SISTEMAS OPERATIVOS (2017-2018)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Guión Practicas

Antonio Rodríguez Alaminos

14 de octubre de 2017

Índice

Índice de figuras

Índice de tablas

Parte I.
Módulo 2: Uso de los Servicios del SO mediante la API

1. Sesion 1: Llamadas al sistema para el SA(Parte 1)

Ejercicio 1. ¿Qué hace el siguiente programa? Probad tras la ejecución del programa las siguientes órdenes del shell: $ >cat archivo y $ >od -c archivo. Descarga file

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

char buf1[]="abcdefghij";
char buf2[]="ABCDEFGHIJ";

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

        //Parte A
        if((fd=open("archivo",O_CREAT|O_TRUNC|O_WRONLY,S_IRUSR|S_IWUSR))<0) {
                printf("\nError␣%d␣en␣open",errno);
                perror("\nError␣en␣open");
                exit(EXIT_FAILURE);
        }
        //Parte B
        if(write(fd,buf1,10) != 10) {
                perror("\nError␣en␣primer␣write");
                exit(EXIT_FAILURE);
        }
        //Parte C
        if(lseek(fd,40,SEEK_SET) < 0) {
                perror("\nError␣en␣lseek");
                exit(EXIT_FAILURE);
        }
        //Parte D
        if(write(fd,buf2,10) != 10) {
                perror("\nError␣en␣segundo␣write");
                exit(EXIT_FAILURE);
        }

        return EXIT_SUCCESS;
}
```

Este programa lo que hace es almacenar las dos cadenas de caracteres una tras otra en un archivo denominado “archivo”. Esto se realiza en los siguientes etapas o partes del programa:

Parte A: En esta parte el programa abre (si no existe lo genera) el fichero archivo. En caso de tener algún problema, nos informara de él mediante un mensaje, este es generado por la función perror: “Error en open”.

Parte B: En este apartado se escribirá la cadena *buf1* (“abcdefghij”) al inicio del archivo abierto o generado “archivo”. Si se produce algún error, el programa como antes, gracias a la función perror genera un mensaje de error: “Error en el primer write”.

Parte C: Ahora continua realiza un desplazamiento del contador del buffer de lectura del archivo que tenemos abierto. Si sucede algún error perror no devuelve: “Error en lseek”.

Parte D: Se almacena en el fichero el *buf2*, en caso contrario perror nos avisa con: “Error en segundo write”.

Ahora que ya tenemos generado el fichero “archivo pasamos a la segunda parte del ejercicio. Como podemos observar parece que las cadenas se almacenaron una seguida de la otra.

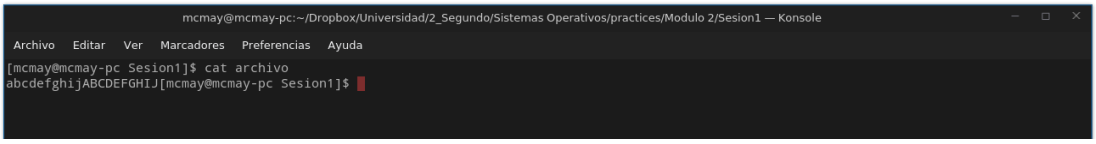

Figura 1.1: Resultado obtenido tras ejecutar $cat archivo

Pero como podemos ver con el siguiente comando esto no es de verdad lo que ha sucedido, ya que en este caso podemos ver como esta separadas una de otras por una cadena de 40 caracteres vacíos.

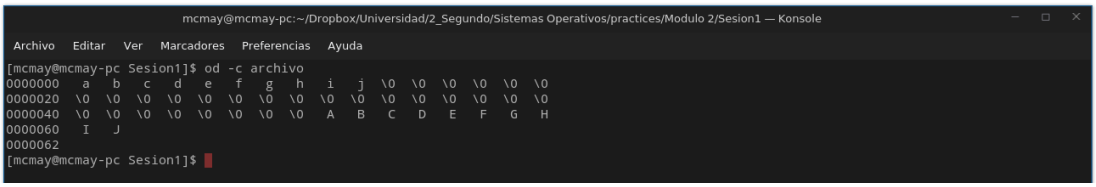

Figura 1.2: Resultado obtenido tras ejecutar $od -c archivo

Ejercicio 2. Implementa un programa que realice la siguiente funcionalidad. El programa acepta como argumento el nombre de un archivo (pathname), lo abre y lo lee en bloques de tamaño 80 Bytes, y crea un nuevo archivo de salida, salida.txt, en el que debe aparecer la siguiente información:

```
Bloque 1
//Aqui van los primeros 80 Bytes del archivo pasado como argumento.
Bloque 2
//Aqui van los segundos 80 Bytes del archivo pasado como argumento.
...
Bloque n
//Aqui van los siguientes 80 Bytes del archivo pasado como argumento.
```

Si no se pasa un argumento al programa se debe utilizar la entrada estándar como archivo de entrada.

Modificación adicional. ¿Cómo tendrías que modificar el programa para que una vez finalizada la escritura en el archivo de salida y antes de cerrarlo, pudiésemos indicar en su primera línea el número de etiquetas "Bloque i" escritas de forma que tuviese la siguiente apariencia?: Descarga file

```
        El numero de bloques es <n_bloques>
```

Aquí tenemos el código que se desarrolla para el programa solicitado.

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

int main(int argc, char const *argv[]) {
        //declaramos los ficheros que utilizaremos
        int fread, fwrite, numbytestmp = 0, contador = 1;
        char buffer[80];


        //comprobamos los argumentos pasados al programa
        if(argc < 2) {
                fread = STDIN_FILENO;
        }else{
                //apertura o generacion del fichero de salida
                if((fread=open(argv[1],O_RDONLY)) < 0) {
                        perror("\nError en open read");
                        exit(EXIT_FAILURE);
                }
        }
```

```
//apertura o generacion del fichero de salida
if((fwrite=open("salida.txt",O_CREAT|O_TRUNC|O_WRONLY,S_IRUSR|S_IWUSR)) < 0) {
        perror("\nError en open write");
        exit(EXIT_FAILURE);
}

//reservamos un trozo para el codigo que se requiere introducir al final
if(lseek(fwrite,80,SEEK_SET) < 0) {
        perror("\nError en el desplazamiento(lseek)");
        exit(EXIT_FAILURE);
}

//lectura mientras de pueda del fichero se almacena en ftmp
while ((numbytestmp = read(fread, &buffer, 80)) > 0){
        //introducimos el numero de bloque
        char titulo[16];
        if((sprintf(titulo, "\nBloque %i \n", contador)) < 0){
                perror("\nError en la inserci n del titulo");
                exit(EXIT_FAILURE);
        }
        if (write(fwrite, &titulo, sizeof(titulo)) < 0) {
                perror("\nError en el write de la inserci n del tituto ");
                exit(EXIT_FAILURE);
        }

        //pasamos a introducir los 80 bytes correspondientes por bloque
        if ((write(fwrite, &buffer, numbytestmp)) < 0) {
                perror("\nError en la inserci n del buffer");
                exit(EXIT_FAILURE);
        }

        contador++;
}

//introducimos el titulo inicial
char buffer2[40];
if((sprintf(buffer2, "El n mero de bloques total es de: %i.\n", contador-1)) <
    0){
        perror("\nError en la inserci n del titulo inicial.");
        exit(EXIT_FAILURE);
}
if(lseek(fwrite,0,SEEK_SET) < 0){
        perror("\nError en el desplazamiento de la inserci n del titulo final
            .");
        exit(EXIT_FAILURE);
}
if((write(fwrite, &buffer2, sizeof(buffer2))) < 0) {
        perror("\nError en la inserci n del titulo final.");
        exit(EXIT_FAILURE);
}
```

```
        //cerramos los fichero abiertos
        close(fread);
        close(fwrite);

        return EXIT_SUCCESS;
}
```

Capturas de pantalla que muestran lo sucedido al ejecutar el programa introduciendo como argumento el código del ejercicio anterior.

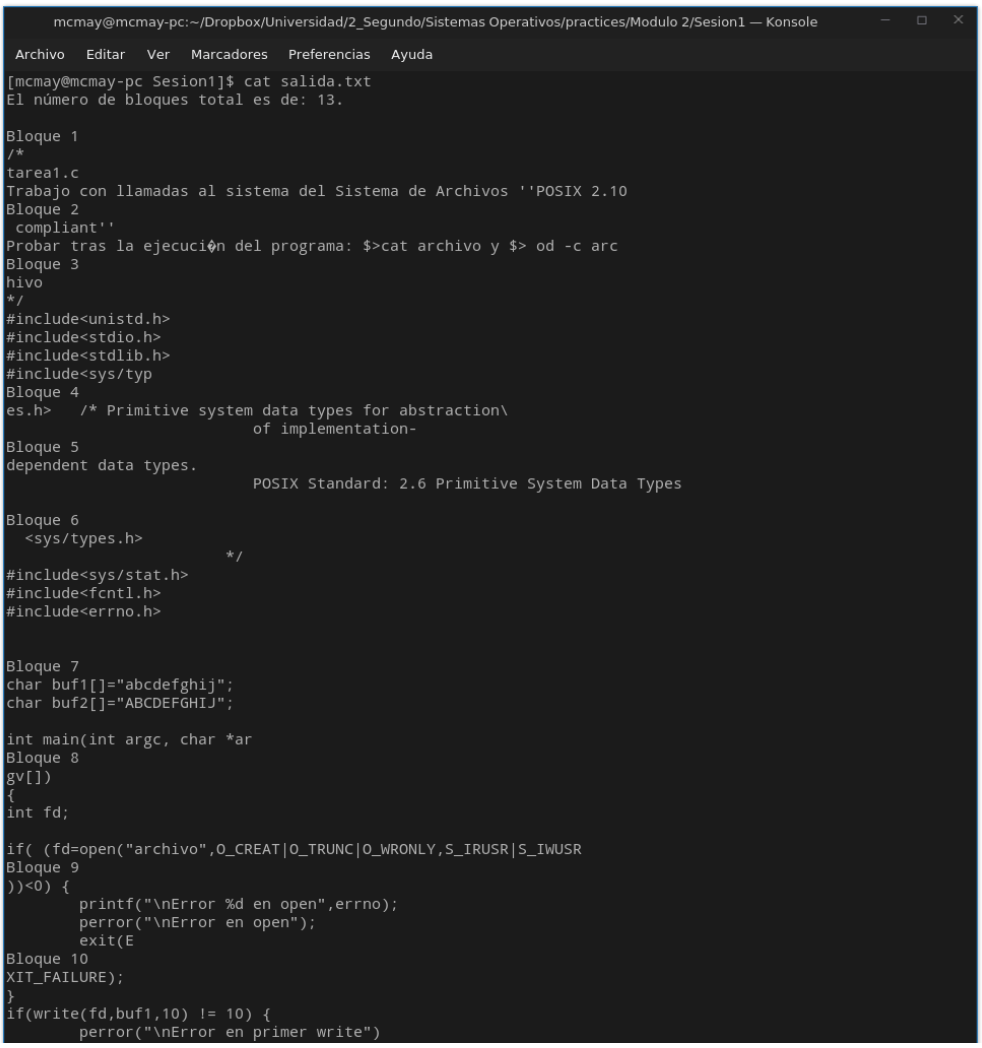

Figura 1.3: Primera parte de la muestra por pantalla del ejercicio.

```
Bloque 11
;
        exit(EXIT_FAILURE);
}

if(lseek(fd,40,SEEK_SET) < 0) {
        perror("\nError en ls
Bloque 12
eek");
        exit(EXIT_FAILURE);
}

if(write(fd,buf2,10) != 10) {
        perror("\nError en
Bloque 13
 segundo write");
        exit(EXIT_FAILURE);
}

return EXIT_SUCCESS;
}
[mcmay@mcmay-pc Sesion1]$
```

Figura 1.4: Primera parte de la muestra por pantalla del ejercicio.

Ejercicio 3. ¿Qué hace el siguiente programa? Descarga file

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

int main(int argc, char *argv[])
{
        int i;
        struct stat atributos;
        char tipoArchivo[30];

        //Parte A
        if(argc<2) {
                printf("\nSintaxis de ejecucion: tarea2 [<nombre_archivo>]+\n\n");
                exit(EXIT_FAILURE);
        }

        for(i=1;i<argc;i++) {
                //Parte B
                printf("%s: ", argv[i]);
                if(lstat(argv[i],&atributos) < 0) {
                        printf("\nError al intentar acceder a los atributos de %s",argv[
                            i]);
                        perror("\nError en lstat");
                }
```

```
                //Parte C
                else {
                        if(S_ISREG(atributos.st_mode)) strcpy(tipoArchivo,"Regular");
                        else if(S_ISDIR(atributos.st_mode)) strcpy(tipoArchivo,"
                            Directorio");
                        else if(S_ISCHR(atributos.st_mode)) strcpy(tipoArchivo,"Especial
                             de caracteres");
                        else if(S_ISBLK(atributos.st_mode)) strcpy(tipoArchivo,"Especial
                             de bloques");
                        else if(S_ISFIFO(atributos.st_mode)) strcpy(tipoArchivo,"Tuberia
                             con nombre (FIFO)");
                        else if(S_ISLNK(atributos.st_mode)) strcpy(tipoArchivo,"Enlace
                            relativo (soft)");
                        else if(S_ISSOCK(atributos.st_mode)) strcpy(tipoArchivo,"Socket
                            ");
                        else strcpy(tipoArchivo,"Tipo de archivo desconocido");
                        printf("%s\n",tipoArchivo);
                }
        }

        return EXIT_SUCCESS;
}
```

Parte A: En esta parte el programa comprueba que se le introducen correctamente los argumentos. En caso de que se produzca algún error se muestra el mensaje “Error de sintaxis de ejecucion”

Parte B: Ahora lo que hace el programa es imprimir el nombre del fichero pasado por argumento. Y en este caso si se produce algun error se muestra “Error al intentar acceder a los atributos de fichero pasado por argumentos”

Parte C: Por ultimo recorre todas estas condiciones buscando el tipo de fichero que se le paso por argumento y indica de que tipo es. En caso de no encontrarlo devuelve que no se encontró.

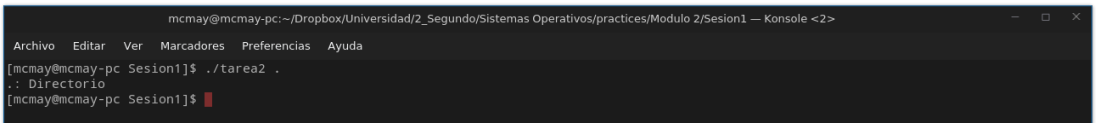

Figura 1.5: Resultado del ejecutable introduciendo el directorio origen

Ejercicio 4. Define una macro en lenguaje C que tenga la misma funcionalidad que la macro S_ISREG(mode) usando para ello los flags definidos en <sys/stat.h>para el campo st_mode de la struct stat, y comprueba que funciona en un programa simple. Consulta en un libro de C o en internet cómo se especifica una macro con argumento en C. Descarga file

El nuevo código que se pide, con las modificaciones realizadas:

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

//Parte A
#define S_ISREG2(mode) (((mode) & S_IFMT) == 0100000)

//Parte B
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(EXIT_FAILURE);
        }

        for(i=1;i<argc;i++) {
                printf("%s: ", argv[i]);
                if(lstat(argv[i],&atributos) < 0) {
                        printf("\nError al intentar acceder a los atributos de %s",argv[
                            i]);
                        perror("\nError en lstat");
                }
                else {
                        if(S_ISREG2(atributos.st_mode)) strcpy(tipoArchivo,"Regular");
                        else if(S_ISDIR2(atributos.st_mode)) strcpy(tipoArchivo,"
                            Directorio");
                        else if(S_ISCHR2(atributos.st_mode)) strcpy(tipoArchivo,"
                            Especial de caracteres");
                        else if(S_ISBLK2(atributos.st_mode)) strcpy(tipoArchivo,"
                            Especial de bloques");
                        else if(S_ISFIFO2(atributos.st_mode)) strcpy(tipoArchivo,"
                            Tuberia con nombre (FIFO)");
                        else if(S_ISLNK(atributos.st_mode)) strcpy(tipoArchivo,"Enlace
                            relativo (soft)");
                        else if(S_ISSOCK(atributos.st_mode)) strcpy(tipoArchivo,"Socket
                            ");
                        else strcpy(tipoArchivo,"Tipo de archivo desconocido");
                        printf("%s\n",tipoArchivo);
                }
        }

        return EXIT_SUCCESS;
}
```

Como se puede opserbar en el apartado A, se realiza la definición del nuevo S_ISREG2.

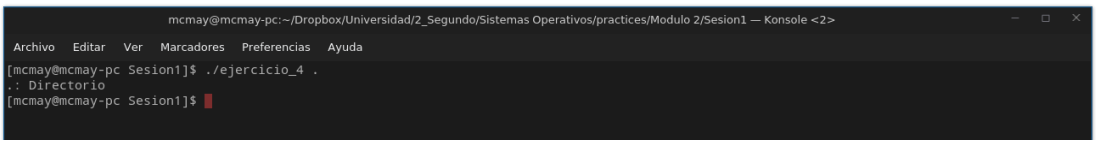

Figura 1.6: Resultado del ejecutable introduciendo el directorio origen