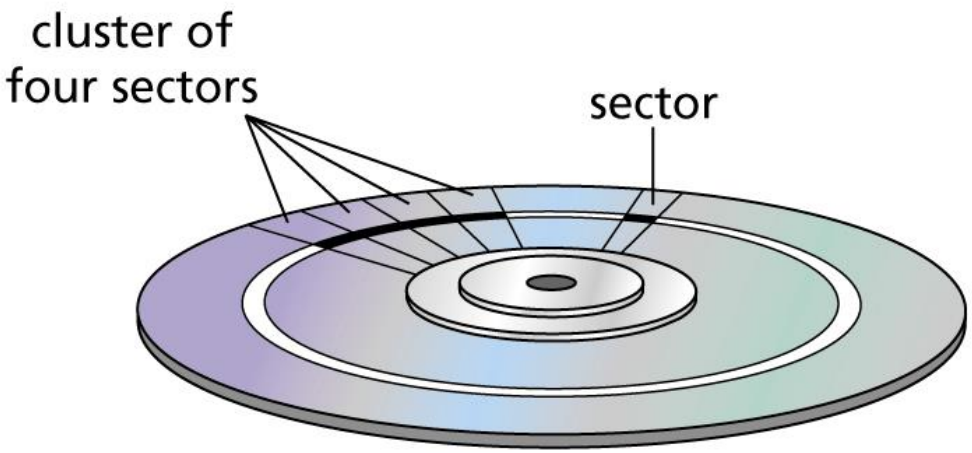
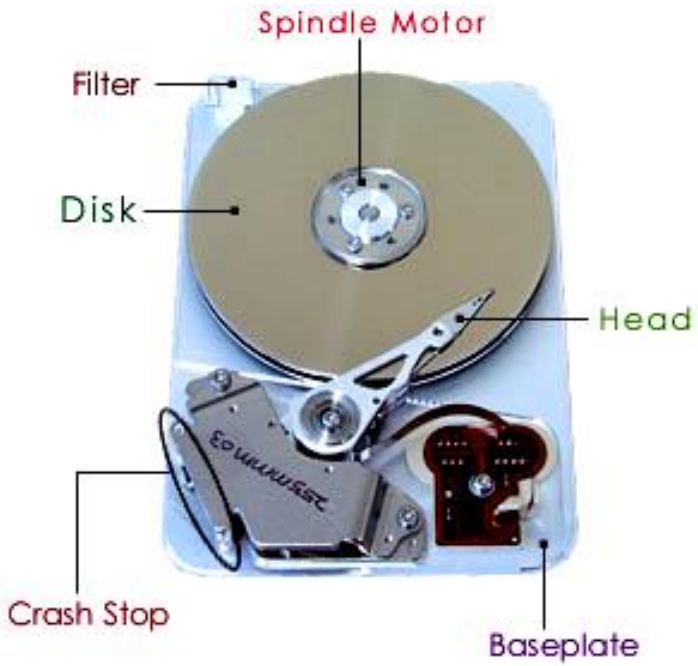
Text Files

Module H: Files

Objectives

- What is a file?
- How are data stored in files?
- How to access data in a text file?





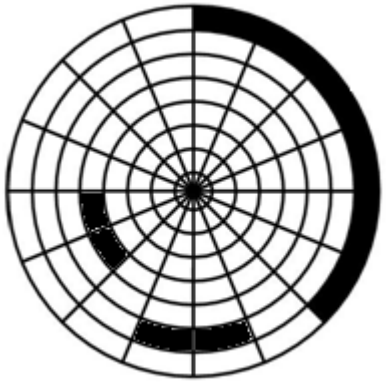
Contents

1- What is a file?

- A complete, named collection of information, such as a program, a set of data used by a program, or a user-created document. A file is the basic unit of storage that enables a computer to distinguish one set of information from another. A file is the “glue” that binds a conglomeration of instructions, numbers, words, or images into a coherent unit that a user can retrieve, change, delete, save, or send to an output device (from MS Computer Dictionary)

What is a file?...

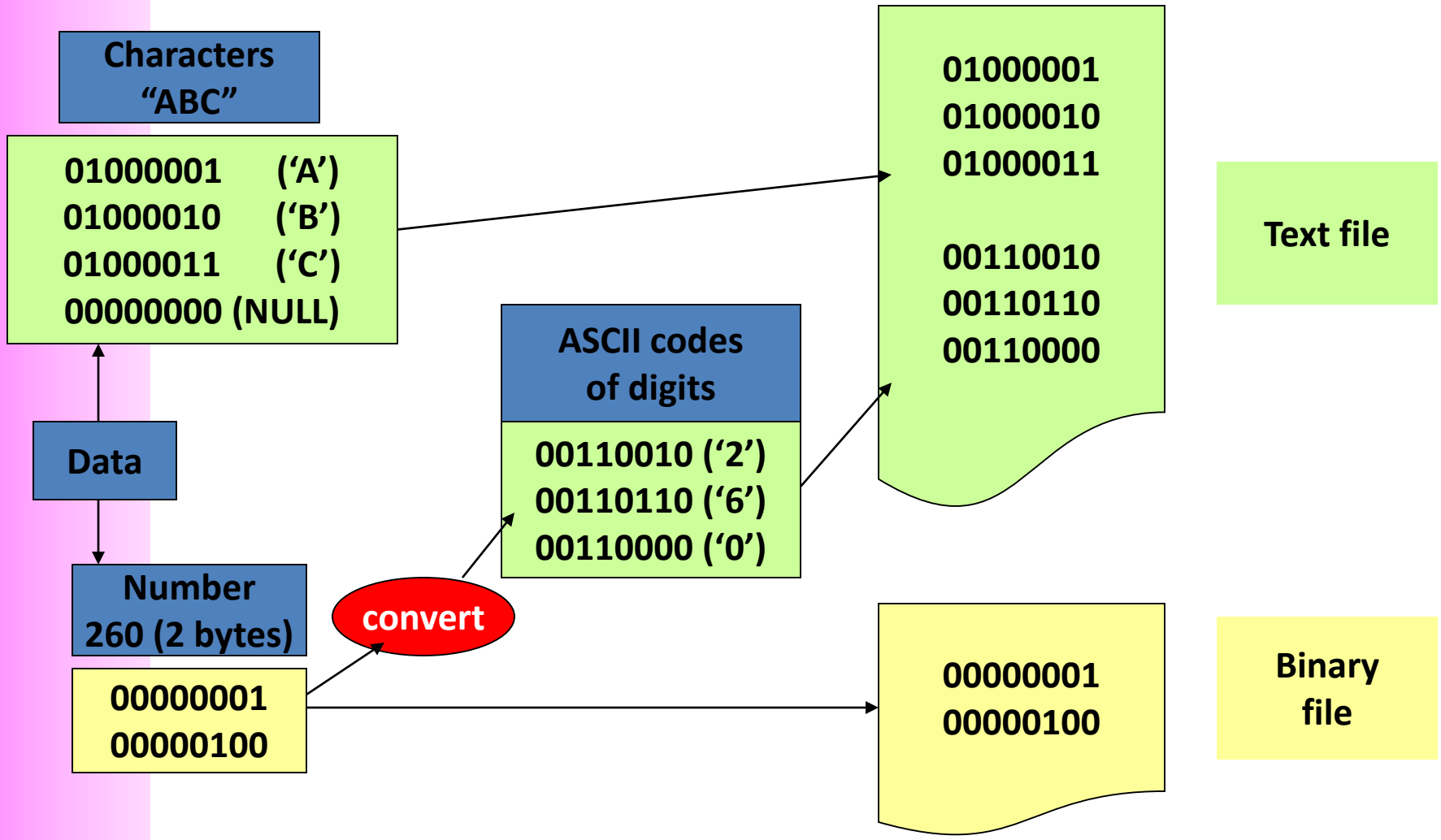
- A file is not necessarily stored contiguously on a secondary storage device .
- Disk → Track → Sector
- Some sectors → Cluster
- Unit of disk allocation: Cluster
- This cluster contains data to point to the next one.
- The contents of a file is accessible after we have turned the power off and back on at a later time.



2- File Types

- The fundamental unit of a file is a byte.
- A file is a stream of bytes.
- A file concludes with a special mark called the end of file mark (EOF).
- Based on the way used to store data:
  - Text file: Unit of data in a file is an ASCII code of character.
  - Binary file: Unit of data in a binary byte → Each byte on the file is a direct image of the corresponding byte in memory

File Types…



Text format is more portable than binary format
But binary format is more efficient than text format

3- Ways for Accessing Files

- Typically, a file consists of records that we can access in either of two ways
  - randomly (like CD's, hard disks) or
  - sequentially (like Cassette Tapes).

sequential file access

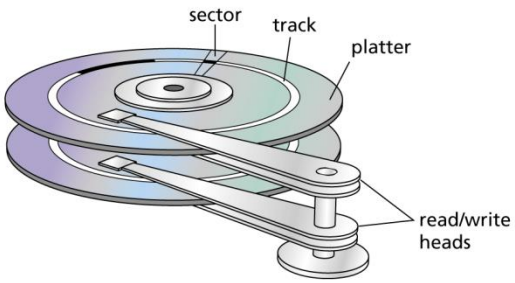
SR1	SR2	SR3	SR4	SR5	SR6	SR7	SR8	SR9	SR10

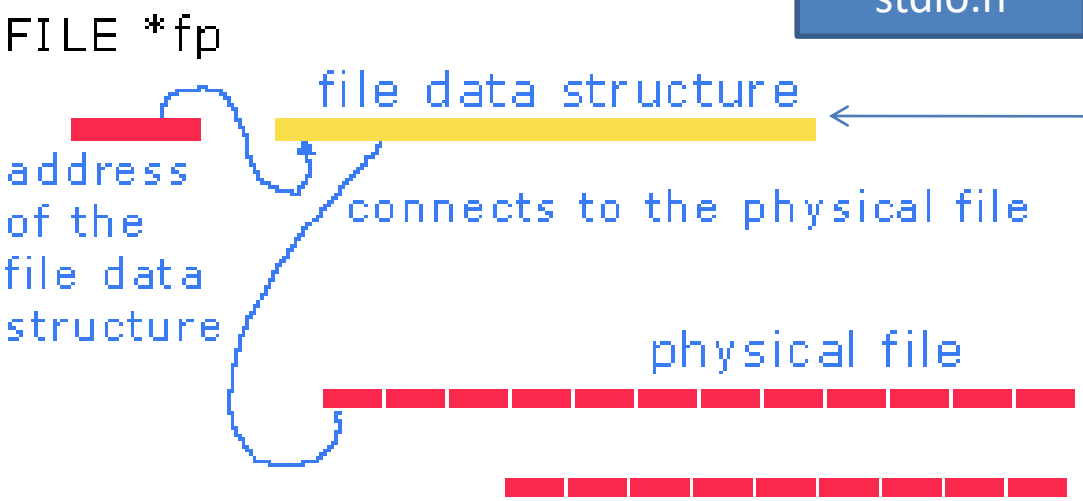
random file access

RR1	RR2	RR3	RR4	RR5

4- Connecting to a File

- Os manages hardware → Connecting a file in a program should be announced to the OS.
- OS can manage some files concurrently. At a time, only one file can be accessed → Information about the opened file must be maintained for next read or write.





stdio.h

```
typedef struct   _iobuf
{
    char*   _ptr;
    int     _cnt;
    char*   _base;
    int     _flag;
    int     _file;
    int     _charbuf;
    int     _bufsiz;
    char*   _tmpfname;
} FILE;
```

5- Declaration

- FILE* identifier ;
- Example

```
#include <stdio.h>
FILE* f=NULL;
```

- The variable f will be updated when a specific file is opened.
- f points to a memory block having the pre-defined structure

```
typedef struct _iobuf
{
   char*   _ptr;
   int     _cnt;
   char*   _base;
   int     _flag;
   int     _file;
   int     _charbuf;
   int     _bufsiz;
   char*   _tmpfname;
} FILE;
```

6- Steps for Accessing a File

Reading file to variables

1) Select file by filename

// function
2) Open the file.
3) Determine the position in the file will be read.
4) Loop
   Read file contents to variables.
   Process variables.
5) Close file

Writing variables to file

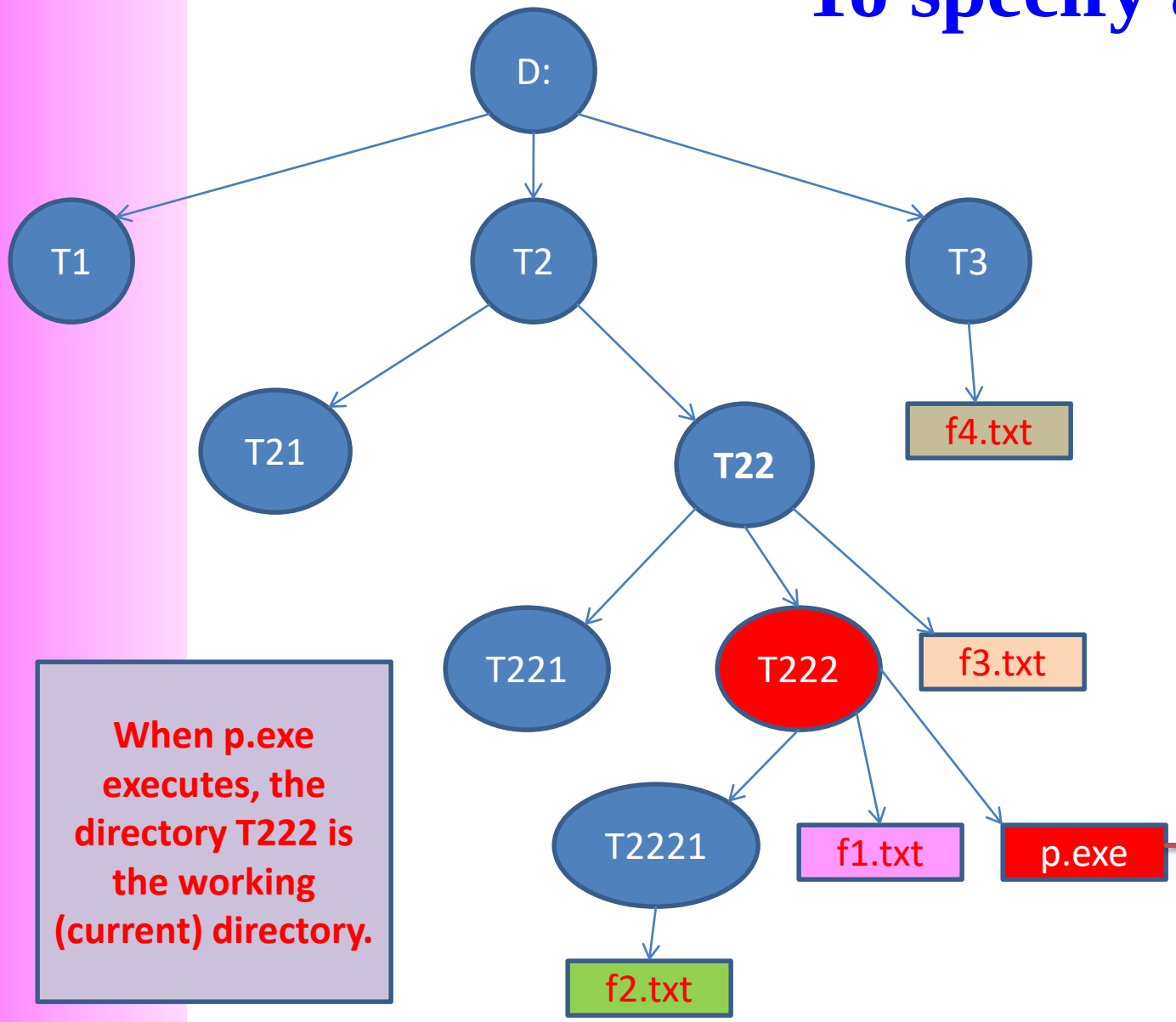
1) Select file by filename

// function
2) Open the file
3) Determine position in the file will be written.
4) Loop
   - Set data to variables (if needed)
   - Write data of variables to file
5) Close file

Generally, we used to process a file from the begin of the file.

Specify a filename (a string):
Absolute pathname:
"c:/t1/t11/f1.txt" or
"c:\\t1\\t11\\f1.txt"
File in the current folder: "f1.txt"

To specify a filename



When p.exe executes, the directory T222 is the working (current) directory.

"f1.txt"
"./f1.txt"
".\\f1.txt"
"T2221/f2.txt"
"./T2221/f2.txt"
"../f3.txt"
"D:/T3/f4.txt"

7- Some Common File Functions

Purpose STDIO.H	Syntax
Open a file	**FILE* fopen(char fname[], char mode[])**
Close a opening file	**int fclose(FILE*)**
Read a character	**int fgetc(FILE*)**
Write a character	**int fputc(char, FILE*) → EOF (-1)**
Read a string	**fgets(char S[], int nbytes, FILE* f); → NULL if EOF**
Write a string	**fputs (char*, FILE*)**
Read a number	**fscanf (FILE*, char* format, PointerList)**
Write a number	**fprintf (FILE*, char* format, VarList)**
Test whether the file is EOF?	**int feof(FILE*)**
Rewind to the beginning	**void rewind (FILE*)**
Get the current file position	**long ftell(FILE*)**
Move the current position	**int fseek (FILE*, long offset, int fromPos)**
Rename a closed file	**rename (char fName[], char newName[])**
Remove a closed file	**remove (char fName[])**

7.1- The fopen() function

FILE *fopen(char file_name[], char mode[]);

- file_name parameter is a null-byte terminated string containing the name of the file.
- mode parameter is a null-byte terminated string containing the connection mode
  - **"r"** - read from the file,
  - **"w"** - write to the file: if the file exists, truncate its contents and then write; if the file does not exist, create a new file and then write to that file,
  - **"a"** - write to the end of the file: if the file exists, append to the end of the file; if the file does not exist, create it and then write.

The fopen() function…

- The other connection modes for text files are
  - "r+" - opens the file for reading and possibly writing,
  - "w+" - opens the file for writing and possibly reading; if the file exists, truncates its contents and then writes to the file; if the file does not exist, creates a new file and then writes to that file,
  - "a+" - opens the file for writing to the end of the file and possibly reading; if the file exists, appends to the end of the file; if the file does not exist, creates it and then writes to the file.
  - *Modes for binary files: “rb”, “wb”, “r+b”, “w+b”,”a+b”*
- fopen returns ***NULL*** if the attempt to connect to the file fails.

7.2- The fclose() function

int fclose(FILE *); 0: successful, EOF (-1): fail

- File opened writing, **fclose** writes any data remaining in the file stream's buffer to the file and concludes by appending an end of file mark immediately after the last character.
- File opened reading, **fclose** ignores any data left in the file stream's buffer and closes the connection.
- **fclose** can fail if the secondary storage medium is full, an I/O error occurs or the medium has been prematurely removed.

7.3- The fgetc(), fputc() Functions

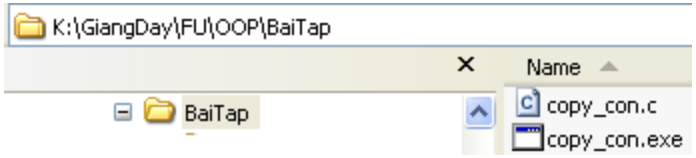
Return	Function	Parameter
The next byte read (ASCII code) or EOF(-1) End of File	int **fgetc(**FILE* fp)	Pointer of the file opened
The character written or EOF	int **fputc**(int ch, FILE* fp)	Character will be written to the file

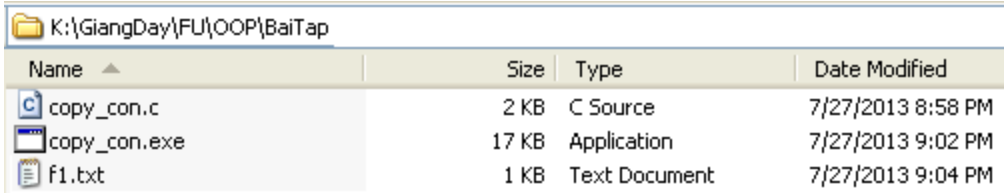
Demonstration 1

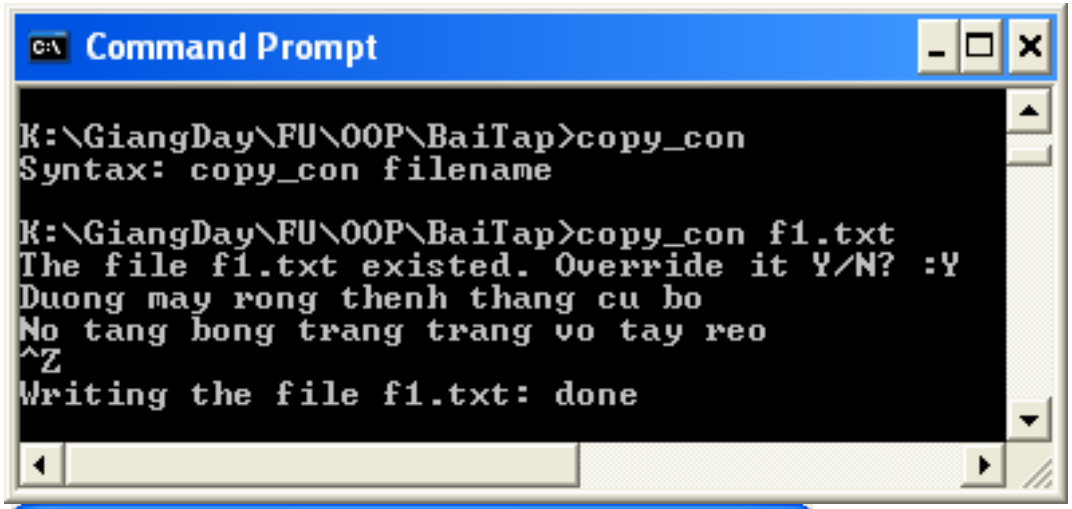
Write a C-program that will use command line to perform writing a text file from characters inputted by user until the keys Ctrl+Z then ENTER are pressed.

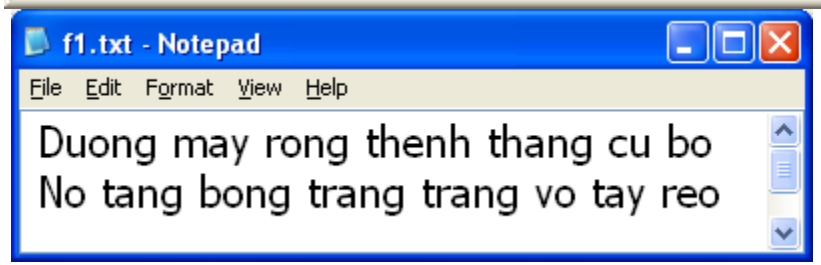
Syntax of the program:

copy_con filename









Demonstration 1…

```c
/*copy-con.c */
#include <stdio.h>
#include <ctype.h>
#define TRUE 1
#define FALSE 0
/* Check whether the file existed or not */
int exist (char* filename)
{   int existed = FALSE;
    /* Try opening it for reading */
    FILE* f= fopen(filename,"r");
    if (f!=NULL)
    {   existed=TRUE;
        fclose(f);
    }
    return existed;
}
```

Demonstration 1…

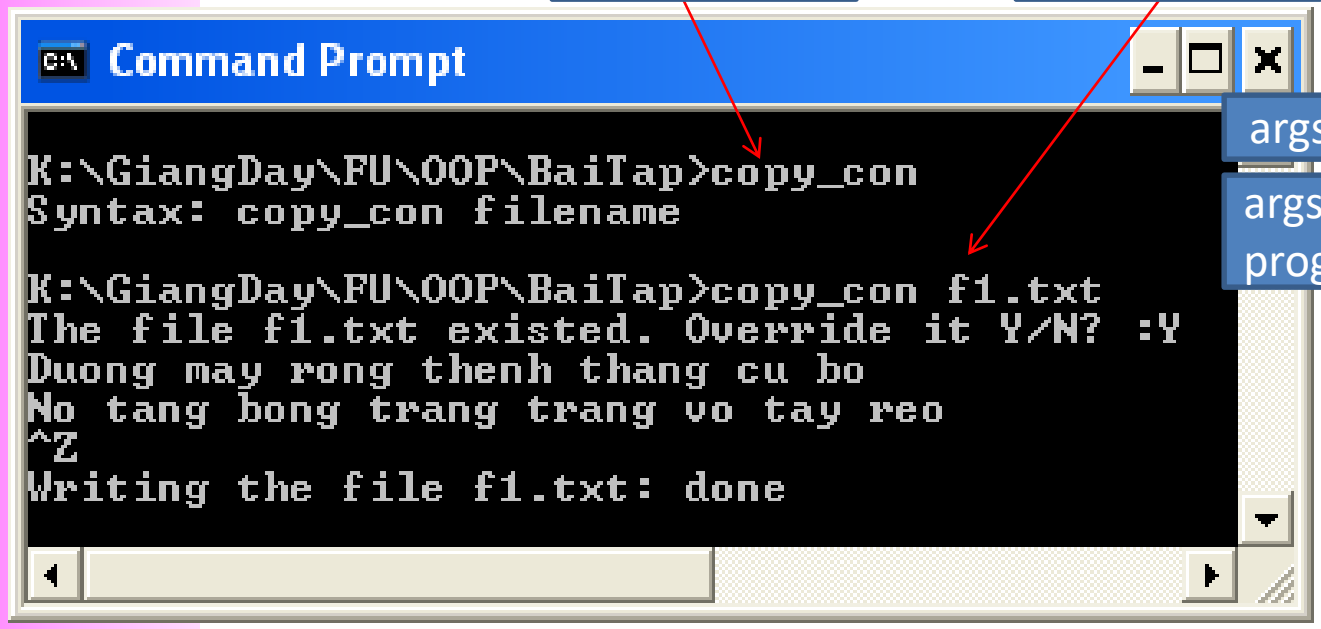
```c
/* Write a file from character inputted until Ctrl+Z is pressed */
int writeFile( char* filename)
{   char c; /* inputted character */
    int CTRL_Z = -1;
    if (exist (filename)==TRUE)
    {   printf("The file %s existed. Override it Y/N? :", filename);
        if (toupper(getchar())=='N') return FALSE;
    }
    /* open the file for writing */
    FILE* f= fopen(filename,"w");
    fflush(stdin); /* Clear input buffer */
    do
    {   c=getchar(); /* get a character */
        if (c!=CTRL_Z) fputc(c,f); /* Write it to file */
    }
    while (c!=CTRL_Z);
    fclose(f);
    return TRUE;
}
```

Demonstration 1…

```
36 int main(int argCount, char* args[])
37 {   if (argCount!=2)printf ("Syntax: copy_con filename\n");
38     else if (writeFile(args[1])== TRUE)
39                printf("Writing the file %s: done\n", args[1]);
40          else printf("Can not write the file %s\n", args[1]);
41     return 0;
42 }
```

argCount=1

argCount=2



args[0]: name of the program

args[1]: parameter of the program

Demonstration 1…

Function for printing the content of a text file

```
int printFile( char* filename)
{  char c; /* inputted character */
   if (exist (filename)==FALSE)
   {  printf("The file %s does not exist.\n", filename);
      return FALSE;
   }
   /* open the file for reading */
   FILE* f= fopen(filename,"r");
   /* When data can be read from the file to variable, process variable */
   while ((c=fgetc(f))!=EOF) putchar(c);
   fclose(f);
   return TRUE;
}
```

7.4- The fgets(), fputs() Functions

Return	Function	Parameter
Success: str Fail: NULL	char* **fgets(char* str, int num,** FILE* fp)	num: Maximum number of characters will be read
Success: >=0 Fail: EOF (-1)	int **fputs**(char* str, FILE* fp)	str: string is written will be written to the file

The **fputs()** function: The null that terminates str is not written and it does not automatically append a carriage return/linefeed sequence.

The **fgets()** function reads characters from the file associated with fp into a string pointed to by str until num-1 characters have been read, a newline character is encountered, or the end of the file is reached. The string is null-terminated and the newline character is retained. The function returns str if successful and a null pointer if an error occurs.

Demonstration 2

Write a C-program that will performs the following operations:

- User enters a filename of a text file
- User will enter data to the file line-by-line until a null string is inputted.
- User will see the content of the file.

Demonstration 2…

K:\GiangDay\FU\OOP\BaiTap\string_file_demo.exe

```
Program for writing then reading a file
Enter a filename:f5.txt
Write file:
The file f5.txt existed. Override it Y/N? :Y
Dinh Tien Hoang
Le Dai Hanh
Ly Cong Uan
Tran Hung Dao
Le Loi
Nguyen Hoang
Nguyen Hue

Data in the file f5.txt:
Dinh Tien Hoang
Le Dai Hanh
Ly Cong Uan
Tran Hung Dao
Le Loi
Nguyen Hoang
Nguyen Hue
```

f5.txt - Notepad

File Edit Format View Help

```
Dinh Tien Hoang
Le Dai Hanh
Ly Cong Uan
Tran Hung Dao
Le Loi
Nguyen Hoang
Nguyen Hue
```

```c
/* string_file_demo.c */
#include <stdio.h>
#include <string.h>
int exist (char* filename)
{   int existed = 0;
    /* Try opening it for reading */
    FILE* f= fopen(filename,"r");
    if (f!=NULL)
    {   existed=1;
        fclose(f);
    }
    return existed;
}
```

Demonstration 2…

```c
/* Write a file from lines inputted until the 0-line entered */
int writeFile( char* filename)
{   if (exist (filename)==1)
    {   printf("The file %s existed. Override it Y/N? :", filename);
        if (toupper(getchar())=='N') return 0;
    }
    char line[201]; /* inputted string */
    int length=0; /* length if inputted line */
    /* open the file for writing */
    FILE* f= fopen(filename,"w");
    fflush(stdin); /* Clear input buffer */
    do
    {   gets(line); /* get a line - If user presses ENTER only --> length=0*/
        length = strlen(line);
        if (length>0)
          { fputs(line,f); /* Write the line to file */
            /* fputs() does not write the new line mark automatically */
            fputs("\n", f);
          }
    }
    while (length>0);
    fclose(f);
    return 1;
}
```

Demonstration 2…

```c
/* Print out the content of a text file to the monitor */
int printFile( char* filename)
{   char c; /* inputted character */
    if (exist (filename)==0)
    {   printf("The file %s does not exist.\n", filename);
        return 0;
    }
    /* open the file for reading */
    FILE* f= fopen(filename,"r");
    /* When data can be read from the file to variable, process variable */
    while ((c=fgetc(f))!=EOF) putchar(c);
    fclose(f);
    return 1;
}
int main(int argCount, char* args[])
{   char filename[81];
    printf("Program for writing then reading a file\n");
    printf("Enter a filename:");
    gets(filename);
    printf("Write file:\n");
    if (writeFile(filename)==1)
    { printf("Data in the file %s:\n", filename);
      if (printFile(filename)==0)printf("File error!\n");
    }
    else printf("Writing file fail!\n", filename);
    getchar();
    return 0;
}
```

7.5- The fscanf(), fprintf() Functions

Return	Function	Parameter
- Success: Number of data items which are read - Fail: 0 or EOF(-1)	int **fscanf(**FILE* fp, char* format, ListOfVarAddresses)	They are the same in the function scanf()
	int **fprintf**(FILE* fp, char* format, VarList)	They are the same in the function printf()

Demonstration 3

Write a C-program that will perform the following operations:

- User enters a filename of a text file
- User will enter data to the file line-by-line until a null string is inputted.
- User will see the content of the file.

(The problem of the previous demo. But, the functions fscanf() and fprintf() are used.)

Demonstration 3….

K:\GiangDay\FU\OOP\BaiTap\string_file_demo2...

```
Program for writing then reading a file
Enter a filename:f6.txt
Write file:
The file f6.txt existed. Override it Y/N? :Y
Study hard -> Pass
Work hard -> success
To prefer pleasure to work -> poor

Data in the file f6.txt:
Study hard -> Pass
Work hard -> success
To prefer pleasure to work -> poor
```

```c
/* string_file_demo2.c */
#include <stdio.h>
#include <string.h>
int exist (char* filename)
{   int existed = 0;
    /* Try opening it for reading */
    FILE* f= fopen(filename,"r");
    if (f!=NULL)
    {   existed=1;
        fclose(f);
    }
    return existed;
}
```

f6.txt - Notepad

File Edit Format View Help

Study hard -> Pass
Work hard -> success
To prefer pleasure to work -> poor

Demonstration 3…

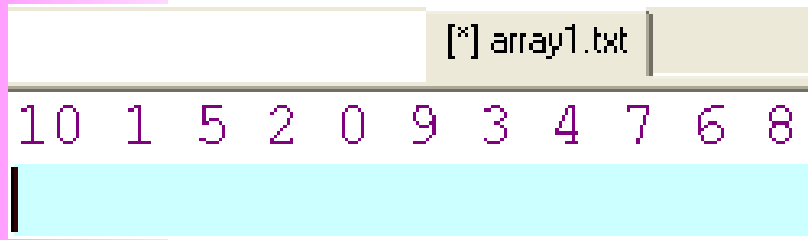
```c
/* Write a file from lines inputted until the 0-line entered */
int writeFile( char* filename)
{   if (exist (filename)==1)
    {  printf("The file %s existed. Override it Y/N? :", filename);
       if (toupper(getchar())=='N') return 0;
    }
    char line[201]; /* inputted string */
    int length=0; /* length if inputted line */
    /* open the file for writing */
    FILE* f= fopen(filename,"w");
    fflush(stdin); /* Clear input buffer */
    do
    {  gets(line); /* get a line - If user presses ENTER only --> length=0*/
       length = strlen(line);
       if (length>0) fprintf(f, "%s\n", line);/* write \n to file */
    }
    while (length>0);
    fclose(f);
    return 1;
}
```

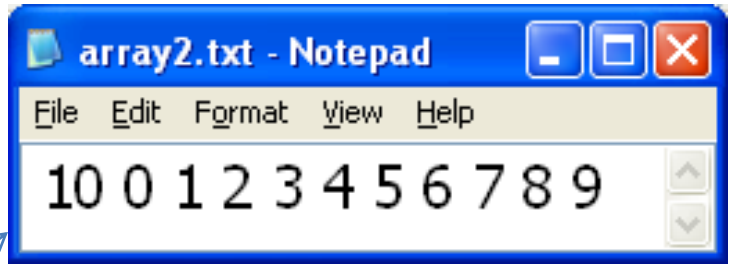
Demonstration 3…

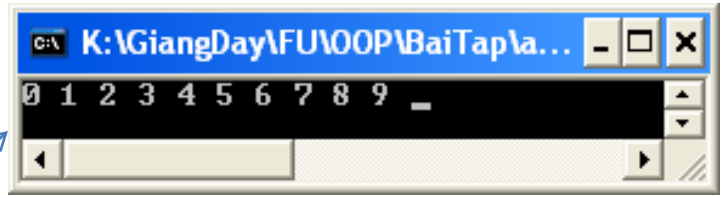
```c
/* Print out the content of a text file to the monitor */
int printFile( char* filename)
{   if (exist (filename)==0)
    {  printf("The file %s does not exist.\n", filename);
       return 0;
    }
    /* open the file for reading */
    FILE* f= fopen(filename,"r");
    char line[201]; /* a line from the file */
    /* When data can be read from the file to variable, process variable */
    /* [^\n]%*c" => read a line to \n then remove \n */
    while (fscanf(f,"%[^\n]%*c",line)>0) puts(line);
    fclose(f);
    return 1;
}
int main(int argCount, char* args[])
{   char filename[81];
    printf("Program for writing then reading a file\n");
    printf("Enter a filename:");
    gets(filename);
    printf("Write file:\n");
    if (writeFile(filename)==1)
    { printf("Data in the file %s:\n", filename);
      if (printFile(filename)==0)printf("File error!\n");
    }
    else printf("Writing file fail!\n", filename);
    getchar();
    return 0;
}
```

Demonstration 4

- Create a file, named **array1.txt**. The first number in the file is number of elements of an integer array. The later numbers are values of elements.







- Write a C-program that will:
  - Read the array contained in the above file.
  - Print it's elements in ascending order to monitor.
  - Write it to the file **array2.txt** using the same format with the file **array1.txt**.

Demonstration 4…

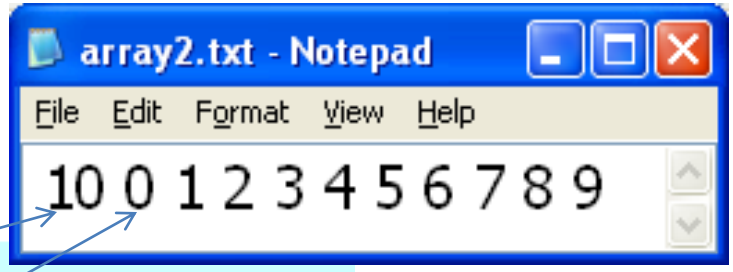
```c
/*array_file01.c */
#include <stdio.h>
/* Read data in a fiel to array */
void fileToArray( char* fname, int*a, int*pn)
{   FILE* f= fopen(fname, "r");
    /* Read the first number in the file -> number of elements */
    fscanf(f, "%d", pn);
    /* Read elements */
    int i;
    for (i=0; i<*pn; i++) fscanf(f, "%d", &a[i]);
    fclose (f);
}
void asc_sort( int *a, int n)
{   int i,j, t;
    for (i=0; i<n-1; i++)
        for(j=n-1; j>i; j--)
          if (a[j]<a[j-1])
          { t=a[j];
            a[j]=a[j-1];
            a[j-1]=t;
          }
}
```

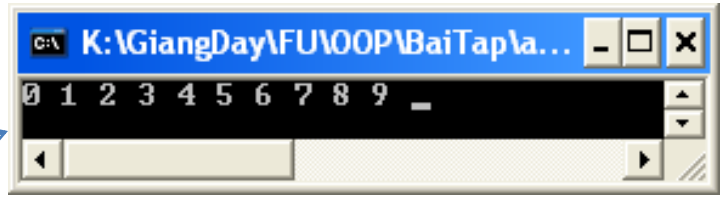
[*] array1.txt

```
10 1 5 2 0 9 3 4 7 6 8
```

Demonstration 4…

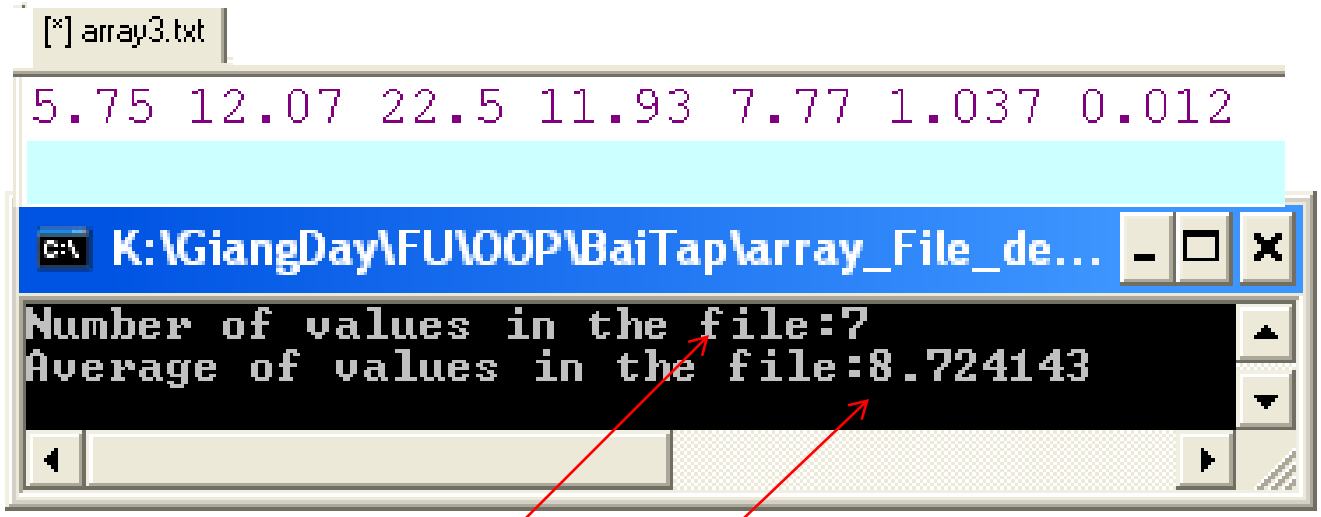
```c
/* Print out array to monitor */
int print( int *a, int n)
{   int i;
    for (i=0;i<n; i++) printf("%d ", a[i]);
}
/* Write array to file */
int printToFile ( char* fname, int *a, int n)
{   FILE* f= fopen(fname, "w");
    fprintf(f, "%d ", n); /* Write number of elements to file */
    int i;
    for (i=0;i<n; i++) /* write elements to file */
      fprintf(f,"%d ", a[i]);
    fclose(f);
}
int main()
{   char infName[] = "array1.txt";
    char outfName[] = "array2.txt";
    int a[200];
    int n=0;
    fileToArray(infName, a, &n);
    asc_sort(a,n);
    /* Print array to monitor - stdout: monitor */
    print(a, n);
    /* Print array to file */
    printToFile(outfName, a, n);
    getchar();
    return 0;
}
```





Demonstration 5

- Create a file, named **array3.txt** containing real numbers.



- Write a C-program that will:
  - Print out number of values
  - Print out the average of values contained in the above file.

Demonstration 5…

```c
/*array_file_demo2.c */
#include <stdio.h>
/* Processing data in a file is slow. So, if possible,
   all operations should be performed in one time.*/
void processFile( char* fname, int* pCount, double *pSum)
{   FILE* f= fopen(fname, "r");
    *pCount = 0 ; /* reset count  */
    *pSum=0;      /* reset sum  */
    double x; /* variable containing a value from file */
    while (fscanf(f, "%lf", &x)==1)
      {   (*pCount)++;
          (*pSum) += x;
      }
    fclose (f);
}
int main()
{   char infName[] = "array3.txt";
    int count =0; /* number of values in file */
    double sum=0; /* sum of values in file */
    processFile(infName, &count, &sum);
    printf("Number of values in the file:%d\n", count);
    printf("Average of values in the file:%lf\n", sum/count);
    getchar();
    return 0;
}
```

[*] array3.txt

```
5.75 12.07 22.5 11.93 7.77 1.037 0.012
```

K:\GiangDay\FU\OOP\BaiTap\array_File_de...

```
Number of values in the file:7
Average of values in the file:8.724143
```

Demonstration 6: rewind(FILE*)

```c
/*test_rewind.c */
#include <stdio.h>
int main()
{   char fname[] = "test_rewind.txt";
    char c; /* a chacracter from file */
    int i;
    FILE * f= fopen(fname, "r");
    printf("10 first characters:\n");
    for (i=0;i<10;i++) putchar(fgetc(f));
    rewind(f);
    printf("\n\nAfter rewind:\n");
    while ((c=fgetc(f))!=EOF) putchar(c);
    fclose(f);
    getchar();
    return 0;
}
```

test_rewind.txt

```
content for testing rewind function
```

test_rewind.txt

```
content for testing rewind function
```

K:\GiangDay\FU\OOP\BaiTap\test_rewind....

```
10 first characters:
content fo

After rewind:
content for testing rewind function
```

Demonstration 7: fseek(…)

test_fseek.txt

```
content for testing fseek function
```

```c
/*test_fseek.c */
#include <stdio.h>
int main()
{   char fname[] = "test_fseek.txt";
    char c; /* a chacracter from file */
    int i;
    FILE * f= fopen(fname, "r");
    printf("15 first characters:\n");
    for (i=0;i<15;i++) putchar(fgetc(f));
    puts("\n");
    fseek(f,-5,SEEK_CUR); /* from CURRENT position */
    for (i=0;i<5;i++) putchar(fgetc(f));
    puts("\n");
    fseek(f,-10,SEEK_END); /* from END position */
    for (i=0;i<5;i++) putchar(fgetc(f));
    puts("\n");
    fseek(f,10,SEEK_SET); /* from BEGINNING position */
    for (i=0;i<5;i++) putchar(fgetc(f));
    fclose(f);
    getchar();
    return 0;
}
```

content for testing fseek function
content for testing fseek function
content for testing fseek function
content for testing fseek function
content for testing fseek function

EOF (2bytes)

content for testing fseek function
content for testing fseek function

```
15 first characters:
content for tes

r tes

funct

r tes_
```

Summary

- File: Related data that are stored in a mass storage (disks).
- Files are managed by the operating system (OS).
- OS identifies a file through it's name.
- To specify a absolute filename in C:

  "C:\\f1\\f11\\file1.dat" or "C:/f1/f11/file1.dat"
- To process data in a file: We need to know format and meaning of each data in file.

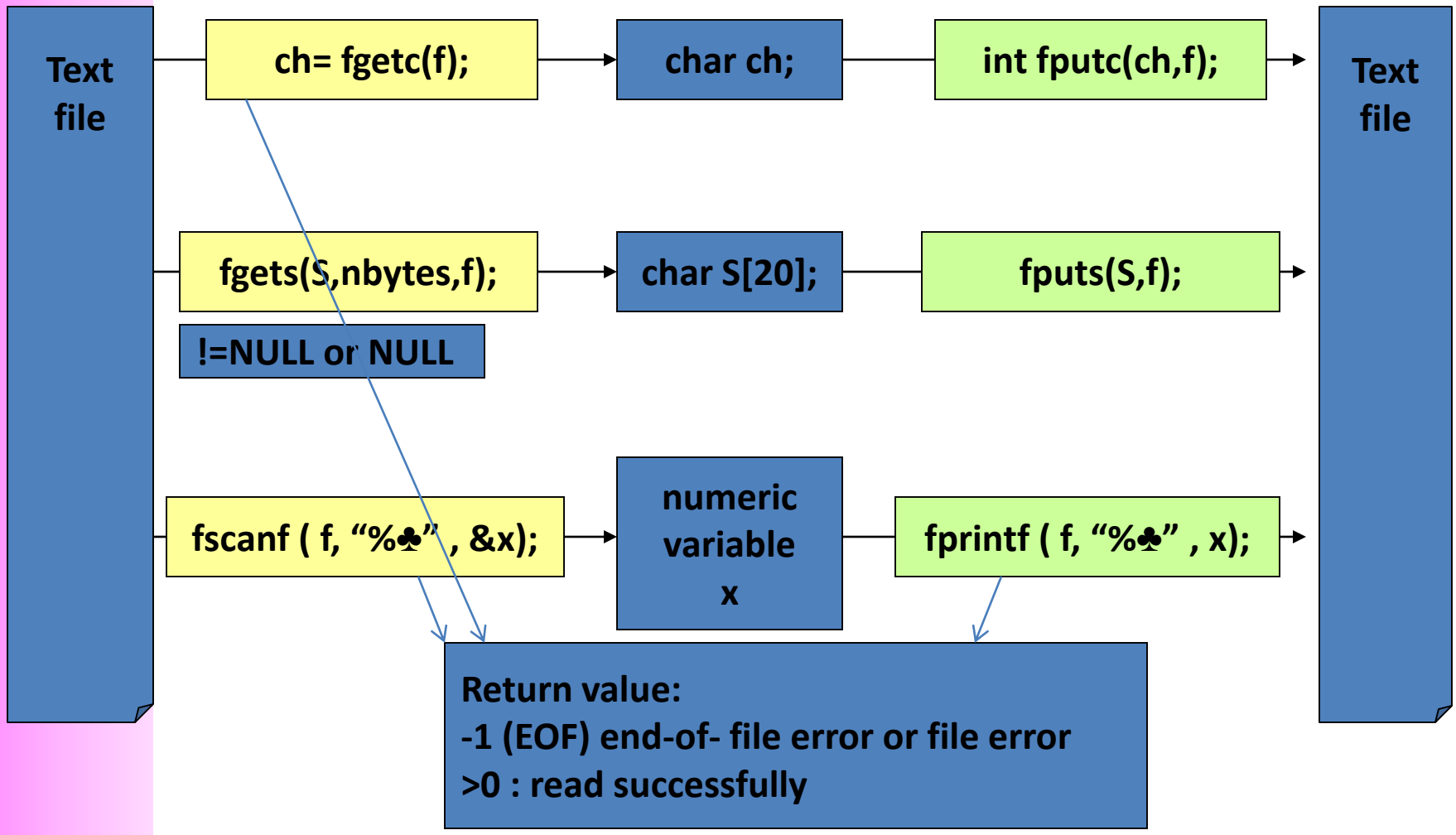
Summary

Purpose STDIO.H	Syntax
Open a file	**FILE* fopen(char fname[], char mode[])**
Close a opening fle	**int fclose(FILE*)**
Read a character	**int fgetc(FILE*)**
Write a character	**int fputc(char, FILE*) → EOF (-1)**
Read a string	**fgets(char S[], int nbytes, FILE* f); → NULL if EOF**
Write a string	**fputs (char*, FILE*)**
Read a number	**fscanf (FILE*, char* format, PointerList)**
Write a number	**fprint (FILE*, char* format, VarList)**
Test whether the file is EOF?	**int feof(FILE*)**
Rewind to the beginning	**void rewind (FILE*)**
Get the current file position	**long ftell(FILE*)**
Move the current position	**int fseek (FILE*, long offset, int fromPos)**
Rename a closed file	**rename (char fName[], char newName[])**
Remove a closed file	**remove (char fName[])**

Summary

Type	Standard I/O	File I/O
int	getchar()	fgetc(fp)
int	putchar(ch)	fputc(ch, fp)
char *	gets(str)	fgets(str, max, fp)
int	puts(str)	fputs(str, fp)

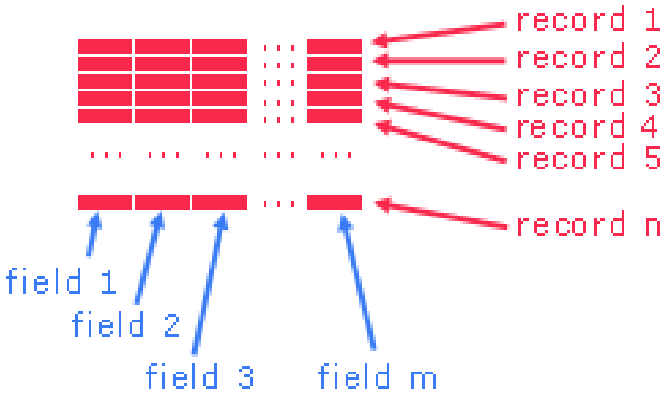
Summary


Text file
ch= fgetc(f);
char ch;
int fputc(ch,f);
Text file
fgets(S,nbytes,f);
char S[20];
fputs(S,f);
!=NULL or NULL
fscanf (f, "%♣" , &x);
numeric variable x
fprintf (f, "%♣" , x);
Return value:
-1 (EOF) end-of- file error or file error
>0 : read successfully

Thank You

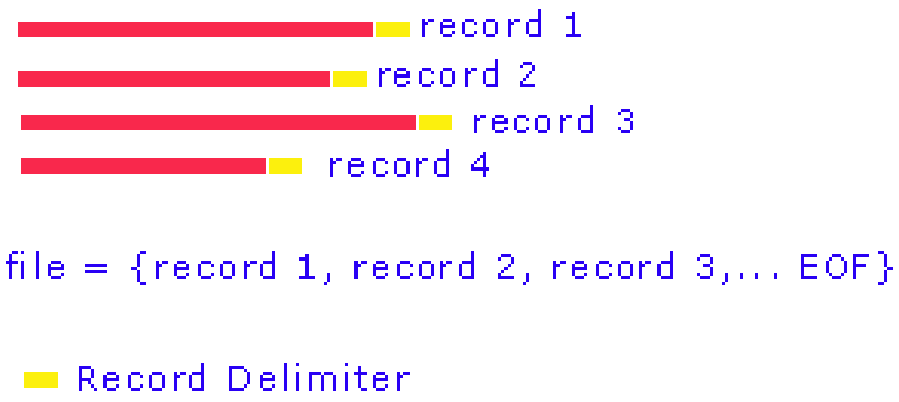
Bonus:
Text Files and Parallel Arrays

- Actually, each real object contains some data, such as details of students include name, address and mark. Some arrays can be used to manage a list of objects.
- Data of a class (group of students) are usually presented in a file as a table.
- A row in a data table is called as a record.
- Each column in the table is call a field.



Bonus…

- We call each line in a text file a record.
- A record is a sequence of characters that ends with a **newline** delimiter.
- Typically, one record refers to one entity of information.



students.txt

```
Joseph;12 Le Loi, Q1, TPHCM;7
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Miranda;123 Calmette, District 1, HCM City;5
Celine Dion;124 street 8, district 7, HCM City;9
```

Bonus…

- To manage a list of records, some arrays are needed. All elements at the same position present a record.
- If one change is performed on an array (such as sorting), others may be changed appropriately.

students.txt

```
Joseph;12 Le Loi, Q1, TPHCM;7
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Miranda;123 Calmette, District 1, HCM City;5
Celine Dion;124 street 8, district 7, HCM City;9
```

Joseph	12 Le Loi, Q1, TPHCM	7
Dinh Tan Vu	12/66 duong so 3, Qo Vap, TPHCM	8
Miranda	123 Calmette, District 1, HCM City	5
Celine Dion	124 street 8, district 7, HCM City	9

Bonus…

- Data representing a student include: name, address, mark.
- A list of students are stored in the file **students.txt** as below:

students.txt

```
Joseph;12 Le Loi, Q1, TPHCM;7
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Miranda;123 Calmette, District 1, HCM City;5
Celine Dion;124 street 8, district 7, HCM City;9
```

- Write a C-program that will print out the list of students in descending order based on their marks then the list will be written to the file students_2.txt with the same format as the previous file.

Bonus…

students.txt

```
Joseph;12 Le Loi, Q1, TPHCM;7
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Miranda;123 Calmette, District 1, HCM City;5
Celine Dion;124 street 8, district 7, HCM City;9
```

K:\GiangDay\FU\OOP\BaiTap\para_Arr_demo.exe

```
Sorted List:
Celine Dion          124 street 8, district 7, HCM City          9
Dinh Tan Vu          12/66 duong so 3, Qo Vap, TPHCM             8
Joseph               12 Le Loi, Q1, TPHCM                        7
Miranda              123 Calmette, District 1, HCM City          5

Result file:students_2.txt
```

students_2.txt - Notepad

File Edit Format View Help

```
Celine Dion;124 street 8, district 7, HCM City;9
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Joseph;12 Le Loi, Q1, TPHCM;7
Miranda;123 Calmette, District 1, HCM City;5
```

Bonus…

```c
/*para_Arr_demo.c*/
#include <stdio.h>
#include <string.h>
/* Read data in a file to 3 arrays */
void readFile(char* fname, char names[][41], char adds[][41], int*marks, int *pn);
/* sort the list based on mark descendingly */
void sort(char names[][41], char adds[][41], int*marks, int n);
/* Print out the list to monitor */
void print(char names[][41], char adds[][41], int*marks, int n);
/* Write the list to file*/
void writeFile(char* fname,char names[][41], char adds[][41], int*marks, int n);
int main()
{   char inFilename[] = "students.txt";
    char outFilename[] = "students_2.txt";
    /* 3 arrays represent a list of students */
    char names[50][41];  char adds[50][41]; int marks[50];
    int n=0; /* number of students */
    /* read data from file to arrays */
    readFile(inFilename, names, adds, marks, &n);
    /* sort the list based on mark descendingly */
    sort(names, adds, marks, n);
    /* Print out result */
    printf("Sorted List:\n");
    print(names, adds, marks, n);
    /* Write the list to file */
    writeFile(outFilename, names, adds, marks, n);
    printf("\nResult file:%s\n", outFilename);
    getchar();
    return 0;
}
```

Bonus…

```c
/* Read data in a file to 3 arrays */
void readFile(char* fname, char names[][41], char adds[][41], int*marks, int *pn)
{   *pn=0; /* reset number of elements */
    FILE* f= fopen(fname, "r");
    if (f!=NULL)
    {   /* While read successfully a whole data line */
        while (fscanf(f, "%40[^;];%40[^;];%d%*c",names[*pn],adds[*pn],&marks[*pn])==3)
            (*pn)++;
        fclose(f);
    }
}
```

students.txt

```
Joseph;12 Le Loi, Q1, TPHCM;7
Dinh Tan Vu;12/66 duong so 3, Qo Vap, TPHCM;8
Miranda;123 Calmette, District 1, HCM City;5
Celine Dion;124 street 8, district 7, HCM City;9
```

```c
/* Print out the list to monitor */
void print(char names[][41], char adds[][41], int*marks, int n)
{   int i;
    for (i=0;i<n; i++)
      printf("%-20s%-41s%4d\n", names[i], adds[i], marks[i]);
}
```

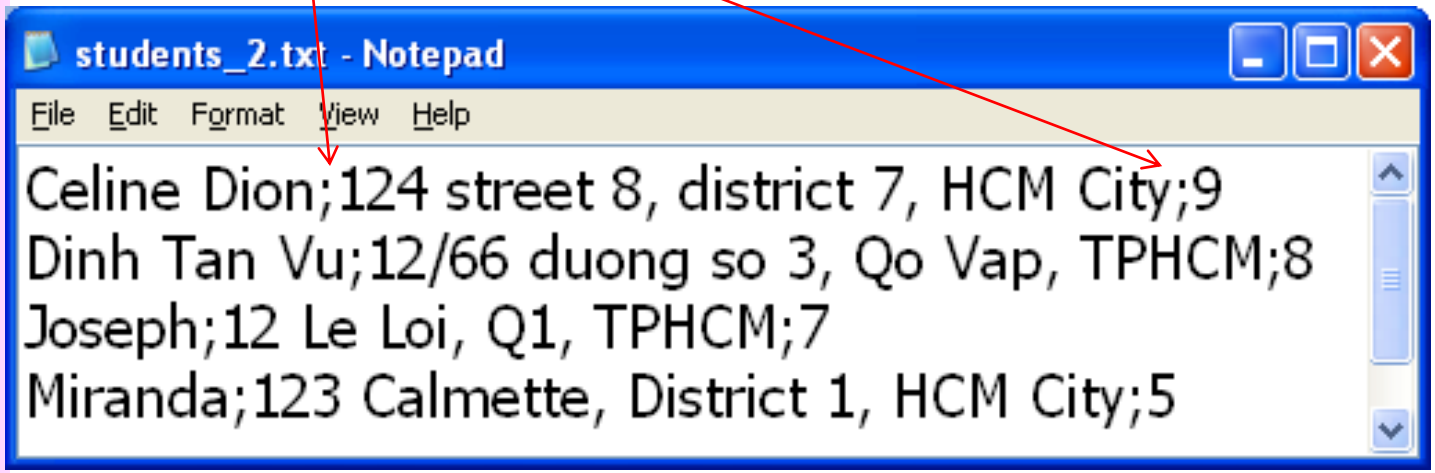
```
Celine Dion          124 street 8, district 7, HCM City          9
Dinh Tan Vu          12/66 duong so 3, Qo Vap, TPHCM             8
Joseph               12 Le Loi, Q1, TPHCM                        7
Miranda              123 Calmette, District 1, HCM City          5
```

Bonus…

```c
/* sort the list based on mark descendingly - Bubble sort*/
void sort(char names[][41], char adds[][41], int* marks, int n)
{   int i, j;
    for (i=0;i<n-1; i++)
      for (j=n-1; j>i; j--)
        if (marks[j]>marks[j-1])
         {  /* swap array names */
            char tName[41];
            strcpy(tName, names[j]);
            strcpy(names[j], names[j-1]);
            strcpy(names[j-1], tName);
            /* swap array addss */
            char tAdd[41];
            strcpy(tAdd, adds[j]);
            strcpy(adds[j], adds[j-1]);
            strcpy(adds[j-1], tAdd);
            /* swap array marks */
            int tMark= marks[j];
            marks[j]=marks[j-1];
            marks[j-1]=tMark;
         }
}
```

Bonus…

```c
/* Write the list to file*/
void writeFile(char* fname,char names[][41], char adds[][41], int*marks, int n)
{   FILE* f= fopen (fname, "w");
    int i;
    for (i=0;i<n; i++)
      fprintf(f,"%s;%s;%d\n", names[i], adds[i], marks[i]);
    fclose(f);
}
```



Thank You