CSE 115L: Programming Language I Lab (Section: 06)

Spring 2020

Lab-12 (File I/O)

A file is a container in computer storage devices used for storing data.
Types of Files: Text Files (**.txt** files) and Binary files (.bin files).

File Operations
In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

for file i/o, you need to keep track of the file being accessed and for that you need to use a file pointer, for example: **FILE *fp

File Opening (create and edit)

fp = fopen(“fileNameOrDirectory”, “mode”)
It opens the file if it exists otherwise it creates a new file with the given name.

Example:

```
FILE *fp = fopen(“Data.txt”, “w+”);
FILE *fptr;
fptr = fopen(“E:\\cprogram\\newprogram.txt", “w”);
FILE *bp = fopen("E:\\cprogram\\oldprogram.bin", "rb");
```

File Opening Modes
r: open for reading **w: open for writing (file need not exist)** **a: open for appending (file need not exist)** **r+: open for reading and writing, start at beginning** **w+: open for reading and writing (overwrite)** **a+: open for reading and writing (append if file exists)** **rb: open an existing file for reading in binary mode** **wb: create a file for writing in binary mode. If the file already exists, discard the current contents** **ab: append: Open o r create a file for writing at the end of the file in binary mode** **rb+: open an existing file for update (reading and writing) in binary mode** **wb+: create a file for update in binary mode. If the file already exists, discard the current contents** **ab+: append: Open or create a file for update in binary mode, content is written at the end of the file**

Append mode is used to append or add data to the existing data of file(if any). Hence, when you open a file in Append(a) mode, the cursor is positioned at the end of the present data in the file.

Text file (stores characters)	**Other File Functions**

int num =7;
char str = “Hello”;
FILE *fp = fopen(“data.txt”, “w+”);
 	
Writing data to a text file:
fprintf(fp, “%d %s”, num, str);
 	
Reading data from a text file:
fscanf(fp, “%d %s”, &num, str);
 	
Changing position:
fseek(fp, sizeof(int), SEEK_SET);
	feof(file pointer):
	detects end of file marker in a file
	
	fgets(char *str, int n, FILE *stream):
	read a string from file
	
	fputs(const char *str, FILE *stream):
	write a string of character on a file
	
	getc(file pointer):
	read a character from a file
	
	putc(char c, file pointer):
	Append

Example: File read and write	

#include<stdio.h>
 	
struct emp
{
  char name[10];
  int age;
};
 	
int main(void)
{
  struct emp e;
  FILE *p;
  p = fopen("one.txt", "w");
 	
  printf("Enter Name and Age: ");
  scanf("%s %d", e.name, &e.age);
  fprintf(p,"%s %d ", e.name, e.age);
  fclose(p);
	#include <stdlib.h>
	#include<stdio.h>
	
	struct emp
	{
	  char name[10];
	  int age;
	};
	
	int main(void)
	{
	  struct emp e;
	  FILE *q;
	
	  if ((q = fopen("one.txt","r")) == NULL)
	  {
	  printf("Error! opening file");
	  // Program exits if the file pointer returns NULL.

```
        return 0;
}
```

```
        exit(1);
        }

        do
        {
        fscanf(q,"%s %d ", e.name, &e.age);
        printf("%s %d\n", e.name, e.age);
        }while( !feof(q) );

        fclose(q);

        return 0;
}
```

Example: Read name and marks of n number of students and store them in a file.

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  char name[50];
  int marks, i, num;

  printf("Enter number of students: ");
  scanf("%d", &num);

  FILE *fptr;
  fptr = (fopen("student.txt", "w"));
  if(fptr == NULL){
        printf("Error!");
        exit(1);
  }

  for(i = 0; i < num; ++i){
        printf("For student%d\nEnter name: ", i+1);
        scanf("%s", name);

        printf("Enter marks: ");
        scanf("%d", &marks);

        fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
  }

  fclose(fptr);
  return 0;}
```

Binary file Read and Write	**Example**

int num =7;
char str = “Hello”;
FILE *fp = fopen(“data.txt”, “wb+”);
 	
Writing data to a binary file:
fwrite(&num, sizeof(int), 1, fp);
 	
num (1 object) will be written to fp file and it’ll need 4 bytes. The function will return a number of successfully written objects.
 	
Reading data from a binary file:
fread(&num, sizeof(int), 1, fp);
 	
The fwrite() and fread() functions take four arguments:
 	
1. address of data to be written in the disk
2. size of data to be written in the disk
3. number of such type of data
4. pointer to the file where you want to write
 	
Changing position:
Same as text file
	#include <stdlib.h>
	#include<stdio.h>
	
	struct emp
	{
	  char name[10];
	  int age;
	};
	
	int main(void)
	{
	  struct emp e;
	  FILE *p,*q;
	  p = fopen("two.bin", "ab");
	  printf("Enter Name and Age: ");
	  scanf("%s %d", e.name, &e.age);
	  fwrite(&e, sizeof(struct emp), 1, p);
	  fclose(p);
	
	  if ((q = fopen("two.bin","r")) == NULL)
	  {
	  printf("Error! opening file");
	  // Program exits if the file pointer returns NULL.
	  exit(1);
	  }
	
	  while((fread(&e, sizeof(struct emp), 1, q))!=0)
	  {
	  printf("%s %d \n", e.name, e.age);
	  }
	
	}

fseek()
The C library function fseek() sets the file position of the stream to the given offset.

fseek(fp, sizeof(int), SEEK_SET);

The above statement means that the current position in fp file stream is: SEEK_SET + 4 bytes

SEEK_SET: beginning of file,
SEEK_CUR: current position of the file pointer
SEEK_END: end of file

**This function returns zero if successful, or else it returns a non-zero value.

```
#include <stdio.h>

int main () {
   FILE *fp;

   fp = fopen("file.txt","w+");
   fputs("This is a class", fp);

   fseek( fp, 7, SEEK_SET );
   fputs(" C Programming Language", fp);
   fclose(fp);

   return(0);
}
```