# Project 1 Warmup to C and Unix Programming

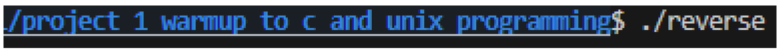
## Running the program:

1. Go to the directory where reverse.c is
2. Write ”gcc reverse.c -o reverse” which compiles the program
3. Then run ”./reverse”

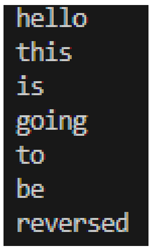
## Different inputs and outputs of the program:

### 1. ./reverse

If only the ./reverse program is run, this will give the user the ability to write into the stdin, for as many lines as they want adn when they press ctrl D, the program stops and reverses the input into stdout:

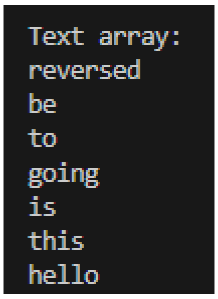


```
/project 1 warmup to c and unix programming$ ./reverse
```



```
hello
this
is
going
to
be
reversed
```

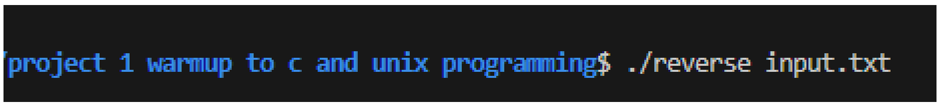
CTRL D



```
Text array:
reversed
be
to
going
is
this
hello
```

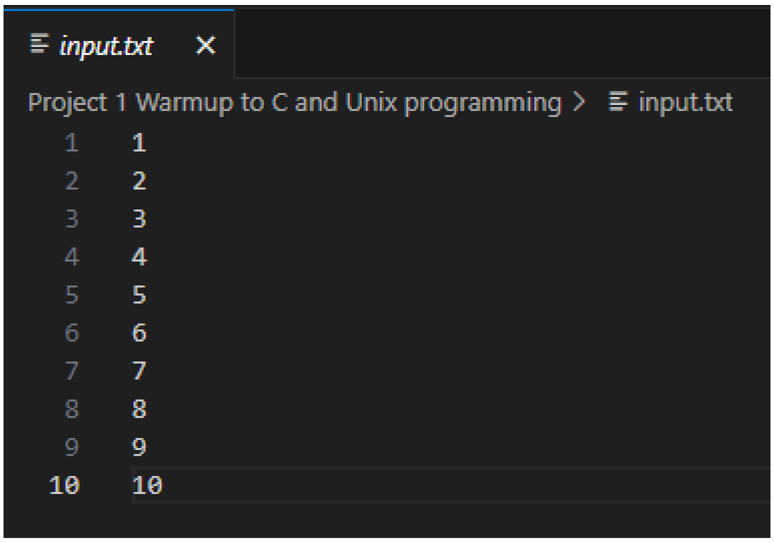
### 2. ./reverse [input.txt]

If the ./reverse program is given an extra argv, it interprets it as an input file. Then it reads the input file and reverses it to stdout:



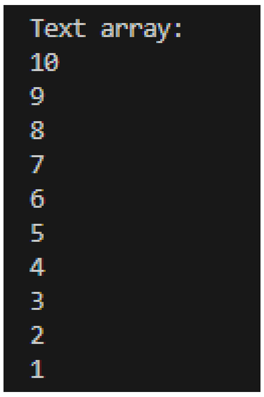
```
/project 1 warmup to c and unix programming$ ./reverse input.txt
```

Input:



```
1
2
3
4
5
6
7
8
9
10
```

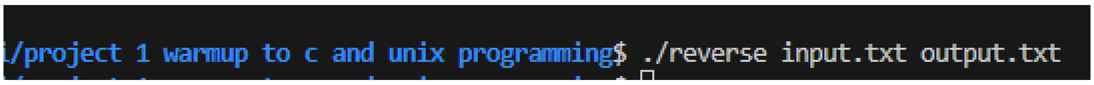
Output:



```
Text array:
10
9
8
7
6
5
4
3
2
1
```

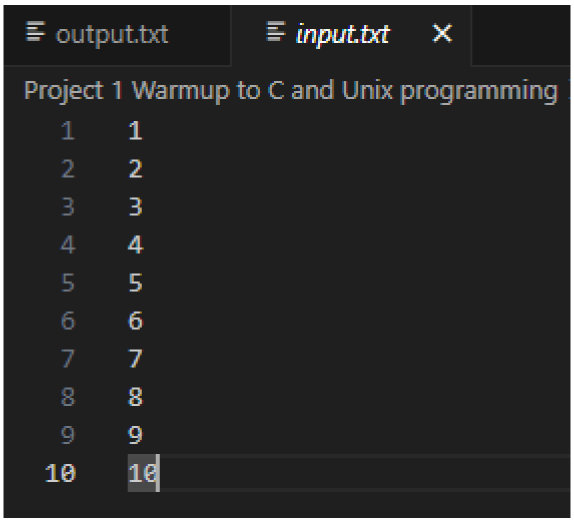
## 3. ./reverse [input.txt] [output.txt]

If the program is given two arguments in addition ot the command, it interprets the last given file as out put. So this time instead of stdout, it writes the reverse order into the output file:



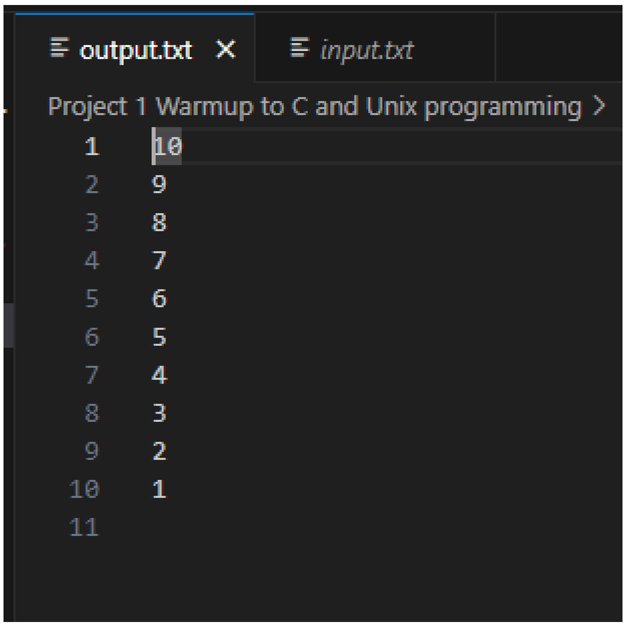
```
i/project 1 warmup to c and unix programming$ ./reverse input.txt output.txt
```

Input:



```
output.txt    input.txt ×
Project 1 Warmup to C and Unix programming
1
2
3
4
5
6
7
8
9
10
```

Output:



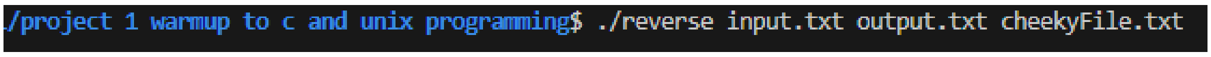
```
output.txt ×    input.txt
Project 1 Warmup to C and Unix programming >
10
9
8
7
6
5
4
3
2
1

```

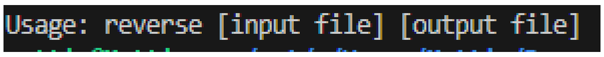
## 4. Error cases

I cannot test each error case like if malloc fails or if output file opening fails, but for the file inputs we can:

Too many arguments:

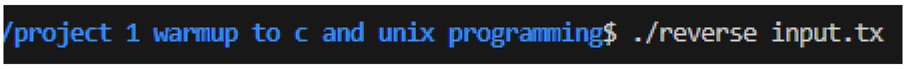


```
/project 1 warmup to c and unix programming$ ./reverse input.txt output.txt cheekyFile.txt
```

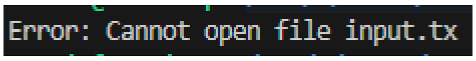


```
Usage: reverse [input file] [output file]
```

bad input file:



```
/project 1 warmup to c and unix programming$ ./reverse input.tx
```



```
Error: Cannot open file input.tx
```

## How the program works:

**Initializing the variables:**

```
FILE *rfile;

FILE *wfile;

char *line = NULL;

size_t len = 0;

ssize_t read;

int size = 5;

int count = 0;

char **text = (char **)malloc(size * sizeof(char *));
```

then it checks the different amounts of arguments that are given and assigns the rfile(read file) and wfile(write file) to different outputs and inputs based on the arguments given.

The basic principle of this program is that it reads lines from rfile and then adds each line to the array, and allocating more memory as needed, while keeping track of how many items have been added. Then we just based on the amount of items added to the array go backwards the array by using the variable that keeps track of the amount of items in the array. We write each item in the array to the chosen output, free the memory and exit the program.