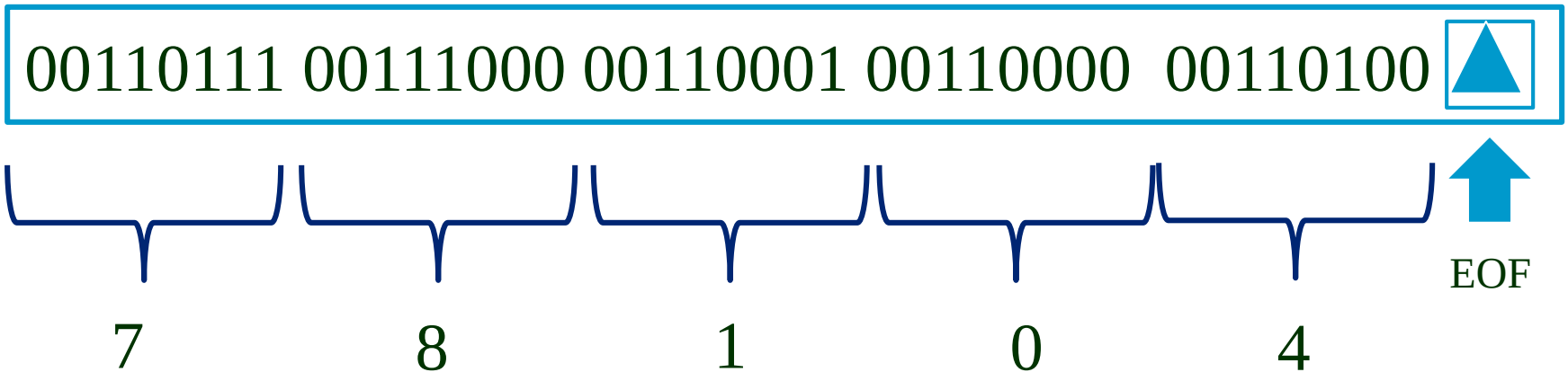
GE6151 COMPUTER PROGRAMMING
LECTURE 10

Prof. Dr. M. Paulraj PhD.,
SRIT
Coimbatore-10

FILES

Text File



Binary File

0 (48), 1 (40), 4(52), 7(55), 8(56)

123	Siti	30
231	Ravi	25
111	Devi	17
112	Abdul	23

File

111	Devi	17

Record

Field

00000001 **Byte**

Bit

Bit: The smallest data item processed by a computer is bit.
Character: 8 bits form a character.
Field: A group of characters forms a field.
Record: A record is composed of many fields.
File: A file is a group of records.

File: File is a collection of related data stored in an auxiliary device.
Files are classified based on the way in which the stored characters are interpreted. Files are classified as text file and binary file.

A text file is a file of characters. To store these data types they must be converted to their equivalent character format.

A binary file is a collection of file stored in the internal format of the computer. (Recall a char is one byte, int is two bytes and float is 4 bytes). The data in a binary file is meaningful only if they are properly interpreted.

When data are written to a **binary file no conversion takes place**. The only problem is file transportability.

C views each file simply as a sequence of bytes. Each file ends with an end of file marker (EOF).

Note: You cannot read binary data type file using text data type file or vice versa.

In a C program, a file is always referenced by a variable name and it must be declared within the program. The variable is a pointer to a special file structure and must be declared as a pointer a file structure. The syntax is:

```
FILE *finp;
```

In the above declaration, 'FILE' is the name of the special data structure used by C for storing information about the file, including whether the file is available for reading or writing.

The pointer name finp is selected by the programmer. The pointer variable name 'finp' will be referenced by the name of the data file.

```
typedef struct {
short level ; /* fill or empty level of the buffer */
short token ; /* Validity check */
short bsize ;  /*  buffer size */
char fd ;
unsigned flags ;
unsigned char hold ;
unsigned char *buffer ;
unsigned char * curp ;
unsigned istemp; }FILE ;
```

Function fopen:

Function fopen makes a connection between an external file and the program. It creates a program file table to store the information needed to process the file.

The syntax of fopen is : fopen("file name","mode");

Mode is a string that represents how we want to access the file.

r – Open the file for reading, if the file exist the marker is positioned at beginning. If the file doesn't exist an error is returned.
w- Open text for writing, If the file exists, it is emptied. If file doesn't exist, it is created.
a – Open text for append. If file exists, the marker is positioned at end. If file doesn't exist, it is created.

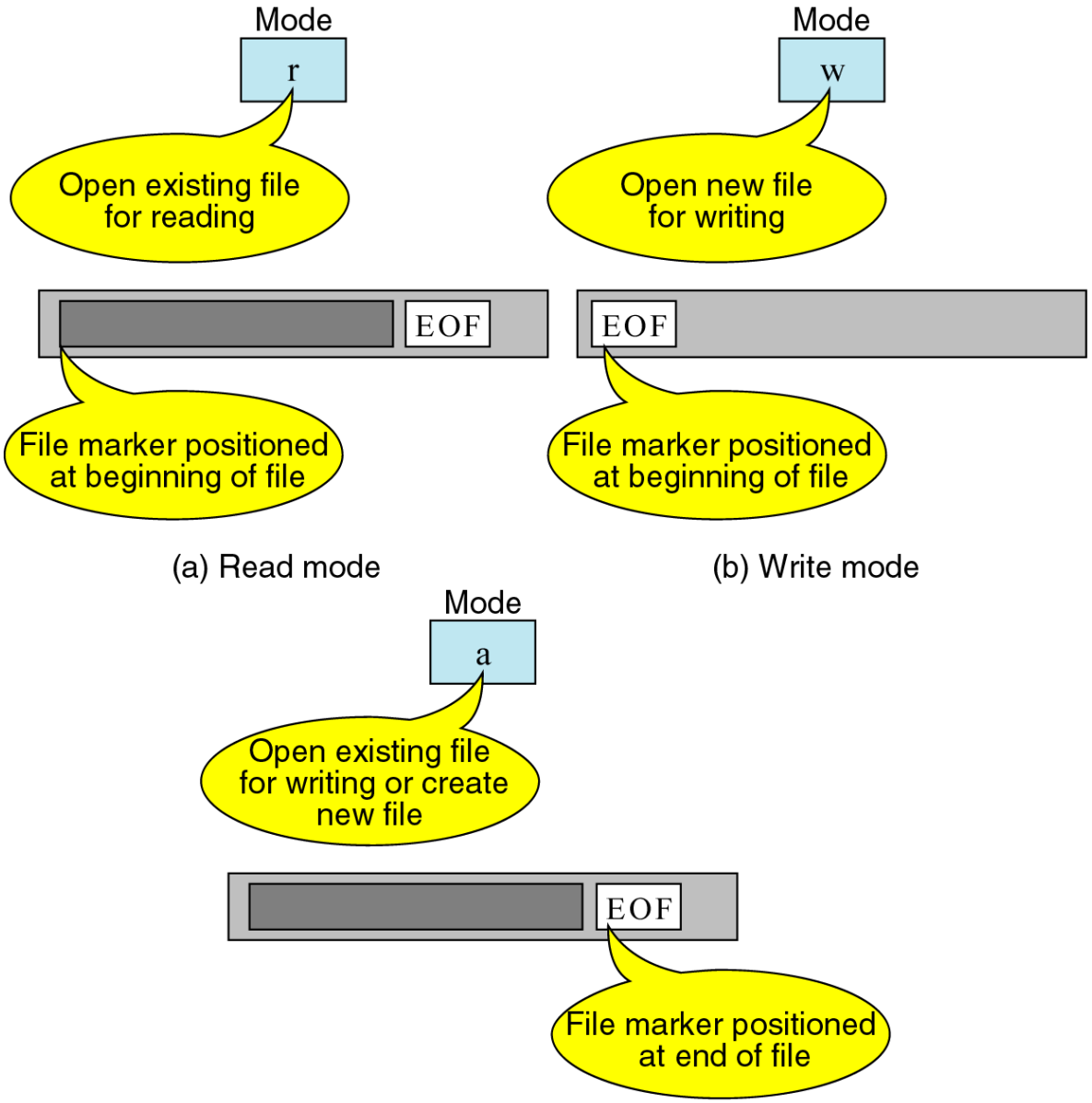
When a file is opened, a file pointer to a FILE structure is returned. If there is any error then it returns a NULL pointer.

Read Mode:
It is used to open an existing file for reading. When the file is opened, the file marker is positioned at the beginning of the file. The file must exist. If it does not exist, NULL is returned as an error.

Write Mode:
The write mode 'w' is designated to open a file for writing. If the file exist, it is emptied and the file marker is placed at the beginning of the file. If the file doesn't exist, a new file is created and the file marker is placed at the beginning of the file. It is an error to read from a file opened in write mode.

Append Mode:
It is designated to open an existing file for writing. The writing starts after the last byte; ie new data are added at the end of the file. If the file doesn't exist, it is created and opened.

The data can be stored in a file either in binary format or text format. **The symbol "b" represents the file is of binary format and "t" represents the file is of text format**.



(a) Read mode

(b) Write mode

(c) Append mode

Examples:

```
FILE  *finp;
finp = fopen("test.dat","rt");

FILE *fout;
fout = fopen("donor.dat","wt");

FILE *fp;
fp = fopen("rain.dat","at");

FILE *fpt;
fpt = fopen("mark.dat","wb");
```

How to check the error status of a opened file?

```
FILE *finp;
finp = fopen("test.dat","r");
if(finp==NULL)  {
printf("Error in opening the file\n");
exit(1); }
```

How to close a file?

A binary file can be closed if it is not required by calling the function fclose.

The proto type of the function fclose is:

```
int  fclose(FILE *fp);
```

Example:

```
fclose(finp);
```

How to write into a text file?

Data can be written into a file using the function ‘fprintf ‘.
The syntax of the function fprintf is:
fprintf(file pointer, format string, address list);

Example:
fprintf(finp, “%4d %f\n”,intnum, floatnum);

The above functional call writes the content of the memory cell intnum and floatnum into the file pointed by the file pointer finp according to the format %4d and %f.

How to Read from a text file?

Data can be read from a file using the function 'fscanf'.
The syntax of the function 'fscanf' is:
fscanf(file pointer, format string, address list);

Example:
fscanf(finp, "%d %f\n",&intnum, &floatnum);

The above functional call writes the content of the memory cell intnum and floatnum into the file pointed by the file pointer finp according to the format %d and %f.

feof() function:
This function is used to check for the end of file. If all the data have been read, the function returns a non zero value. If the end of file have not reached then it will return 0.

The proto type of feof is : int feof(FILE *stream);

int feof(FILE *);

Description:
Used to determine if the end of the file (stream) specified, has been reached.
When a file is being read sequentially one line, or one piece of data at a time (by means of a loop, say), this is the function you check every iteration, to see if the end of the file has come.

Return Values:
Nonzero (**true**) if the end of file has been reached,
zero (**false**) otherwise.

```
#include<stdio.h>
int main(void)
{
FILE *finp, *fout;
int mno;
float mark;
int index;
finp = fopen("test.dat","wt");
if(finp==NULL))
  printf("Error in opennifg file");
for(index=1;index<=10;index++)
{
printf("Enter Matrik Number ");
scanf("%d",&mno);
printf("Enter  Mark ");
scanf("%f",&mark);
fprintf(finp,"%d %f\n",
mno,mark);
}
fclose(finp);
```

```
fout  = fopen("test.dat","rt");
if(fout==NULL)
printf("Error in opening file\n");
while(!feof(fout))
{
fscanf(fout,"%d %f\n",&mno,&mark);
printf("%d\t%.2f\n",mno,mark);
}
fclose(fout)
return 0;
}
```

Binary Files:

File status:

An opened file will be either in read state or in the write state or in the error state. When a file is in the error state, it can not be used to read or write operation. If we want to read the content of a file, then it must be in the read state. If we try to read from a file in the write state an error will occur.

If we want to write into a file, then it must be in the write state. If we try to write into a file, which is in read state then an error, will occur.

When a file is opened in the write state, it can be in two modes namely write mode or append mode.

In write mode, if the file already exists, then writing starts at the beginning of the file and the already written data are lost.

In append mode, if the file already exist, the data are added after the existing data. If the file does not exist, then a new file will be created.

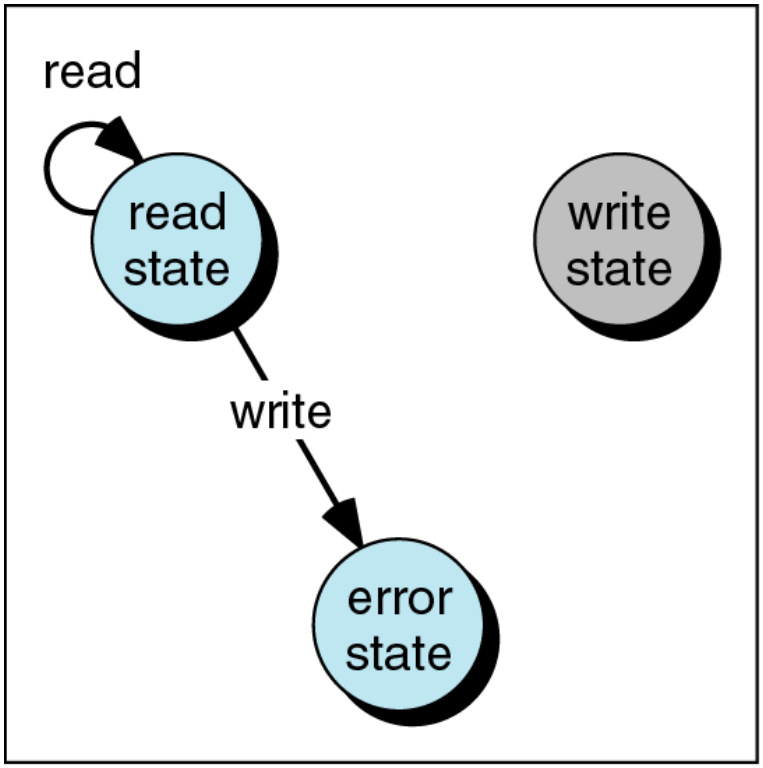
A file can be opened in update mode also. Even when the file is opened in update mode, it will be only at one of the file state either read state or write state.

The table summarizes the mode in which a binary file can be accessed.

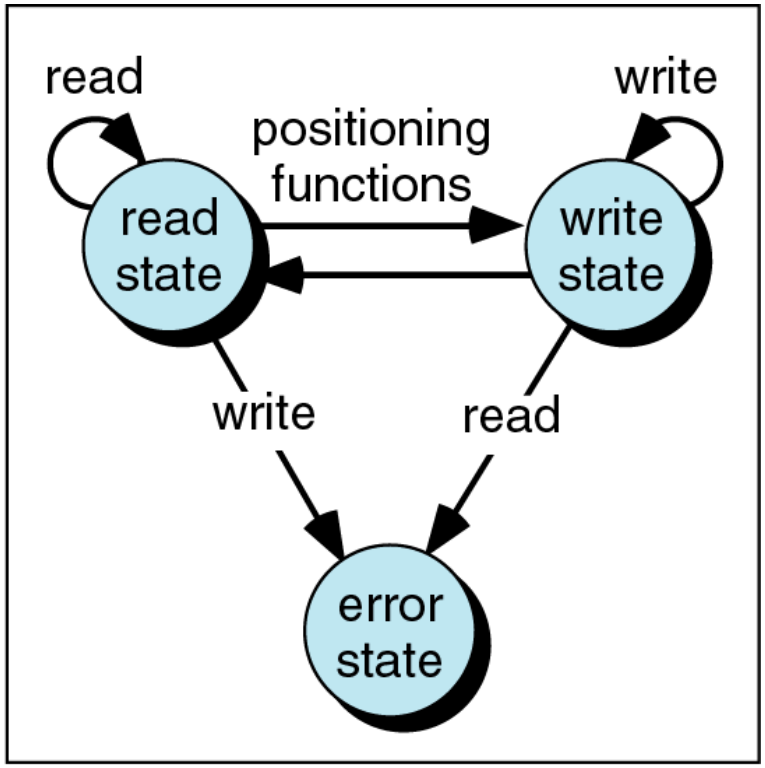
Mode	Meaning
rb	Open file for reading Read starting at beginning If file doesn't exist, error is returned.
wb	Open the file for writing If file exists, it is overwritten If file doesn't exist, It is created.
ab	Open the file for append New records inserted at the end. If file doesn't exist, it is created.

The file mode determines the initial state when a file is opened. To change between the mode, file positioning functions are used. To give the capability of reading and writing a + symbol is added to the file mode.

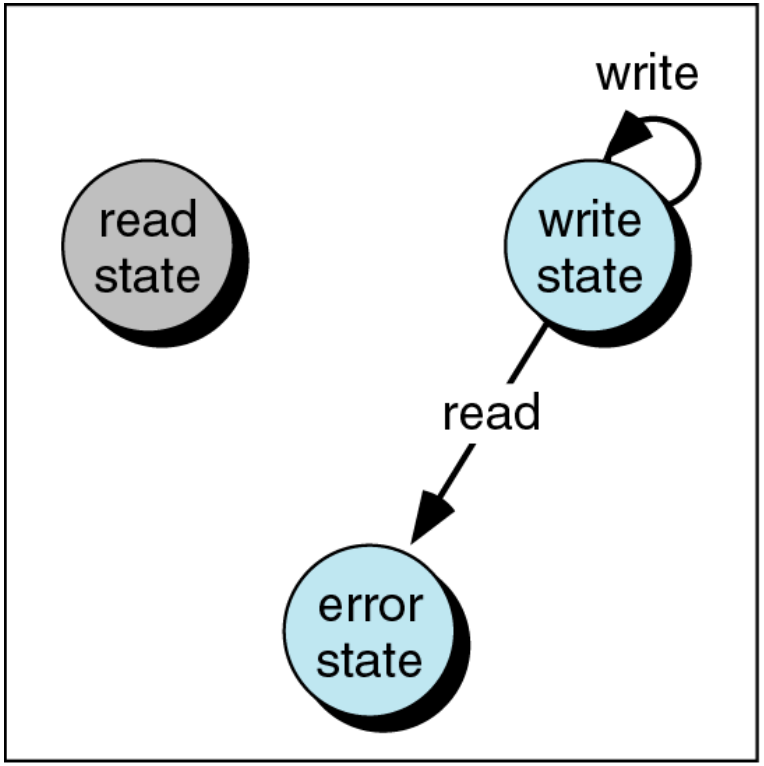
Mode	Meaning
r+b	Open the file for both read and write. Initial state is read. Read starts at the beginning. If the file doesn't exist, error occurs.
w+b	Open file for both read and write. Initial state: write If the file exists, it is emptied If the file doesn't exist, it is created.
a+b	Open the file for read and write Initial state: write Write starts at end. If file doesn't exist, it is created.


read
read state
write state
write
error state

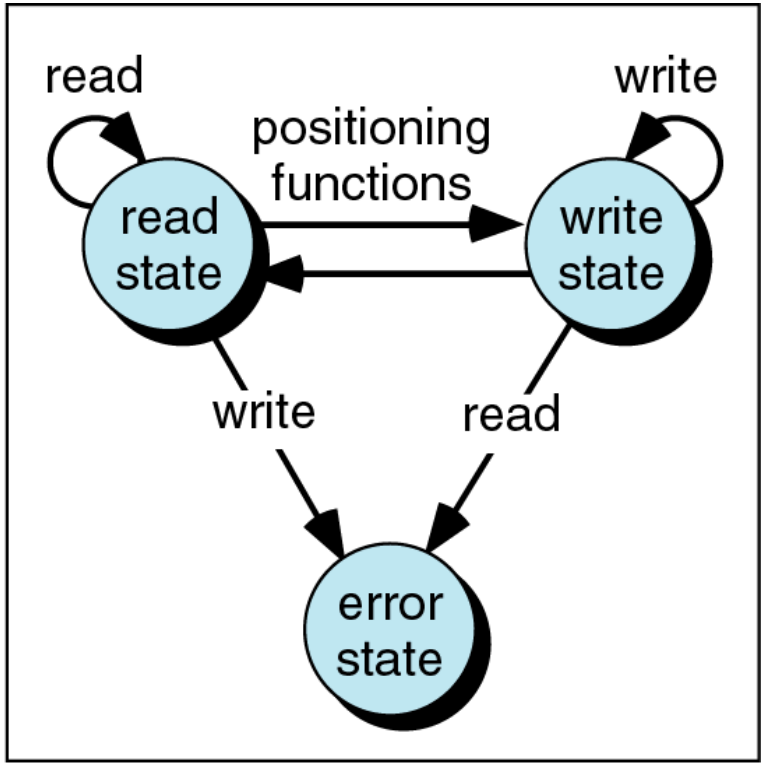
rb


read
write
positioning functions
read state
write state
write
read
error state

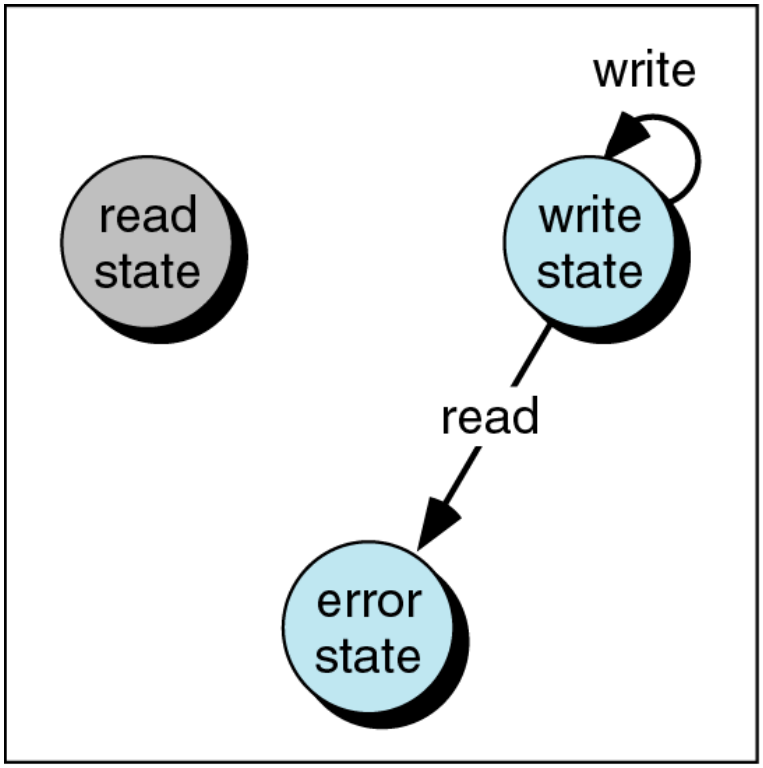
r+b


write
read
state
write
state
read
error
state

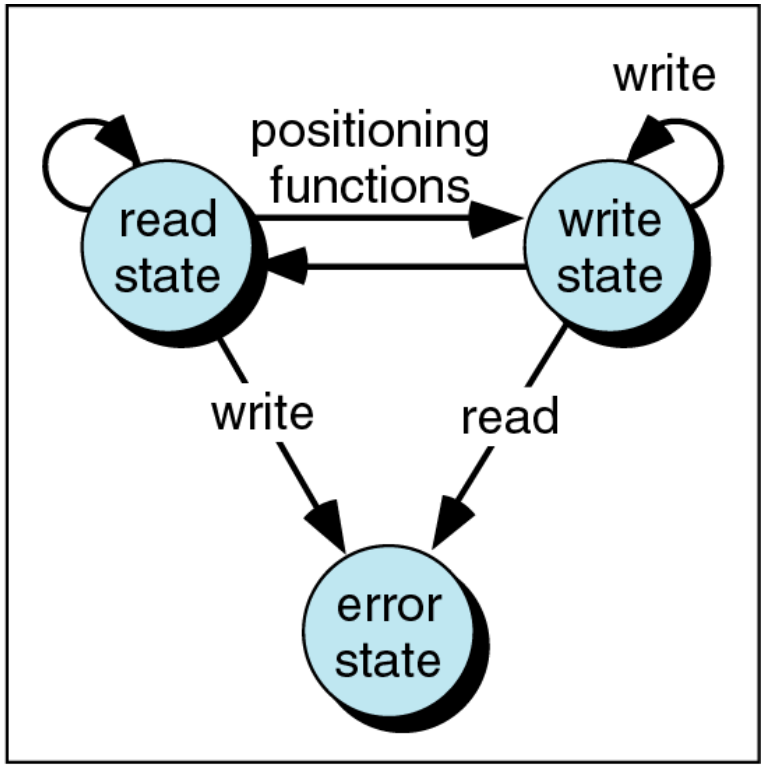
wb


read
write
positioning
functions
read
state
write
state
write
read
error
state

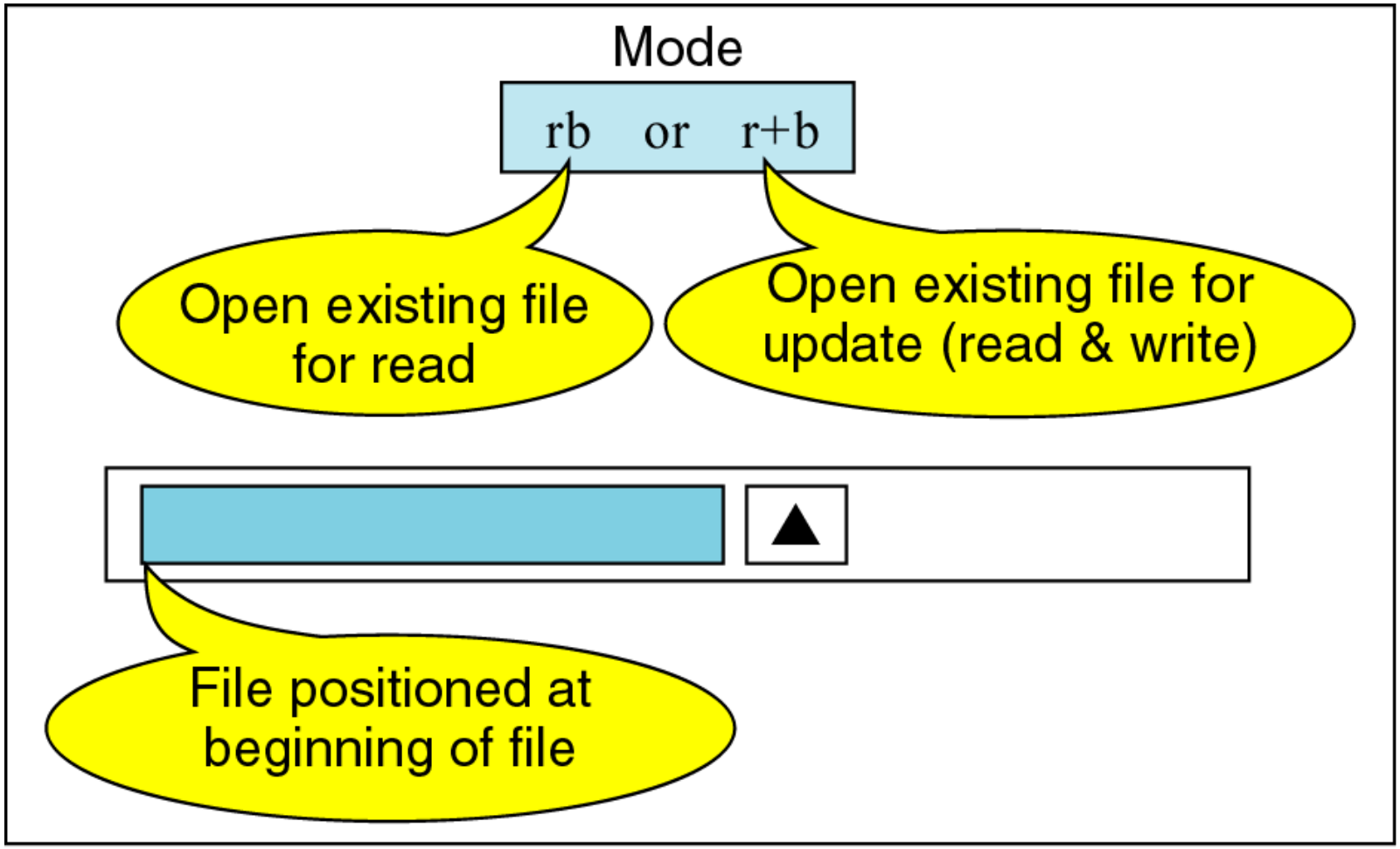
w+b


write
read
state
write
state
read
error
state

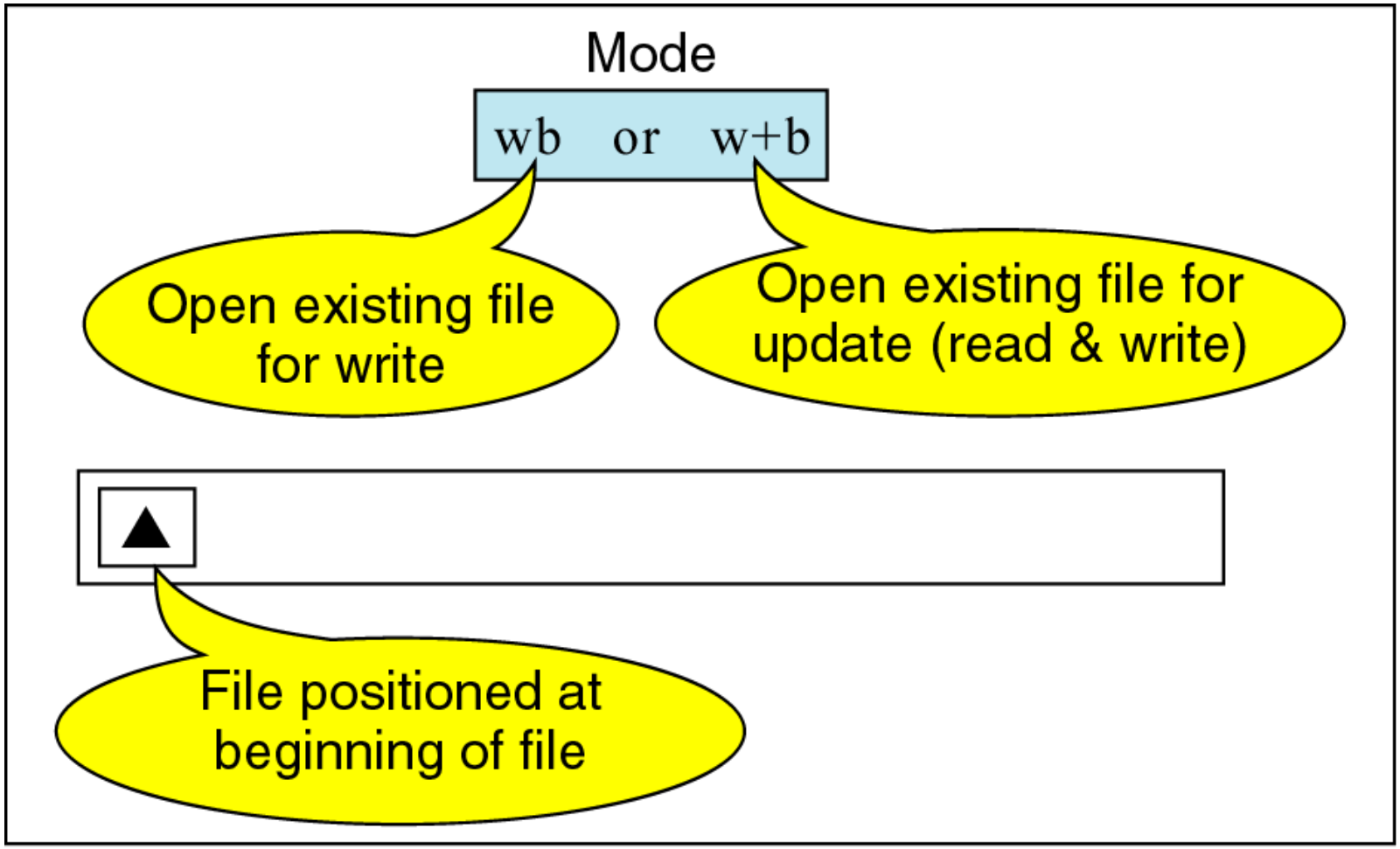
ab


write
positioning
functions
read
state
write
state
write
read
error
state

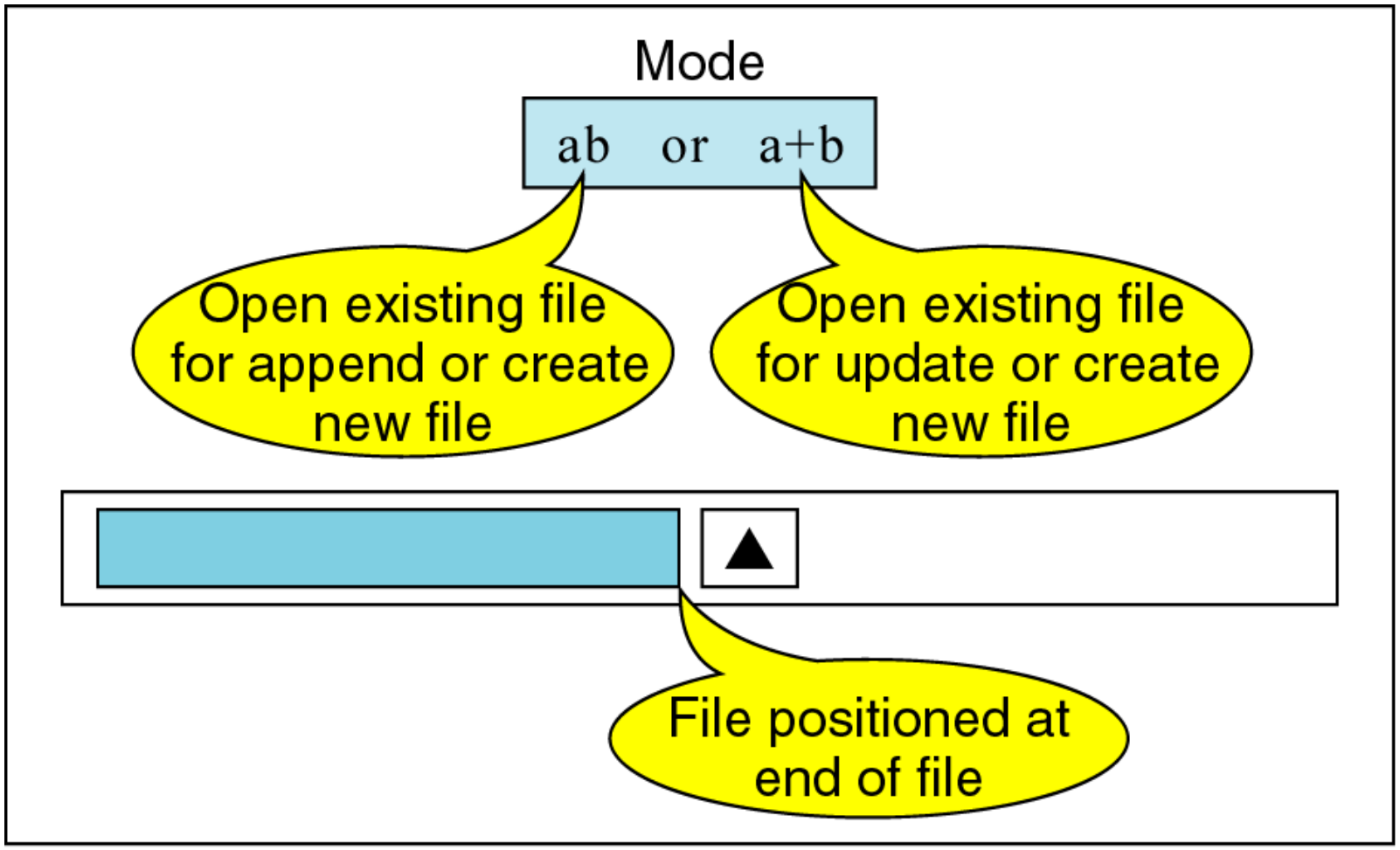
a+b


Mode
rb or r+b
Open existing file for read
Open existing file for update (read & write)
File positioned at beginning of file

(a) Read mode


Mode
wb or w+b
Open existing file for write
Open existing file for update (read & write)
File positioned at beginning of file

(b) Write mode



(c) Append mode

Block read

How the binary formatted data can be transferred to a file from memory or vice versa?

Data found in the memory can be transferred to a file with out any conversion. The data are in hieroglyphical form. Using block input and output functions, the binary data can be read from or written to a file.

The fread function is used to read the **data from a binary file**. The proto type of the function is shown blow:

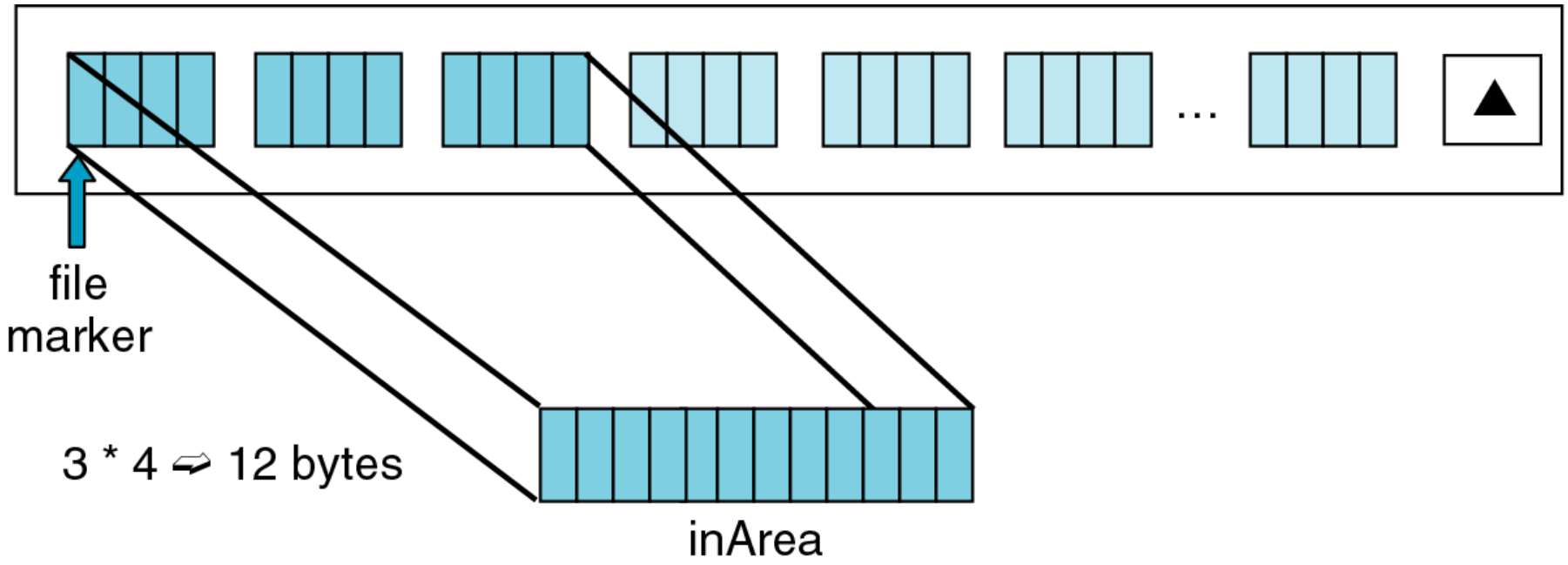
```
int fread( void *ptarea,
                    int element_size,
                    int count,
                    File *fp);
```

ptarea is a pointer to a input area in the memory. The pointer type is void (a generic type).The element_size and count are used to determine the size of data to be transferred. Normally element_size is the sizeof the data type and count specifies the number of data to be transferred.

Example :

```
FILE *finp;
int data;
finp = fopen(“sample.dat”,”rb”);
fread(&data,  sizeof(int), 1,finp);
```

```
FILE *fpData;
int inArea[3];
fread(inArea, sizeof(int), 3, fpData);
```



fread (inArea, sizeof (int), 3, fpData) ;

Example 2:

```
FILE *fp;
int data[5];
fp = fopen(“sample.dat”,”rb”);
fread(data,sizeof(int),5,fp);
```

Note :

The function fread returns the number of items read by it.
fread doesn’t return EOF. It returns the number of elements read.

Example 3:
Assumed:
The file sample contains 5 integer type data.

```
#include<stdio.h>
void main(void)
{
FILE *finp;
int i,count,data[5];
finp = fopen("sample1.dat","rb");
count = fread(data,sizeof(int),5,finp);
printf("The number of data read is
%d\n",count);
for(i=0;i<5;i++)
   printf("%d\n",data[i]);
return;
 }
```

The fwrite function is used to assign the data from the memory to a binary file. The proto type of the function is shown below:

```
int fwrite( void *ptarea,
                    int element_size,
                    int count,
                    File *fp);
```

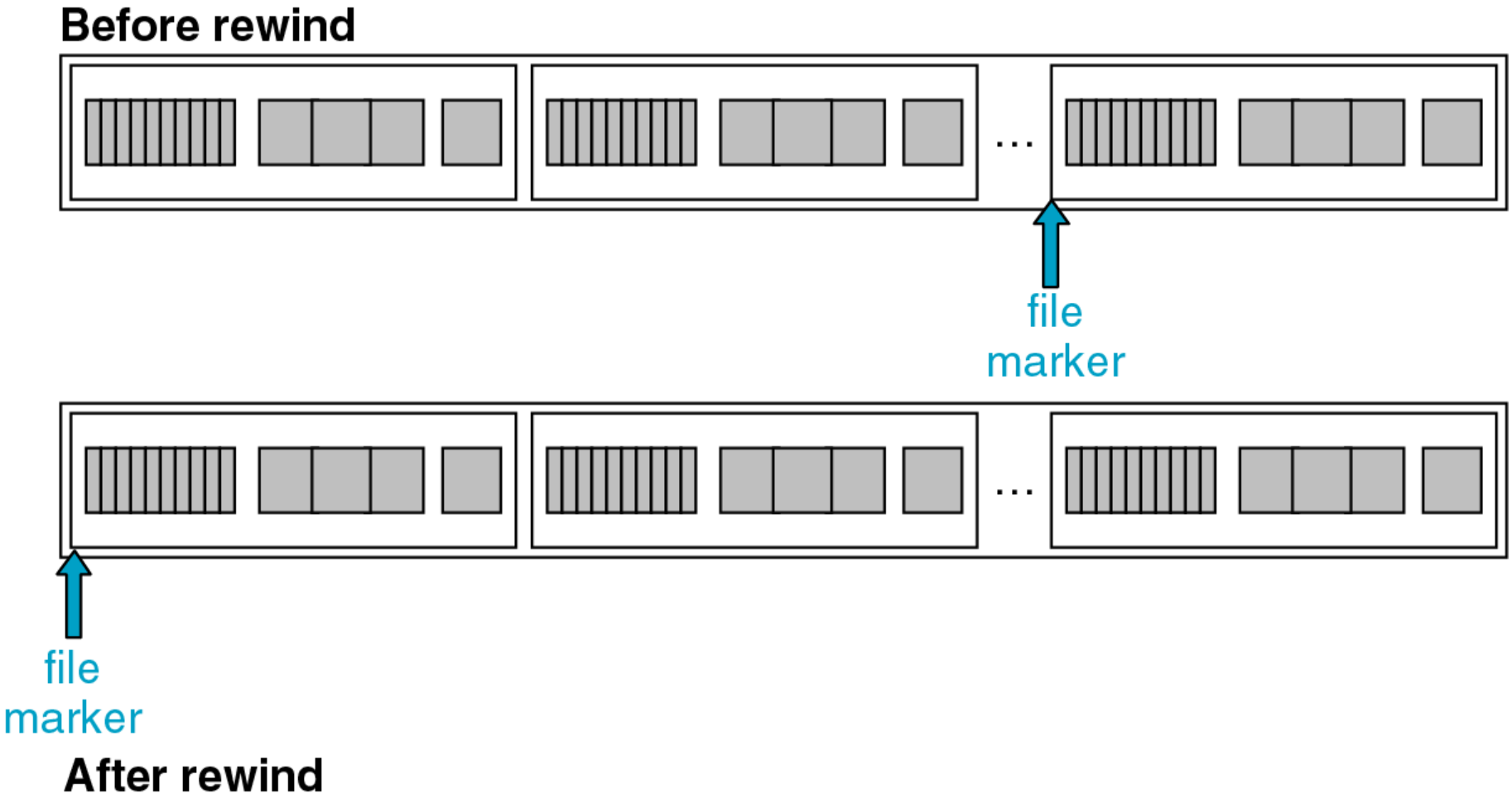
The parameters for file write corresponds to the parameters for the file read function.

Example:

```
FILE *finp;
int  array[] = {1,2,3,4,5,6,7,8,9,10};
fwrite(array, sizeof(int), 10, finp);
```

File positioning Functions: rewind()

Syntax: void return(FILE *finp);



Rewind function is used to set the file marker to the beginning of the File. This function also changes a work file from a write state to read state. The same effect can be accomplished by closing the file and opening it in read mode.

```
rewind(finp);
```

```
#include<stdio.h>
void main(void)
{
        FILE *finp;
        typedef struct
        {
                char number[7];
                float f50;
                float f100;
                float f200;
                float f400;
                float f800;
                float f1600;
        } noise;
        noise nrecord;
        int index, itemsread;
        float max = 0.0;
        char maxcar[8];
```

```
finp = fopen("bdata.dat",“wb");
for(index = 1; index <= 1; index++)
        {
        printf("Enter Car Number ");
        scanf("%s",nrecord.number);
        printf("Enter Noise level for 50 ");
        scanf("%f", &nrecord.f50);
        printf("Enter Noise level for 100 ");
        scanf("%f", &nrecord.f100);
        printf("Enter Noise level for f200 ");
        scanf("%f", &nrecord.f200);
        printf("Enter Noise level for 400 ");
        scanf("%f", &nrecord.f400);
        printf("Enter Noise level for 800 ");
        scanf("%f", &nrecord.f800);
        printf("Enter Noise level for 1600 ");
        scanf("%f", &nrecord.f1600);
        fwrite(&nrecord, sizeof(nrecord), 1, finp);
}
fclose(finp);
```

```
   finp = fopen("bdata.dat","rb");
    while((itemsread = fread(&nrecord, sizeof(nrecord), 1, finp)) !=0)
            {
            if(nrecord.f200 > max)
            {
                    max = nrecord.f200;
                    for(index=0;index<7;index++)
                        maxcar[index] = nrecord.number[index];
                         maxcar[7] = '\0';
            }
            if(nrecord.f100 < 50.0)
            printf("Car noise level less than 50 dB is %s\n",
nrecord.number);
            }
            printf("Car Number with maximum Noise is %s and the noise
level is %f\n", maxcar, max);
            fclose(finp);
            return;   }
```

```
#include<stdio.h>
typedef struct { int day;
        float temp;
        float rainfall;
        } weather;
void main(void) {
FILE *finp, *fout;
weather wrecord;
int index,ndata,tempv;
float minr;
finp = fopen("rain.dat", "wb");
printf("Enter Number of Data ");
scanf("%d", &ndata);
for(index=0;index<ndata;index++) {
printf("Enter Day number ");
scanf("%d", &wrecord.day);
printf("Enter Mean Temperature ");
scanf("%f", &wrecord.temp);
printf("Enter the rainfall ");
scanf("%f", &wrecord.rainfall);
fwrite(&wrecord, sizeof(wrecord), 1, finp);
}
fclose(finp);
fout = fopen("rain.dat","rb");
fread(&wrecord, sizeof(wrecord), 1, fout);
minr = wrecord.rainfall;

while( (tempv = fread(&wrecord,
sizeof(wrecord), 1, fout)) !=0) {
printf("%d %f %f\n", wrecord.day,
wrecord.temp, wrecord.rainfall);
if (wrecord.rainfall < minr) minr =
wrecord.rainfall; }
fclose(fout);
printf("Minimum rain fall = %.2f\n", minr);
return;
}
```