Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 1. Llamadas al sistema para el S.Archivos. Parte I

Nombre y apellidos:

Ana Alicia Vílchez Ceballos

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de 30 minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: no . En caso de haber contestado “no”, indica los motivos por los que no las has resuelto

He tenido alguna duda al construir el primer ejercicio que resolveré en clase.

2. Tengo que trabajar algo más los conceptos sobre:

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

Mi solución al **ejercicio 1** ha sido:

Este programa abre o crea un txt llamado “archivo” con sus correspondientes flags que indican el modo de apertura, en este caso abre o crea el fichero en modo escritura y lectura. Esciribe en él los 10 caracteres que se encuentran en el vector buf1, luego con lseek desplaza el puntero lógico del fichero unos 40 “caracteres nulos” y vuelve a escribir los 10 caracteres en el fichero que se corresponden con los almacenados en buf2. La diferencia que existe cuando lo mostramos con cat o con od -c es que cat no muestra los caracteres nulos producidos como consecuencia de ejecutatar lseek y sin embargo, od -c muestra con mas detalle el contenido del fichero incluyendo los caracteres nulos.

Mi solución a la **ejercicio 2** ha sido:

```
/*
ejer2.c
Trabajo con llamadas al sistema del Sistema de Archivos ''POSIX 2.10 compliant''
*/
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>       /* Primitive system data types for abstraction\
                               of implementation-dependent data types.
                               POSIX Standard: 2.6 Primitive System Data Types
                               <sys/types.h>
                            */
#include<sys/stat.h>
#include<fcntl.h>
#include<errno.h>
```

```
int main(int argc, char *argv[])
{
        char cadena[7];
        char cadena1[22];
        char buf1[80];
        int fd;
        int fx;
        int grupo = 1;
        // aqui detectamos si se trata de la salida estandar o si es de un fichero
        if(argc > 1){
                if( (fd=open(argv[1],O_RDONLY))<0) {
                                printf("\nError %d en open",errno);
                                perror("\nError en open");
                                exit(EXIT_FAILURE);
                }
        }
       else
        fd = STDIN_FILENO;

        if(          (fx=open("salida.txt",O_CREAT|O_TRUNC|O_WRONLY,S_IRUSR|S_IWUSR          |
S_IRWXU))<0) {
                        printf("\nError %d en open",errno);
                        perror("\nError en open");
                        exit(EXIT_FAILURE);
        }

        if((write(fx,"\n\n\n",3)) < 0) {   // este \n lo pongo para que luego cuando volvamos al
principio del fichero con lseek pueda escribir el numero de grupos
                        perror("\nError en el tercer write");
                        exit(EXIT_FAILURE);
        }
        while(read(fd,&buf1, 80) > 0){ // mientras lea caracteres...
```

```
		sprintf(cadena, "Grupo %i", grupo);
		if((write(fx,cadena,7)) < 0) {
			perror("\nError en el primer write");
			exit(EXIT_FAILURE);
		}

		if((write(fx,"\n",1)) < 0) {
			perror("\nError en el 5 write");
			exit(EXIT_FAILURE);
		}

		if((write(fx,buf1,80)) < 0) {
			perror("\nError en segundo write");
			exit(EXIT_FAILURE);
		}

		if((write(fx,"\n\n",2)) < 0) {
			perror("\nError en el tercer write");
			exit(EXIT_FAILURE);
		}

		grupo += 1;
	}

	// MODIFICACIÓN ADICIONAL:
	sprintf(cadena1, "grupos producidos: %i\n", grupo);

	if(lseek(fx,0,SEEK_SET) < 0){ // seek_set marca el inicio del fichero
		perror("\nError en lseek");
		exit(-1);
	}
```

```
        if((write(fx,cadena1,sizeof(cadena1))) < 0) {
                perror("\nError en el 6 write");
                exit(EXIT_FAILURE);
        }
        return EXIT_SUCCESS;
}
```

Mi solución a la **ejercicio 3** ha sido:

En este ejercicio le pasamos una serie de archivos como argumento y nos va a decir qué tipo de archivos son: regular, de carácter especial, directorio, etc.

Para ello llama a la función lstat que recogerá los atributos de los archivos, y luego para identificar el tipo, llama a las macros que le correspondan pasándole el atributo st_mode que indica el tipo de archivo y los atributos. Strcpy podría haber sido sustituido por sprintf. Las macros POSIX a las que llama devolverán un booleano. En el guión se definen 7 macros para detectar 7 tipos de archivos.

Mi solución a la **ejercicio 4** ha sido:

```
/*
ejer4.c
Trabajo con llamadas al sistema del Sistema de Archivos ''POSIX 2.10 compliant''
*/

#include<unistd.h>  /* POSIX Standard: 2.10 Symbolic Constants <unistd.h>
```

```
                    */
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>  /* Primitive system data types for abstraction          \
                          of implementation-dependent data types.
                          POSIX Standard: 2.6 Primitive System Data Types
                          <sys/types.h>
                       */
#include<sys/stat.h>
#include<stdio.h>
#include<errno.h>
#include<string.h>
                          // aqui construimos la macro
#define S_REGULAR(mode) (((mode) & S_IFREG) == 0100000) // un numero multiplicado por si mismo, da ese numero


int main(int argc, char *argv[])
{


int i;
struct stat atributos;
char tipoArchivo[30];


if(argc<2) {
        printf("\nSintaxis de ejecucion: tarea2 [<nombre_archivo>]+\n\n");
        exit(-1);
}
for(i=1;i<argc;i++) {
        printf("%s: ", argv[i]);
        if(lstat(argv[i],&atributos) < 0) { // aqui es donde asociamos el struct de tipo stat con los atributos del archivo que le pasamos comoargumento
                printf("\nError al intentar acceder a los atributos de %s",argv[i]);
```

```
                perror("\nError en lstat");
        }
        else {
                if(S_REGULAR(atributos.st_mode))
                        sprintf(tipoArchivo, "el archivo %s es regular\n", argv[i]);


                else
                        sprintf(tipoArchivo, "el archivo %s NO es regular\n", argv[i]);


                printf("%s\n",tipoArchivo);
        }
}


return 0;
}
```