UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #18 Files (1)

Instructor: Dr. Liudong Xing
SENG-2134C, lxing@umassd.edu
ECE Dept

Administrative Issues

- Lab#7 solution posted
- Homework#4 assigned
  - Due **9am, Wednesday, March 22**
- Today's topics
  - Repetitive algorithms (L#17, Cont'd)
  - Files (L#18)

Review of Lectures #17

- Two approaches to writing repetitive algorithms
  - Using loops (for, while, do…while; iterative way)
  - Using recursion: is a repetitive process where a function calls itself
    - Recursive solution involves a two-way journey
      - First we decompose the problem from top to bottom
      - Then we solve it from bottom to top
    - Base case:
      - The statement that "solves" the problem
      - Every recursive function must have a base case
      - Once the base case has been reached, the solution begins

Review: factorial(n)

```
long factorial(int n)
{
    int i;
    long fact=1;
    for(i=1; i<= n; i++)
     {
          fact = fact * i;
     }
    return fact;
}
```

```
long factorial(int n)
{
    if (n == 0)
          return 1;
    else
          return(n*factorial(n-1));
}
```

$$factorial(n) = \begin{cases} 1 & \text{if } n = 0 \\ 1*2*...*(n-1)*n & \text{if } n > 0 \end{cases}$$

$$factorial(n) = \begin{cases} 1 & \text{if } n = 0 \\ n* factorial(n-1) & \text{if } n > 0 \end{cases}$$

Iterative Solution

Recursive Solution

Review: Fibonacci(n)

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,……

- Base cases: 0, 1
- General case:

$$fib(n)=fib(n-1)+ fib(n-2)$$

```
long fib(long n)
{
        if ((n == 0) || (n == 1))
                return n;
        return(fib(n-1)+fib(n-2));
}
```

Recursive Solution

```
long fib(long n)
{
        int i;
        long cn = 1;
        long pn = 0;
        long ppn;

        for (i=1; i<n; i++)
        {
                ppn = pn;
                pn = cn;
                cn = ppn + pn;
        }
        return cn;
}
```

Iterative Solution

Agenda

- Files
  - Concepts
  - To create, open, close files

An Example

```
#include "stdio.h"
void main(void)
{
   int a;
    a=0;
    printf(“Please input the value of variable a:\n”);
    scanf_s(“%d”, &a);
    printf(“The new value of variable a is %d\n.”, a);
}
```

When we store data in variables in our programs, they are lost after the program ends. These data are temporarily stored in main memory

How can we store data permanently?

Store them in files.

File

- A collection of information/related data treated as a unit
- Saved in secondary (auxiliary) memory like disks.
- Examples
  - Your personal data files
  - Video files

File Types

Two categories of files

- Text file
  - All data are human-readable characters.
  - Each line of data ends with a newline character.
- Binary file
  - They are not human-readable.
  - They store data in the computer's internal computer formats.

Streams

- All files in C are considered as byte streams.
- Each file ends with an EOF marker or at a specific byte number in a system-maintained administrative data structure

- Define a file:

```
FILE *file_pointer;
```

  - file_pointer is a pointer to a FILE structure
  - A file pointer is a variable whose memory cell contains an address instead of an *int* or *float* value

About FILE

- FILE is a C derived data type defined in the C standard header file stdio.h
  - Include file: #include <stdio.h>

- No direct relation between the C data type FILE and your actual file
- To manipulate a disk file, use the C data type FILE to declare a file_pointer, then use this file_pointer to handle your file

FILE → file_pointer → actual_file on the disk

Naming File Pointers

- Naming convention for file pointers is the same as the naming rules for other C identifiers
- Legal examples
  - FILE *ECE160;
  - FILE *apple;
- Illegal examples
  - FILE *2005ECE160;
  - FILE *+apple;

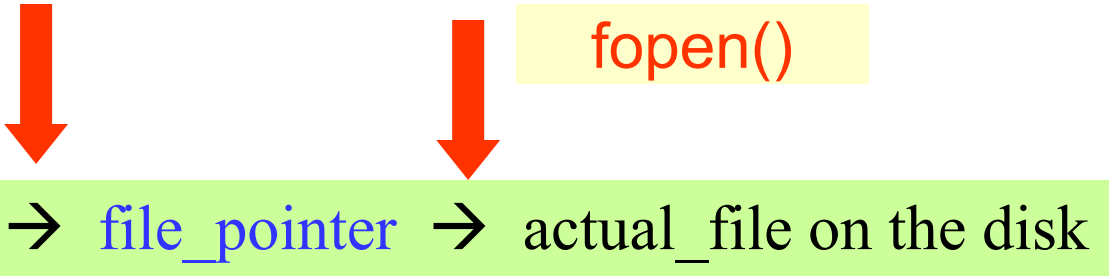
After we have declared the file pointer, how do we make a file available for us to read?

Function fopen()

- Use C standard library function fopen() (in stdio.h)

- "fopen() gives us the ability to create a link between a file stored in the secondary memory and a file pointer. Once the link is created we can work with the file pointer in our program to give us access to the file to which it is linked."

```
FILE *file_pointer;
```



How to Open a file?

- Format:

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

- mode:
  - r: Open file for reading.
    - If file exists, the marker is positioned at the beginning of the file.
    - If the file doesn't exist, then error is returned.
  - w: Open text file for writing.
    - If file exists, it is emptied.
    - If file doesn't exist, it is created.
  - a: Open text file for appending.
    - If file exists, the marker is positioned at the end.
    - If file doesn't exist, it is created.

Note: in Microsoft Visual Studio, we use
fopen_s(&file_pointer, "file_name", "mode");

Example

```
FILE *example_ptr;
example_ptr = fopen("lab3.cpp", "r");
```

- The file_pointer is named example_ptr
- The access_mode is "r" → the files lab3.cpp is opened for reading
- *Note:* file_name and access_mode are in string literals, they must be enclosed by double quotes!

```
Note: in Microsoft Visual Studio, we use
fopen_s(&example_ptr, "lab3.cpp", "r");
```

How to Close a file?

- After the program finishes execution, C will automatically close all opened files
- It is a good practice to close files (to free system resources) after they have been used!
- To close a file manually, use fclose()
- Format/prototype:

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

- Example:

```
fclose(example_ptr);
```

- *Note:* use file_pointer, not the file name to close a file!

Note!

- fopen() returns
  - a valid address in your file variable if the open succeeds,
  - NULL (a C-defined constant for no address) if the open failed

- fclose() returns
  - an integer that is ZERO if the close succeeds,
  - EOF (-1) if there is an error

An Example

```
#include "stdio.h"
void main(void)
{
   FILE *fp;
   if((fp = fopen("my160file.txt","r")) == NULL)
   {
            printf("I was not able to open file\n");
   }
   if(fclose(fp) == EOF)
   {
            printf("I was not able to close file\n");
   }
}
```

Exercises (1)

- True/False

  _____You must create a link between an external disk file and a file pointer before you can read your input data from a file

  _____It is a good practice to close an input file when you need no further access to the file

  _____A file pointer is an *int* data type and can be declared with other *int* type variables

Exercises (2)

- Find error, if any, in each statement

```
#Include <Stdio.h>;

File myfile;

*myfile=fopen(lab6.dat, r);

close(“myfile”);
```

Summary

- 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)

Next…

- How to read from a file
- How to write output to a file

How to read from a file (an example)

```
fscanf(example_ptr, "%d%lf", &a, &b);
```

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

Note: in Microsoft Visual Stdio, we use

```
fscanf_s(example_ptr, "%d%lf", &a, &b);
```

How to write output to a file (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.

```c
#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);
}
```

```c
//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 (Preview)

Summary of Lectures #18

- Files
  - A collection of information/related data treated as a unit
  - How to declare a file_pointer (FILE)
  - How to open a file (fopen())
  - How to close a file (fclose())

Next Topics …

- How to read from a file
- How to write output to a file

Things to Do

- **Homework #4 due Wednesday, March 22**