



SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 125 - Introduction to Computing Science and Programming

Lab 02

Thanks to: Sepid (sepidh@sfu.ca)



- Passing arguments to `main()`
- Redirecting `stdin` and `stdout`
- Example: Reading letters from input and calculating the frequencies.



Passing arguments to main()

Until now :

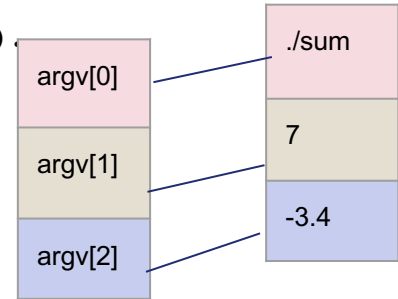
Our **main()** functions were not receiving any arguments.

But **main()** similar to other functions can receive arguments, just there is some notes to follow:

- **main()** function can receive two arguments.
- To do that, the **main()** function should be defined as **int main(int argc, char* argv[])**.
- **argv**: Argument vector, size argc+1
- let say our **main()** function is in **test.c**, we can pass arguments to it using :

```
gcc test.c -o test
```

```
./ test argv1 argv2 ... argvn
```



ex: `./sum 7 -3.4`



Passing arguments to main()

Let's clear thing up with an example:

Write a function that receives two integers as main function arguments and return sum of them.

```
#include <stdlib.h> // for atoi() and friends
#include <stdio.h>
```

```
int main( int argc, char* argv[] )
{
    printf("what is in argv[0]? %s\n",argv[0]);
    //The arguments will be type of char*
    // To convert them to int and float you cant use atoi and atof respectively.
    int sum=atoi(argv[1])+atoi(argv[2]);
    printf("First number : %s, second one : %s\n", argv[1], argv[2]);
    printf("sum is: %d. \n", sum);
    return sum;
}
```

```
1 #include <stdlib.h> // for atoi() and friends
2 #include <stdio.h>
3
4
5 int main( int argc, char* argv[] )
6 {
7     printf("what is in argv[0]? %s\n",argv[0]);
8     //The arguments will be type of char*
9     // To convert them to int and float you cant use atoi and atof respectively.
10    int sum=atoi(argv[1])+atoi(argv[2]);
11    printf("First number : %s, second one : %s\n", argv[1], argv[2]);
12    printf("sum is: %d. \n", sum);
13    return sum;
14 }
15
16
```

PROBLEMS 4 TERMINAL DEBUG CONSOLE

TERMINAL

```
(base) sepidh@cs-vml-42:~/Downloads/lab3$ gcc example1.c -o example
(base) sepidh@cs-vml-42:~/Downloads/lab3$ ./example 5 9
what is in argv[0]? ./example
First number : 5, second one : 9
sum is: 14.
(base) sepidh@cs-vml-42:~/Downloads/lab3$
```



Exercise 1:

Now write a function the receives three number and return the biggest one.

```
Answer is in the next slide!
```



Passing arguments to main()

Exercise 1- Solution:

```
#include <stdlib.h> // for atoi() and friends
#include <stdio.h>
int main( int argc, char* argv[] )
{
    float x= atof(argv[1]);
    float y= atof(argv[2]);
    float z= atof(argv[3]);
    float max = x;
    if (y > max)
        max = y;
    if (z > max)
        max = z;
    printf("max is: %f. \n", max);
    return 0;
}
```

```
1 #include <stdlib.h> // for atoi() and friends
2 #include <stdio.h>
3
4
5 int main( int argc, char* argv[] )
6 {
7     float x= atof(argv[1]);
8     float y= atof(argv[2]);
9     float z= atof(argv[3]);
10
11     float max = x;
12
13     if(y > max)
14         max = y;
15     if(z > max)
16         max = z;
17     printf("max is: %f. \n", max);
18     return 0;
19
20 }
21
22
```

PROBLEMS 4 TERMINAL DEBUG CONSOLE

✓ TERMINAL

```
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ gcc ex1.c
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ ./ex1 -9 1.
max is: 10.000000.
(base) sepidh@cs-vml-42:~/Downloads/Lab3$
```



Redirecting stdin and stdout

you can use `>` and `<` to direct your stdin and stdout.

for example:

`./hello > myfile.txt` will redirect your printf to myfile.txt

so if you have a program that do printf("Hello World"), instead of seeing output on terminal you can find it in `myfile.txt` .

similarly an other example: `./read_numbers <numbers.txt` will redirect stdin to `numbers.txt` and it will read inside the file .

```
1
2
3
4
```

`number.txt`

```
char str[256];

/* opening file for reading */
while( fgets(str, 256, stdin)!=NULL ) {
    /* writing content to stdout */
    printf("%s", str);
}
```

`read_numbers.c`

The program will read file `numbers.txt` and print out it line by line



Redirecting stdin and stdout

Lest clear thing up with an example:

Write a function that read a file and write it in the other file.

- First for making a file in linux there are several ways one of them is using cat.
 - for making a file and writing in it you can use **cat > filename.txt**
 - for reading that file you can use **cat filename.txt**

check this example:

After you do **cat >file.txt** you can write what you want in the file

for going next line in file you can press enter, to finish writing

press **ctrl+d**.

```
(base) sepidh@cs-vml-42:/local-scratch/localhome/sepidh/lab3$ ls
(base) sepidh@cs-vml-42:/local-scratch/localhome/sepidh/lab3$ cat >file.txt
Hi
My name is Sepid
to end the cat press ctrl+d
(base) sepidh@cs-vml-42:/local-scratch/localhome/sepidh/lab3$ ls
file.txt
(base) sepidh@cs-vml-42:/local-scratch/localhome/sepidh/lab3$ cat file.txt
Hi
My name is Sepid
to end the cat press ctrl+d
(base) sepidh@cs-vml-42:/local-scratch/localhome/sepidh/lab3$ █
```



Redirecting stdin and stdout

Let's clear things up with an example:

Write a function that reads a file and writes it to another file.

Now let's write a function that receives our file.txt and prints it line by line in file2.txt

```
#include <stdio.h>

int main () {
    // let assume is line has maximum 256 char
    char str[256];

    /* opening file for reading */
    while( fgets(str, 256, stdin)!=NULL ) {
        /* writing content to stdout */
        printf("%s", str);
    }

    return 0;
}
```

```
1  #include <stdio.h>
2
3  int main () {
4      // let assume is line has maximum 256 char
5      char str[256];
6
7      /* opening file for reading */
8      while( fgets(str, 256, stdin)!=NULL ) {
9          /* writing content to stdout */
10         printf("%s", str);
11     }
12
13     return 0;
14 }
```

PROBLEMS 50 TERMINAL DEBUG CONSOLE

✓ TERMINAL

```
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ cat file.txt
Hi
My name is Sepid
to end the cat press ctrl+D
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ gcc example2.c -o example2
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ ./example2 <file.txt >file2.txt
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ cat file2.txt
Hi
My name is Sepid
to end the cat press ctrl+D
(base) sepidh@cs-vml-42:~/Downloads/Lab3$ cat file.txt
Hi
My name is Sepid
to end the cat press ctrl+D
(base) sepidh@cs-vml-42:~/Downloads/Lab3$
```



Exercise 2:

Using cat make a file with integer inside called number.txt (picture 1). write a function that read those number add 1 to each and write it to the other file.

Answer is in the next slide!

```
(base) septh@cs-vml-42:~/Downloads/lab3$ cat > number.txt  
4  
5  
5  
7  
9  
(base) septh@cs-vml-42:~/Downloads/lab3$
```

picture 1



Redirecting stdin and stdout

Exercise 2_ Soloution:

```
#include <stdio.h>
#include <stdlib.h> // for atoi() and friends

int main () {
    // let assume is line has maximim 256 char
    char str[256];

    /* opening file for reading */
    while( fgets(str, 256, stdin)!=NULL ) {
        /* writing content to stdout */
        int number= atoi(str)+1;

        printf("%d\n", number);
    }
    return 0;
}
```

```
1  #include <stdio.h>
2  #include <stdlib.h> // for atoi() and friends
3
4  int main () {
5      // let assume is line has maximim 256 char
6      char str[256];
7
8      /* opening file for reading */
9      while( fgets(str, 256, stdin)!=NULL ) {
10         /* writing content to stdout */
11         int number= atoi(str)+1;
12
13         printf("%d\n", number);
14     }
15     return 0;
16 }
17
```

PROBLEMS 50 TERMINAL DEBUG CONSOLE

✓ TERMINAL

```
(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat number.txt
4
5
6
7
0
(base) sepidh@cs-vml-42:~/Downloads/lab3$ gcc ex2.c -o ex2
(base) sepidh@cs-vml-42:~/Downloads/lab3$ ./ex2 <number.txt >number2.txt
(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat number2.txt
5
6
7
8
1
(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat number.txt
4
5
6
7
0
(base) sepidh@cs-vml-42:~/Downloads/lab3$
```



Reading letters from input and calculating the frequencies.

```
(base) sepidh@cs-vnl-42:~/Downloads/lab3$ cat > myfile.txt
aabbcc(base) sepidh@cs-vnl-42:~/Downloads/lab3$ cat myfile.txt
aabbcc(base) sepidh@cs-vnl-42:~/Downloads/lab3$
```

Exercise 3:

1. Write a program that calculates the frequency of letter occurrences in text.
2. Read ASCII text from standard input.
3. On reaching EOF, print to stdout the normalized frequency of occurrence for each letter a-z that appeared in the input, one per line, in alphabetical order using the format produced by `printf("%c %.4f\n", letter, freq)`
4. You need to receive inputs from myfile.txt
5. use `cat > myfile.txt` to create the file and to end the file with EOF press `ctrl+d` same picture 1. (in the other word instead of pressing enter which save it as char in the file after you are done just hold ctrl and press d (you may need to press d two times)
6. Letters that occur zero times should not appear in the output.
7. Characters other than lower and upper case letters should be ignored.
8. Lower and upper case instances count as the same letter, e.g. 'a' and 'A' are both reported for the letter 'a' on the output.
9. The frequencies reported should sum to approximately 1 (with a little slack for accumulation of printf rounding errors).
10. By the way, you cannot implement this function by writing 26 "if" statements (1 for each letter). Hint: Each letter has numerical ASCII value. Can this numerical value be used at all?

picture 1

Answer is in the next slide!



Reading letters from input and calculating the frequencies.

Exercise 3_ Solution part 1, code:

```
1  #include <stdio.h>
2  #include <stdlib.h> // for atoi() and friends
3  #include <ctype.h> //for using to lower to lowercase all chars
4  int main () {
5      // let assume is line has maximim 256 char
6      char str[256];
7      // array of length 26 for saving the frequencies (float)
8      float freq[26]={0};
9      /* opening file for reading */
10     int i=0;
11     while(fgets(str, 256, stdin)!=NULL )
12     {
13
14         /* writing content to stdout */
15         while(str[i]!='\0')
16         {
17
18             //printf("%c\n",str[i]); //to check if we are reading correctly
19             char temp= tolower(str[i]); //lower casing the char
20             int index= (int)temp-'a'; //conver asci to int
21             // ascii will start from 'a'=97 https://www.rapidtables.com/code/text/ascii-table.html
22             freq[index]++; //increasing the entry in freq that is relates to read char
23             i++;
24         }
25     }
26     float length= i;
27     for (int k=0; k<26; k++)
28     {
29         if(freq[k]!=0)
30         {
31             float fr = freq[k]/length;
32
33             printf("%c : %f\n",(k+97), fr);
34         }
35     }
36     return 0;
37 }
```



Reading letters from input and calculating the frequencies.

Exercise 3_ Solution part2 , terminal:

```
PROBLEMS 50 TERMINAL DEBUG CONSOLE
> ✓ TERMINAL
(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat myfile.txt
aabbcc(base) sepidh@cs-vml-42:~/Downloads/lab3$ gcc example2.c -o example2
(base) sepidh@cs-vml-42:~/Downloads/lab3$ ./example2 <myfile.txt
a : 0.333333
b : 0.333333
c : 0.333333
(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat <myfile.txt
aabbcc(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat > myfile.txt
aadcf(base) sepidh@cs-vml-42:~/Downloads/lab3$ cat myfile.txt
aadcf(base) sepidh@cs-vml-42:~/Downloads/lab3$ ./example2 <myfile.txt
a : 0.400000
c : 0.200000
d : 0.200000
f : 0.200000
(base) sepidh@cs-vml-42:~/Downloads/lab3$
```



Exercise 3_ Solution part 3, code in text format:

```
#include <stdio.h>
#include <stdlib.h> // for atoi() and friends
#include <ctype.h> //for using to lower to lowercase all chars
int main () {
    // let assume is line has maximim 256 char
    char str[256];
    // array of length 26 for saving the frequencies (float)
    float freq[26]={0};
    /* opening file for reading */
    int i=0;
    while(fgets(str, 256, stdin)!=NULL )
    {
        /* writing content to stdout */
        while(str[i]!='\0')
        {
            //printf("%c\n",str[i]); //to check if we are reading correctly
            char temp= tolower(str[i]); //lower casing the char
            int index= (int)temp-97 ; //conver ascii to int
            // ascii will start from 97 https://www.rapidtables.com/code/text/ascii-table.html
            freq[index]++; //increasing the entry in freq that is relates to read char
            i++;
        }
    }
    float length= i;
    for (int k=0; k<26; k++)
    {
        if(freq[k]!=0)
        {
            float fr = freq[k]/length;
            printf("%c : %f\n", (k+97), fr);
        }
    }
    return 0;
}
```

