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

```

/* Reading and printing 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", "r" ) ) == NULL )
      printf( "File could not be opened\n" );
   else {
      printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
      fscanf( cfPtr, "%d%s%lf", &account, name, &balance );

      while ( !feof( cfPtr ) ) {
         printf( "%-10d%-13s%7.2f\n", account, name, balance );
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
      }

      fclose( cfPtr );
   }

   return 0;
}
```

```
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");
```

```
 /* As long as there are numbers in both files, read and
compare numbersone by one. Write the smaller number to the
output file and read the next number in the file from which
the smaller number is read. */

   f1 = fscanf(fileA, "%d", &num1);
   f2 = fscanf(fileB, "%d", &num2);

   while ((f1!=EOF) && (f2!=EOF)){
      if (num1 < num2){
         fprintf(fileC,"%d\n", num1);
         f1 = fscanf(fileA, "%d", &num1);
      }
      else if (num2 < num1) {
         fprintf(fileC,"%d\n", num2);
         f2 = fscanf(fileB, "%d", &num2);
      }
      else {  /* numbs are equal:read from both files */
         fprintf(fileC,"%d\n", num1);
         f1 = fscanf(fileA, "%d", &num1);
         f2 = fscanf(fileB, "%d", &num2);
      }
   }
```

```
   while (f1!=EOF){/* if reached end of second file, read
          the remaining numbers from first file and write to
          output file */
      fprintf(fileC,"%d\n", num1);
      f1 = fscanf(fileA, "%d", &num1);
   }
   while (f2!=EOF){ if reached the end of first file, read
          the remaining numbers from second file and write
          to output file */
      fprintf(fileC,"%d\n", num2);
      f2 = fscanf(fileB, "%d", &num2);
   }

   /* close files */
   fclose(fileA);
   fclose(fileB);
   fclose(fileC);
   return 0;
}  /* end of main */
```

Character I/O

- Suppose you want to store a set of quotations from Shakespeare in a file named `hamlet.txt.`

```
To be, or not to be: that is the question.
Whether ‘tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?
```

Text Files

- You might think of the file `hamlet.txt` as consisting of five lines. Internally however, text files are represented as a sequence of characters including the ‘\n’.

```
/* This program copies one file to another using
   character I/O
*/
#include <stdio.h>

void CopyFile(FILE *infile, FILE *outfile);
FILE *OpenUserFile(char *prompt, char *mode);

int main()
{
   FILE *infile, *outfile;

   printf("This program copies one file to another.\n");
   infile = OpenUserFile("Old file: ", "r");
   outfile = OpenUserFile("New file: ", "w");
   CopyFile(infile, outfile);
   fclose(infile);
   fclose(outfile);
}
```

```
void CopyFile(FILE *infile, FILE *outfile)
{
   int ch;
   while((ch = fgetc(infile)) != EOF) {
       fputc(ch, outfile);
   }
}
```

```
   FILE *OpenUserFile(char *prompt, char *mode)
   {
      char filename[20];
      FILE *result;
      int f = 1;

      while (f){
         printf("%s", prompt);
         scanf("%s", filename);
         result = fopen(filename, mode);
         if (result == NULL)
            printf("Can't open the file %s.\n",
                                        filename);
         else f = 0;
      }

      return result;
   }
```