Exercise 1

a) There is a disk with a file system for Unix. Assuming that the size of the block chosen for the file system is 2KBytes, that the addresses of the blocks are 4 bytes and the inodes have the traditional structure (10 direct pointers, 1 simple indirect, 1 double indirect and 1 triple indirect) . It indicates the number of blocks that a 4500 KByte file occupies, including both data and address blocks.

b) There is a Unix-like filesystem with the following information:

I-nodes table:

Inode No.	1	2	3		
Type	Directory	Directory	File		
Physical links counter	3	2	1		
Data Block Address	11	12	13		
…….					

Data Blocks:

Block No.	11	12	13		
Contents	**. 1**	**. 2**	File Data		
	.. 1	**.. 1**			
	d 2	**f1 3**			

Modify the previous tables to reflect the execution of the following 2 operations:

ln f1 / d / f2 Physical link from / d / f2 to File f1

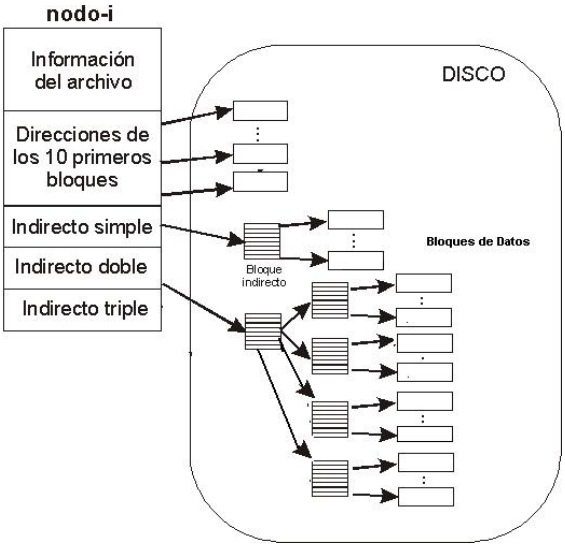
ln –s f1 / d / f3 Symbolic link from / d / f3 to File f1

Solution 1

a)

There are 512 block addresses in each address block: 2KBytes / 4 bytes = 512 positions.
The 10 simple pointers will point to the first 20 KBytes of the File.

Each block of addresses needed will point to 1 MByte (1024Kbytes) of the File: 512 block addresses * 2 Kbytes per block = 1024 Kbytes = 1Mbyte of the File.
Therefore to reach 4500 Kbytes we need 5 blocks of addresses.
The first one we get from the simple indirect pointer, which points directly to one of them. For the remaining four we need the double indirect pointer that will point to 1 address block that will point to those 4 necessary address blocks.
Then 5 + 1 = 6 address blocks are needed.
The File Data block size is: 4500 KBytes / 2Kbytes = 2250 Blocks
Then the total size is 2256 blocks plus the inode.



b)

Inode Table:

Inode No.	1	2	3	**4**	
Type	Directory	Directory	File	**Enlace Simbólico**	
Physical links counter	3	2	2	**1**	
Data Block Address	11	12	13	**14**	
…….					

Data Blocks:

Block No.	11	12	13	**14**	
	. 1	**. 2**	File Data	**f1**	
	.. 1	**.. 1**			

Contents	**d 2**	**f1 3**			
		f2 3			
		f3 4			

Exercise 2

It has been decided to purchase a 100MB hard drive for high performance tasks.
The system administrator of the company has decided to give a series of parameters in the configuration of the Files Type UNIX system (volume size, block size and number of i-nodes) in order to maximize the performance of the system as refers to number of disk accesses.
The final characteristics that will be applied when creating the Files system using the mkfs command are:

- Block size: 1024 bytes.
- Size of the address of the blocks: 4 bytes.
- Number of i-nodes: 50
- A load block (Boot) that occupies 2 blocks.
- A Superblock that occupies 8 Kbytes and a Bitmap is used to indicate which blocks are used and which are free.
- Fields of an i-node:
  - File attributes:
    - ♣ Owner and group ID.
    - ♣ Read permissions for the owner, group and the rest of the world.
    - ♣ Write permissions for the owner, group and the rest of the world.
    - ♣ Execution permits for the owner, group and rest of the world.
  - o Link counter (1 byte).
  - o 4 indirect pointers.
  - o 4 simple indirect pointers.
  - o 4 double indirect pointers.

Answer the following questions:

a) What is the maximum size of a File stored on this disk? Reason for the answer.
b) If the objective is to minimize access to disk blocks, what parameter do you think the administrator should increase / decrease to achieve this purpose?
c) Assuming that the Files system is capable of using hard and symbolic links. Calculate:

- Maximum number of hard links (in addition to the initial name) that can be made from a File on this disk.

• Maximum number of symbolic links that can be made from a File.

Solution 2

a)The maximum size of a File depends on the inode references and the number of block addresses that fit in a data block (indirect reference) which is equal to the block size (1024 bytes) between the size of the address of block (4 bytes), like this:

REFERENCES	Block No.S
4 Blocks directos	4
4 referencia indirecta simple:	4*1024 / 4 = 1024
4 referencia indirecta doble:	4*(1024 /4)*(1024 / 4) = 262144

Therefore the No. of blocks is: 4 + 1024 + 262144 = 263172 blocks;
Thus, the maximum size (in bytes) of a File in this File system is:
263172 * 1024 = 269488128 bytes => 263172 KB (approximately 263 Mbytes)
Since the disk is 100 MB, a File of maximum size would not fit; therefore the size of a File is limited by the capacity of the disk. For this case, the maximum disk size must be calculated as follows:
Maximum size of a File = (Disk size / Block size) - Number of blocks occupied by the rest of the structures.

Las estructuras típicas que nos define el enunciado son:

Boot	SuperBlock	Mapas de Bits	Nodos-I	Datos y Directorys

Thus:

Boot size = 2 blocks
Superblock size = 8 blocks
Bitmap Size = (100 * 2 ^ 20) / 2 ^ 10 = 100 bits => 1 block
Node-I Size = 50
Maximum file size = (100 * 2 ^ 20/2 ^ 10) - (2 + 8 + 1 + 50) = 100 - 61 = 39 blocks
Maximum size in bytes = 39 * 1024 bytes = 39 kbytes

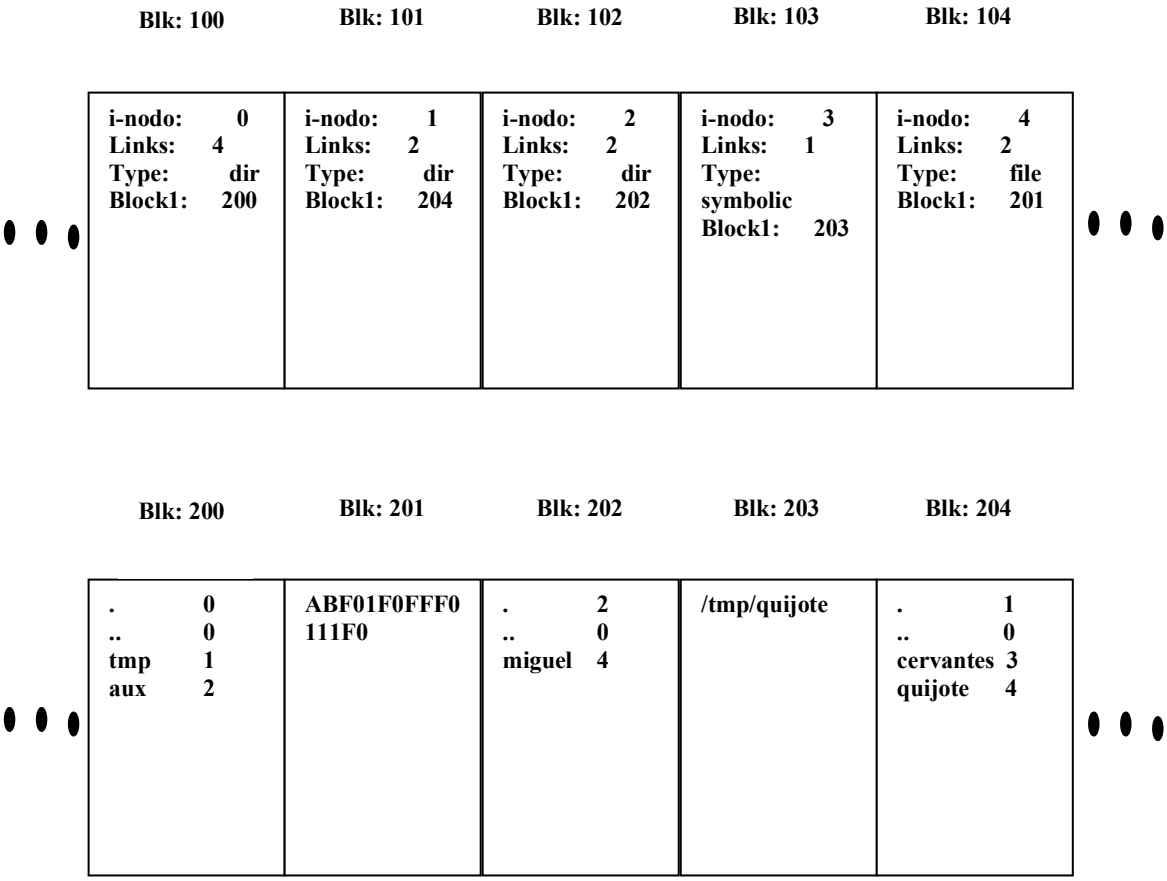
b) In general, the parameter that is used to achieve the least number of disk accesses is the block size. The transfer of a File requires looking for each block that forms it on disk, waiting for the latency time and making the data transfer.
The trade-off to the large block size is the amount of internal fragmentation is high and that causes the percentage of disk usage to decrease.

c) The number of hard links only depends on the size of the link counter. that indicates the number of hard links made, the link counter of an inode of this Files system has 1 byte, so there can be from 0 to 255 hard links. Since the initial name is also a hard link, the No. of possible hard links, in addition to the initial name, will be 254 links.

The number of symbolic links only depends on the number of Files that can be created. Since the disk only has 50 inodes, the maximum number of Files will be 50. There are at least two occupied inodes on the disk (the one in the root Directory and the one in the "original" File). Therefore the maximum number of symbolic links to a File can be 48.

Exercise 3

Given the file system in the figure, whose block size is 512 bytes.



It is requested to execute the following instructions in the order indicated, indicating what information from the Files system is used at all times and what it is used for, and what are the modifications that must be made at each moment.

a) cat /aux/miguel
b) rm /tmp/quijote
c) cat /tmp/cervantes
d) rm /aux/miguel
e) rm /tmp/cervantes

NOTE: cat command: displays a text file on the screen.
Command rm: delete a File.

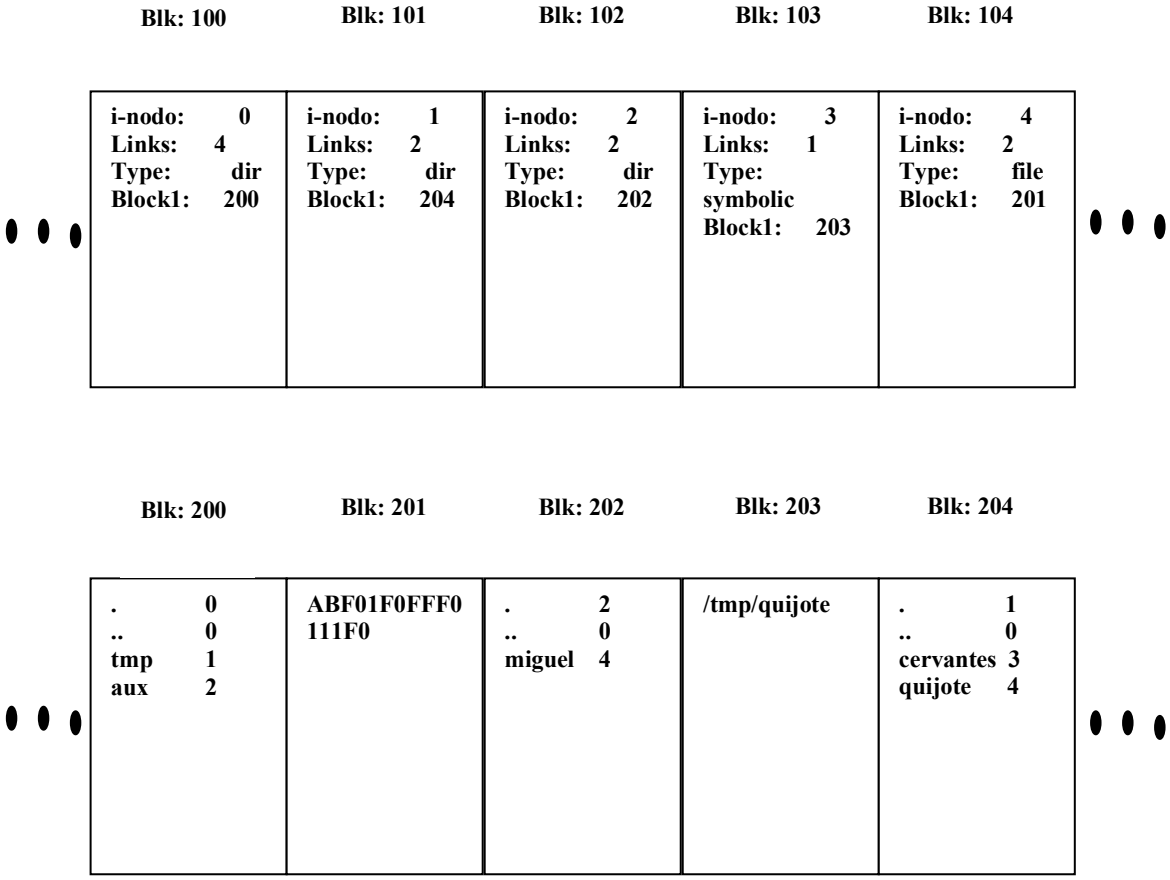
Solution 3

1. cat /tmp/cervantes
   a) Read the inode of the root Directory (inode 0 - Block 100) and then the first Block of data of said Directory (Block 200) is read, as there is the entry of 'tmp' (inode `1) there is no need to read more Blocks.
   b) Read the inode of tmp (inode 1 - Block 101), it is seen that it is a Directory and the first Block of data is read (Block 204), as there is the entry of 'cervantes' (inode 3) it is not necessary to read more Blocks.
   c) Read the inode of cervantes (inode 3 - Block 103), it is seen that it is a symbolic link and the first Block of data (Block 203) is read obtaining the path of the linked File '/ tmp / quijote'
   d) Read the inode of the root Directory (inode 0 - Block 100) and then the first Block of data of said Directory (Block 200) is read, as there is the entry of 'tmp' (inode `1) there is no need to read more Blocks.
   e) Read the inode of tmp (inode 1 - Block 101), it is seen that it is a Directory and the first Block of data (Block 204) is read as there is no entry equal to 'quijote', an error is returned for File not found.

NOTA: no changes occur in the Files system.

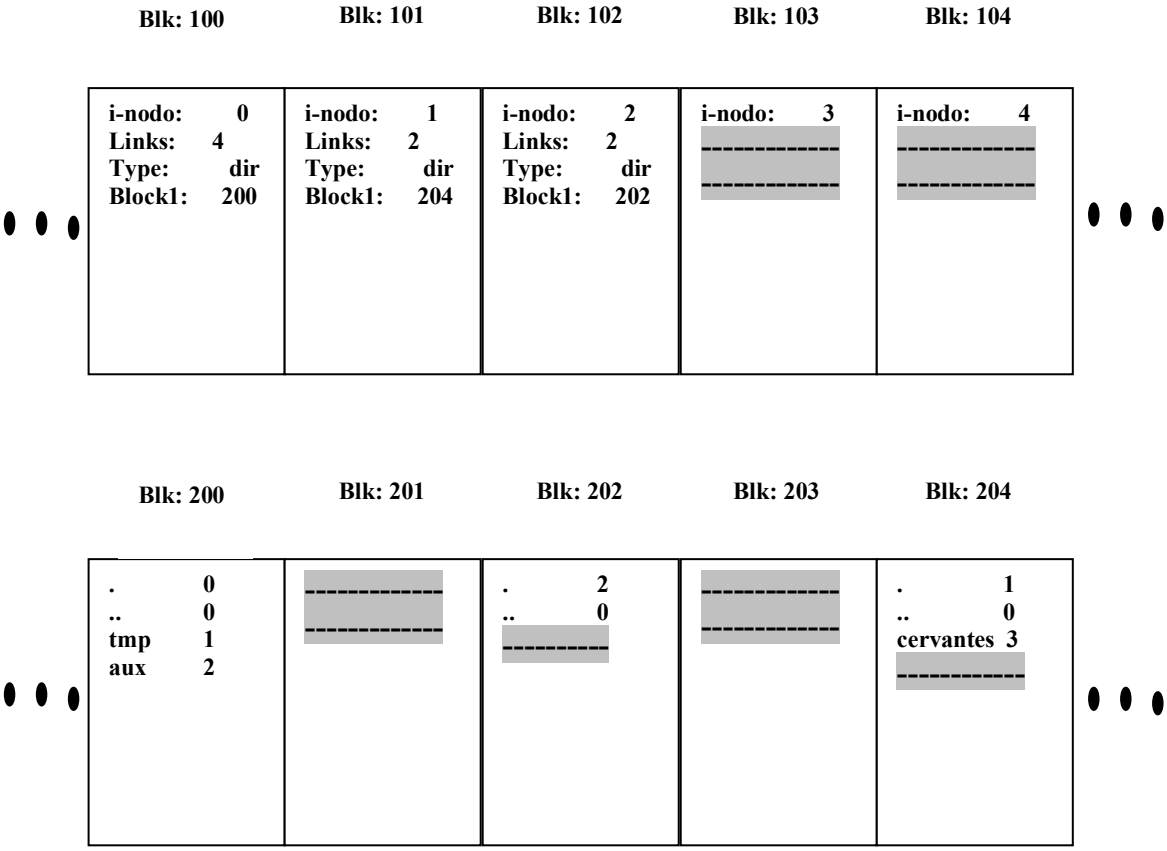
2. rm /aux/Miguel
   a) Read the inode from the root Directory (inode 0 - Block 100) and then read the first Block of data from
   b) Said Directory (Block 200), as there is the entry of ‘aux’ (inode 2), there is no need to read more Blocks.
   c) Read the aux inode (inode 2 - Block 102), it is seen that it is a Directory and the first block of data (Block 202) is read, as there is the entry of 'miguel' (inode 4) it is not necessary to read more Blocks. As the File must be deleted, its entry is deleted in this Directory.
   d) Reading miguel's inode (inode 4 - Block 104), it is seen that it is a File, to delete it the number of Links is decremented (remaining at 0), since it is equal to zero, its inode and its Data Blocks are deleted. (Block 201) for this the bitmap will have to be updated.

Final state of the File System:



3. rm /tmp/Cervantes
   a) Read the inode of the root Directory (inode 0 - Block 100) and then the first Block of data of said Directory (Block 200) is read, as there is the entry of 'tmp' (inode 1) there is no need to read more Blocks .
   b) Read the aux inode (inode 1 - Block 101), it is seen that it is a Directory and the first data Block (Block 204) is read, as there is the entry of 'cervantes' (inode 3) it is not necessary to read more Blocks. As the File must be deleted, its entry is deleted in this Directory.
   c) Read the inode of cervantes (inode 3 - Block 103), it is seen that it is a symbolic link (although that does not matter), to delete it the number of Links is decremented (remaining at 0), since it is equal to zero its inode is deleted and its Data Blocks. (Block 203) to do this, the bitmap will have to be updated.

Final state of the File Systen



Exercise 4.

a) There is a disk with Unix files system with the following characteristics:
- Block size of 4 KBytes.
- Block addresses: 4 bytes.
- Structure of the i-node:
  - ♣ 10 direct pointers.
  - ♣ 1 simple indirect pointer.
  - ♣ 1 double indirect pointer.
  - ♣ 1 triple indirect pointer.

Indicate the number of Blocks that a 10 MByte File occupies, including both the Data Blocks and the addresses.

b) There is a Unix File Type system with the following information:

Inode Table:

Inode No.	2	3	4	5	
Type	Directory	Directory	File	Symbolic Link	
Physical links counter	3	2	2	1	
Dirección Block Datos	11	12	13	14	
…….					

Data Blocks:

Block No.	11	12	13	14	
Contents	. 2	. 3	File Data	/d/f1	
	.. 2	.. 2			
	d 3	f1 4			
		f2 4			
		f3 5			

Explain how the i-nodes and Data Blocks look after performing each of the following operations (Make a table of i-nodes and Data Blocks for each section):

- rm /d/f1
- rm /d/f2
- rm /d/f3
- mkdir /d/d1

SOLUTION.

a)

In each Address Block, 1024 Block addresses can fit: 4KBytes / 4 bytes = 1024 positions.
The 10 simple pointers will point to the first 40 Kbytes of the File.
Each block of addresses that is needed will point to 1024 * 4 = 4 Mbyte of the File
Therefore to reach 10 Mbyte we need 3 address blocks.
The first one we get from the simple indirect pointer, which points directly to one of them. For the remaining 2 we need the double indirect pointer that will point to 1 Block of addresses that will point to those 2 Blocks of necessary addresses.
Then 3 + 1 = 4 Address Blocks are needed
The size in File Data Blocks is: 10 Mbyte / 4Kbytes = 10240/4 = 2560 Blocks
Then the total size is 2563 Blocks plus the inode.

b)

1-

Inode table after rm /d/f1:

Inode No.	2	3	4	5	
Type	Directory	Directory	File	Enlace Simbólico	
Physical links counter	3	2	1	1	
Dirección Block Datos	11	12	13	14	
…….					

Data Blocks after rm /d/f1:

Block No.	11	12	13	14	
	. 2	. 3	File Data	/d/f1	

Contents	.. 2	.. 2			
	d 3	f2 4			
		f3 5			

2-

Inode table after rm /d/f2:

Inode No.	2	3		5	
Type	Directory	Directory		Enlace Simbólico	
Physical links counter	3	2		1	
Dirección Block Datos	11	12		14	
…….					

Data Blocks after rm /d/f2:

Block No.	11	12	13	14	
Contents	. 2	. 3	Free	/d/f1	
	.. 2	.. 2			
	d 3	f3 5			

3-

Inode table after rm /d/f3:

Inode No.	2	3			
Type	Directory	Directory			
Physical links counter	3	2			
Dirección Block Datos	11	12			
…….					

Data Blocks after rm /d/f3:

Block No.	11	12	13	14	
Contents	. 2	. 3	Free	Free	
	.. 2	.. 2			
	d 3				

4-

Inode table after mkdir /d/d1:

Inode No.	2	3	4		
Type	Directory	Directory	Directory		
Physical links counter	3	2	2		
Dirección Block Datos	11	12	13		
…….					

Data Blocks after mkdir /d/d1:

Block No.	11	12	13	14	
Contents	. 2	. 3	. 4	Free	
	.. 2	.. 2	.. 3		
	d 3	d1 4			

Exercise 5.

a) Write a program that writes in a File "records.dat" a collection N of records that include a code and a customer name, ordered by code. If the File does not exist, the program must create it.

b) Write a program that reads the records from the previous file and displays them in order on the screen.

SOLUTION

a)

```
// file:  ej5-registro.h
struct  registro
{
   int codigo;
   char nombre[30];
};

typedef struct registro TypeRegistro ;

// file ej5-escribir.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

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

#include "registro.h"

#define N 4


int main ( int argc, char *argv[] )
{
    int file1 ;
    int i ;
    TypeRegistro registro1 ;
    int posicion1 ;
    char nombreFich[30]="registros.dat" ;


    /* Abrir File */
        file1 = open (nombreFich,O_APPEND|O_WRONLY) ;
    if (-1 != file1) {
      printf ("El File ya existe, anyado al final\n");
    }

    if (-1 == file1) {
      printf ("El File no existe\n");
      printf ("Se va a crear el File\n");
          file1=open (nombreFich, O_CREAT| O_WRONLY,
S_IWUSR|S_IRUSR) ;
      if (-1 == file1) {
          printf ("Error en la creacion :1\n");
          exit (-1);
      }
    }

    /* Escribir registros */
        registro1.codigo=0;
        for (i=1; i< N; i++) {

        registro1.codigo++;
        strcpy (registro1.nombre, "nombre dos");
        write (file1, &registro1, sizeof (registro1) );
        }

    /* Imprimir la posici?n */
        posicion1 = lseek(file1,0,SEEK_CUR) ;
        printf ("Estoy en la posicion %d del File\n", posicion1
);

    /* Cerrar File */
    close(file1);
```

```
}
```

b)

// file ej5-registro.h

```
struct  registro
{
   int codigo;
   char nombre[30];
};

typedef struct registro TypeRegistro ;
```

// file ej5-leer.c

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

#include "registro.h"

int main ( int argc, char *argv[] )
{
    int file1;
    int bytes_leidos;
    char nombreFich[40]="registros.dat";
    TypeRegistro registro1;


    file1 = open (nombreFich, O_RDONLY) ;
    if (file1 == -1) {
        fprintf (stderr, "No se ha podido abrir el File\n");
        exit (-1) ;
        }

    bytes_leidos = read (file1, &registro1, sizeof(registro1));
    while ( bytes_leidos !=0 ){
            printf  ("registro leido -> codigo:%d:
nombre:%s:\n",
             registro1.codigo,
             registro1.nombre);
            bytes_leidos = read (file1, &registro1,
sizeof(registro1));
        }
```

```
    close(file1);
}
```

Exercise 6.

a) Write a program that writes in a File "records.dat" a collection N of records that include a code and a customer name, ordered by code. The program must ask for the client code and name, jump to the corresponding position in the File and write it. If the File does not exist, the program must create it.

b) Write a program that asks for the code of a client, reads it from the File and displays it in order on the screen. If the customer does not exist, you will need to write an empty record.

SOLUTION

a)

```
// file ej6-registro.h
typedef struct {
     int codigo;
      char nombre[30];} TypeRegistro;

// file ej6-escribirregAccDirecto.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "registro.h"

/*
Programa que pide un código/posición de persona y escribe los
datos en laposición del File que corresponden
*/
int main () {
        int f;
        int numbyteescritos;
        TypeRegistro reg;
        char nombreFich[30]="registros.dat";
        mode_t mode = S_IRUSR | S_IWUSR;

        /*  Tratamos de abrir el File.  Si no existe se crea  */
        if ((f=open (nombreFich, O_WRONLY|O_CREAT, mode)) == -1)
{
          perror ("Error en la apertura del File \n");
          exit (-1);
```

```
        }

          printf( "Dar codigo del usuario:");
          scanf ("%d", &reg.codigo);
          printf( "Dar nombre del usuario:");
          scanf ("%s", &reg.nombre);

        //resto -1 porque la persona de código n está en la
posición n-1
          lseek (f, (reg.codigo-1) * sizeof (reg), SEEK_SET);
          numbyteescritos=write (f, &reg, sizeof (reg) );
         if (numbyteescritos == -1)
          printf  ("Error al escribir registro
codigo:%d\n",reg.codigo);
         else
          printf  ("registro escrito,  codigo:%d:
nombre:%s:\n",reg.codigo, reg.nombre);

          close(f);
}
```

b)

```
// file ej6-registro.h
typedef struct {
    int codigo;
     char nombre[30];} TypeRegistro;
```

// file ej6-leerregDescAccDirecto.c

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

#include "registro.h"

/*
Programa que pide un código/posición de persona y muestra los
datos que se encuentran en esa posición  en el File
*/
int main () {
        int f;
        int numbyteleidos,cod;
        char nombreFich[40]="registros.dat";
        TypeRegistro reg;

        if ((f=open (nombreFich, O_RDONLY)) == -1)
          fprintf (stderr, "No se ha podido abrir el File\n");
        else {
```

```
          printf( "Dar codigo del usuario:");
          scanf ("%d", &cod);
        //resto -1 porque la persona de código n está en la
posición n-1
          lseek (f, (cod-1) * sizeof (reg), SEEK_SET);
          numbyteleidos=read (f, &reg, sizeof(reg));
          printf  ("registro leido,  codigo:%d:
nombre:%s:\n",reg.codigo, reg.nombre);
          close(f);
        }
}
```

Exercise 7.

a) Write a program that uses the write call to write one message to standard output and another to standard error.
b) Write a possible command to see the output in 2 different files for stdout and stderr.

SOLUTION

a)

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
int i=3;
  char cad[10];
int j=3;

  write (STDOUT_FILENO, "Escrito en salida estandar \n",
    strlen ("Escrito en salida estandar \n"));
  write (STDERR_FILENO, "Escrito en salida errores \n",
    strlen ("Escrito en salida errores \n"));
}
```

b)

./a.out > std_out 2>std_err

Exercise 8.

Write a C program to demonstrate the use of the "dup" system call. The program should redirect the error output to a file "f.dat" and duplicate it on the standard output using dup and dup2. To see the effect you must print something at each step.

SOLUTION.

```
int main () {

   int d,daux;

   /* Cierro la salida de errores */
   close (2);
   /* Escrito en pantalla */
   printf ("UNO\n");
   /* Abro un File en el descriptor asociado a la salida de
errores ya que está Free porque lo acabo de cerrar*/
   d=open ("f.dat", O_CREAT|O_WRONLY,0777);
   if (d<0) exit (1); /* Si hay error finalizo el programa*/
   /* copio la salida estandar al primer descritor Free ( en el
descriptor 3 está "la pantalla") */
   daux=dup(1);
   /* Asocio la salida estandar al File f.dat */
   dup2 (d,1);
   /* Se escribe en el File f.dat*/
   printf ("DOS\n");
   /* Asocio la salida estandar a lo que hay en daux (la
pantalla)*/
   dup2(daux,1);
   /* escrito en pantalla*/
   printf ("TRES\n");
}
```

Exercise 9.

Write a C program that creates a new Directory "DirNuevo" in the current Directory with read and write permissions for the owner user. Next you must execute a call to read the path of the current Directory and show all the Directories (absolute paths) that exist from the current Directory to the root Directory

SOLUTION.

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

int main(){
  char nombreDir [80];

  printf ("Creo un Directory\n");
  mkdir ("./DirNuevo", S_IWUSR|S_IRUSR);
  if (errno != 0 )
          perror("Error creacion ./DirNuevo");
```

```
  strcpy (nombreDir,"");
  getcwd(nombreDir,80);
  printf ("Nombre del Directory actual=%s\n", nombreDir);
  while ( strcmp(nombreDir,"/")!=0){
    chdir ("..");
    getcwd(nombreDir,80);
    printf ("Nombre del Directory actual=%s\n", nombreDir);
  }
}
```

Exercise 10.

Write a C program that receives a name from Directory and shows on the screen the names of Files and Directories it contains, their mode, whether or not they have read permission for the owner, if they are Directorys, and for Files modified in the last 10 days shows your access date.

SOLUTION.

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>

int main () {
int er;
char nomdir[100], nomfich[100], resp[30];
struct stat atr;
DIR *d;
struct dirent *rd1;
time_t fecha;

printf ("Nombre Directory\n");
fgets (nomdir, sizeof (nomdir), stdin);
/* hay que quitar el \n del nombre del Directory*/
nomdir[strlen(nomdir)-1]='\0';

fecha=time(&fecha);
if ((d=opendir(nomdir))==NULL) {
  printf ("No existe ese Directory \n");
  return -1;
}
else {
  while (( rd1 =readdir(d)) != NULL) {
    if ( (strcmp(rd1->d_name, ".")!=0 )&& (strcmp(rd1->d_name,
```

```
"..")!=0 )){
      strcpy (nomfich, nomdir);
      strcat (nomfich, "/");
      strcat (nomfich, rd1->d_name);
      printf ("File :%s:", nomfich);
      er=stat (nomfich, &atr);
      printf ("modo :%#o:", atr.st_mode);
      if ((atr.st_mode & 0400) != 0)
         printf (" permiso R para propietario\n");
      else
         printf (" No permiso R para propietario\n");
      if (S_ISDIR(atr.st_mode)) printf (" Es un Directory \n");
      if (S_ISREG(atr.st_mode))
         /* Files modificados en los ultimos 10 dias */
        if ( (fecha - 10*24*60*60) <  atr.st_mtime) {
          printf ("FILE:%s: fecha acceso %s, en sgdos %d\n",
rd1->d_name,
                           ctime (&atr.st_mtime),(int)
atr.st_mtime );
        }
    }
  } /* while*/
  closedir (d);
}
}/* main*/
```

Exercise 11.

Write a C program that allows you to check the operation of the "umask" call. To do this, you need to apply the file mask to Type "rwxrwx ---" and create a file. Then you must change the mask to "rw-r --- w-" and create another File applying that mask. Check the result using "ls –l".

SOLUTION

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

int main () {
  mode_t permisos, mascara_nueva, mascara_anterior;
  FILE *f;
  int fd;

  /* ponemos como mascara  rwxrwx--- = 0770 */
  mascara_nueva=S_IRUSR | S_IWUSR |S_IXUSR | S_IRGRP | S_IWGRP|
S_IXGRP;
  mascara_anterior=umask (mascara_nueva);
```

```
  /* creamos un File Type FILE *  */
  if ((f=fopen("nuevo_fich_str","w"))!= NULL)
    fclose(f);

  /* creamos un File Type descriptor con los permisos rw-r---w-
=0642 */
  permisos =S_IRUSR|S_IWUSR|S_IRGRP|S_IWOTH;
  if ((fd=open("nuevo_fich_Des",O_CREAT, permisos))> 0)
    close(fd);
  umask (mascara_anterior);
}
```

Exercise 12.

Writing a program modifies the access date of "existing_fich" by placing it 2 hours compared to the current date and modifies the modification date by placing it 3 hours before the one it had.
It can be checked with an ls -l before and after the execution of the program.
"Existing_file" must be created first.

SOLUTION

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

int main () {
  int er=0;
  struct stat atributos;
  struct utimbuf tiempo_fich;
  time_t hora_actual;


  er =stat ("fich_existente", &atributos);
  if (er!=0) printf ("Error\n");
  else {
    hora_actual =time (&hora_actual);
    /* adelanto la fecha de acceso al File 2 horas respecto a la
actual*/
   printf ("Hora actual: %d \n", (int)hora_actual);
    tiempo_fich.actime =hora_actual -2*60*60;
   printf ("Cambio hora actual: %d \n",
```

```
(int)tiempo_fich.actime);
    /* adelanto la fecha de modificación del File 3 horas
respecto a la que tenía */
    tiempo_fich.modtime=atributos.st_mtime- 3*60*60;
   printf ("Cambio hora modificacion: %d \n",
(int)tiempo_fich.modtime);
    utime ("fich_existente", &tiempo_fich);
  }
}
```

Exercise 13

Given a hard disk with a Unix-like filesystem with the following specifications: the block size is 4KB, the block address length is 4 bytes and the i-nodes have a traditional structure (10 direct pointers, 1 single indirect pointer, 1 double indirect pointers and 1 triple indirect pointer).

Answer the following questions:

a) What is the number of blocks (including both the data and address blocks) that have the following files:
   - File A with a size of 20 KBytes
   - File B with a size of 200 KBytes
   - File C with a size of 2000 KBytes
   - File D with a size of 20000 KBytes

b) What is the maximum file size?

Solution

In each address block 1024 block addresses can be allocated: 4KBytes /4 bytes =1024 addresses

a. Number of blocks

- File A with a size of 20 KBytes

  A 20KB file will use 5 data blocks, using the first 5 direct pointers.

  That is, it will use **5 blocks**

- File B with a size of 200 KBytes

  A 200Kb file will use 50 data blocks. The first 10 will be addressed by the direct pointers. The remaining 40 will be pointed by the single indirect pointers (an extra block is needed).

  That is, it will use 50 blocks + 1 block (related to the single indirect pointer) = **51 blocks**

- File C with a size of 2000 KBytes

A 2000KB file will use 500 data blocks. The first 10 will be addressed by the direct pointers. The remaining 490 will be pointed by the single indirect pointers (an extra block is needed).

That is, it will use 500 blocks + 1 block (related to the single indirect pointer) = **501 blocks**

- File D with a size of 20 000 Kbytes

A 20000KB file will use 5000 data blocks. The first 10 will be addressed by the direct pointers. The next 1024 blocks will be pointed by the single indirect pointers (an extra block is needed). The remaining 3966 blocks will be addressed using the double indirect pointers (using 1 block to keep the pointer block addresses (1024 address block) plus 4 blocks to point the 3966 data blocks.

That is: 5000 blocks + 1 block (single indirect) + 5 blocks (double indirect) = **5006 blocks**

c) What is the maximum possible file size?

The maximum number of blocks of a file is:

- 10 blocks addressed by direct pointers
- 1024 blocks addressed by single indirect pointers
- 1024*1024 blocks addressed by double indirect pointers
- 1024*1024*1024 blocks addressed by triple indirect pointers

The overall number of block is 1G+1M+1K+10 blocks which is approximately **4 TBytes** (not considering 4GB, 4MB and 40KB because are much smaller than the first term).

Exercise 14

Write a program that receives a directory name and prints the names of the containing files and directories, their mode, their owner read right, their type (if they are or not directories), and, for the files modified within the last 10 days, their access time.

Solution

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```
/*This program receives a directory name and prints the names of the containing files
and directories, their mode, their owner read right, their type (if they are or not
directories), and, for the files modified within the last 10 days, their access time.
*/

main () {
int er;
char dirname[100], filename[100], resp[30];
struct stat atr;
DIR *d;
struct dirent *rd1;
time_t access_date;

printf (“Directory name\n");
fgets (dirname, sizeof (dirname), stdin);
/* Delete \n from  the directory name*/
dirname[strlen(dirname)-1]='\0';

access_date=time(&access_date);
if ((d=opendir(dirname))==NULL) {
  printf ("This directory does not exist \n");
  return -1;
}
else {
  while (( rd1 =readdir(d)) != NULL) {
    if ( (strcmp(rd1->d_name, ".")!=0 )&& (strcmp(rd1->d_name, "..")!=0 )){
      strcpy (filename, dirname);
      strcat (filename, "/");
      strcat (filename, rd1->d_name);
      printf ("fichero :%s:", filename);
      er=stat (filename, &atr);
      printf ("modo :%#o:", atr.st_mode);
      if ((atr.st_mode & 0400) != 0)
            printf ("Read access for owner \n");
      else
            printf (" No Read access for owner \n");
      if (S_ISDIR(atr.st_mode)) printf (" is a directory \n");
      if (S_ISREG(atr.st_mode))
        /* files modifies withing the last 10 days */
          if ( (access_date - 10*24*60*60) <  atr.st_mtime) {
        printf ("FILE:%s: access date  %s, in seconds %d\n", rd1->d_name,
                                ctime (&atr.st_mtime), atr.st_mtime );
          }
    }
  } /* while*/
  closedir (d);
}
```

```
}/* main*/
```