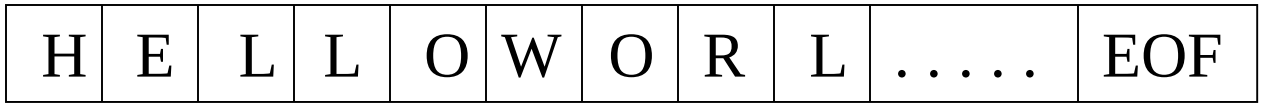
Lecture 17

File Processing

CSE115: Computing Concepts

Files and Streams

- C views a file as a sequential stream of bytes.
- A file ends with EOF (end-of-file) marker
- When a file is opened, a **stream** is associated with a file.
- Streams provide communication channels between files and the programs.



Files and Streams

- In addition to providing access to a file, a stream can also be used to access devices.
- For example, when a program (any program) is executed, 3 streams are automatically opened:
  - standard input (`stdin`)
    - enable the program to read data from keyboard
  - standard output (`stdout`)
    - enable the program to print data on the screen
  - standard error (`stderr`)
    - enable program to print errors on the screen
- They are all manipulated using file pointers.

Declaring a file

- The standard library <stdio.h> provides some of the file manipulation function.
- Declaring file:

```
FILE *the_file;
```

  - States that *the_file* is a pointer variable that points to a file structure
  - If there is more than one file, each file needs to have its own FILE pointer.

File operation: `fopen()`

- Before we can process a file, we must either open or create it.
- Syntax for fopen( ):

  file_pointer_name = fopen(file_name, OpenMode);

- The fopen() function takes 2 arguments: the file name and the file mode.

File operation: `fopen()`

- Example:

```
FILE *the_file;
the_file = fopen("my_file.txt","r");
```

  - This statement will try to open a file called “my_file.txt”.
  - my_file.txt is the file name referring to a phisycal file on the disk.
  - If the file is to be placed in a different directory than the program directory, the full path need to be specified. For example: “D:/my_file.txt”
  - The function returns a pointer to the successfully opened file. But if the file cannot be opened, a NULL is returned.

File OpenMode:

- The following is the list of open mode that can be associated with fopen( ).
  - **“r”** : to open existing file in input mode so data can be read from it. The file to be opened must exist physically on disk for this open mode to be successful.
  - **“w”** : the file is intended to be an output file and we are planning to write data to it.
  - **“a”** : to append data to the end of existing file.
  - **“r+”** : to open existing file for update (both in input and output mode). If the file already exist, the content is not destroyed.
  - **“w+”** : destroy the file if already exist and open a new file for update.
  - **“a+”** : open file for update (adding data at the end of the file.

fopen()

- You could also ask the users to specify the name of the file s/he want to open.
- Example:

```
char filename[50];
FILE *the_file;
printf("Enter file name: ");
gets(filename);

the_file = fopen(filename,"r");
```

- The users must enter the full filename, including the extension (.txt, .doc etc) and path.

File operation: failed `fopen()`

- If `fopen()` returns a NULL value, it means that `fopen()` has failed.
- This is due to any of the following reasons:
    - Opening a non-existing file for reading
    - Opening a file for reading or writing without having granted the appropriate access to the file by the operating system.
    - Opening a file for writing when no disk space is available.
- Therefore, in our program, we need to write statements that would handle this failure.

File operation: `fclose()`

- Syntax for fclose( ):

  fclose(file_pointer_name);

- Closes the file associated with *file_pointer_name*.

Example

```
#include <stdio.h>
int main()
{
    FILE *fp;

    fp = fopen("myFile.txt","r");
    if(fp == NULL)
        printf("The file is not there.");
    else
    {
        printf("File successfully opened.");
        fclose(fp);
    }
    return 0;
}
```

fgetc()

- `fgetc()` reads a character from a file.
- Syntax:

  variable = fgetc(file_pointer_name);
- Where:
  - variable is a character variable to hold the value returned by fgetc( ).
  - file_pointer_name is the pointer to an open file.

fgetc(): Example

```
#include <stdio.h>
int main(void)
{
    char ch;
    FILE *thefile;

    thefile = fopen(“myFile.txt","r");

    if (thefile == NULL)
    {
       printf("File failed to open\n");
    }
    else
    {
       ch = fgetc(thefile);/* read a character from the file */
       printf("%c",ch);/* print the value of ch*/
       fclose(thefile);
    }

    return 0;
}
```

`fputc()`

- `fputc()` writes a character value to the file.
- Syntax:
  - Using character variable, ch:

    fputc(ch, file_pointer_name);
  - Using character constant:

    fputc(‘a’, file_pointer_name);

fputc(): Example

```
#include <stdio.h>
int main(void)
{
    char ch = 'X';
    FILE *thefile;

    thefile = fopen("myFile.txt","w");

    if (thefile == NULL)
    {
       printf("File failed to open\n");
    }
    else
    {
       fputc(ch, thefile);/* write a character in the file */
       fclose(thefile);
    }

    return 0;
}
```

fscanf()

- `fscanf()` reads a value, or values of mix data type from a file.
- Syntax:

```
fscanf(file_pointer_name, formatControl, variableList);
```

- Example:

```
char name[50];
int age;
fscanf(theFile, "%s %d", name, &age);
```

fprintf()

- `fprintf()` prints values to the file
- Syntax:

```
fprintf(file_pointer_name, formatControl, variableList);
```

- Example 1:

```
fprintf(theFile, "This is an example");
```

- Example 2:

```
char name[50] = "Ahmad";
fprintf(theFile, "Name: %s", name);
```

- Example 3:

```
char str[10] = "Two";
int a = 2;
fprintf(theFile, "%d %s", a, str);
```

feof()

- `int feof(FILE *stream)` tests the end-of-file indicator for the given stream.
- Returns: a non-zero value when End-of-File indicator is found (e.g. when everything in the file has already been read), otherwise returns 0

- Example:

```
#include <stdio.h>
int main ()
{
  FILE *fp;
  int c;
  fp = fopen("file.txt","r");
  if(fp == NULL) {
    printf("Error in opening file"); return(-1);
  }
  while(1) {
    c = fgetc(fp);
    if( feof(fp) )
              break;
    printf("%c", c);
  }
  fclose(fp);
}
```

fgets()

- `fgets()` reads a string of characters from a file.

- Syntax:

```
fgets(stringVariable, size, file_pointer_name);
```

- Example:

```
char content[50];
fgets(content, 50, theFile);
```

- `fgets()` will read one line or ***size – 1*** characters (wchichever comes first) from the file pointed by *theFile*, and saves the string into variable *content*.

fgets() Example

```c
#include <stdio.h>
int main ()
{
  FILE *fp;
  int c;
  char content[50];

  fp = fopen("file.txt","r");
  if(fp == NULL) {
    printf("Error in opening file"); return(-1);
  }
  while(1) {
    fgets(content, 50, fp);//read a line
    if( feof(fp) )
           break;
    printf("%s", content);
  }
  fclose(fp);
}
```