Input/Output and Files

Introduction

- **Data files**
  - When you use a file to store data for use by a program, that file usually consists of text (alphanumeric data) and is therefore called a **text file.**
  - Can be created, updated, and processed by C programs
  - Are used for permanent storage of large amounts of data
    - Storage of data in variables and arrays is only temporary

Files and Streams

- C views each file as a sequence of bytes
  - File ends with the *end-of-file marker*
- Stream created when a file is opened
  - Provide communication channel between files and programs
  - Opening a file returns a pointer to a `FILE` structure
    - Example file pointers:
    - `stdin` - standard input (keyboard)
    - `stdout` - standard output (screen)
    - `stderr` - standard error (screen)

Files and Streams

- Read/Write functions in standard library
  - `fgetc`
    - Reads one character from a file
    - Takes a `FILE` pointer as an argument
    - `fgetc( stdin )` equivalent to `getchar()`
  - `fputc`
    - Writes one character to a file
    - Takes a `FILE` pointer and a character to write as an argument
    - `fputc( 'a', stdout )` equivalent to `putchar( 'a' )`
  - `fscanf` / `fprintf`
    - File processing equivalents of `scanf` and `printf`

Creating a Sequential File

- Creating a File
  - `FILE *myPtr;`
    - Creates a `FILE` pointer called `myPtr`
  - `myPtr = fopen("myFile.dat", openmode);`
    - Function `fopen` returns a `FILE` pointer to the file specified
    - Takes two arguments – file to open and file open mode
    - If open fails, `NULL` returned
  - `fprintf`
    - Used to print to a file
    - It is like printf, except first argument is a `FILE` pointer (pointer to the file you want to print in)

Creating a Sequential File

- Typical file open modes:

Mode	Description
`r`	Open a file for reading.
`w`	Create a file for writing. If the file already exists, discard the current contents.
`a`	Append; open or create a file for writing at end of file.

Creating a Sequential File

- `feof( FILE pointer )`
  - Returns true if end-of-file indicator (no more data to process) is set for the specified file
- `fclose( FILE pointer )`
  - Closes specified file
  - Performed automatically when program ends
  - Good practice to close files explicitly
- Details
  - Programs may process no files, one file, or many files
  - Each file must have a unique name and should have its own pointer


```c
/*
   Create a sequential file */
#include <stdio.h>

int main()
{
   int account;
   char name[ 30 ];
   double balance;
   FILE *cfPtr;   /* cfPtr = clients.dat file pointer */

   if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL )
      printf( "File could not be opened\n" );
   else {
      printf( "Enter the account, name, and balance.\n" );
      printf( "Enter EOF to end input.\n" );
      printf( "? " );
      scanf( "%d%s%lf", &account, name, &balance );

      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n",
                  account, name, balance );
         printf( "? " );
         scanf( "%d%s%lf", &account, name, &balance );
      }

      fclose( cfPtr );
   }

   return 0;
}
```

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
?
```

Program Output

Reading Data from a File

- Reading a sequential access file
  - Create a `FILE` pointer, link it to the file to read

```
myPtr = fopen( "myFile.dat", "r" );
```

  - Use `fscanf` to read from the file
    - Like `scanf`, except first argument is a `FILE` pointer

```
fscanf( myPtr, "%d%s%f", &myInt,
   &myString, &myFloat );
```

  - Data read from beginning to end


```
1
2  /* Reading and printing a sequential file */
3  #include <stdio.h>
4
5  int main()
6  {
7     int account;
8     char name[ 30 ];
9     double balance;
10    FILE *cfPtr;    /* cfPtr = clients.dat file pointer */
11
12    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL )
13       printf( "File could not be opened\n" );
14    else {
15       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
16       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
17
18       while ( !feof( cfPtr ) ) {
19          printf( "%-10d%-13s%7.2f\n", account, name, balance );
20          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
21       }
22
23       fclose( cfPtr );
24    }
25
26    return 0;
27 }
```

```
Account    Name          Balance
100        Jones           24.98
200        Doe            345.67
300        White            0.00
400        Stone          -42.16
500        Rich           224.62
```

Example: Merge two files

```
#include <stdio.h>
int main()
{   FILE *fileA,  /* first input file */
         *fileB,  /* second input file */
         *fileC; /*  output file to be created */
   int num1,  /* number to be read from first file */
       num2;  /* number to be read from second file */
   int f1, f2;

   /* Open files for processing */
   fileA = fopen("class1.txt","r");
   fileB = fopen("class2.txt","r");
   fileC = fopen("class.txt","w");
```