CS19001: Programming and Data Structures Laboratory

Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

http://cse.iitkgp.ac.in/~aritrah/course/lab/PDS/Autumn2018/CS19101_PDS-Lab_Autumn2018.html

12-Nov-2018

Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

Tutorial

File Handling and Command Line Arguments

Files

- Help in permanent storage of data in HDD
- Data stored as sequence of bytes, "logically" contiguous
- The last byte of a file contains the end-of-file character (EOF)
- Two types –
  - text (ASCII only, EOF terminated),
  - binary (Can have non-ASCII chars)
- Basic Operations – Open/Close, Read/Write

Files

```
FILE *fptr;
char filename[ ]= "file2.dat";
fptr = fopen (filename,"w");
if (fptr == NULL) {
      printf ("ERROR IN FILE CREATION");
      /* DO SOMETHING */
}
```


Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

File opening modes

The second argument of fopen is the mode in which we open the file.

- "r" : opens a file for reading (can only read). Error if the file does not already exists
- "r+" : allows write also
- "w" : creates a file for writing (can only write). Will create the file if it does not exist.
  Caution: writes over all previous contents if the file already exists
- "w+" : allows read also
- "a" : opens a file for appending (write at the end of the file)
- "a+" : allows read also

Writing to a file

```
FILE *fptr;
fptr = fopen ("file.dat","w");
fprintf (fptr, "Hello World!\n");
fprintf (fptr, "%d %d", a, b);
```

Reading from a file

```
FILE *fptr;
fptr = fopen ("input.dat", "r");
/* Check whether it is open */
if (fptr == NULL)
  {
     printf("Error in opening file \n");
     exit(-1);
  }
fscanf (fptr,"%d %d",&x, &y);
```

Reading from a file

```
char ch;
while (fscanf(fptr,"%c", &ch) != EOF)
{
    /* not end of file; read */
}
/* EOF checking in loop */
```

Reading lines from a file

```
FILE *fptr;
char str[1000];
/* Open file and check it is open */
while (fgets(str,1000,fptr) != NULL)
{
   printf ("Read line %s\n",str);
}
```

- Takes three parameters : a character array "str", maximum number of characters to read "size", and a file pointer "fp"
- Reads until any one of these happens - (i) number of characters read = size - 1, (ii) '\n' is read (char '\n' is added to "str"), (iii) *EOF* is reached or an error occurs
- '\0' added at end of "str" if no error. Returns *NULL* on error or *EOF*, otherwise returns pointer to "str"

Writing lines to a file

- use `fputs()`
- Takes two parameters : A string "str" (null terminated) and a file pointer "fp"
- Writes the string pointed to by "str" into the file
- Returns non-negative integer on success, *EOF* on error

Reading/Writing a character

```
//as declared in header
char fgetc(FILE *fp);

//as declared in header
int fputc(char c, FILE *fp);

//usage
char c;
c = fgetc(fp1);
fputc(c, fp2);
```

Moving File Pointers


```
//as declared in header
int fseek(FILE *stream, long int offset,
int whence)
```

The function takes the following three arguments and moves the file pointer an *offset* number of bytes from *whence*.

- stream: Pointer to a FILE object
- offset: Number of bytes offset from whence.
- whence: Constants that point to a position of the file
  1. SEEK_SET: Beginning of file
  2. SEEK_CUR: Current position of file
  3. SEEK_END: End of file

Moving File Pointers Example

Consider a file containing "abcdefghijklmnop"

```
fseek(fp,4,SEEK_SET);
fscanf(fp,"%c",&c);
printf("%c\n",c);
fseek(fp,2,SEEK_CUR);
fscanf(fp,"%c",&c);
printf("%c\n",c);
fseek(fp,-3,SEEK_CUR);
fscanf(fp,"%c",&c);
printf("%c\n",c);

//Output
e
h
f
```

Locate file pointer position


```
long ftell(FILE *stream);
```

The function obtains the current value of the position of the file pointer *stream*(in terms of number of bytes) with respect to the starting of the file.nsider the same file containing "abcdefghijklmnop"

```
fgets(string,10,fp);
printf("%ld", ftell(fp));
//Output
9
```

Closing a file

```
FILE *fptr;
char filename[]= "myfile.dat";
fptr = fopen (filename,"w");
fprintf (fptr,"Hello World of filing!\n");
... ... ... ... ...
fclose (fptr);
```

- Should close a file when no more read/write to a file is needed in the rest of the program

Command Line Arguments

```
int main (int argc, char *argv[]);
```

- User input:
  `./a.out s.dat d.dat`
- Effect:
  argc=3,
  argv[0] = ./a.out
  argv[1] = s.dat
  argv[2] = d.dat

Convert Argument string to usable data

```
int main(int argc, char *argv[ ]) {
  int i, n1, n2;
  printf("No. of arg is %d\n", argc);
  for (i=0; i<argc; ++i)
    printf("%s\n", argv[i]);
  sscanf(argv[1], "%d", &n1);
  sscanf(argv[2], "%d", &n2);
  printf("Sum is %d\n", n1+n2);
  return 0;
}
```

Execution:

```
$ ./a.out  32  54
No. of arg is 3
./a.out
32
54
Sum is 86
```

Bitwise Operators

Bitwise operators perform logical operations (OR, AND, XOR) at the bit-level. Operators include:

- '&' → bitwise-AND; '|' → bitwise-OR; '^'→ bitwise-XOR;
- '∼' → bitwise-NOT; '>>' → right-shift; '<<' → left-shift.

```
int a=7, b=9; //a:00000111, b:00001001
printf("%d", a|b)//Output is 15(00001111)
int a=0, b=1; //a:0000000, b:00000001
printf("%d\n", a&b) //Output is 0(00000000)
```

Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

Programming Assignments

Complete and submit during lab

Assignment 11: Binary Convolution Filter

Write a single C program which will parse command line arguments and decide accordingly to either create a file or process an existing file. The argument specification is as follows.

1. Creation of file: `./a.out create n filename.txt`
   Your program should open an empty file called "filename.txt" and write a randomly generated binary string of 0s and 1s of size/length n to that file.
2. Processing file: `./a.out process filename.txt n m`
   Your program should open an existing file called "filename.txt" that contains a binary string of length n, process that string on an element-by-element basis using a *convolution operation* and overwrite the existing string in the same file given as "filename.txt".
   The usage of m is described in future slides!

Assignment 11: Binary Convolution Filter

Convolution Operation: For every element b in the binary string, 4 neighouring elements are as follows:

- two elements immediately preceding b (say, e_1 and e_2)
- two elements immediately succeeding b (say, f_1 and f_2)

Perform a logical OR operation on these 5 values, i.e. $(e_1|e_2|b|f_1|f_2)$ and overwrite the value of b with the result in the file.

Note that, for elements residing at the boundaries of the string, only consider the neighbours of the element that are there in the string.

Example: Consider the binary string 000111000000. If we apply the operation for every element in the string, the final output is 011111111000.

Note that, for the element at position 1 which is 0 has one preceding neighbour 0 and two succeeding neighours 0 and 1. For this element, perform OR operation on these 4 elements only.

Soumyajit Dey, Aritra Hazra CSE, IIT Kharagpur

Assignment 11: Binary Convolution Filter

NOTE: You cannot read the entire string from the file in a character array of size n and perform this operation.
You are allowed to use a character array of size $m < n$ (given as command line argument) for storing a chunk of size m from the file.
Using this array, perform the following operations for every chunk in the file.

1. read a chunk of size m from the file
2. process m elements using the *convolution operation*
3. write back the modified chunk to the same file

Use **only one extra character array** of size m for storing the output of the convolution operation at each step and extra variables for obtaining neighbours of elements that were not a part of the chunk obtained.
Assume that, n is a multiple of m.

Assignment 11: Helper Functions

```
// convert a character to an integer
int ctoi(char c)
{
   int x;
   x = c - '0';
   return x;
}

// convert an integer to a character
int itoc(int i)
{
   char x;
   x = i + '0';
   return x;
}
```


Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

Illustrative Example

Say, $m = 10, n = 100$. The first 10 elements shall be read to a char array A="0000011010". We are going to perform the convolution and store the result in a output array B.

- you need to convert each character to an integer
- for the first element, consider only the next two neighbours, you have B[0]=A[0]|A[1]|A[2]=0|0|0=0
- Similarly, B[1]=A[0]|A[1]|A[2]|A[3]=0|0|0|0=0 (one preceding and two succeeding), B[2]=A[0]|A[1]|A[2]|A[3]|A[4]=0|0|0|0|0=0, B[3]=A[1]|A[2]|A[3]|A[4]|A[5]=0|0|0|0|0|1=1
- For the last two elements you need to separately read two more elements from the file, e_1, e_2. B[8]= A[6]|A[7]|A[8]|A[9]|e_1, B[9]= A[7]|A[8]|A[9]|e_1|e_2
- Continuing like this, B="0001111111". We now write back B to the file and then again read the next 10 elements to A and continue

Thank You