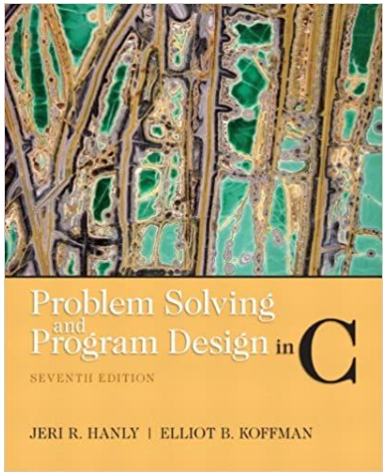
BIRZEIT UNIVERSITY



Faculty of Engineering and Technology
Department of Computer Science

Introduction to Computers and Programming (Comp 133)

References :
Book : Problem Solving and Program Design in C (7th Edition) 7th Edition
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Text and Binary File Processing

Chapter 11

Chapter 11

- Text and Binary File Processing
  - Text File

Text File

- Text file a named collection of characters saved in secondary storage. (e.g., on a disk)
- To mark the end of a text file, the computer places a special end-of-file character, which we will denote **EOF**

```
This is a text file!<newline>
It has two lines.<newline><eof>
```

- **input (output) stream** continuous stream of character codes representing textual input (or output) data (on disk)

```
This is a text file!<newline>It has two lines.<newline><eof>
```

Read Text File

```c
int main()
{
    /* Pointer to the file */
    FILE *fp1;
    /* Character variable to read the content of file */
    char c;

    /* Opening a file in r mode*/
    fp1= fopen ("newfile.txt", "r");

    /* Infinite loop -I have used break to come out of the loop*/
    while(1)
    {
        c = fgetc(fp1);
        if(c==EOF)
            break;
        else
            printf("%c", c);
    }
    fclose(fp1);
    return 0;
}
```

Write to Text File

```c
int main()
{
    char ch;
    FILE *fpw;
    fpw = fopen("C:\\test.txt","w");


    printf("Enter any character: ");
    scanf("%c",&ch);


    fprintf(fpw,"%c",ch);
    fclose(fpw);

    return 0;
}
```

Chapter 11

- Text and Binary File Processing
  - Binary File

Binary File

- **Binary file** :file containing **binary numbers (0,1)** that are the computer's internal representation of each file component.
- For example, the following code fragment creates a binary file named "nums.bin" , which contains the even integers from 2 to 500.

FIGURE 11.3 Creating a Binary File of Integers

```
1. FILE *binaryp;
2. int   i;
3.
4. binaryp = fopen("nums.bin", "wb");
5.
6. for  (i = 2; i <= 500; i += 2)
7.     fwrite(&i, sizeof (int), 1, binaryp);
8.
9. fclose(binaryp);
```

"wb" (write binary) for output files
"rb" (read binary) for input files.
"ab" Append to binary file.

Binary File

- The **fread** and **fwrite** use with Binary file instead of **fscanf** , **fprintf**
- **fwrite** requires four arguments:
  - Address of value to be written to a memory.
  - Size of each element.
  - Maximum number of elements to be written to the binary file
  - File pointer to a binary file opened in mode "**wb**" using function fopen
  - **fwrite(&i, sizeof (int), 1, binaryptr);** //write one integer to the binary file
  - **fwrite(score, sizeof (int), 10, binaryptr);** / write an array of 10 integers

Function Output
- Returns the number of elements written
- If return value is different than count, there was an error

Binary File

- Function **fread** also requires four arguments
  - Address of first memory cell to fill.
  - Size of each element to read value.
  - Maximum number of elements to copy from the file into memory.
  - File pointer to a binary file opened in mode "rb" using function fopen .

Function Output

- Returns number of elements read.
- If return value is different than count, there was an error or the end of the file was reached.

Creating a Binary File of Integers

```
FILE *binaryp;
int i;

binaryp=fopen("num.bin","wb");
for(i=2;i<500;i+=2)
    fwrite(&i,sizeof(int),1,binaryp);

fclose(binaryp);
```

sizeof operator used to finds the number of bytes used for storage of a data type.

Writing to a binary file

```c
#include <stdio.h>
#define SIZE 100
int main()
{
    int x=20,A[SIZE]={0,1,2,3};
    FILE* fptr_out=fopen("out.bin", "wb");

    fwrite(&x, sizeof(int),1,fptr_out);
    fwrite(A, sizeof(int),SIZE,fptr_out);
    fclose(fptr_out);
    return 0;
}
```

sizeof operator used to finds the number of bytes used for storage of a data type.

Reading from a binary file

```c
#include <stdio.h>
#define SIZE 100
int main()
{
    int x,A[SIZE];
    FILE* fptr_inp=fopen("in.bin", "rb");

    fread(&x, sizeof(int),1,fptr_inp);
    fread(A, sizeof(int),SIZE,fptr_inp);
    fclose(fptr_inp);
    return 0;
}
```

Binary File example

```c
struct rec
  {
      int x,y,z;
  };

  int main()
  {
      int counter;
      FILE *ptr_myfile;
      struct rec my_record;

      ptr_myfile=fopen("test.bin","wb");
      if (!ptr_myfile)
      {
          printf("Unable to open file!");
          return 1;
      }
      for ( counter=1; counter <= 10; counter++)
      {
          my_record.x= counter;
          fwrite(&my_record, sizeof(struct rec), 1, ptr_myfile);
      }
      fclose(ptr_myfile);
      return 0;
  }
```

Binary File example

```c
struct rec
    {
        int x,y,z;
    };

    int main()
    {
        int counter;
        FILE *ptr_myfile;
        struct rec my_record;

        ptr_myfile=fopen("test.bin","rb");
        if (!ptr_myfile)
        {
            printf("Unable to open file!");
            return 1;
        }
        for ( counter=1; counter <= 10; counter++)
        {
            fread(&my_record,sizeof(struct rec),1,ptr_myfile);
            printf("%d\n",my_record.x);
        }
        fclose(ptr_myfile);
        return 0;
    }
```

Text File Vs Binary File

Example	Text File I/O	Binary File I/O	Purpose
1	`plan_txt_inp = fopen("planets.txt", "r");` `doub_txt_inp = fopen("nums.txt", "r");`	`plan_bin_inp = fopen("planets.bin", "rb");` `doub_bin_inp = fopen("nums.bin", "rb");`	Open for input a file of planets and a file of numbers, saving file pointers for use in calls to input functions.
2	`plan_txt_outp = fopen("pl_out.txt", "w");` `doub_txt_outp = fopen("nm_out.txt", "w");`	`plan_bin_outp = fopen("pl_out.bin", "wb");` `doub_bin_outp = fopen("nm_out.bin", "wb");`	Open for output a file of planets and a file of numbers, saving file pointers for use in calls to output functions.
3	`fscanf(plan_txt_inp, "%s%lf%d%lf%lf", a_planet.name, &a_planet.diameter, &a_planet.moons, &a_planet.orbit_time, &a_planet.rotation_time);`	`fread(&a_planet, sizeof (planet_t), 1, plan_bin_inp);`	Copy one planet structure into memory from the data file.
4	`fprintf(plan_txt_outp, "%s %e %d %e %e", a_planet.name, a_planet.diameter, a_planet.moons, a_planet.orbit_time, a_planet.rotation_time);`	`fwrite(&a_planet, sizeof (planet_t), 1, plan_bin_outp);`	Write one planet structure to the output file.

(continued)

Text File Vs Binary File

Example	Text File I/O	Binary File I/O	Purpose

5	`for (i = 0; i < MAX; ++i)`
	`  fscanf(doub_txt_inp,`
	`  "%lf", &nums[i]);`
		`fread(nums, sizeof (double),`
		`  MAX, doub_bin_inp);`
			Fill array **nums** with type **double** values from input file.

6	`for (i = 0; i < MAX; ++i)`
	`  fprintf(doub_txt_outp,`
	`  "%e\n", nums[i]);`
		`fwrite(nums, sizeof (double),`
		`  MAX, doub_bin_outp);`
			Write contents of array **nums** to output file.

7	`n = 0;`
	`for (status =`
	`  fscanf(doub_txt_inp,`
	`  "%lf", &data);`
	`  status != EOF &&`
	`  n < MAX;`
	`  status =`
	`  fscanf(doub_txt_inp,`
	`  "%lf", &data))`
	`  nums[n++] = data;`
		`n = fread(nums,`
		`  sizeof (double),`
		`  MAX, doub_bin_inp);`
			Fill **nums** with data until **EOF** encountered, setting **n** to the number of values stored.

8	`fclose(plan_txt_inp);`
	`fclose(plan_txt_outp);`
	`fclose(doub_txt_inp);`
	`fclose(doub_txt_outp);`
		`fclose(plan_bin_inp);`
		`fclose(plan_bin_outp);`
		`fclose(doub_bin_inp);`
		`fclose(doub_bin_outp);`
			Close all input and output files.

Thank You.

جامعة بيرزيت
BIRZEIT UNIVERSITY