Sistemas Operativos

Formulario de auto-evaluación

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

Nombre y apellidos:

Jesús Manuel García Palma

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 ..... minutos.

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

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 crea un nuevo archivo llamado archivo1 el cuál tiene permiso de escritura, lectura y ejecución para el grupo. Después se llama a la orden umask(0) para poner la máscara a 0 y después se crea un archivo2, que tiene las mismas características que el anterior. Después , con la orden stat comprobamos que los atributos del primer archivo sean accesibles.

Con la orden chmod se hace la modificación de los permisos de archivo1. También se activa la asignación del GID del propietario al GID del proceso que ejecuta archivo1.

Con la orden chmod se cambian también los permisos del archivo2 para tener los permisos de lectura y escritura para el grupo, y lectura para el resto de usuarios.

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

```
#include<sys/stat.h>

#include<fcntl.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

#include<errno.h>

#include<unistd.h>

#include<dirent.h>


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

{

  DIR *direct;

  unsigned int permisos;

  char *pathname;

  struct stat atributos;

  struct dirent *ed;
```

```
char cadena[100];
char cadena2[100];
extern int errno;
if (argc==3)
{
	pathname=argv[1];
	direct=opendir(pathname);
	permisos=strtol(argv[2],NULL,8);
}
else{
	printf("Uso: ejercicio2.c <pathname> <permisos>\n");
	exit(-1);
}
readdir(direct);
while((ed=readdir(direct))!=NULL){
	sprintf(cadena,"%s/%s",pathname,ed->d_name);
	if(stat(cadena,&atributos) < 0) {
		printf("\nError al intentar acceder a los atributos de archivo");
		perror("\nError en lstat");
		exit(-1);
	}
	if(S_ISREG(atributos.st_mode)){
		sprintf(cadena2,"%s",ed->d_name);
		printf("%s: %o ",cadena2,atributos.st_mode);
		chmod(cadena,permisos);
		if(chmod(cadena,permisos) < 0) {
			printf("Error: %s\n",strerror(errno));
		}
		else{
			stat(cadena,&atributos);
			printf("%o \n",atributos.st_mode);
```

```
                }
        }
  }

  closedir(direct);
  return 0;
}
```

g++ -o ejercicio2 ejercicio2.c y ejecutamos con ./ejercicio2 <pathname> <permisos>

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

```
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#include<errno.h>
#include<unistd.h>
#include<dirent.h>
#define mymask(mode) ((mode) & ~S_IFMT)

// Permisos de ejecución para grupo y otros.
#define S_IFXGRPOTH 011

// Se define la macro con la regla para comprobar si tiene permiso x en grupo y otros.
#define regla1(mode) ((((mode) & ~S_IFMT) & 011) == S_IFXGRPOTH)
```

```
void buscar_dir(DIR *direct, char pathname[], int *reg, int *tamanio){
struct stat atributos;
struct dirent *ed;
DIR *direct_act;
char cadena[500];
while((ed=readdir(direct)) != NULL){
// Ignorar el directorio actual y el supoerior
        if (strcmp(ed->d_name, ".") != 0 && strcmp(ed->d_name, "..") != 0){
                sprintf(cadena,"%s/%s",pathname,ed->d_name);
                if(stat(cadena,&atributos) < 0) {
                        printf("\nError al intentar acceder a los atributos del archivo");
                        perror("\nError en lstat");
                        exit(-1);
                }
                if (S_ISDIR(atributos.st_mode)){
                        if ((direct_act = opendir(cadena)) == NULL)
                                printf("\nError al abrir el directorio: [%s]\n",
                                cadena);
                        else
                                buscar_dir(direct_act, cadena, reg, tamanio);
                        }else{
                                printf("%s %ld \n", cadena, atributos.st_ino);
                                if (S_ISREG(atributos.st_mode)){
                                        if (regla1(atributos.st_mode)){
                                                (*reg)++;
                                                (*tamanio) += (int) atributos.st_size;
                                        }
                                }
                        }
                }
        }
```

```
        closedir(direct);
 }
 int main(int argc, char *argv[])
 {
 DIR *direct;
 char pathname[500];
 int reg=0,tamanio=0;
 if (argc==2)
 {
        strcpy(pathname,argv[1]);
 }
 else{
        strcpy(pathname,".");
 }
 if((direct=opendir(pathname)) == NULL){
        printf("\nError al abrir el directorio\n");
        exit(-1);
 }
 printf("Los inodos son: \n\n");
 buscar_dir(direct,pathname,&reg,&tamanio);
 printf("Hay %d archivos regulares con permiso x para grupo y otros\n",reg);
 printf("El tamaño ocupado por los archivos es %d bytes\n",tamanio);
 return 0;
 }
```

```
g++ -o ejercicio3 ejercicio3.c y ./ejercicio3
Los inodos son:

./ejercicio2.cpp 5776330
./ejercicio3.c 5776336
./tarea2.c 5771615
```

./ejercicio3.c~ 5776335

./tarea3.c 5771613

./ejercicio3 5776334

./ejercicio2 5776333

./ejercicio2.c 5776332

./tarea1.c 5771612

./ejercicio2.c~ 5776331

Hay 2 archivos regulares con permiso x para grupo y otros

El tamaño ocupado por los archivos es 26819 bytes