlines respectivamente, en caso contrario se nos dirá por pantalla.

En el head, al querer mostrar las n primeras líneas, coincidimos en no necesitar memoria dinámica, y al no saber cuándo se produce el *End Of File,* el bluce elegido fue un while que leyera hasta que la línea en cuestión fuera null y simplemente imprimiéramos las n que queríamos. Añadimos la funcionalidad equivalente a introducir head -n *número negativo *, para la cual necesitamos memoria dinámica para poder imprimir las líneas totales menos las n líneas finales. Por lo tanto, usamos otro while al no conocer el número de líneas a leer con anterioridad y nos creamos un array de strings dinámico que creciera por cada iteración del while. Tras esto solo tenemos que mostrar las líneas hasta ese límite que hayamos introducido.

En el tail tenemos que imprimir las n últimas líneas, por lo tanto, no sabemos cuántas vamos a tener. Sabemos que necesitamos reservar memoria dinámica en un array de strings de longitud n, ya que no tenemos que quedarnos con todas las líneas nos introduzcan, solo con esas últimas, y cuando este array se llene, si tenemos otra línea, pasamos todos los elementos una posición hacia atrás e introducimos en la última posición que ahora ha quedado vacía.

Por ello tenemos un while nuevamente, un contador, y un if al que solo se entra cuando este array se llene y tengamos la siguiente línea, para hacer lo que hemos mencionado de pasar los elementos atrás y añadir la última posición.

En nuestro longlines intentamos realizar una implementación que solo guardara n líneas, las rellenara, y, de forma similar al tail, cuando estuvieran llenas mirase la longitud de la nueva línea leída y si esta era mayor que la última, eliminara esta, guardara la nueva y la volviese a ordenar. Pero nos resultó en muchos problemas y por ello decidimos hacerlo de una forma más eficiente a nivel de cómputo, pero menos eficiente a nivel de memoria, al guardarse todas las líneas que leemos, ordenarlas y luego imprimir las n que queríamos. La ordenación la hicimos con un método de la burbuja formado por dos for y la lectura nuevamente con un while.

Principales Funciones

	Main	**Nombre**	**Tipo**	**Descripción**
Argumentos	Argumento 0	**El que le asignes al ejecutable**	**String**	**Es el nombre del ejecutable**
	Argumento 1	**option**	**String**	**Según introduzcas 1,2 o 3 se ejecutará una función u otra**
	Argumento 2	**lines**	**String**	**Será el valor de n que le pasaremos a cada función**
	Argumento 3	**ninguno**	**ninguno**	**Se mostrará mensaje de número de argumentos incorrecto si metemos más de 3 (0,1,2)**
Variables Locales	Variable 1	**arguments**	**int**	**Almacena el número de argumentos introducidos gracias a argc**
	Variable 2	**lines**	**int**	**Transformamos el argv[2] a int para obtener el número de líneas deseadas por el usuario.**
	Variable 3	**option**	**int**	**Transformamos el argv[|1] a int para saber que función desea ejecutar el usuario 1, 2 o 3 y si es mayor se muestra mensaje de error.**
Valor Devuelto				
Descripción de la Función	Archivo del programa principal que debe ser compilado y ejecutado junto con librería.c y librería.h. Disponemos de un control de errores para la selección de la funcion deseada. 1 para la función head, 2 para la función tail y 3 para la función longliness			

	Head	**Nombre**	**Tipo**	**Descripción**
Argumentos	Argumento	**n**	**int**	**Es el número de lineas introducidas por el usuario o las 10 por defecto si no introduce nada**

Variables Locales	Variable 1	**lineBuffer**	**char**	**Va almacenando la cadena de caracteres actual para ir usándola según se necesite.**
	Variable 2	**counter**	**int**	**Variable para ir contando las líneas introducidas**
	Variable 3	**i**	**int**	**Variable de control para los bucles**
	Variable 4	**length**	**int**	**Variable que controla la longitud de las cadenas.**
	Variable 5	**topLimit**	**int**	**Variable de control en caso de que nos pasen un número negativo para imprimir desde o hasta counter - n**
	Variable 6	**list**	**Char****	**Variable para ir almacenando valores**
Valor Devuelto			**int**	**Ejecución Satisfactoria**
Descripción de la Función	Lee líneas de la entrada estándar y las muestra según el número introducido por el usuario.			

	Tail	**Nombre**	**Tipo**	**Descripción**
Argumentos	Argumento	**n**	**Int**	**Es el número de lineas introducidas por el usuario o las 10 por defecto si no introduce nada**
Variables Locales	Variable 1	**lineBuffer**	**char**	**Va almacenando la cadena de caracteres actual para ir usándola según se necesite.**
	Variable 2	**counter**	**int**	**Variable para ir contando las líneas introducidas**
	Variable 3,4	**j,i**	**int**	**Variables para el control de bucles**
	Variable 5	**length**	**int**	**Variable que controla la longitud de las cadenas.**
	Variable 6	**list**	**Char****	**Variable para ir almacenando valores**

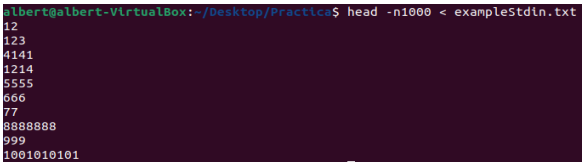
Valor Devuelto			**int**	**Ejecución satisfactoria**
Descripción de la Función	Muestra las n últimas líneas introducidas por la entrada estándar, almacena siempre las n últimas únicamente para ser más eficientes y no tener que almacenar todas las introducidas.			

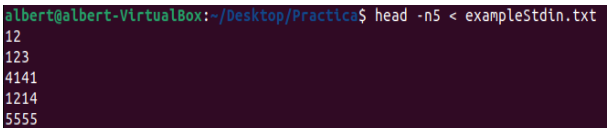
	Longlines	**Nombre**	**Tipo**	**Descripción**
Argumentos	Argumento 0	**n**	**Int**	**Es el número de lineas introducidas por el usuario o las 10 por defecto si no introduce nada**
Variables Locales	Variable 1	**lineBuffer**	**char**	**Va almacenando la cadena de caracteres actual para ir usándola según se necesite.**
	Variable 2	**list**	**Char****	**Variable para ir almacenando valores**
	Variable 3	**counter**	**int**	**Variable para ir contando las líneas introducidas**
	Variable 4,5	**I,k**	**int**	**Variables para el control de bucles**
	Variable 6	**aux**	**Char***	**Variable para almacenar temporalmente un valor a la hora de ordenar.**
	Variable 7	**origin**	**int**	**Variable de control a la hora de imprimir la lista de una forma u otra.**
Valor Devuelto			**int**	**Ejecución satisfactoria**
Descripción de la Función	**Muestra de mayor a menor longitud las n líneas introducidas por el usuario. Para ello se ordenan en una lista y luego se recorre desde counter hasta origin.**			

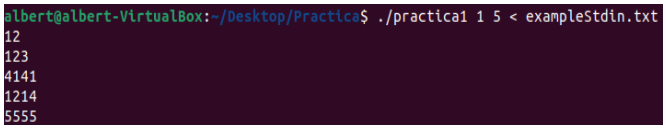
Casos de Prueba

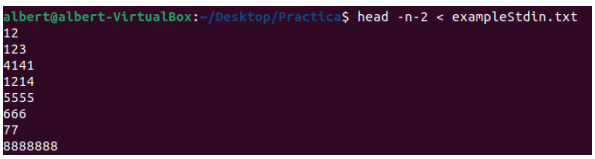
Para los casos de prueba de todos los apartados siempre probamos con valores límite, por ello tenemos positivos, negativos, iguales, menores y mayores a la longitud de los ficheros, así como exceso de argumentos y argumentos erróneos. Algunos de estos ejemplos serían:

Para el Head:

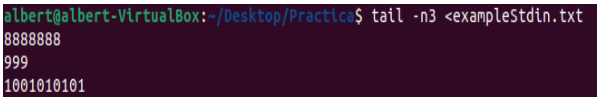




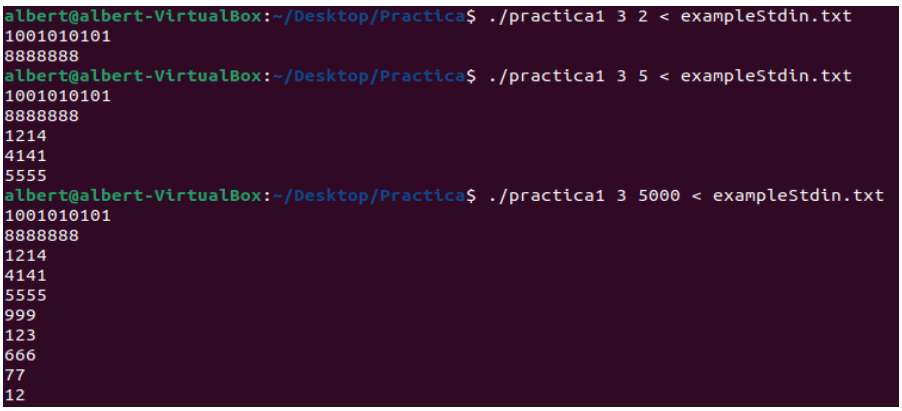




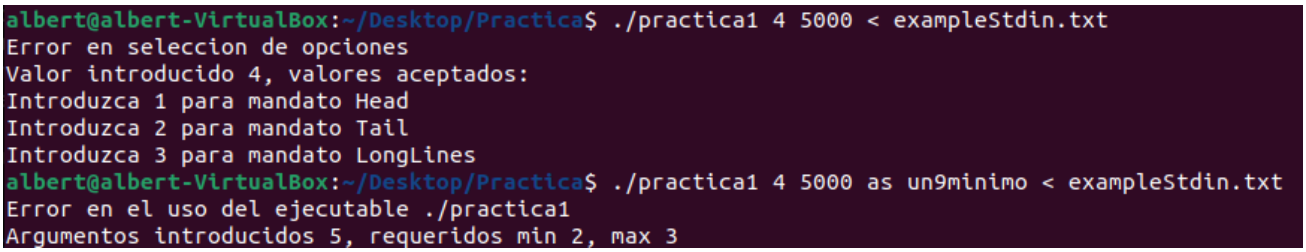
Para el Tail:



Para el LongLines:



En caso de meter demasiados argumentos u opciones no válidas:



Comentarios Personales

PROBLEMAS ENCONTRADOS

Dentro de los problemas que más han lastrado nuestro desarrollo de la práctica tenemos como principales la relativamente compleja entrada inicial a C como lenguaje de programación y el hecho del trabajo con memoria dinámica. Me gustaría poder cuantificar la de veces que hemos tenido Segmentation Fault en nuestra consola, pero nos pasaríamos del límite de páginas de la memoria seguramente.

En cuanto a compleja entrada inicial al lenguaje, tuvimos problemas para poder crear un main que se enlazara con los otros ficheros .c, nos costó poder empezar a simplemente probar nuestro programa y hacer que este tuviera algo que mostrar, lo cual fue interesante ya que al tener buenos conocimiento en Java, JavaScript y Python nos pensábamos que todo sería mucho más sencillo.

En este caso pudimos comprobar que no era simplemente sintaxis diferente y ya, se nota definitivamente que estamos en un nivel más bajo en cuanto al lenguaje se refiere.

CRÍTICAS CONSTRUCTIVAS

La práctica me parece una muy buena iniciación al lenguaje, nos ha dado soltura en c y lo que se pedía era asequible y estaba bien explicado, por lo tanto y hasta para nuestra sorpresa, no tenemos nada que criticar de ella.

PROPUESTA DE MEJORAS

No tenemos ninguna propuesta muy sugerente, como mucho al principio dudamos sobre qué cantidad de parecido debían de tener las funciones head y tail, ya que, al comprobar algunos valores, como al introducir head -n -3, nos dimos cuenta de que no solo devolvía las n primeras líneas, si no su funcionalidad era diferente. Por ello en parte la implementamos al considerar que este caso era demasiado común como para que no fuera necesario, y luego nos confirmaron en la tutoría que no era necesario, pero sí bienvenido.

Pese a ello es una trivialidad y no creo que haya mucho margen de mejora.

EVALUACIÓN DEL TIEMPO DEDICADO

En general hicimos más a medida que se nos acercaba la fecha de entrega, tuvimos 2 tutorías clave que nos resolvieron dudas importantes y tras las cuales pudimos completar gran parte de las funciones que nos faltaban o mejorar las que ya teníamos. Como mucho podríamos decir que dejamos el longlines demasiado para el final confiando en que sería mucho más parecida a la función tail, pero logramos tenerlo todo hecho sin mucha demora o presión por no poder acabarlo.

Por lo tanto, estamos muy satisfechos del trabajo realizado y de cómo lo hemos llevado a cabo.