# Tipos de datos

struct OPENFILES {

int inuse; // 0 cerrado, 1 abierto

unsigned short inode;

int currpos;

int currbloqueenmemoria;

char buffer[1024];

unsigned short buffindirect[512]; //

};

typedef int VDDIR;

struct vddirent

{

char \*d\_name;

};

struct vddirent \*vdreaddir(VDDIR \*dirdesc);

VDDIR \*vdopendir(char \*path);

# Funciones adicionales para el manejo de bloques en los nodos i

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Lectura y escritura de bloques

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int writeblock(int num\_block,char \*buffer)

{

// Les toca hacerla

// Un bloque es un grupo de sectores lógicos

// Determinar cuál es el primer sector lógico que

// le corresponde a ese número de bloque, para eso

// se necesitan los datos que están en el sector de

// boot.

// La operación de escritura se hace sobre el sector

// lógico calculado y los siguientes sectores lógicos

// que corresponden a ese bloque.

}

int readblock(int num\_block,char \*buffer)

{

// Les toca hacerla

// Un bloque es un grupo de sectores lógicos

// Determinar cuál es el primer sector lógico que

// le corresponde a ese número de bloque, para eso

// se necesitan los datos que están en el sector de

// boot.

// La operación de lectura se hace sobre el sector

// lógico calculado y los siguientes sectores lógicos

// que corresponden a ese bloque.

}

/\* Esta función se utilizará con las funciones seek, write y read, con estas operaciones el nodo i del archivo debe estar cargado en memoria RAM. Por ejemplo si escribimos información en el archivo se le empiezan a asignar bloques, ¿cuáles?, los que encontremos disponibles, y esos bloques disponibles es necesario que los escribamos en el nodo i que está en memoria, y con esta función sabemos cuál es la dirección donde debe ir ese apuntador\*/

unsigned short \*postoptr(int fd,int pos)

{

int currinode;

unsigned short \*currptr;

unsigned short indirect1;

currinode=openfiles[fd].inode;

// Está en los primeros 10 K

if((pos/1024)<10)

// Está entre los 10 apuntadores directos

currptr=&inode[currinode].blocks[pos/1024];

else if((pos/1024)<522)

{

// Si el indirecto está vacío, asígnale un bloque

if(inode[currinode].indirect==0)

{

// El primer bloque disponible

indirect1=nextfreeblock();

assignblock(indirect1); // Asígnalo

inode[currinode].indirect=indirect1;

}

currptr=&openfiles[fd].buffindirect[pos/1024-10];

}

else

return(NULL);

return(currptr);

}

unsigned short \*currpostoptr(int fd)

{

unsigned short \*currptr;

currptr=postoptr(fd,openfiles[fd].currpos);

return(currptr);

}

# Funciones para el manejo de directores (sólo directorio ráiz)

VDDIR dirs[2]={-1,-1};

struct vddirent current;

VDDIR \*vdopendir(char \*path)

{

int i=0;

int result;

if(!secboot\_en\_memoria)

{

result=vdreadseclog(0,(char \*) &secboot);

secboot\_en\_memoria=1;

}

Aquí se debe calcular la variable inicio\_nodos\_i con los datos que están en el sector de boot de la partición

// Determinar si la tabla de nodos i está en memoria

// si no está en memoria, hay que cargarlos

if(!nodos\_i\_en\_memoria)

{

for(i=0;i<secboot.sec\_tabla\_nodos\_i;i++)

result=vdreadseclog(inicio\_nodos\_i+i,&inode[i\*8]);

nodos\_i\_en\_memoria=1;

}

if(strcmp(path,".")!=0)

return(NULL);

i=0;

while(dirs[i]!=-1 && i<2)

i++;

if(i==2)

return(NULL);

dirs[i]=0;

return(&dirs[i]);

}

// Lee la siguiente entrada del directorio abierto

struct vddirent \*vdreaddir(VDDIR \*dirdesc)

{

int i;

int result;

if(!nodos\_i\_en\_memoria)

{

for(i=0;i<secboot.sec\_tabla\_nodos\_i;i++)

result=vdreadseclog(inicio\_nodos\_i+i,&inode[i\*8]);

nodos\_i\_en\_memoria=1;

}

// Mientras no haya nodo i, avanza

while(isinodefree(\*dirdesc) && \*dirdesc<4096)

(\*dirdesc)++;

// Apunta a donde está el nombre en el inodo

current.d\_name=inode[\*dirdesc].name;

(\*dirdesc)++;

if(\*dirdesc>=24)

return(NULL);

return( &current);

}

int vdclosedir(VDDIR \*dirdesc)

{

(\*dirdesc)=-1;

}