File I/O

J.W. Choi
2024

File I/O

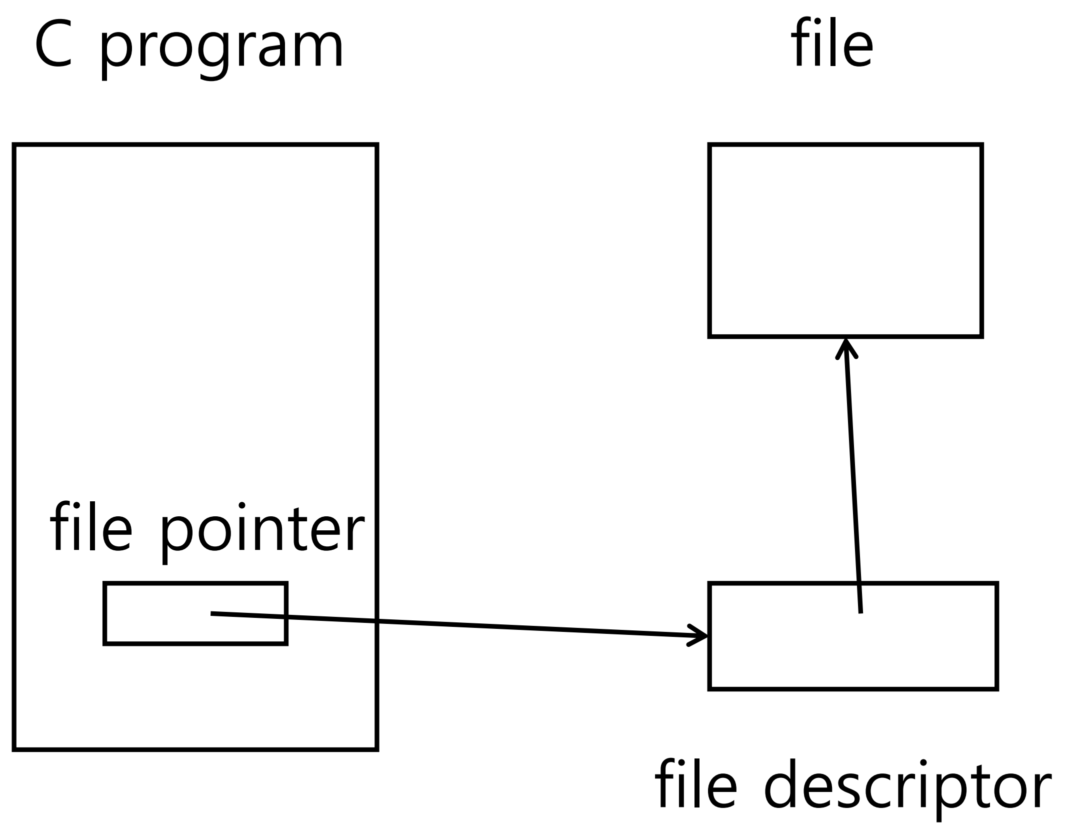
File I/O

- File is a steam of data
- Create a data file
- Declare a file
  - FILE *myFile;
  - ** pointer to an internal file descriptor data structure
- Open the file
  - #include <stdio.h>
  - char myFileName[15] = “prices.dat”;
  - myFile = fopen (myFileName, “r”);
  - ** r(read), w(write), a(append) modes
- Close the file
  - fclose (myFile)

File I/O

- File Open
  - Memory is allocated for the file descriptor data structure.
- File Close
  - Memory allocated for the file descriptor data structure is released (just like free()).
- After opening the file, read and write data as if you read and write to the terminal.

File I/O



Example

```
#include <stdio.h>

int main ()
{
   FILE *myFile;
   char myFileName[15] = “price.dat”;

   myFile = fopen(myFileName, “r”);
   if (myFile == NULL)
      printf (“₩nFile Could Not Be Opened”);
   else
      {
       .....
       }

    return 0;
}
```

Writing and Reading Data From a File

- Similar to writing and reading data from the terminal
- fprintf, fputs, fputc to write data to a file
- fscanf, fgets, fgetc to read data from a file
  - No Need for a User Prompt
- #include <stdio.h>

- Differences
  - A file pointer is needed.
  - At the end of a file, EOF is written.
    - similar to ₩0 at the end of a string

fprintf, printf

- fprintf (file_ptr, control_string, other_arguments)

- printf (cs, oa) is the same as
  - fprintf (stdout, cs, oa)

fscanf, scanf

- fscanf (file_ptr, control_string, other_arguments)
- scanf (cs, oa) is the same as
  - fscanf (stdin, cs, oa)

Testing the End of a File

- Look for EOF
- See if fscanf returned a number > 0
  - fscanf returns the number of items read successfully.
  - At the end of a file, fscanf returns EOF, not a number > 0

Testing for EOF

- scanf normally returns the number of items read.
- But at end of file, fscanf returns EOF

- Use this fact to test the end of file when reading a file.

- if (fscanf(inFile, “%d”, &a) != EOF) or
- if (fscanf(inFile, “%d”, &a) > 0) or
- if (fscanf(inFile, “%d”, &a) == 1)

  means “1 item is read from a file

  pointed to by inFile”

gets, fgets, puts, fputs

- gets (string-name);
- fgets (string-name, n, file-pointer);
  - read n-1 characters from file, and store them in string.
- puts (string-name);
- fputs (string-name, file-pointer);
  - write string to file.

getchar, fgetc, putchar, fputc

- getchar ();
- fgetc (file-pointer);
  - read 1 character from file.
- putchar (‘char’);
- fputc (char, file-pointer);
  - write 1 character to file.

Example 1

```
void main ()
{
   int i;
   FILE *myFile;
   double  price[2] = {139.25, 19.03};
   char   *description[2] = {“glove”, “CD”};

/* Write the product data stored in the description array and price
    array to the output file  */

   myFile = fopen(“price.dat”, “w”);
   if (myFile == NULL)
      printf (“₩nFile Could Not Be Opened”);
   else
   {
      for (i=0; i<2; i=i+1)
         fprintf (myFile, “%-9s %6.2f₩n”, description[i], price[i]);
      fclose (myFile);
    }
}
```

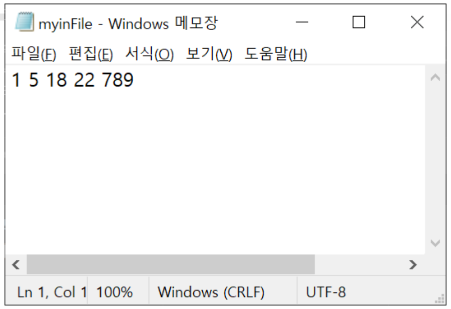
Example 2

```
#include  <stdio.h>
void main ()
{
   int a, sum=0;
   FILE *inFile, *outFile;

   inFile = fopen(“myinFile.txt”, “r”);
   if (inFile == NULL) {
      printf (“Input File Could Not Be Opened.”);
      exit(1);}
   outFile = fopen(“myoutFile.txt”, “w”);
   if (outFile == NULL) {
      printf (“Output File Could Not Be Opened.”);
      exit(1);}

  /* Read and sum the integers from the input file and
      write the sum to the output file */
   while (fscanf(inFile, “%d”, &a) == 1) /* loops until EOF  */
      sum = sum + a;
   fprintf (outFile, “The sum is %d ₩n”, sum);

   fclose (inFile);
   fclose (outFile);
}
```


myinFile - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
1 5 18 22 789
```

Example 3

```
FILE *infile;
char line[100];
int lcount = 0;

if((infile = fopen("charstream.txt", "r")) == NULL) {
    printf("File Could Not Be Opened.₩n");
    exit(1);}

/* Get each line from the input file and write the line and
     line number to the screen */
while (fgets(line, sizeof(line), infile) != EOF ) {
    lcount++;
    printf("Line %d: %s", lcount, line); }

fclose(infile);
```

Example 4

```
void main () {
   char ch;
   FILE *myFile;
   myFile = fopen(“charstream.txt”, “r”);

   /* Read the input file one character at a time,
       and write the character to the screen.    */
   if (myFile == NULL)
      printf (“₩nFile Could Not Be Opened”);
   else {
      ch = fgetc (myFile);
      while (ch != EOF) {
          putchar (ch);
          ch = fgetc (myFile); }
      printf (“₩nFinished printing the file ₩n”);
      fclose (myFile); }
 }
```

Example 5 (Variant of Example 4)

```
FILE *myFile;
char ch;

myFile =fopen("charstream.txt", "r");
if (myFile==NULL) {
    printf("Could not open charstream.txt!₩n");
    exit(1);}

/* Read the input file one character at a time, and write
     the character to the screen.   */
while ((ch=fgetc(myFile)) != EOF)
    putchar(ch);
printf (“₩nFinished printing the file ₩n”);
fclose(myFile);
```

Example 6

```
FILE *myInFile, *myOutFile;
char ch;

myInFile =fopen(" charstream.txt", "r");
if (myInFile==NULL) {
   printf("Could not open charstream.txt!₩n");
   exit(1);}
myOutFile =fopen(“samedata.txt", “w");
if (myOutFile==NULL) {
   printf("Could not open samedata.txt!₩n");
   exit(1);}

/* Read the input file one character at a time, and write
    the character to the output file.   */
while ((ch=fgetc(myInFile)) != EOF)
   fputc(ch, myOutFile);
fclose(myInFile);
fclose(myOutFile);
```

Exercise 1

- Create a file named “cars.txt”. Then write a C program that stores the following data to a struct array, then writes the data to a file named “cars.txt”

```
struct CAR {
    char name[20];
    int year;
} car[2];
```

```
car[0] = {“Avante”, 2007 }
car[1] = {“Sonata”, 2008 }
struct CAR car[2] = {{“Avante”, 2007 },
                     {“Sonata”, 2008 }};
```

expected output
“cars.txt”

```
Avante    2007
Sonata    2008
```

- After writing the file, to verify, open the file for “r”, and read the data in the file and print the data to the terminal.

Exercise 2

- Create a file named “cars.txt”. Then write a C program that stores the following data to a struct array, then writes the data to a file named “cars.txt”

```
struct CAR {
    char name[20];
    int year;
    double price;
} car[4];
```

```
car[0] = {“Avante”, 2007, 13000.00 };
car[1] = {“Sonata”, 2008, 18000.00 };
car[2] = {“SM7”, 2009, 22000.00 };
car[3] = {“Equus”, 2010, 35000.00 };
```

expected output “cars.txt”

```
------------------------------------
|Name      |Year      |Price       |
------------------------------------
|Avante    |      2007|    13000.00|
|Sonata    |      2008|    18000.00|
|SM7       |      2009|    22000.00|
|Equus     |      2010|    35000.00|
------------------------------------
```

- After writing the file, to verify, open the file for “r”, and read the data in the file and print the data to the terminal.

End of Class