CS101 Lab-4

Today's Linux Commands

- Work in a directory named `~/lab4g1`
- We've learned how to redirect input from a file (`<`) and output to a file (`>`). The next step is to connect commands together using a pipe (`|`).
    a. A pipe takes the standard output of the command on its left and uses it as the standard input for the command on its right.
    b. We can chain many commands using pipes
    c. Syntax: `command1 | command2`
- `grep` is searches for lines that contain a specific pattern of text
    a. Find all occurrences of "`include`" in `hello.c`
        - `grep "include" hello.c`
    b. List all files/directories in your home directory (`~`) and then filter that list to show only the ones with "`lab`" in their name.
        - `ls ~ | grep "lab"`
- `wc` command counts the number of lines, words, and characters in its input
    a. Count lines, words, chars in `hello.c`
        - `wc hello.c`
    b. Count the number of C files in the current directory
        - `ls *.c | wc -l`
    c. In all `.c` files in the current directory, find every line that uses the `printf` function and count how many such lines exist.
        - `cat *.c | grep "printf" | wc -l`

Instructions for the lab assignment:

We do not have resources to grade a question everyday for you all. We'll take a round and see how everyone is doing. We'll do formal grading later on in the semester once you have experience writing a decent number of C programs. Solve the first two problems in the lab, and solve other problems as practice whenever you have time.

1. Write a program to take input of an expression two signed integers and an operator: `a c b`. Here, `c` is an operator in `{+, -, *, /, % }` and `a` and `b` are integers with absolute values less than `1000`. The program should then evaluate the expression and print the result. If the user wants to divide by `0`, you should print an error message. In the last lab, you likely used if-else statements or else-if ladder. In this lab, use the `switch` expression. For checking absolute values, use `abs()` from `math.h`. To compile without warning, you might also need to include `stdlib.h`.

2. Copy the program for Problem-1 to a new file. Edit it so it accepts input as follows: first, it reads the number of expressions, say, `num_expressions`. It then reads that many expressions, each separated by a new line. Use `for` loop to achieve this. Then, use input redirection to read the input from `input.txt`. After writing the code, verify `while(num_expressions--)` instead of `for` also works. Convince yourself that both the approaches are identical. Sample `input.txt` file contents:

   3
   45 / 56
   12 - 21
   9 % 0

3. Build a countdown launch timer. Read the timer duration in seconds from the user and save it as int, say, `time_to_launch`. For input `time_to_launch = 10`, the program should print "`T-minus 10... 9... 8... 7... 6... 5... 4... 3... 2... 1... BLASTOFF!`"

4. Copy the program for Problem-3 to a new file. Modify it so as to include time units. Assume that `time_to_launch` is in seconds. While printing output, convert the time in **HH:MM:SS** format. For example, when `time_to_launch = 1000000`, countdown should start from T-minus 27:46:40 and go all the way to 00:00:00. Use output redirection to save output to `output.txt`. Verify that your output has exactly `1000000` lines and `2000000` colons when `time_to_launch = 1000000`.
   Hint: use %02d for printing as discussed in the class.

Solution for Lab-3 Problem 1:

Use proper indentations, comments, and descriptive naming of variables.

```c
#include<stdio.h>
#include<math.h>

int main() {
    int a, b, ans;
    char c;

    //enter an expression like 3 + 4 and press enter
    //note that 3+4 and pressing enter also works
    scanf("%d %c %d", &a, &c, &b);

    //to use abs() function, need to include math.h
    //to prevent compiler warning, need to include stdlib.h
    if(abs(a) > 1000 || abs(b) > 1000) {
        //preventing large input values avoids integer overflow errors
        //1000*1000 = 1,000,000 which is within the range of int
        printf("Input numbers must be between -1000 and 1000!\n");
        return 1;
    }

    if(c == '+') {
        ans = a + b;
    } else if(c == '-') {
        ans = a - b;
    } else if(c == '*') {
        ans = a * b;
    } else if(c == '/') {
        if(b != 0) {
            ans = a / b;
        } else {
            printf("Division by zero error!\n");
            return 1;
        }
    } else if(c == '%') {
        if(b != 0) {
            ans = a % b;
        } else {
            printf("Division by zero error!\n");
            return 1;
        }
```

```
    } else {
        printf("Invalid operator!\n");
        return 1;
    }
    printf("%d %c %d = %d\n", a, c, b, ans);
    return 0;
}
```