j1albertolp_473623
www.wuolah.com/student/j1albertolp_473623

Examenmodulo2SOresuelto.pdf

Prácticas_Modulo2_ResueltoConEnunciados

2º Sistemas Operativos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

EXAMEN – Resuelto 11/2019

```
//j1albertolp
//NO es necesario hacer tantos includes, de hecho aquí hay algunos que no se usan.
#include<limits.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<dirent.h>
#include<sys/wait.h>
#include<ftw.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
```

```
//Ejercicio1
#define TAM 7
#define CHECK 0777
#define S_ISREG_CHECK_TAM(mode,len) S_ISREG(mode) && len==TAM &&
(((mode) & CHECK)==CHECK)

int main(int argc, char** argv){
DIR *dir_abierto;
struct dirent *puntero_dir;
struct stat atributos;
char ruta[PATH_MAX];

//Abrimos directorio de trabajo actual
if((dir_abierto=opendir("./"))==NULL){
        printf("ERROR abriendo directorio de trabajo");
        exit(-1);
}

//Empezamos lectura de archivos del directorio
while ((puntero_dir = readdir(dir_abierto)) != NULL) {
        if (strcmp(puntero_dir->d_name, ".") != 0 && strcmp(puntero_dir->d_name,
"..") != 0){
                sprintf(ruta, "./%s",puntero_dir->d_name);
                //Realizamos stat sobre archivo
                if(stat(ruta, &atributos) < 0){
                        printf("ERROR, no se ha podido acceder a los metadatos de ");
                        printf("%s\n",ruta);
```

```
                exit(-1);
            }

            //Comprobamos críterio de seleccion
            if(S_ISREG_CHECK_TAM(atributos.st_mode,strlen(puntero_dir->d_name))){
                write(STDOUT_FILENO,puntero_dir->d_name,TAM+1);//Pone el solo el /0

            }



        }
}

//Cerramos directorio
closedir(dir_abierto);
exit(0);

}
```

```
//Ejercicio2

#define TAM 7
#define permisos S_IRWXU|S_IRWXG //Macro extra, para ponerlo más rapido == (0770)


int main(int argc, char const *argv[]) {
  int fd[2]; //Entradas salidas del cauce
  pid_t pid;
  int leidos;
  int fd_crear;
  int orden=1;
  char name[NAME_MAX]; //Nombre de los archivos que vayamos obteniendo de ej1
  char escribir[NAME_MAX];
  char tmp[PATH_MAX];

  umask(0); //Pongo la máscara a 0, para que después no me influya en la creación


  pipe(fd); //Creo la tuberia
```

```
 if((pid=fork())<0){
   printf("Error en la creacion de un hijo" );
   exit(-1);
 }

 //HIJO -> Execlp(ej1)
 if(pid==0){
   close(fd[0]); //Cierro lectura con padre
   dup2(fd[1],STDOUT_FILENO); //Dup2 en la salida estandar para escribir
   execlp("./ej1","ej1",NULL); //Despues de exec no vuelve hijo
   exit(0);
 }

 //PADRE
 else{
   close(fd[1]); //Cierro la escritura con hijo1
   dup2(fd[0],STDIN_FILENO); //Dup2 en la entrada estandar para leer

   //Lectura
   while((leidos=read(STDIN_FILENO,name,TAM+1))==TAM+1){
        //Apartado A Escribir orden y nombre
        sprintf(escribir,"Orden:%d-%s\n",orden,name);
        write(STDOUT_FILENO,escribir,strlen(escribir));
        orden++;

        //Apartado B
        sprintf(tmp,"/tmp/%s",name);
        if((fd_crear=open(tmp,O_CREAT|O_WRONLY|O_TRUNC,permisos))<0){
                printf("ERROR en open");
                exit(-1);
        }

        //Cerramos el archivo creado
        close(fd_crear);
    }

 }

 return 0;
}
```

```
//Ejercicio3

#define TAM 7
```

```
#define permisos S_IRWXU|S_IRWXG //Macro extra, para ponerlo más rapido ==
(0770)
#define permisos_netflix S_IRWXU


int main(int argc, char const *argv[]) {
  int fd[2]; //Entradas salidas del cauce
  pid_t pid;
  int leidos;
  int fd_crear;
  int orden=1;
  char name[NAME_MAX]; //Nombre de los archivos que vayamos obteniendo de
ej1
  char escribir[NAME_MAX];
  char serie[PATH_MAX];
  char tmp[PATH_MAX];
  int leido_netflix=0; //Entero que funcionara de bool, 0=false, 1=true.

  if(argc!=2){
        printf("Numero argumentos incorrecto, ponga un solo argumento, indicando
serie\n");
        exit(-1);
  }

  sprintf(serie,"%s\n",argv[1]); //Formo la cadena del nombre de la serie.
  umask(0); //Pongo la máscara a 0, para que después no me influya en la creación


  pipe(fd); //Creo la tuberia

  if((pid=fork())<0){
    printf("Error en la creacion de un hijo" );
    exit(-1);
  }

  //HIJO -> Execlp(ej1)
  if(pid==0){
    close(fd[0]); //Cierro lectura con padre
    dup2(fd[1],STDOUT_FILENO); //Dup2 en la salida estandar para escribir
    execlp("./ej1","ej1",NULL); //Despues de exec no vuelve hijo
    exit(0);
  }

  //PADRE
  else{
```

```
    close(fd[1]); //Cierro la escritura con hijo1
    dup2(fd[0],STDIN_FILENO); //Dup2 en la entrada estandar para leer

    //Lectura
    while((leidos=read(STDIN_FILENO,name,TAM+1))==TAM+1){
        //Apartado A Escribir orden y nombre
        sprintf(escribir,"Orden:%d-%s\n",orden,name);
        write(STDOUT_FILENO,escribir,strlen(escribir));
        orden++;
        if(strcmp(name, "netflix") ==0)
            leido_netflix=1; //Se pone a true

        //Apartado B
        sprintf(tmp,"/tmp/%s",name);
        if((fd_crear=open(tmp,O_CREAT|O_WRONLY|O_TRUNC,permisos))<0){
            printf("ERROR en open");
            exit(-1);
        }

        //Cerramos el archivo creado
        close(fd_crear);
    }

    //Netflix
    if(leido_netflix){
        sprintf(tmp,"/tmp/series_recomendadas.txt");
        if((fd_crear=open(tmp,O_CREAT|O_WRONLY,permisos_netflix))<0){
            printf("ERROR en open Netflix");
            exit(-1);
        }
        lseek(fd_crear,0,SEEK_END);//Ponemos el puntero al final para escribir
        write(fd_crear,serie, strlen(serie));
        close(fd_crear);
    }

  }

  return 0;
}
```