Sistemas Operativos

Formulario de auto-evaluación

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

Nombre y apellidos:

David Criado Ramón

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 15 minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: Sí (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:

El código crea o escribe (sobrescribiendo el archivo) si ya existe, en caso de crear lo crearía con permiso de lectura y escritura para el usuario actual. Escribiría al principio del mismo la cadena abcdefghij y a partir del byte 40 desde el principio del archivo la cadena de caracteres ABCDEFGHIJ. En caso de producirse algún error mostraría por salida estándar la variable errno, y donde se produce el error. Además mostraría por la salida de error en qué parte se ha producido el error. Cualquier error mataría inmediatamente al proceso.

Al mostrarlo con cat sólo vemos ambas cadenas seguidas ya que el espacio que hemos dejado entre dónde terminamos de escribir la primera vez y dónde pusimos lseek se rellana con '\0'. Esto lo podemos comprobar con od -c que nos muestras el código carácter ASCII a carácter ASCII por lo que podemos comprobar que en el carácter 40 empieza la segunda cadena.

Nota: En el código C proporcionado falta la llamada a close(fd) que el pdf muestra.

Salidas:

cat archivo

abcdefghijABCDEFGHIJ

od -c archivo

```
0000000   a   b   c   d   e   f   g   h   i   j  \0  \0  \0  \0  \0  \0
0000020  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040  \0  \0  \0  \0  \0  \0  \0  \0   A   B   C   D   E   F   G   H
0000060   I   J
0000062
```

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

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char* argv[]){
   char buffer[80];
   char* aux;
   char* aux2;
   int fd_in;
   int fd_out;
   ssize_t rstatus=80;
   int bloque;
   ssize_t contador=0; /* Para modificación adicional, cuenta bytes para el tamaño del
                  buffer */

    /* Configuro fd_in como archivo o entrada estándar */
```

```
if(argc==2){
  if((fd_in=open(argv[1],O_RDONLY))<0){
    printf("Error: %d, al abrir el archivo %s\n", errno, argv[1]);
    perror("Error al abrir el archivo\n");
    exit(EXIT_FAILURE);
  }
}
else {
  fd_in=STDIN_FILENO;
}

/* Creo o sobreescribo archivo de salida */
if((fd_out=open("salida.txt",O_CREAT|O_TRUNC|O_RDWR,S_IRUSR|S_IWUSR))<0){
  printf("Error %d, al crear archivo salida.txt.\n", errno);
  perror("Error al crear archivo de salida.\n");
  exit(EXIT_FAILURE);
}

/* Lectura/escritura a través de buffer */
for (bloque=1; rstatus==80; ++bloque){
    rstatus=read(fd_in,buffer,80);
    if (rstatus<0){
      printf("Error: %d, al leer.\n", errno);
      perror("Error al leer.\n");
      exit(EXIT_FAILURE);
    }

    if (bloque==1){
      asprintf(&aux, "Bloque %d\n", bloque);
      }
    else {
      asprintf(&aux, "\nBloque %d\n", bloque);
    }

    if ((write(fd_out,aux,strlen(aux)))<0){
      printf("Error: %d, al escribir el número de bloque %d.\n",errno, bloque);
      perror("Error al escribir número de bloque.\n");
      exit(EXIT_FAILURE);
    }
    contador+=strlen(aux);


    if ((write(fd_out,buffer,rstatus))<0){
      printf("Error: %d, al escribir bloque número %d.\n",errno, bloque);
      perror("Error al escribir un bloque.\n");
      exit(EXIT_FAILURE);
    }
    contador+=rstatus;

}
/* Modificación adicional. Guardo el archivo en memoria dinámica
   y sobreescribo desde el principio */
    aux=(char*) realloc(aux,contador);
    asprintf(&aux2, "El número de bloques es %d\n",bloque-1);
    if((lseek(fd_out,0,SEEK_SET))<0){
      printf("Error: %d, al reposicionar puntero.\n", errno);
      perror("Error de reposicionamiento de puntero\n");
      exit(EXIT_FAILURE);
    }
    if((read(fd_out,aux,contador))<0){
      printf("Error: %d, al releer el archivo.\n", errno);
      perror("Error de relectura de archivo\n");
```

```
        exit(EXIT_FAILURE);
      }
      if((lseek(fd_out,0,SEEK_SET))<0){
        printf("Error: %d, al reposicionar puntero.\n", errno);
        perror("Error de reposicionamiento de puntero\n");
        exit(EXIT_FAILURE);
      }
      if(write(fd_out,aux2,strlen(aux2))<0){
        printf("Error: %d, al escribir el número de bloques.\n", errno);
        perror("Error al escribir el número de bloques.\n");
        exit(EXIT_FAILURE);
      }
      if(write(fd_out,aux,strlen(aux))<0){
        printf("Error: %d, al reescribir el resto del archivo.\n", errno);
        perror("Error al reescribir el resto del archivo.\n");
        exit(EXIT_FAILURE);
      }

    /* Cerramos archivos y liberamos memoria dinámica */
      free(aux2);
      free(aux);
      if (argc==2 && close(fd_in)<0){
        printf("Error: %d, al cerrar archivo de entrada.\n", errno);
        perror("Error al cerrar archivo de entrada.\n");
        exit(EXIT_FAILURE);
      }

      if (close(fd_out)<0){
        printf("Error: %d, al cerrar archivo de salida.\n", errno);
        perror("Error al cerrar archivo de salida.\n");
        exit(EXIT_FAILURE);
      }



   return EXIT_SUCCESS;

}
```

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

El ejercicio 3 analiza las rutas proporcionadas como argumento al programa (puede haber más de una) y nos indica qué tipo de archivo es, en caso de no poder acceder al archivo devuelve el error correspondiente o no haber indicado archivos como argumento. Utilizando lstat (que evita que nos devuelva los metadatos del fichero al que enlace simbólicamente la ruta proporcionada) y utilizando las macros disponibles nos muestra por pantalla el tipo de pantalla si es conocido o “Tipo de archivo desconocido” en caso contrario.

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

```
#define S_ISREG2(mode) mode&S_IFREG
```