File I/O

Advanced C

File Input / Output

- Sequence of bytes
- Could be regular or binary file
- Why?
  - Persistent storage
  - Theoretically unlimited size
  - Flexibility of storing any type data

Advanced C

File Input / Output

Regular File

A regular file looks normal as it appears here.

regular files are generally group of ASCII characters hence the Sometimes called as text files which is human readable.

The binary file typically contains the raw data. The contents of a Binary file is not human readable

Binary File

^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^F^@^
@^@^P^@^@^@RåtdÔ<91>^Z^
@^@^Pw·^X^Aù¿^@^@^@^@ØÞø
¿0Éu·^@^@^@^@^@^@^@^@^@
^@^@^@^C^@^@^@^@^@^@^
@ðÈu·H^Aù¿&ìu·^X^Aù ¿^@^
@^@^@^D^@^@^@x^Xw·^@^@^
@^@^T^Aù¿^P^Aù¿Í<8d>u·ýßX·^L^
Fu·^P^Aù¿^O^Aù¿^@ÿø¿

EMERTXE

Advanced C

File Input / Output – Via Redirection

- General way for feeding and getting the output is using standard input (keyboard) and output (screen)
- By using redirection we can achieve it with files i.e

  ```
  ./a.out < input_file > output_file
  ```

- The above line feed the input from input_file and output to output_file
- The above might look useful, but its the part of the OS and the C doesn't work this way
- C has a general mechanism for reading and writing files, which is more flexible than redirection

Advanced C

File Input / Output

- C abstracts all file operations into operations on streams of bytes, which may be "input streams" or "output streams"
- No direct support for random-access data files
- To read from a record in the middle of a file, the programmer must create a stream, seek to the middle of the file, and then read bytes in sequence from the stream
- Let's discuss some commonly used file I/O functions

Advanced C

File IO - File Pointer

- stdio.h is used for file I/O library functions
- The data type for file operation generally is

Type

```
FILE *fp;
```

- FILE pointer, which will let the program keep track of the file being accessed
- Operations on the files can be
  - Open
  - File operations
  - Close

Advanced C

File IO - Functions - fopen() & fclose()

Prototype

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *stream);

Where mode are:
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 file)
a+ - open for reading and writing (append if file exists)
```

001_example.c

```
#include <stdio.h>

int main()
{
    FILE *fp;

    fp = fopen("test.txt", "r");
    fclose(fp);

    return 0;
}
```

EMERTXE

Advanced C

File IO - Functions - fopen() & fclose()

002_example.c

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

int main()
{
    FILE *input_fp;

    input_fp = fopen("text.txt", "r");

    if (input_fp == NULL)
    {
        return 1;
    }

    fclose(input_fp);

    return 0;
}
```

Advanced C

File IO - DIY

- Create a file named **text.txt** and add some content to it.
  - WAP to print its contents on standard output
  - WAP to copy its contents in **text_copy.txt**

Advanced C

File IO - Functions - feof()

003_example.c

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

int main()
{
    FILE *fptr;
    char ch;

    fptr = fopen("/etc/shells", "r");

    /* Need to do error checking on fopen() */

    while (ch = fgetc(fptr))
    {
        if (feof(fptr))
            break;

        fputc(ch, stdout);
    }

    fclose(fptr);

    return 0;
}
```

EMERTXE

Advanced C

File IO - Functions - ferror() and clearerr()

004_example.c

```c
#include <stdio.h>

int main()
{
    FILE *fptr;
    char ch;

    fptr = fopen("file.txt", "w");

    ch = fgetc(fptr);/* This should fail since reading a file in write mode*/
    if (ferror(fptr))
        fprintf(stderr, "Error in reading from file : file.txt\n");

    clearerr(fptr);

    /* This loop should be false since we cleared the error indicator */
    if (ferror(fptr))
        fprintf(stderr, "Error in reading from file : file.txt\n");

    fclose(fptr);

    return 0;
}
```

Advanced C

File IO - Functions - ftell()

005_example.c

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

int main()
{
    FILE *fptr;
    char ch;

    fptr = fopen("/etc/shells", "r");

    /* Need to do error checking on fopen() */

    printf("File offset is at -> %ld\n\n", ftell(fptr));
    printf("--> The content of file is <--\n");

    while ((ch = fgetc(fptr)) != EOF)
        fputc(ch, stdout);

    printf("\nFile offset is at -> %ld\n", ftell(fptr));

    fclose(fptr);

    return 0;
}
```

EMERTXE

Advanced C

File IO - DIY

- Create a file named **text.txt** and add “ab**c**dab**c**b**c**daab**c**”.
  - WAP program to find the occurrences of character 'c' using ftell()

Advanced C

File IO - Functions - fprintf(), fscanf() & rewind()

006_example.c

```c
#include <stdio.h>

int main()
{
    int num1, num2;
    float num3;
    char str[10], oper, ch;
    FILE *fptr;

    if ((fptr = fopen("text.txt", "w+")) == NULL)
    {
        fprintf(stderr, "Can't open input file text.txt!\n");
        return 1;
    }

    fprintf(fptr, "%d %c %d %s %f\n", 2, '+', 1, "is", 1.1);
    rewind(fptr);
    fscanf(fptr, "%d %c %d %s %f", &num1, &oper, &num2, str, &num3);

    printf("%d %c %d %s %f\n", num1, oper, num2, str, num3);

    fclose(fptr);

    return 0;
}
```

EMERTXE

Advanced C

File IO - Functions - fseek()

007_example.c

```c
#include <stdio.h>

int main()
{
    int num1, num2;
    float num3;
    char str[10], oper, ch;
    FILE *fptr;

    if ((fptr = fopen("text.txt", "w+")) == NULL)
    {
        fprintf(stderr, "Can't open input file text.txt!\n");
        return 1;
    }

    fprintf(fptr, "%d %c %d %s %f\n", 2, '+', 1, "is", 1.1);
    fseek(fptr, 0L, SEEK_SET);
    fscanf(fptr, "%d %c %d %s %f", &num1, &oper, &num2, str, &num3);

    printf("%d %c %d %s %f\n", num1, oper, num2, str, num3);

    fclose(fptr);

    return 0;
}
```

Advanced C

File IO - Functions - fwrite() & fread()

008_example.c

```c
#include <stdio.h>

int main()
{
    int num1, num2, num3, num4;
    FILE *fptr;

    if ((fptr = fopen("text.txt", "w+")) == NULL)
    {
        fprintf(stderr, "Can't open input file text.txt!\n"); return 1;
    }

    scanf("%d%d", &num1, &num2);

    fwrite(&num1, sizeof(num1), 1, fptr);
    fwrite(&num2, sizeof(num2), 1, fptr);
    rewind(fptr);
    fread(&num3, sizeof(num3), 1, fptr);
    fread(&num4, sizeof(num4), 1, fptr);

    printf("%d %d\n", num3, num4);

    fclose(fptr);

    return 0;
}
```

EMERTXE

Advanced C

File IO - Functions - fwrite() & fread()

009_example.c

```c
#include <stdio.h>

int main()
{
    struct Data d1 = {2, '+', 1, "is", 1.1};
    struct Data d2;
    FILE *fptr;

    if ((fptr = fopen("text.txt", "w+")) == NULL)
    {
        fprintf(stderr, "Can't open input file text.txt!\n");
        return 1;
    }

    fwrite(&d1, sizeof(d1), 1, fptr);
    rewind(fptr);
    fread(&d2, sizeof(d2), 1, fptr);

    printf("%d %c %d %s %f\n", d2.num1, d2.oper, d2.num2, d2.str, d2.num3);

    fclose(fptr);

    return 0;
}
```

```c
struct Data
{
    int num1;
    char oper;
    int num2;
    char str[10];
    float num3;
};
```

EMERTXE

Advanced C

File IO - DIY

- WAP to accept students record from user. Store all the data in as a binary file
- WAP to read out entries by the previous program

Screen Shot

```
user@user:~]./students_record_enrty.out
Enter the number of students : 2
Enter name of the student : Tingu
Enter P, C and M marks : 23 22 12
Enter name of the student : Pingu
Enter P, C and M marks : 98  87  87
user@user:~]./read_students_record.out
-------------------------------------------------------------
Name            Maths          Physics        Chemistry
-------------------------------------------------------------
Tingu              12               23               22
Pingu              87               98               87
-------------------------------------------------------------
Average         49.50            60.50            54.50
-------------------------------------------------------------
user@user:~]
```

EMERTXE