Computer Programming Using C
COP 3275 - Summer 2017

Lecture 23: File I/O

By Ahmed E. Khaled

/* Programming */
Declaring a Structure Tag

• A *structure tag* is a name used to identify a *particular kind of structure*.

• The declaration of a structure tag named `part`:

```c
struct part {
    int number;
    char name[NAME_LEN+1];
    int on_hand;
};
```
Pass structure by value

```c
struct student {
    int id;
    char name[20];
};

void func(struct student record){
    printf(" Id is: %d \n", record.id);
    printf(" Name is: %s \n", record.name);
}

void main() {
    struct student record;
    record.id=1;
    strcpy(record.name, "Raju");

    func(record);
}
```
Pass structure by reference

```c
struct student {
    int id;
    char name[20];
};

void func(struct student *record) {
    printf(" Id is: %d \
", record->id);
    printf(" Name is: %s \
", record->name);
}

void main() {
    struct student record;
    record.id=1;
    strcpy(record.name, "Raju");

    func(&record);
}
```
File I/O

- The `<stdio.h>` header is the primary repository of input/output functions, including printf, scanf, putchar, getchar, puts, and gets.

- This library also provides functions that deals with files.

- In C, the term *stream* means any source of input or any destination for output.

- Programs obtain their input from keyboard or file and write their output to screen or file.

- Streams often represent files stored on various media.
File Pointers

- Accessing a file stream is done through a file pointer, which has type FILE *.
- The FILE type is declared in <stdio.h>.
- Certain streams are represented by file pointers with standard names.
- Additional file pointers can be declared as needed:
  FILE *fp1, *fp2;
Text Files versus Binary Files

- `<stdio.h>` supports two kinds of files: text and binary.
- The bytes in a *text file* represent characters, allowing humans to examine or edit the file.
- In a *binary file*, bytes don’t necessarily represent characters.
• Text files have two characteristics that binary files don’t possess.

• **Text files may contain a special “end-of-file” marker** (according to the OS).

• **Text files are divided into lines.**

• In a binary file, there are no end-of-line or end-of-file markers; all bytes are treated equally.
Opening a File

- Opening a file for use as a stream requires a call of the `fopen` function.
- Prototype for `fopen`:
  ```c
  FILE *fopen(const char * filename, const char * mode);
  ```
- `filename` is the name of the file to be opened.
- `mode` is a “mode string” that specifies what operations we intend to perform on the file.
• One way to call fopen is:
  
  `fopen("c:\\project\\test1.dat", "r")`

• An alternative is to use the / character:
  
  `fopen("c:/project/test1.dat", "r")`

• `fopen` returns a file pointer that the program save in a variable:
  
  `fp = fopen("in.dat", "r");`
  
  `/* opens in.dat for reading */`

• When it can’t open a file, `fopen` returns a null pointer.
Modes

- Factors determine which mode string to pass to `fopen`:
  - Which operations are to be performed on the file
  - Whether the file contains text or binary data

- Mode strings for text files:

<table>
<thead>
<tr>
<th>String</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;r&quot;</td>
<td>Open for reading</td>
</tr>
<tr>
<td>&quot;w&quot;</td>
<td>Open for writing</td>
</tr>
<tr>
<td>&quot;a&quot;</td>
<td>Open for appending</td>
</tr>
<tr>
<td>&quot;r+&quot;</td>
<td>Open for reading and writing, starting at beginning</td>
</tr>
<tr>
<td>&quot;w+&quot;</td>
<td>Open for reading and writing (truncate if file exists)</td>
</tr>
<tr>
<td>&quot;a+&quot;</td>
<td>Open for reading and writing (append if file exists)</td>
</tr>
</tbody>
</table>
Closing a File

• The `fclose` function allows a program to close a file that it’s no longer using.

• The argument to `fclose` must be a file pointer obtained from a call of `fopen`.

• `fclose` returns zero if the file was closed successfully, and returns the error code `EOF` otherwise (a macro defined in `<stdio.h>`).
The outline of a program that opens a file for reading:

```c
#include <stdio.h>
#define FILE_NAME "example.dat"

int main(void) {
    FILE *fp;

    fp = fopen(FILE_NAME, "r");

    if (fp == NULL) {  
        printf("Can't open %s\n", FILE_NAME);
        return -1;
    }

    fclose(fp);
    return 0;
}
```
#include <stdio.h>

void main() {
    char sentence[1000];
    int value;

    FILE *fptr;
    fptr = fopen("program.txt", "w");  /* opening file for writing*/

    printf("Enter a sentence:\n");
    scanf("%s", &sentence);
    printf("Enter integer value:\n");
    scanf("%d", &value);

    fprintf(fptr, "%s", sentence);  //write to file
    fprintf(fptr, "%d", value);  //write to file

    fclose(fptr);
}
#include <stdio.h>
void main() {
    FILE *fp;
    char str[60];

    fp = fopen("file.txt", "r"); /* opening file for reading */

    while( fgets (str, 60, fp)!=NULL ) {
        printf("%s", str);
    }
    fclose(fp);
    return(0);
}
Convert string to integer

#include <stdio.h>

void main() {
    char myarray[5] = "-110";
    int i;
    sscanf(myarray, "%d", &i);
    printf("%d", i);
}