PRÁCTICA 1: LLAMADAS AL SISTEMA OPERATIVO

SISTEMAS OPERATIVOS

PABLO AMOR MOLINA	100495855
SERGIO VERDE LLORENTE	100495899
HUGO CUEVAS ROMERA	100495962

8 DE MARZO DE 2024

UNIVERSIDAD CARLOS III DE MADRID

ÍNDICE

Descripción

Esta práctica consiste en la creación de una serie de funciones relacionadas con las llamadas al sistema operativo siguiendo el estándar POSIX, apoyandose en llamadas sobre ficheros como *open*, *read*, *write*, *lseek* o *close*; o llamadas sobre directorios como *opendir*, *readdir* o *closedir*.

Nuestra labor consiste en implementar tres programas en el lenguaje C que se apoyen en las llamadas al sistema anteriormente mencionadas. Estos programas se llaman **mywc**, **myls** y **myishere**, y sus funciones son las siguientes:

mywc

En esta función debemos abrir un fichero como argumento y contar el número de líneas, palabras y bytes que contenga este mismo, mostrando el resultado por la salida estándar.

Para llevar a cabo esta función, primero hemos debido declarar las variables de conteo, además de la variable identificativa del fichero y una variable booleana que usaremos posteriormente.

Tras las declaraciones, hemos hecho una serie de comprobaciones para evitar posibles errores de lectura, apertura de ficheros o escasez de argumentos. Una vez comprobado y evitado los posibles errores, comenzamos con la función principal comprobando el primer byte del fichero, teniendo la variable booleana inicializada a 1 (Si esta variable es 1 estamos buscando una palabra, es decir, el byte anterior era un carácter vacío. Si es 0, estábamos en una palabra). En cada comprobación, dependiendo de la variable booleana y del byte correspondiente, realizaremos las sumas a los contadores de bytes, y de palabras y líneas si es el caso.

Finalmente, una vez hayamos comprobado todos los bytes del fichero .txt, imprimiremos los contadores seguidos del nombre del fichero correspondiente.

myls

En esta función debemos abrir un directorio como argumento y mostrar todas las entradas de este por la salida estándar, teniendo por defecto el directorio actual en caso de no tener ningún directorio argumentado.

Para llevar a cabo esta función concreta hemos declarado las variables de nombre del directorio, el puntero dirección que nos permitirá movernos por el directorio y la variable que guarda la entrada interna. Posteriormente hemos comprobado que el número de argumentos es el válido y inicializado la variable del nombre del directorio según si hay o no argumento.

Tras este inicio hemos abierto el directorio con *opendir* y guardado su dirección en la variable puntero, previamente habiendo comprobado que este directorio no sea nulo.

Finalmente, tras la apertura del directorio, hemos ido imprimiendo por pantalla la variable de entrada hasta que en algún momento fuese nula. Esta variable en cada iteración ha sido inicializada con la función *readdir* de la dirección guardada en el puntero, lo que permite imprimir todas de las entradas contenidas en el directorio correspondiente.

myishere

En esta función debemos abrir el directorio incluido como primer argumento y buscar dentro el fichero recibido como segundo argumento para comprobar si se encuentra en él o no.

Para llevar a cabo esta función, al igual que en el caso anterior, hemos declarado las variables del nombre de directorio (en este caso debe venir obligatoriamente de un argumento), el puntero de dirección, la variable que guarda la entrada interna y, además, una variable booleana que nos permitirá interpretar si hemos encontrado o no el fichero argumentado. Además, una vez hayamos declarado las variables, comprobamos que el número de argumentos sea correcto.

Tras todo esto hemos comprobado que el directorio argumentado no sea nulo y, en el caso de que así sea, abrimos el directorio con la función *opendir* y guardado su dirección en el puntero.

Posteriormente, tras la apertura del directorio, hemos iniciado un bucle en el que en cada iteración hemos ido inicializando la variable de la entrada con la función *readdir* de la dirección guardada en el puntero. Si la variable booleana de búsqueda era igual a 0, comprobabamos si la entrada era igual al segundo argumento, en cuyo caso afirmativo modificaría dicha variable booleana a 1, deteniendo así la búsqueda.

Finalmente, según si la variable booleana es igual a 0 o a 1, imprimimos por pantalla el resultado correspondiente de la búsqueda de nuestro fichero en el directorio argumentado.

Batería de Pruebas

Pruebas de MYWC

Prueba	Descripción	Salida	Salida modelo (wc)
./mywc Prueba0.txt	Prueba genérica utilizando un fichero con el texto: Hola Hugo ¿Qué tal?	1 4 22 Prueba0.txt (0)	1 4 22 Prueba0.txt (0)
./mywc Prueba1.txt	Prueba sobre fichero de 1 línea con un espacio al inicio y al final	0 4 23 Prueba1.txt (0)	0 4 23 Prueba1.txt (0)
./mywc Prueba2.txt	Prueba sobre fichero con un salto de línea al principio y al final	2 4 23 Prueba2.txt (0)	2 4 23 Prueba2.txt (0)
./mywc Prueba3.txt	Prueba sobre fichero que combina las propiedas de Prueba1 y Prueba2	2 4 25 Prueba1.txt (0)	2 4 25 Prueba1.txt (0)
./mywc Empty.txt	Prueba sobre fichero vacío	0 0 0 Empty.txt (0)	0 0 0 Empty.txt (0)
./mywc	Prueba de error por falta de argumentos	Se debe introducir al menos un argumento (255)	Cuelgue de la terminal
./mywc Prueba1.txt Prueba2.txt	Prueba con exceso de argumentos (Caso no contemplado en la práctica, únicamente incluido para valorar si implementar una excepción)	0 4 23 Prueba1.txt (0)	0 4 23 Prueba1.txt 2 4 23 Prueba2.txt 2 8 46 total
./mywc error.txt	Prueba de error por nombre de fichero incorrecto	No se ha podido abrir el fichero (255)	No existe el archivo o el directorio (1)

Pruebas de MYLS

Prueba	**Descripción**	**Salida**	**Salida modelo (ls -f -1)**
./myls Prueba0	Prueba genérica utilizando un directorio con dos ficheros	A.py . .. B.txt (0)	A.py . .. B.txt (0)
./myls	Prueba sin dar un directorio como argumento (situándonos en el directorio donde ubican el resto de programas)	Makefile myishere.c probador_ssoo_p1.py myls myls.o myishere.o Prueba0 mywc . .. mywc.c mywc.o myishere myls.c autores.txt p1_tests (0)	Makefile myishere.c probador_ssoo_p1.py myls myls.o myishere.o Prueba0 mywc . .. mywc.c mywc.o myishere myls.c autores.txt p1_tests (0)
./myls PruebaEmpty	Prueba con un directorio vacío	. .. (0)	. .. (0)
./myls PruebaEmpty Prueba0	Prueba de error por demasiados argumentos	Número de argumentos incorrecto (255)	Imprime los ficheros de todos los directorios dados (0)
./myls error	Prueba con un directorio con el nombre equivocado	Error al abrir el directorio (255)	No existe el archivo o el directorio (2)

Pruebas de MYISHERE

Prueba	Descripción	Salida	Salida esperada
./myishere Dir Prueba0.txt	Prueba genérica utilizando un directorio y un fichero que este dentro	File Prueba0.txt is in directory prueba (0)	File Prueba0.txt is in directory prueba (0)
./myishere Dir Prueba1.txt	Prueba con un directorio y un fichero que no este en él	File Prueba1.txt is not in directory prueba (0)	File Prueba1.txt is not in directory prueba (0)
./myishere	Prueba de error por falta argumentos	Se necesitan más argumentos (255)	Se necesitan más argumentos (255)
./myishere error Prueba0.txt	Prueba con un directorio con el nombre equivocado	Error al abrir el directorio (255)	Error al abrir el directorio (255)
./myishere .. PRUEBA	Prueba buscando desde el directorio padre, en este caso, el escritorio	File PRUEBA is in directory .. (0)	File PRUEBA is in directory .. (0)

Conclusiones de la práctica

Esta práctica nos ha sido de gran utilidad debido a que nos ha ayudado conocer más a fondo el entorno y los métodos de llamadas al sistema operativo, siguiendo en este caso el estándar POSIX. Además hay que destacar el aprendizaje de las bases del lenguaje de programación C que nos ha permitido desarrollar correctamente esta práctica.

En cuanto a los problemas que han ido surgiendo a lo largo de la práctica tienen que ver fundamentalmente con la falta de entendimiento de algunos aspectos de este lenguaje de programación, además de una serie de problemas a la hora de aplicar unas funciones de las cuales desconociamos completamente su utilidad y formato en el código, y que eran de vital importancia para seguir con el desarrollo de la práctica.

La solución a estos problemas se han ido obteniendo a base de prueba y error, además de una constante búsqueda en internet y en los distintos foros de programación. También gracias a las distintas pruebas realizadas posteriormente a la escritura del código hemos podido depurar este, haciendolo mucho más eficiente a la vez que efectivo.

Como opinión general del grupo hemos concluido que esta primera práctica sobre las llamadas a los sistemas operativos es bastante interesante a la hora de intentar familiarizarse con estas mismas, debido a que no exigen un nivel muy avanzado de conocimiento sobre el tema e involucran muchas funciones relevnates a la hora de inicializarse en el mundo de los Sistemas Operativos.