Fundamentos de los Sistemas Operativos
Ficha de entrega de práctica

**: campo obligatorio*
IMPORTANTE: esta ficha no debe superar las DOS PÁGINAS de extensión

Grupo de prácticas*: 1-41
Miembro 1: Alejandro Vialard Santana
Miembro 2:

Número de la práctica*: 3	**Fecha de entrega*: 20/04/2022**

Descripción del trabajo realizado*
Esta práctica consiste en conocer el uso y tratamientos de ficheros mediante comandos en la consola. Para ello, se nos pide realizar ciertos métodos que nos permitan hacer cambios a ficheros de texto, utilizando librerías como <sys/types.h>, <errno.h>, <unistd.h>, <pwd.h>, <fcntl.h>; las cuales nos permitirán usar funciones como getopt(), open(), read(), write(), lseek(), que nos será de ayuda para poder obtener las opciones por la línea de comandos, abrir y leer ficheros, escribir en ellos, entre otras funciones.
Para el correcto funcionamiento de la practica, he realizado diferentes funciones:
1. compruebaError(): método que realiza la comprobación con errno del error exacto que ocurre.
2. abrirArchivo(): método al que se le pasa el puntero del archivo que queremos abrir, mediante la función open(), y se comprueba el fid del archivo.
3. cerrarArchivo(): método que se le pasa un identificador de archivo (fid) y con la función close(), lo cierra.
4. compruebaArgumentos(): método al que se le pasa el número de argumentos y un array de punteros, y comprueba si se producen errores como por ejemplo, si hay menos o más de 4 argumentos, las opciones introducidas (s,l,U,C,t) mediante getopt(), y luego comprueba las combinaciones de las opciones imprimiendo sus errores correspondientes (no más de una opción, no introducir fichero, o de no introducir ninguna opción).
5. extraeArgumentos: método al que se le pasa el número de argumentos y un array de punteros, e imprime el numero de argumentos y que cual es cada uno, Ej: argv[0] = ./pcase.
6. sentenceCase(): método al que se le pasa un identificador de archivo y pone a mayúscula la primera letra de cada frase, leyendo con la función read(), y mediante init = 0 (que indica que es la primera letra), mira si el buffer es minúscula, y lo cambia a mayúscula restando 32, y para no sobrescribir, se desplaza con lseek() y se escribe con write(). Si minúscula solo cambia init a 1, y si detecta un '.' pone init a 0.
7. lowerCase(): método al que se le pasa un fid, lee el archivo y comprueba si es mayúscula y las cambia a minúscula sumando 32 al buffer, y si es minúscula, se desplaza el carácter con lseek() y se escribe.
8. upperCase(): *método al que se le pasa un fid, lee el archivo y comprueba si es minúscula, si lo es, la cambia a mayúscula restando 32 al buffer, si no, ya está a mayúscula y solo cambia init a 1, se desplaza el carácter y se escribe.*
9. capitalizeEachWord(): método al que se le pasa un fid, y pone a mayúscula la primera letra de cada palabra. Comprueba si está en minúscula, y la cambia a mayúscula poniendo init a 1 (así vemos que se cambia la primera letra), si encuentra un espacio (buf = 32), pone init a 0 para volver a cambiar a mayúscula. Si la primera está en mayúscula, solo cambia init a 1. Se desplaza y se escribe.
10. toogleCase(): método al que se le pasa un fid y pone en minúscula la primera letra de la palabra y el resto de ella en mayúscula. Comprueba si la primera es mayúscula y la cambia a minúscula, si ya lo es, solo cambia el init a 1. Si init está a 1 (ya no es la primera letra), y comprueba si es minúscula y la cambia a mayúscula, y si encuentra un espacio, pone init a 0 para tratar la siguiente palabra.
11. main(): método principal al que se le pasa como parámetros el número de argumentos del comando y un vector de punteros. En él, se llama a la función extraeArgumentos(), compruebaArgumentos(), y para poder leer uno o dos archivos, hacemos un bucle desde 2 hasta el número de argumentos, ya que argv[2] = fichero 1 (./pcase -option -file1 (-file2)). Luego, dependiendo de qué opción esté a 1, se hace el método correspondiente. Se cierra el archivo y se comprueba si hay errores.

Horas de trabajo invertidas* Miembro 1: 10 horas aprox **Miembro2:**

Grupo de prácticas*: 1-41
Miembro 1: Alejandro Vialard Santana
Miembro 2:

Número de la práctica*: 3	**Fecha de entrega*: 20/04/2022**

Cómo probar el trabajo*
(qué debe hacer el profesor para utilizar el programa entregado: nombre de los programas, instrucciones de compilación, opciones de menú, datos de prueba, argumentos de invocación al programa, etc.)

En la carpeta "practica3-Alejandro-Vialard" encontrará una serie de elementos.
-***pcase.o:*** *es el ejecutable del código, para crearlo, sitúese en la carpeta y escriba "gcc pcase.c -o pcase".*
-***pcase.c:*** código en lenguaje C de los métodos de la práctica.
-**pcase.sh:** script en el que aparece los diferentes comandos para realizar las pruebas correspondientes, ya está en modo ejecutable, pero si lo quiere hacer sólo debe escribir "chmod -x pcase.sh".
-**carpetas de pruebas:** verá 5 carpetas de pruebas cuyos nombres siguen "prueba-método", y dentro de cada una, hay 3 ficheros .txt de pruebas para corroborar la práctica, siguiendo como nombre "prueba-método-n°.txt".

Aclaraciones: si desea ver código, en el terminal puede escribir "cat pcase.c" p "cat pcase.sh". Si quiere editar puede usar gedit o vim "gedit pcase.c pcase.sh".
Para ejecutar el script y ver todas las pruebas que se realizan, debe escribir "./pcase.sh" y en pantalla verá todos los textos correspondientes.

Incidencias
(errores no resueltos, anomalías, cualquier cosa que se salga de lo normal)

Comentarios
(observaciones adicionales que quieran anotar)