Command Line and File Handling

Command line argument

- Look at the following examples:

```
gcc first.c –o output          //command to compile the program first.c
cd C:\MinGW\bin                //command to change directory to bin
```

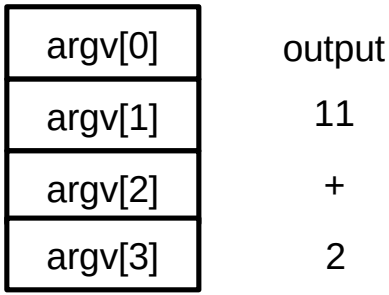
- The values after the command are command line arguments like “first.c”, “-o”, and “output” for first, while “C:\MinGW\bin” for second.
- You can think of command line argument as a way to provide input to a program (similar to scanf).

Contd.

void main(int argc, char *argv[]) //It is same as **argv

- argc: an integer that tells the number of arguments passed on the command line.
- argv: an array of strings, argv[i] is the ith argument as string.

output 11 + 2



Program

```
#include<stdio.h>
void main(int argc, char *argv[]) {
        if(argc>2)
                printf(“Too few args!\n”);
        else if(argc == 2)
                printf(“Hello %s\n”, argv[1]);
        else
                printf(“Too many args!\n”);
}
```

What about other types?

- Everything on command line is read as string!
- How do I convert string to int?
- Using stdlib.h library function atoi and atof
- atoi takes a string and converts to int
- atof takes a string and converts to double
- atol takes a string and converts to long integer

Add 2 Numbers

```
#include<stdio.h>
#include<stdlib.h>
void main(int argc, char * argv[]) {
        if(argc != 3)
                printf(“Bad args!\n”);
        else {
                int a = atoi(argv[1]);
                int b = atoi(argv[2]);
                printf(“%d\n”, a+b);
        }
}
```

File Handling

- File is a collection of bytes stored on a secondary storage device like hard disk.
- Following functions are used to interact with a file in C programming:
- fopen : Open file for reading/writing
- fscanf : Read from file
- fprintf : Write to file
- fclose : close the file after writing

fopen function

FILE * fopen (char *name, char *mode)

- returns a file pointer
- pointer points to an internal structure containing information about the file:
  - location of a file
  - the current position being read in the file, etc
- First argument is name of the file. (only file name or complete path)
- Second argument is the mode in which we want to open the file.

File modes

- r: read-only
- w: write at the beginning. May overwrite the current content. A new file is created if it does not exist.
- a: append or write at the end. File is created if it does not exist.
- r+: open a file for read and update. The file must be present or error.
- w+: write/read. Create an empty file or overwrite an existing one.
- a+: append/read. File is created if it does not exist. File is read from the beginning but written at the end.

fclose function

int fclose(FILE * fp)

- An open file must be closed after last use.

Program to display the content of a file

```
void main() {
        FILE *fp; char filename[128];
        scanf(“%s”, filename);
        fp = fopen( filename, “r”);
        if(fp == NULL) {
                fprintf(stderr, “Opening file %s failed\n”, filename);
                exit(1);
        }
        copy_file(fp, stdout);
        fclose(fp);
}
```

Contd.

```
void copy_file(FILE * fromfp, FILE * tofp) {
        char ch;
        while(fscanf( fromfp, "%c" , &ch ) == 1) {
                fprintf( tofp, "%c", ch);
        }
}
```

fscanf/scanf returns the number of characters read.

Program to copy content to another file

```
void main() {
        FILE *fp; char filename1[128], filename2[128];
        scanf(“%s”, filename1);
        scanf(“%s”, filename2);
        fp = fopen( filename1, “r”);
        fp1 = fopen( filename2, “w”);
        if(fp1 == NULL || fp2 == NULL) {
                fprintf(stderr, “Opening file %s failed\n”, filename);
                exit(1);
        }
        copy_file(fp1, fp2);
        fclose(fp1);
        fclose(fp2);
}
```

Assignment

- Describe the functions fseek and ftell. Write a program using both the functions.

ftell() and fseek()

Function ftell() returns the current position of the file pointer in a stream. The return value is 0 or a positive integer indicating the byte offset from the beginning of an open file. A return value of -1 indicates an error. Prototype of this function is as shown below:
long int ftell(FILE *fp);

Function fseek() positions the next I/O operation on an open stream to a new position relative to the current position.
int fseek(FILE *fp, long int offset, int origin);
Here fp is the file pointer of the stream on which I/O operations are carried on; offset is the number of bytes to skip over. The offset can be either positive or negative, denting forward or backward movement in the file. Origin is the position in the stream to which the offset is applied; this can be one of the following constants:
SEEK_SET : offset is relative to beginning of the file
SEEK_CUR : offset is relative to the current position in the file
SEEK_END : offset is relative to end of the file