Sistemas Operativos

Práctica 1

Grupo 82

Laura Álvarez Flórez	100363965	100363965@alumnos.uc3m.es
Sabrina Riesgo Reyes	100363834	100363834@alumnos.uc3m.es
Laura Yunta García	100363785	100363785@alumnos.uc3m.es

Índice

PrintFile.c

El objetivo del programa printFile.c es imprimir por pantalla el contenido del fichero que recibe como argumento. En primer lugar, declaramos las variables y abrimos el fichero utilizando la función open. Esta función recibe 3 parámetros (ruta del fichero que se desea abrir, formas de apertura, permisos). En este caso, como forma de apertura tendríamos O_RDONLY (apertura del fichero solamente para escritura), y los permisos de lectura, escritura y ejecución. A continuación, se comprueba que el fichero se ha abierto correctamente mediante un bucle if-else. Si ocurre un error se imprimirá por pantalla un mensaje que informa sobre lo ocurrido, de lo contrario se seguirá ejecutando el código. Leemos el fichero mediante la función read que también recibe 3 parámetros (descriptor del fichero que devuelve la función open, puntero al buffer de memoria, número de bytes del buffer que queremos leer). Si el fichero está vacío también lo avisará por pantalla, si esto no ocurre, recorrerá el fichero, imprimiendo para cada persona el nombre, la edad, el DNI, el carácter de control de DNI y el salario. Finalmente, cerramos el fichero.

BATERÍA DE PRUEBAS Y RESULTADOS ESPERADOS

Programa	Fichero(s) de Entrada	Salida por Pantalla	Fichero(s) de Salida	Comp.
printFile.c	fichero personas_1718.per	Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Aitana Gonzalez 21 48596325 Q 18656 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	---	SI
printFile.c	<fichero vacío>	El fichero está vacío.	---	SI
printFile.c	<fichero que no existe>	El fichero no se ha abierto correctamente.	---	SI

Statistics.c

El objetivo del programa statistics.c es imprimir por pantalla la edad media, la renta media, el carácter de control de DNI más frecuente y la edad media de las personas que contienen ese carácter de control de DNI más frecuente.

En primer lugar, declararemos todas las variables que se utilizarán, entre las cuales se encuentran el array "letras" y el array "repeticiones". El array "letras" contendrá los 23 posibles caracteres de control de DNI, y el array "repeticiones" contendrá en todas sus posiciones el valor 0, ya que tiene como objetivo almacenar en la posición correspondiente a cada letra el número de veces que esta aparece una vez leído todo el fichero. Para asegurarnos de que todas las posiciones de "repeticiones" inicialmente contienen un 0, utilizaremos un bucle for que recorre todo el array. A continuación, abriremos el fichero mediante la función open, la cual recibe 3 parámetros (ruta del fichero que se desea abrir, formas de apertura, permisos). En este caso, como forma de apertura tendríamos O_RDONLY (apertura del fichero solamente para escritura), y los permisos de lectura, escritura y ejecución. Como resultado la función devolverá un valor entero, que se guardará en la variable "fichero". Si dicho valor es -1 el fichero no se ha abierto correctamente, por lo que comprobaremos a través de un bucle if cuál es el valor de la variable "fichero". En caso de que el valor sea distinto de -1 se continuará ejecutando el programa, en caso contrario se imprimirá por pantalla un mensaje que nos informará sobre lo ocurrido. Una vez comprobado que el fichero se ha abierto correctamente, utilizando la función read para leerlo, que también recibirá 3 parámetros (descriptor de fichero que devuelve la función open, puntero al buffer de memoria, número de bytes del buffer que queremos leer). Siempre que el número de bytes leídos sea mayor que 0 seguiremos ejecutando. Lo primero que haremos será calcular la edad y el salario total, y aumentaremos el contador cada vez que sumemos la edad y el salario de una persona. Además, comprobaremos cual es la letra de cada persona y aumentaremos en 1 la posición que le corresponde a dicha letra en el array "repeticiones". Ya tendríamos los totales, el número de personas que hay en el fichero y cuántas veces se repite cada carácter, por tanto, dividiremos el total de edades y el total de salarios entre el número de personas y obtendremos el valor de renta y edad media. Por último, recorreremos el array "repeticiones" para sacar el valor máximo y lo guardamos en una variable. Mediante otro bucle recorreremos una vez más el array y nos detendremos en la primera posición que contenga el valor máximo; así obtendremos la primera letra del alfabeto que más se repite. Para finalizar el programa, tenemos que hacer la edad media de las personas con el carácter de control de DNI más frecuente (el obtenido en el apartado anterior). Ahora utilizaremos la función lseek que recibe también 3 parámetros (descriptor de fichero abierto, número de bytes a desplazar, origen = SEEK_SET = inicio del fichero), para movernos al inicio del fichero y poder leerlo una vez más. Volvemos a calcular la edad media como en el segundo apartado, pero esta vez controlando mediante un bucle if que solo se tengan

en cuenta las personas que tienen el carácter de DNI más frecuente. El último paso sería cerrar el fichero.

BATERÍA DE PRUEBAS Y RESULTADOS ESPERADOS

Programa	Fichero(s) de Entrada	Salida por Pantalla	Fichero(s) de Salida	Comp.
statistics.c	fichero personas_1718.per	Renta media: 71174 Edad media: 45 Carácter de control de DNI más frecuente: H Edad media para el carácter de control de DNI más frecuente: 54	---	SI
statistics.c	<fichero vacío>	El fichero está vacío.	---	SI
statistics.c	<fichero que no existe>	El fichero no se ha abierto correctamente.	---	SI
statistics.c	Fichero personas_1718.per (sin contar las mayores de 60 años)	Renta media: 61291 Edad media: 41 Carácter de control de DNI más frecuente: E Edad media para el carácter de control de DNI más frecuente: 56	---	SI

Split.c

Este programa tendrá la función de dividir a las personas según su rango de edad. De tal forma que recibirá por parámetro un número entero positivo que servirá como límite de rango. Finalmente, el usuario recibirá como resultado dos ficheros (uno contendrá las personas con una edad menor al número introducido y otro las personas de edad mayor o igual al número introducido) para ello si estos ficheros existen se les añadirán la información y si no directamente crearán el fichero y lo escribirán sobre él.

Inicialmente para la implementación de este método se deberán declarar e introducir las librerías que van a ser usadas posteriormente. Tras esto, siguiendo a la cabecera del main se han inicializado todas las variables que se van a utilizar para la implementación de la función, estas son:

-fichero_1: En él se almacenará el primer fichero introducido por parámetro.

-fichero_2 y fichero_3: En ellos se almacenarán el segundo y el tercer fichero introducidos por parámetro, respectivamente (Que se creará o se truncará dependiendo de si existe o no).

-p1: En él se almacenará el objeto Persona que se está leyendo en cada caso.

-bytes: Almacena el número de bytes leídos del fichero.

-comparador: Guarda en él el entero convertido desde ascii gracias a la función atoi que ha sido introducido por parámetro para compararlo con las edades de las personas del fichero.

-edad: Almacenará la edad de la persona leída para ser comparada con el introducido por parámetro.

A continuación, abriremos los ficheros mediante la función open, la cual recibe 3 parámetros (ruta del fichero que se desea abrir, formas de apertura, permisos). En el caso del primer fichero, como forma de apertura se tendría O_RDONLY (apertura del fichero solamente para escritura), y los permisos de lectura, escritura y ejecución. Para el segundo y tercer fichero, como forma de apertura se tendrá O_CREAT (Creación del archivo si este no existe), O_WRONLY (Apertura solo para escritura) y O_TRUNC (Si el archivo no existe o no es normal su longitud se trunca a 0 y el modo y el propietario permanecen sin cambios) y por último se tendrán los permisos de lectura, escritura y ejecución.

Llegados a este punto de la ejecución de la función split se ejecutará un primer condicional que informará de si el programa ha tenido algún problema para abrir cualquiera de los tres ficheros. Una vez se ha comprobado que no ha habido errores también se comprueba que el primer fichero no esté vacío y posteriormente se introduce en el condicional donde se encontrará una primera implementación del código del bucle posterior, que nos dará el valor de la edad de la primera persona leída. Una vez ejecutado esto el programa se introduce en el bucle que va a ser el encargado de la lectura del fichero. Además, por cada persona leída este bucle va a guardar el valor de la edad de la persona y la va a almacenar en la variable edad ya explicada anteriormente. Se comparará si este número es más pequeño que el número introducido por parámetro, si esta instrucción se cumple se escribirán los datos de la persona correspondiente en el segundo fichero cuyo directorio se ha introducido por parámetro; en caso contrario, se introducirá en el tercer fichero en el cual se

almacenarán los datos de las personas mayores al número de edad que se ha introducido por parámetro.
Por último, gracias a unos contadores que han sido inicializados anteriormente el programa será capaz de identificar si alguno de los dos ficheros de salida se ha quedado sin rellenar porque no haya ninguna de las personas del fichero que cumpla su condición.
Finalmente, se cerrarán los ficheros utilizados.

BATERÍA DE PRUEBAS Y RESULTADOS ESPERADOS

Programa	Fichero(s) de Entrada	Salida por Pantalla	Fichero(s) de Salida	Comp.
split.c	54 fichero personas_1718.per fichero: menores.c fichero:mayores.c	---	-Fichero menores.c con contenido: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18656 -Fichero mayores.c con contenido: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	SI
split.c	fichero vacío	El fichero está vacío.	---	SI
split.c	fichero que no existe	El fichero no se ha abierto correctamente.	---	SI
split.c	2 fichero personas_1718.per fichero: menores.c fichero: mayores.c	No hay ninguna persona menor al número introducido	Fichero menores.c vacío -Fichero mayores.c con contenido: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Aitana Gonzalez 21 48596325 Q 18656 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	SI
split.c	90 fichero personas_1718.per fichero: menores.c fichero: mayores.c	No hay ninguna persona mayor al número introducido	Fichero mayores.c vacío -Fichero menores.c con contenido: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Aitana Gonzalez 21 48596325 Q 18656 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	SI

Filter.c

Este programa tendrá la función de seleccionar a las personas según su carácter de control del DNI. De tal forma que recibirá por parámetro un carácter que se utilizará como filtro. Finalmente, el usuario recibirá como resultado un fichero donde se escribirán las personas con esa letra de control en el DNI para ello si estos ficheros existen se truncarán para añadir la información y si no directamente crearán el fichero y lo escribirán sobre él.

Inicialmente para la implementación de este método se deberán declarar e introducir las librerías que van a ser usadas posteriormente. Tras esto, siguiendo a la cabecera del main se han inicializado todas las variables que se van a utilizar para la implementación de la función, estas son:

-fichero_entrada: En él se almacenará el primer fichero introducido por parámetro.

-fichero_salida: En ellos se almacenarán el segundo introducido por parámetro (Que se creará o se truncará dependiendo de si existe o no).

-persona: En él se almacenará el objeto Persona que se está leyendo en cada caso.

-bytes: Almacena el número de bytes leídos del fichero.

-cont: Variable que ayudará a la función a identificar si hay algún carácter que no coincida con el de ninguna persona del fichero.

A continuación, abriremos los ficheros mediante la función open, la cual recibe 3 parámetros (ruta del fichero que se desea abrir, formas de apertura, permisos). En el caso del primer fichero, como forma de apertura se tendría O_RDONLY (apertura del fichero solamente para escritura), y los permisos de lectura, escritura y ejecución. Para el segundo fichero, como forma de apertura se tendrá O_CREAT (Creación del archivo si este no existe), O_WRONLY (Apertura solo para escritura) y O_TRUNC (Si el archivo no existe o no es normal su longitud se trunca a 0 y el modo y el propietario permanecen sin cambios) y por último se tendrán los permisos de lectura, escritura y ejecución.

Llegados a este punto de la ejecución de la función filter se ejecutará un primer condicional que informará de si el programa ha tenido algún problema para abrir cualquiera de los dos ficheros. Una vez se ha comprobado que no ha habido errores también se comprueba que el primer fichero no esté vacío y posteriormente se introduce en el condicional donde se encontrará una primera implementación del código del bucle posterior, que nos dará el valor de la letra de control del DNI de la primera persona leída. Una vez ejecutado esto el programa se introduce en el bucle que va a ser el encargado de la lectura del fichero. Además, por cada persona leída este bucle va a comparar si el valor del carácter de control del DNI de la persona es igual al carácter introducido por parámetro. En caso de que esta condición se cumpla el programa escribirá los datos de la persona en el fichero de salida (segundo fichero introducido por parámetro); en caso contrario, se descartarán estos datos y se pasará a la siguiente persona del fichero.

Por último, gracias al contador que ha sido inicializado en el comienzo del código el programa será capaz de identificar si el fichero de salida se ha quedado sin escribir debido a que no haya cumplido ninguna de las personas la condición.

Finalmente, se cerrarán los ficheros utilizados.

BATERÍA DE PRUEBAS Y RESULTADOS ESPERADOS

Programa	Fichero(s) de Entrada	Salida por Pantalla	Fichero(s) de Salida	Comp.
filter.c	H fichero personas_1718.per fichero: filtro.c	---	-Fichero filtro.c con contenido: Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	SI
split.c	<fichero vacío>	El fichero está vacío.	---	SI
split.c	<fichero que no existe>	El fichero no se ha abierto correctamente.	---	SI
split.c	X fichero personas_1718.per fichero: filtro.c	No hay ninguna persona con esa letra de DNI	Fichero filtro.c vacío	SI

Combine.c

Lo primero que haremos será abrir los ficheros de entrada y salida utilizando la función open, que recibe 3 parámetros (ruta del fichero que se desea abrir, formas de apertura = O_RDONLY, permisos = lectura, escritura y ejecución). Calculamos el tamaño de los ficheros de entrada mediante la función stat. Comprobamos que los ficheros se han abierto correctamente, teniendo en cuenta el valor que devuelve la función open para cada uno de ellos. Una vez terminadas las comprobaciones pasamos a ejecutar el código que crea el nuevo fichero, pero puede haber varios casos que explicaremos a continuación.

1- Si el primer fichero de entrada está vacío tendríamos que escribir en el fichero de salida todas las personas del segundo fichero, pero empezando por el final. Utilizaremos la función lseek para posicionarnos en el final del fichero que vamos a leer y leemos la última persona. Continuaremos repitiendo el proceso hasta que lleguemos al inicio del fichero y siempre que leamos una persona, escribiremos toda su información en el fichero de salida.
2- El segundo caso sería muy parecido al primero, sólo que en este caso el fichero de entrada vacío no sería el primero, sino el segundo. Seguiríamos el mismo algoritmo que en el caso 1, pero leyendo y escribiendo la información del primer fichero.
3- Si tenemos dos ficheros de entrada vacíos, no tendría sentido escribir nada en el fichero de salida, por lo que este también estará vacío.
4- La última de las opciones es tener dos ficheros de entrada con contenido, por tanto, tendremos que leer ambos empezando por el final, pero mezclando las personas del fichero 1 con las del fichero 2. Para ello nos colocaremos en el final de los ficheros, leeremos la última persona de uno de ellos, la escribimos en el fichero de salida y repetimos los mismos pasos con el otro fichero. Este algoritmo se ejecutará constantemente hasta que se llegue al inicio de ambos ficheros.

Finalmente cerraremos todos los ficheros.

BATERÍA DE PRUEBAS Y RESULTADOS ESPERADOS

Programa	Fichero(s) de Entrada	Salida por Pantalla	Fichero(s) de Salida	Comp.
combine.c		El archivo no existe	---	SI
combine.c	fichero no existente	El archivo no existe		SI
combine.c	<fichero vacío>,< fichero vacio>	El fichero no se ha abierto correctamente.	---	SI
combine.c	<fichero vacio>, personas_1718.per	---	Soledad Garcia 54 6582351 G 136955 Benedicto Augusto 56 5486235 E 56896 Aitana Gonzalez 21 48596325 Q 18656 Ernesto Perez 66 6535859 H 120586 Silvia Lopez 32 56288595 M 65365 Juan Martin 43 4562321 H 28583	
combine.c	Personas_1718.per,	---	Soledad Garcia 54 6582351	

	<fichero vacio>		G 136955 Benedicto Augusto 56 5486235 E 56896 Aitana Gonzalez 21 48596325 Q 18656 Ernesto Perez 66 6535859 H 120586 Silvia Lopez 32 56288595 M 65365 Juan Martin 43 4562321 H 28583	
combine.c	Fichero1 Fichero2	---	Aitana Gonzalez 21 48596325 Q 18656 Soledad Garcia 54 6582351 G 136955 Benedicto Augusto 56 5486235 E 56896 Ernesto Perez 66 6535859 H 120586 Silvia Lopez 32 56288595 M 65365 Juan Martin 43 4562321 H 28583	SI

Aclaración: en la última comprobación del método combine.c, se ha realizado el método sobre dos ficheros resultantes de utilizar Split (30, personas_1718.per, Fichero1, Fichero2).

Conclusiones

Durante el desarrollo de las funciones hemos intentado realizar una exhaustiva batería de pruebas con cada una para evitar posibles fallos. Se ha tenido especial cuidado en el tratado de los posibles ficheros que pueda recoger la función main por parámetro, habiéndose hecho subrutinas que controlan si este no existe, si esta vacío, si presenta algún fallo o si no se ha introducido ninguna referencia a la estructura esperada.

Este diagrama muestra como hemos organizado nuestro código para los diferentes apartados de la práctica.