PRÁCTICA 1: LLAMADAS AL SISTEMA

Universidad Carlos III de Madrid

Jaime de Esteban Uranga - 100363590

[100363590@alumnos.uc3m.es]

Álvaro González de la Vega - 100363686

[100363686@alumnos.uc3m.es]

Sofía Najarro Almaraz - 100363545

[100363545@alumnos.uc3m.es]

ÍNDICE DE CONTENIDOS.

EJERCICIO 1 - PRINT FILE

En este apartado nos pedían desarrollar un programa que mostrará por pantalla el contenido de un fichero de personas.

Para ello hemos creado un Nodo que referencia a una lista doblemente enlazada. Esta lista sigue la estructura del objeto persona. Aquí almacenaremos los datos del fichero de entrada. En este Nodo insertamos desde la cola.

En la función principal del programa lo que haremos será abrir el fichero que se pasará como argumento por argv[1]. Tras esto se irá leyendo el contenido del fichero e insertándolo en el Nodo.

Tras esto mostramos el contenido del Nodo, sin usar la función printf(), usando sprintf() y después escribiendo a pantalla con la constante STDOUT_FILENO.

Quedan contemplados en el código un tratamiento de errores para que cuando se lee el fichero de entrada, se pasen valores correctos al programa y en caso contrario este devuelve "-1" y acaba el programa.

BATERÍA DE PRUEBAS

Fichero de entrada	Resultado esperado	Resultado obtenido
Fichero personas_1718.per (incluido en el zip de la práctica)	Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Aitana Gonzalez 21 48596325 Q 18657 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Aitana Gonzalez 21 48596325 Q 18657 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955
Fichero inexistente	Error opening file	Error opening file: no such file or directory

Resultado de la prueba de leak de memoria Valgrind:

```
-1==16==
==16== HEAP SUMMARY:
==16==     in use at exit: 0 bytes in 0 blocks
==16==   total heap usage: 3 allocs, 3 frees, 1,576 bytes allocated
==16==
==16== All heap blocks were freed -- no leaks are possible
```

EJERCICIO 2 - STATISTICS

Aquí nos pedían un programa que dado un fichero de personas mostráramos: Renta media, edad media, carácter de DNI más frecuente y edad media de las personas con el carácter de DNI más frecuente.

Aquí, al igual que en el anterior ejercicio, hemos utilizado un Nodo que referencia a una lista doblemente enlazada que inserta sus nodos por la cabeza (mas eficiente, no tiene que recorrerse toda la lista para insertar)

Básicamente, el funcionamiento del programa es igual al anterior hasta que se han almacenado los datos en el Nodo. Después de esto, creamos diferentes variables para los 4 datos que nos piden. La edad y renta media son relativamente sencillos, ya que es recorrer la lista e ir sumándolo todo para después dividirlo entre el contador de programa (número de elementos que conforman la lista). Para sacar el redondeo a entero del salario lo que hacemos es convertir la variable salario a entero y luego restarlo de la variable en double. Si el resultado de esa resta es mayor o igual 0.5 se le suma 1 al salario para completar el redondeo, si no se mantiene.

Para hallar el carácter de control hemos creado un array de chars de 26 elementos (número de letras en el alfabeto) y mientras recorremos la lista aumentamos el valor de la posición del array a la que referencia el valor entero del carácter de control. Después se comprueba en qué posición del array hay un valor más alto y ese será el carácter más repetido. La edad media de las personas con ese carácter de DNI se hace igual que la edad media normal, solo que comprobando mientras se recorre la lista que ese carácter coincide con el más frecuente.

Para finalizar se muestran los datos por pantalla.

BATERÍA DE PRUEBAS

Fichero de entrada	Resultado esperado	Resultado obtenido
Fichero personas_1718.per (incluido en el zip de la práctica)	Renta media: 71174 Edad media: 45 Carácter de control de DNI mas frecuente: H Edad media para el carácter de control de DNI mas frecuente: 54	Renta media: 71174 Edad media: 45 Carácter de control de DNI mas frecuente: H Edad media para el carácter de control de DNI mas frecuente: 54
Fichero inexistente	Error opening file	Error opening file: no such file or directory

Resultado de la prueba de leak de memoria Valgrind:

```
==18== HEAP SUMMARY:
==18==     in use at exit: 0 bytes in 0 blocks
==18==   total heap usage: 2 allocs, 2 frees, 1,064 bytes allocated
==18==
==18== All heap blocks were freed -- no leaks are possible
```

EJERCICIO 3 – SPLIT

En este ejercicio nos pedían separar a un grupo de personas en dos grupos dada una edad, por un lado, los menores y por otro los mayores o de igual edad.

El funcionamiento de paso de parámetros es igual a los anteriores ejercicios, solo que esta vez se meten más datos de entrada. No solo tenemos un fichero de entrada (que pasaremos por argv[1] como en los anteriores apartados), sino que también se pasa una edad límite y dos ficheros de salida, sobre los que habrá que escribir. Controlamos durante el programa comprobamos si la edad pasada por argumento es un entero y está entre los límites permitidos (entre 0 y 150).

Una vez comprobada la edad límite se recorre la lista y se decide si escribir sobre el fichero de edad menor o en el de edad mayor o igual. Los ficheros de salida han sido creados si no existían o se han abierto y se ha escrito en ellos, respetando lo anteriormente escrito utilizando el parámetro O_APPEND para mover el puntero de fichero al final del documento al empezar a escribir.

BATERÍA DE PRUEBAS

Fichero de entrada	Edad pasada	Ficheros de salida pasados	Resultado obtenido
Fichero personas_1718.per (incluido en el zip de la práctica)	45	menores.per (Vacío) mayores.per (Vacío)	menores.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657 mayores.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955
Fichero personas_1718.per (incluido en el zip de la práctica)	151	menores.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657 mayores.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	Input error
Fichero inexistente	10	menores.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657 mayores.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	Error opening input file: no such file or directory
Fichero personas_1718.per (incluido en el zip de la práctica)	22	menores.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657 mayores.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	menores.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657 Aitana Gonzalez 21 48596325 Q 18657 mayores.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955 Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955

Resultado de la prueba de leak de memoria Valgrind:

```
==18== HEAP SUMMARY:
==18==     in use at exit: 0 bytes in 0 blocks
==18==   total heap usage: 2 allocs, 2 frees, 1,064 bytes allocated
==18==
==18== All heap blocks were freed -- no leaks are possible
```

EJERCICIO 4 - FILTER

En este ejercicio nos pedían que, dado un fichero de entrada y carácter de DNI, escribiéramos en un fichero de salida las personas que tuvieran ese mismo carácter de DNI.

Aquí igual que en Split, tenemos varios datos de entrada. Un fichero de entrada (pasado por argv[1]), el carácter del DNI y un fichero de salida.

Lo primero que hacemos es comprobar si el carácter del DNI corresponde con un símbolo del alfabeto, hemos implementado código para que no importe si el carácter es minúscula o mayúscula, el funcionamiento es igual.

Una vez comprobado esto e insertado el fichero de entrada en el Nodo, recorremos la lista buscando a las personas con ese mismo carácter de DNI. Las personas que coincidan serán añadidas al fichero de salida.

El fichero de salida se abrirá antes de ejecutar el bucle. Al abrirlo se creará un fichero o se abrirá un fichero con ese nombre y se borrará toda la información anterior.

BATERÍA DE PRUEBAS

Fichero de entrada	Carácter de c	Fichero de salida pasado	Resultado obtenido en el fichero de salida
Fichero personas_1718.per (incluido en el zip de la práctica)	H	control.per (Vacío)	Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586
Fichero personas_1718.per (incluido en el zip de la práctica)	h	control.per: Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586
Fichero inexistente	H	control.per: Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	Error opening input file: no such file or directory
Fichero personas_1718.per (incluido en el zip de la práctica)	1	control.per: Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	Input error
Fichero personas_1718.per (incluido en el zip de la práctica)	A	control.per: Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	(Fichero vacío)

Resultado de la prueba de leak de memoria Valgrind:

```
==18== HEAP SUMMARY:
==18==     in use at exit: 0 bytes in 0 blocks
==18==   total heap usage: 2 allocs, 2 frees, 1,064 bytes allocated
==18==
==18== All heap blocks were freed -- no leaks are possible
```

EJERCICIO 5 - COMBINE

En este ejercicio nos daban dos ficheros de entrada y teníamos que mezclarlos en un único fichero de salida de manera que se vayan alternando. Además, se tenía que hacer en orden inverso, es decir desde el final del fichero hasta el inicio.

Creamos dos listas doblemente enlazadas, con dos punteros que van insertando en dos listas distintas las personas de los dos ficheros de entrada, insertándolos en la cabeza de la lista para invertir el orden de las personas.

Una vez rellenas las listas, lo que se hará es recorrer ambas listas de manera secuencial comprobando que el puntero de la cabeza es distinto de NULL y escribir en el fichero de salida.

BATERÍA DE PRUEBAS

Fichero de entrada 1	Fichero de entrada 2	Fichero de salida pasado	Resultado obtenido en el fichero de salida
a.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657	b.per: Ernesto Perez 66 6535859 H 120586 Benedicto Augusto 56 5486235 E 56896 Soledad Garcia 54 6582351 G 136955	solucion.per (Vacío)	Aitana Gonzalez 21 48596325 Q 18657 Soledad Garcia 54 6582351 G 136955 Silvia Lopez 32 56288595 M 65365 Benedicto Augusto 56 5486235 E 56896 Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586
a.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657	c.per: (Vacío)	solucion.per Aitana Gonzalez 21 48596325 Q 18657 Soledad Garcia 54 6582351 G 136955 Silvia Lopez 32 56288595 M 65365 Benedicto Augusto 56 5486235 E 56896 Juan Martin 43 4562321 H 28583 Ernesto Perez 66 6535859 H 120586	Aitana Gonzalez 21 48596325 Q 18657 Silvia Lopez 32 56288595 M 65365 Juan Martin 43 4562321 H 28583
a.per: Juan Martin 43 4562321 H 28583 Silvia Lopez 32 56288595 M 65365 Aitana Gonzalez 21 48596325 Q 18657	Fichero inexistente	solucion.per Aitana Gonzalez 21 48596325 Q 18657 Silvia Lopez 32 56288595 M 65365 Juan Martin 43 4562321 H 28583	Error opening input file 2: no such file or directory

Resultado de la prueba de leak de memoria Valgrind:

```
==18== HEAP SUMMARY:
==18==     in use at exit: 0 bytes in 0 blocks
==18==   total heap usage: 2 allocs, 2 frees, 1,064 bytes allocated
==18==
==18== All heap blocks were freed -- no leaks are possible
```