

EKT120: COMPUTER PROGRAMMING

LAB 10: FILES

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.

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 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.

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.

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.

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.

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

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(fnp);
```

fseek() function:

The fseek function positions the file location indicator to a specified byte position in a file.

The general format of fseek() function is:

```
int fseek(FILE *stream, long offset, int wherefrom);
```

The first parameter is a pointer to a file structure. The second parameter is a signed integer constant that specifies the number of bytes the position indicator must move absolutely or relatively.

If wherefrom is set to zero then the offset is made from the beginning of the file.

If the wherefrom is set to one then the displacement is calculated relatively from the current position.

If wherefrom is set to 2 then the file location indicator is set to the end of the file.

```
#define SEEK_SET 0
```

```
#define SEEK_CUR 1
```

```
#define SEEK_END 2
```

If the offset is positive, move forward towards the end of the file.

If the offset is negative, move backward towards the beginning of the file.

Examples:

```
fopen(...)
```

```
fseek(fp,4*sizeof(structure type), SEEK_SET);
```

```
fseek(fp,-4*sizeof(structure_type),SEEK_END);
```

```
fseek(fp,2*sizeof(structure_type),SEEK_CUR);
```

TASK 1

The following program is used to create a text file named task1.dat. Use the program to read the following data into the file task1.dat and observe the output.

```
12345      12.5
23456      22.3
34567      14.5
12111      18.4
14323      21.6
11223      24.5
12113      19.5
```

```
/* Task 1 */
#include<stdio.h>
int main(void)
{
FILE *finp;
int matno;
float tmark;
int index, nodata;

finp = fopen("task1.dat","wt");

printf("Number of Data ");
scanf("%d",&nodata);

for(index=1;index<nodata;index++)
{
printf("Enter Matrik Number ");
scanf("%d",&matno);
printf("Enter Test Mark ");
scanf("%f",&tmark);
fprintf(finp,"%d %f\n",matno,tmark);
}
}
```

```

fclose(finp);

finp = fopen("task1.dat","r");
while(!feof(finp))
{
    fscanf(finp, "%d %f\n",&matno, &tmark);
    printf("%d %.2f\n",matno,tmark);
}
return 0;
}

```

TASK 2

Using a sound level meter, the noise emanated from 30 Proton Saga Model cars at different frequency levels are measured and shown in Table-1.

- Write a program in C to write the data into a binary file named *noise.dat*.
- Write a program that reads the data from the file *noise.dat* and find the car that emanates the maximum noise at 200Hz.
- Also write a program that reads the data from the file and find the cars whose noise level is less than 50 dB at 100 Hz.

Table-1 Car Noise level

| Car Num | 50Hz | 100Hz | 200Hz | 400Hz | 800Hz | 1600Hz |
|---------|------|-------|-------|-------|-------|--------|
| SAA1513 | 54.9 | 58.6 | 59.9 | 61.0 | 64.0 | 65.6 |
| BEF1526 | 57.1 | 60.4 | 72.1 | 70.0 | 66.7 | 65.6 |
| PBA1845 | 51.2 | 52.8 | 61.0 | 61.9 | 62.8 | 64.8 |
| KFC8421 | 54.3 | 58.0 | 63.7 | 60.0 | 64.0 | 65.4 |
| PHD3934 | 48.3 | 51.2 | 55.5 | 58.0 | 57.1 | 54.7 |
| PCA6685 | 43.9 | 47.4 | 49.6 | 47.9 | 55.8 | 57.5 |
| WHY4153 | 49.9 | 60.6 | 64.1 | 63.9 | 65.8 | 70.2 |
| RF1001 | 44.9 | 53.8 | 58.7 | 56.5 | 65.4 | 62.2 |
| RG1500 | 48.0 | 53.2 | 62.5 | 61.7 | 62.2 | 68.9 |
| KNM3583 | 55.0 | 54.1 | 69.0 | 65.2 | 63.9 | 64.0 |
| PTJ9144 | 50.6 | 54.4 | 58.4 | 57.1 | 60.8 | 62.2 |
| SAB7692 | 44.9 | 51.6 | 53.4 | 54.7 | 62.6 | 63.1 |
| HOW1670 | 50.0 | 62.6 | 62.6 | 61.6 | 61.3 | 63.2 |
| RF2210 | 50.3 | 51.0 | 59.6 | 58.3 | 65.6 | 65.8 |
| SAR9645 | 54.3 | 49.9 | 56.2 | 58.5 | 59.6 | 65.6 |
| SA3298 | 49.1 | 46.5 | 52.2 | 55.7 | 58.7 | 64.0 |
| PFU4012 | 51.3 | 55.6 | 58.8 | 59.3 | 59.8 | 62.7 |
| ABT5750 | 48.5 | 50.6 | 53.7 | 55.0 | 58.1 | 59.1 |
| KAT1670 | 50.0 | 62.6 | 62.6 | 61.6 | 61.3 | 63.2 |
| SAA6220 | 56.5 | 53.1 | 66.6 | 68.1 | 66.9 | 67.2 |

| | | | | | | |
|---------|------|------|------|------|------|------|
| BAT3513 | 62.5 | 57.4 | 71.0 | 61.4 | 69.2 | 69.9 |
| ABT1429 | 49.1 | 51.3 | 62.2 | 60.3 | 60.7 | 64.5 |
| KAS7440 | 51.6 | 54.8 | 58.9 | 58.7 | 64.1 | 62.4 |
| PCE5949 | 46.0 | 50.4 | 55.8 | 56.0 | 58.3 | 60.5 |
| PHD9410 | 52.7 | 53.5 | 59.2 | 61.5 | 60.8 | 59.7 |
| RE2691 | 52.2 | 53.7 | 56.3 | 59.0 | 60.8 | 62.4 |
| KCE4530 | 54.2 | 54.9 | 59.4 | 60.0 | 62.7 | 64.2 |
| PEG8771 | 48.2 | 52.7 | 68.4 | 70.1 | 64.4 | 64.9 |
| SA1272 | 54.9 | 57.9 | 55.5 | 58.6 | 57.9 | 64.7 |

TASK 3

(a) Write a text file named test.dat to store the following table.

| Matrik No | Test1 | Test2 | Test3 |
|-----------|-------|-------|-------|
| 1 | 12.3 | 12.6 | 12.9 |
| 2 | 13.2 | 12.5 | 13.7 |
| 3 | 11.9 | 13.9 | 14.1 |
| 4 | 12.8 | 12.9 | 13.0 |
| 5 | 11.9 | 13.2 | 9.3 |
| 6 | 12.7 | 14.8 | 12.8 |

(b) Write a program in C to read the data from the file test.dat and find the average mark obtained by each student. Print your result in the following format.

| Matrik No | Test1 | Test2 | Test3 | Average Mark |
|-----------|-------|-------|-------|--------------|
|-----------|-------|-------|-------|--------------|

TASK 4

Write a program in C that can be used to create and store student's mailing address. Your program must able to display the mailing address of a student when a user enters the student's matrik number.