Sistemas Operativos

Formulario de auto-evaluación

Molulo 2. Sesión 2. Llamadas al sistema para el S.Archivos Parte II

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 120 minutos.

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

Resolver una duda con el ejercicio 3 sobre si es preciso que el programa recorra los archivos que se encuentran en los subdirectorios del directorio que le pasamos como argumento

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:

En el siguiente programa abrimos o creamos dos ficheros: archivo1 y archivo2. Luego obtenemos los atributos del struck stat de cada uno de los ficheros para obtener el campo st_mode y obtener la cadena de bit que se corresponden con los permisos. Para el archivo1, la operación que realizamos con las cadenas de bit implica que le quitamos la ejecución/búsqueda del archivo para el grupo en caso de que lo tenga con "atributos.st_mode & ~S_IXGRP", pues estamos negando a IX_GRP → 111 111 110 111 111. Posteriomente le hacemos un OR con S_ISGID, es decir, si el archivo1 no tiene activada la asignación del GID del propietario al GID efectivo del proceso que ejecute el archivo, la activa.

En archivo2 simplemente le ponemos los permisos pisando los que tenía anteriormente a través de OR: S_IRWXU → el usuario tiene permisos de lectura, escritura, ejecución

S_IRGRP → lectura para el grupo

S_IWGRP → escritura para el grupo

S_IROTH → lectura para otros

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

```
// en este ejercicio trataremos con la apertura de los ficheros de un directorio,
// la  modificación de los permisos y obtencion de sus atributos.


                                        //POSIX Standard: 2.6 Primitive System Data Types <sys/types.h>
#include <unistd.h>                 //POSIX Standard: 2.10 Symbolic Constants        <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>            //Needed for open
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h> //Primitive system data types for abstraction of implementation-
```

```
dependent data types.



//Función que permite imprimir los permisos de esta forma
void imprimir (struct stat estru){
 printf( (S_ISDIR(estru.st_mode)) ? "d" : "-" );
 printf( (estru.st_mode & S_IRUSR) ? "r" : "-" );
 printf( (estru.st_mode & S_IWUSR) ? "w" : "-" );
 printf( (estru.st_mode & S_IXUSR) ? "x" : "-" );
 printf( (estru.st_mode & S_IRGRP) ? "r" : "-" );
 printf( (estru.st_mode & S_IWGRP) ? "w" : "-" );
 printf( (estru.st_mode & S_IXGRP) ? "x" : "-" );
 printf( (estru.st_mode & S_IROTH) ? "r" : "-" );
 printf( (estru.st_mode & S_IWOTH) ? "w" : "-" );
 printf( (estru.st_mode & S_IXOTH) ? "x" : "-" );
 printf("%-3s");
}

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

        DIR *dirp;
        struct dirent *direntp;
        struct stat atributos;

        if(argc < 3){
                printf("error en el numero de argumentos: %d\n", argc);
                exit(1);
        }
        // abrimos el directorio
        dirp = opendir(argv[1]);
        if (dirp == NULL){
```

```
                printf("Error: No se puede abrir el directorio\n");
                exit(2);
        }

        // convertimos la mascara pasada como argumento en un entero con strol
        int base;
        char *finalPtr;
        strtol(argv[2], &finalPtr, base) ;

        /* Leemos las entradas del directorio */
        printf("nombre_archivo\tpermisos antes\tpermisos despues\n");
        while((direntp = readdir(dirp)) != NULL) {
                //obtenemos los atributos de cada archivo dentro del directorio
                stat(direntp->d_name, &atributos);
                printf("%s\t", direntp->d_name);

                imprimir(atributos);
                printf("\t");
                //modificamos los permisos con chmod:
                if(chmod(direntp->d_name, base) < 0)
                        printf("Error\t");
                else
                        imprimir(atributos);

                printf("\n");
        }

        return 0;
}
```

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

```
// en este ejercicio trataremos con la apertura de los ficheros de un directorio,
// la  modificación de los permisos y obtencion de sus atributos.


                                        //POSIX Standard: 2.6 Primitive System Data Types <sys/types.h>
#include <unistd.h>                     //POSIX Standard: 2.10 Symbolic Constants        <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>              //Needed for open
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h> //Primitive system data types for abstraction of implementation-dependent data types.


#define S_REGULAR(mode) (((mode) & S_IFREG) == 0100000) // un numero multiplicado por si mismo, da ese numero



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


        DIR *dirp;
        struct dirent *direntp;
        struct stat atributos;


        if(argc < 2){
                printf("error en el numero de argumentos: %d\n", argc);
                exit(1);
```

```
	}
	// abrimos el directorio
	dirp = opendir(argv[1]);
	if (dirp == NULL){
		printf("Error: No se puede abrir el directorio\n");
		exit(2);
	}

	int n_archivosr = 0;
	double tam_archr = 0;
	/* Leemos las entradas del directorio */
	printf("Los i nodos son:\n");
	while((direntp = readdir(dirp)) != NULL) {
		//obtenemos los atributos de cada archivo dentro del directorio

		stat(direntp->d_name,&atributos) ;

		if(S_REGULAR(atributos.st_mode) && (atributos.st_mode & (S_IXGRP | S_IXOTH) == (S_IXGRP | S_IXOTH))){ // Si el archivo es regular y tiene permisos de ejecucucion para grupos y para otros
			printf("%s\t%d\n", direntp->d_name, direntp->d_ino);
			n_archivosr += 1;
			tam_archr += direntp->d_reclen;
		}
	}
	printf("\nExisten %d archivos regulares con permiso x para grupo y otros\n", n_archivosr);
	printf("El tamanio total ocupado por dichos archivos es de %7.2f bytes \n", tam_archr/8);


	return 0;
}
```