



UNIVERSITY OF MASSACHUSETTS  
DARTMOUTH

# ECE160: Foundations of Computer Engineering I

## Lecture #20 – **Files (II)**

Instructor: Dr. Liudong Xing  
SENG-213C, [lxing@umassd.edu](mailto:lxing@umassd.edu)  
ECE Dept.



# Administrative Issues (3/29)

- Lab#9
  - Review Exam#2 problems
  - Due 5pm, Wednesday, March 29
- Today's topics
  - Files II (L#20)

# Review of Lectures #18

- Files: a collection of information/related data treated as a unit
- **How to declare a file\_pointer**

```
FILE *file_pointer;
```

- **How to open a file**

```
file_pointer = fopen("file_name", "mode");
```

- To create a link between a file stored in actual disk and a file pointer
- Returns a valid address if the open succeeds, otherwise **NULL** (a C-defined constant for no address)
- **How to close a file**

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

- To free system resources (memory space)
- Returns integer ZERO if the close succeeds, otherwise EOF (-1)

# Outline

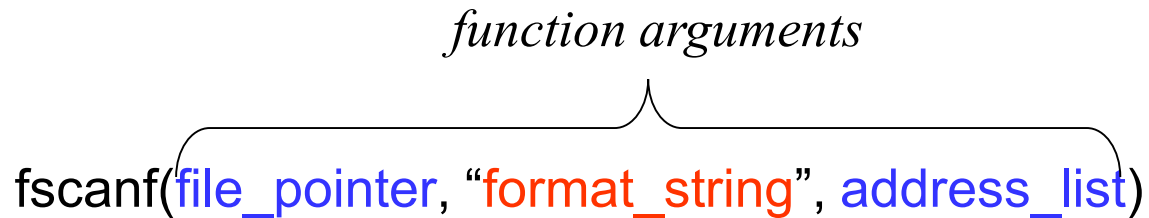
- To read from a file
- To write output to a file
- Character input/output function

# How to Read data from a File?

Using `fscanf()` :

*function arguments*

`fscanf(file_pointer, "format_string", address_list)`



- Reads the contents of the file indicated by the `file_pointer` according to the conversion code in `format_string`.
- Contents read are put into the address given by the `address_list`.

*Read from keyboard:*

```
scanf("format string", address list)
```

**Note: in Microsoft Visual Studio, we use `fscanf_s(...)`**

# An Example

```
FILE *example_ptr;  
example_ptr = fopen("L19test.txt", "r");  
fscanf(example_ptr, "%d%f", &a, &b);
```

- Two values will be read from input file indicated by `example_ptr`
- The integer value → the memory cell reserved for `a`
- The float value → the memory cell reserved for `b`

# Exercise (1)

- Find errors, if any, in the following statement:

```
scanf("myfile", "%4d %6d", week, year);
```

# Example Program

```
#include <stdio.h>
```

```
void main(void)
```

```
{
```

```
double x ;
```

```
int i, k;
```

```
FILE *inptr;
```

variable  
declarations

declaring a file\_pointer variable to be used with in functions *fopen()*, *fscanf()*, and *fclose()*

```
inptr=fopen ("ECE160.DAT","r");
```

```
fscanf(inptr,"%d",&i);
```

```
fscanf(inptr,"%d %lf",&k,&x);
```

Calling *fopen()* to allow the program to access the disk file ECE160.DAT → creating a link between *inptr* and the actual file

Closing the file to free resources pointed by *inptr*

```
fclose(inptr);
```

*fscanf()* works similar to *scanf()*. However, it uses the file pointed by the file\_pointer *inptr*

```
printf("i=%5d\nk=%5d\nx=%9.3lf\n",i, k, x);
```

```
}
```



# Note!

```
fscanf(file_pointer, "format_string", address_list)
```

- The input file indicated by the `file_pointer`
  - Should have an acceptable file name
  - Must be linked with a `file_pointer` using `fopen()` before it is used
  - Should be closed after it is used `fclose(file_pointer);`

```
FILE *file_pointer;
```

```
file_pointer = fopen("file_name", "r");
```



```
FILE → file_pointer → actual_file on the disk
```

# Outline

- To read from a file
- **To write output to a file**
- Character input/output function

# How to Write output to a File

```
printf("format_string", data_list)
```

- The output displayed on the screen is lost when the screen scrolls or clears
- To keep a permanent record of the output, write the output to a file
  - Using **fprintf()**
  - Use a file editor to view the output in a file or print the result on a printer

# Function fprintf()

```
fprintf(file_pointer, "format_string", data_list)
```

Writes the values of data in `data_list` using the given `format_string` to a file that is linked to the program using the `file_pointer`

An example:

```
fprintf(example_ptr, "week = %5d\n year = %5d\n", week, year)
```

The values of `week` and `year` are written to an external file that has a file pointer named `example_ptr` using the format string given in the double quotes.

## Exercise (2)

- Find errors, if any, in the following statement:

```
printf(*myfile, "week=%4d\n year= %6d", &week, &year);
```

# Example Program

```
#include <stdio.h>
void main(void)
{
  double x =327.5;
  int  week=7, year=2005;
  FILE *myfile;
```

variable  
declarations

declaring a `file_pointer` variable to be used with in functions `fopen()`, `fprintf()`, and `fclose()`

Calling `fopen()` to allow the program to access the disk file `Example.OUT` → creating a link between `myfile` and the actual file

```
myfile=fopen ("Example.OUT","w");
fprintf(myfile,"week=%6d\neyear=%6d\n",week, year);
fprintf(myfile,"income=%lf\n",x);
```

Closing the file to free resources pointed by `myfile`

```
fclose(myfile);
```

`fprintf()` works similar to `printf()`. However, it uses the file pointed by the `file_pointer` `myfile`

```
}
```

# Note!

```
fprintf(file_pointer, "format_string", data_list)
```

- Like the input file in `fscanf()`, the output file indicated by the `file_pointer` in `fprintf()`
  - Should have an acceptable file name
  - Must be linked with a `file_pointer` before it is used
  - Must be opened before it is used
  - Should be closed after it is used `fclose(file_pointer);`

```
FILE *file_pointer;
```

```
file_pointer = fopen("file_name", "w");
```



```
FILE → file_pointer → actual_file on the disk
```

# Summary

- Use `scanf()` function to read data from keyboard
- Use `fscanf()` function to read data from a disk file
- Use `printf()` function to write output on the screen
- Use `fprintf()` function to write output to an external disk file



```

#include <stdio.h>
int main(void)
{
FILE *fp;
int num1 = 100;
int num2 = 200;
int num3 = 300;
int a = 0, b = 0, c = 0;

//fp = fopen("Xing_file1.txt","w");
fopen_s(&fp, "Xing_file1.txt", "w");

if (!fp)
{
printf("I was not able to open file\n");
return(1);
}

fprintf(fp, "%d\n%d\n%d\n", num1, num2, num3);

if (fclose(fp) == EOF)
{
printf("I was not able to close file\n");
return(2);
}

```

```

//fp = fopen("Xing_file1.txt","r");
fopen_s(&fp, "Xing_file1.txt", "r");

if (!fp)
{
printf("I was not able to open file\n");
return(1);
}

fscanf_s(fp, "%d%d%d", &a, &b, &c);

printf("a is %d\nb is %d\nc is %d\n",a,b,c);

if (fclose(fp) == EOF)
{
printf("I was not able to close file\n");
return(2);
}
}

```

## A Complete Example

# Exercise (3)

- Write a program that writes the following data to a file (`fprintf()`). It then reads the data from the file (`fscanf()`) and prints them out on the screen (one per line) (`printf()`)

10  
3.14 20

```
#include "stdio.h"
void main(void)
{
    int i = 10;
    float j = 3.14;
    int a = 20;

    int ii;
    float jj;
    int aa;

    FILE *fp;

    ... ..

}
```

# Outline

- To read from a file
- To write output to a file
- **Character input/output function**

# Review (from Lecture#4)

- A character is stored in a computer's memory as an integer representing the ASCII code of the corresponding character (<https://www.ascii-code.com/>).
- Examples (ASCII in Dec):
  - '0' – 48 '1' – 49 'a' – 97 '\n' -- 10
- For this reason, a character in C can be interpreted as a small integer; C often treats a character like an integer!

# Character Input/Output Functions

- Character input functions read one character at a time from a text stream
- Character output functions write one character at a time to a text stream

# getchar(), putchar()

- `int getchar(void);`
  - It reads a single character from the standard input stream (a character typed in at the keyboard) and returns its value.
  - `EOF` is returned if an error is detected
  - To call `getchar` function, nothing is enclosed in the parentheses
  - Note the return type is integer
- `int putchar(int mychar);`
  - It writes one character to the standard output.
  - `EOF` is returned if any error occurs during the write operation
  - The character it wrote will be returned in case of success

# Examples

`getchar();`

- A character is read from the keyboard, but not stored in any variable's memory cell

`c1=getchar();`

- A character is read from the keyboard, but stored in the memory cell reserved for variable `c1`

# Examples

`putchar('x');`

– causes the character x to be printed on the screen

- `printf()` and `putchar()`:

Assume

`char c1='a', c2='b';`

```
putchar(c1);  
putchar(' ');  
putchar(c2);  
putchar('\n');
```



```
printf(“%c %c\n”, c1, c2);
```



# Note!

- *getchar()* and *putchar()* read and write the standard input (keyboard) and output (monitor) streams
- To work with disk files, use *getc()/fgetc()* and *putc()/fputc()*

# Character I/O Functions (Cont'd)

- `int getc(FILE *fp);`  
`int fgetc(FILE *fp);`
  - They read the next character from a file with file\_pointer `fp`.
  - EOF is returned when an end of file is detected, or an error occurs
  
- `int putc(int mychar, FILE *fp);`  
`int fputc(int mychar, FILE *fp);`
  - They write a character to a file with file\_pointer `fp`.
  - If the character is successfully written, the function returns it
  - EOF is returned if any error occurs.

```

/* This program writes data from the keyboard into a file */
#include "stdio.h"
int main(void)
{
    FILE *fp;
    int c;

    printf("This program writes your input to a file.\n");
    //fp = fopen("mycopyfile.txt", "w");
    fopen_s(&fp, "mycopyfile.txt", "w");
    if (!fp)
    {
        printf("Error. I couldn't open the file.\n");
        return 1;
    }

    while ((c = getchar()) != EOF)
    {
        fputc(c, fp); /* Note the automatic conversion of c to char */
    }

    if (fclose(fp) == EOF)
    {
        printf("Error. I couldn't close the file.\n");
        return 2;
    }
    printf("I have created your file.\n");
    return 0;
}

```

# An Interesting Problem

- Read 3 integers from the keyboard, add them up, and print their sum.
- 3 solutions:
  - using `scanf`
  - using `scanf`, a function, and pass by reference
  - using `getchar` and `atoi` (a C library function converting from the string argument to an integer)

## Solution 1: using scanf()

- Read 3 integers from the keyboard, add them up, and print their sum.

```
#include "stdio.h"
int main(void)
{
    int k=1; //counter variable
    int a;
    int sum = 0;
    printf("Please enter 3 numbers to add.\n");

    .. .. ..//use a while loop

    return 0;
}
```

```

#include "stdio.h"
void add(int *s, int n);
int main(void)
{
int a;
int k = 1; // counter variable
int sum = 0;

printf("Please enter numbers to add.\n");

while (k<=3)
{
scanf_s("%d", &a);
.. .. .. //function call
k++;
}

printf("The sum result is %d\n", sum);
return 0;
}

.. .. .. //function definition

```

Solution 2:  
using scanf, a  
function, and  
pass by  
reference

```

#include "stdio.h"
#include "stdlib.h"
int main(void)
{
    char c;
    int a;
    int sum = 0;
    int k = 1; //counter variable

    while (k<=3)
    {
        c = getchar();
        if (c != '\n')
        {
            a = atoi(&c);
            sum = sum + a;
            k++;
        }
    }
    printf("The sum is: %d\n", sum);
    return 0;
}

```

Solution 3:  
using  
getchar()  
and atoi()

# Summary of Lectures #20

- Use *fscanf()* function to read data from a disk file
- Use *fprintf()* function to write output to an external disk file
- *getchar()* and *putchar()* read and write the standard input (keyboard) and output (monitor) streams
- *getc()/fgetc()* and *putc()/fputc()* read and write a file stream specified by the `file_pointer`



# Things To Do

- Review lecture notes
- Run the programs on Slides #14, 17, 18, 29-31 and check contents of related files in your disk (refer to the solution file if necessary)

## Next Topics

- Arrays