

CSE 384 Section 1 (Spring 2020) Midterm instructor's solution

1. Define Input/output (I/O) redirection? Be sure your definition is complete! Describe why redirection is so useful? (25 points - full credit will depend on the completeness of your answer).

Solution:

In Linux (Unix) everything is considered a file.

Command line programs send output to a file called **standard output (stdout)** – connected to the screen by default).

Command line programs usually send error and status messages to another file called **standard error (stderr)** – also connected to screen by default).

Command line programs receive input from yet another called file **standard in (stdin)** – connected to the keyboard by default)

1. **stdin or file descriptor 0** – is file connected to the keyboard. Command line programs read input from this file.
2. **stdout or file descriptor 1** – is file connected to the display. Command programs send output to this file
3. **stderr or file descriptor 2** – is a file connected to the display. Programs send status/error messages to this file.

I/O redirection - allows us to alter the input source of shell command as well as where its output and error messages are sent to. This is made possible by the "<" and ">" redirection operators. Specifically, redirection allows us to change where output (**stdout/stderr**) goes (e.g. the screen or another file), and where input (**stdin**) comes from (e.g. the keyboard or another file).

Examples:

```
ls -l /usr/bin > ls-output.txt - redirects stdout to file "ls-output.txt"
```

```
ls -l /bin/usr 2> ls-error.txt - redirects stderr to file "ls-error.txt"
```

```
cat < lazy_dog.txt - redirect stdin to come from text file "lazy_dog.txt"
```

The most useful and powerful thing you can do with I/O redirection is to **connect multiple commands together with what are called pipelines**. With pipelines, the standard output of one command is fed into the standard input of another. This is accomplished with the pipe (|) operator.

Example:

```
ls -l /usr/bin | less - redirect stdout of the ls program to
stdin of less program
```

References:

<https://mwcory79.github.io/MikeCorley/presentations/TLCL-19.01.pdf#page=78>

<https://mwcory79.github.io/MikeCorley/presentations/TLCL-19.01.pdf#page=85>

http://linuxcommand.org/lc3_lts0070.php

2. Assume the following files exist in your home directory: “foo.txt” and “bar.txt”

a. Write a Linux command make “foo.txt” readable and writable for the owner, and only readable for everyone else. (12.5 points)

b. Nine characters of the file attributes (called the file mode), represent the read, write, and execute permissions for the file's owner, the file's group owner, and everybody else (12.5 points)

Owner	Group	World
rWX	rWX	rWX

Internally these file mode attributes are represented by 9 binary bits.

For example: the permissions attributes: “rwxrwxrwx” = 111111111_2 = (binary) and 777_8 (octal)

The permission attributes for “bar.txt” = rw-r-----

Please write the 9 binary permission bits (owner/group/world) for "bar.txt" - - - - -

Solution: (25 total points): a. 12.5, b. 12.5

a. Octal notation: **chmod 644 foo.txt** or Symbolic notation: **chmod u+rw,go+r foo.txt**

b. **110100000₂**

3. Use the Linux (Unix) *find* program and the word count program (*wc*) to write a command to find and count all of the files in your home directory (and the sub directories) that in end with “.c” The output should be the count of the files found. (25 points)

Solution: `find ~ -type f -name "*.c" | wc -l`

4. Complete the function: *void AppendToEndOfList(int list[], int valToInsert, int* len, int max)* shown below (25 points)

- a. accepts an integer *valToinsert*. If space exists in the array *list* (*max is the capacity*) then *valToInsert* is inserted to the back of *list*, and *len* is incremented. If the *max* number of elements is reached (i.e. capacity) then don't insert, instead print an error message on the standard error stream and return
- b. what is the output of the program if run with the following command line:

```
gcc q4a.c && ./a.out 11
```

Solution:

```
mwcory@mwcorley-VirtualBox: ~/Desktop/cse384/midterm
File Edit View Search Terminal Help
#include <stdio.h>
#include <stdlib.h>

void AppendToEndOfList(int list[], int valToInsert, int* len, int max)
{
    if(*len < max)
    {
        list[*len] = valToInsert;
        (*len)++; // need parans becuae ++ has great precedence than *
    }
    else
        fprintf(stderr, "Sorry! capacity reached\n");
}

int main(int argc, char* argv[])
{
    const int MAX_SIZE = 10;
    int list[MAX_SIZE];
    int count_items = 0;

    int num = atoi(argv[1]);

    for(int i = 0; i < num; ++i)
    {
        AppendToEndOfList(list, i+1, &count_items, MAX_SIZE);
    }

    for(int j = 0; j < count_items; ++j)
    {
        printf("%d ", list[j]);
    }
    printf("\n");

    return 0;
}
1,1 Top

mwcory@mwcorley-VirtualBox: ~/Desktop/cse384/midterm
File Edit View Search Terminal Help
mwcorley@mwcorley-VirtualBox:~/Desktop/cse384/midterm$ gcc q4a_solution.c && ./a.out 11
Sorry! capacity reached
1 2 3 4 5 6 7 8 9 10
```