CS205 C/ C++ Programming - Lab Assignment

Name: 张佳晨(Jiachen ZHANG)

SID: 11713020

Part 1 - Analysis

> Firstly, I have to save the commands which the system should have.
>
> Secondly, I need to read the commands user input.
>
> Thirdly, I will compare the commands user input with that system has already had.

> I have used the functions in `string.h` and `math.h` and `goto` for a quick restart of the program.

Part 2 - Code

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define START_CMD               0
#define STOP_CMD                1
#define RESTART_CMD             2
#define STATUS                  3
#define EXIT                    4

int main(){
        printf(">");
        char *commands[] = {"start", "stop", "restart", "status", "exit" };
        int i = -1;
        char command[10];
        while(1){
                fflush(stdin);
                /* Prevent overflow */
                char c;
                int num = 0;
                /* Skip all previous Spaces */
                c =  getchar();
                while(c == ' ')
                        c =  getchar();
                /* Save the character from the first non-space */
                command[0] = c;
                c =  getchar();

                while((c !=' ')&&(c !='\n')){
                        num++;
                        command[num] = c;
                        /* The length of the longest command "restart" is only 7,
      and if it can be read later, it is treated as a invalid information */
                        if(num > 6){
                                printf("Invalid command\n>");
                                goto out;
                        }
                        c =  getchar();
                }
                /* Add '\0' to the ending to change it into a string */
                command[num+1] = '\0';
                /* Iterate to see if the input command exists */
                for(i=0; i<5; i++){
                        if(strcmp(command,*(commands+i))==0) {
                                break;
                        }
                }
                switch(i){
                        case START_CMD:
                        case STOP_CMD:
                        case RESTART_CMD:
                        case STATUS:
                                printf("command %s recognized\n>", commands[i]);
                                break;
                        case EXIT:
                                return 4;
                        default:
                                printf("Invalid command\n>");
                                break;
                }
                out:;
        }
        return 0;
}
```

Part 3 - Result & Verification

Test case #1:

```
Input:
sssssss
Output:
Invalid command
```

Test case #2:

```
Input:
start
Output:
command start recognized
```

Test case #3:

```
Input:
exit
Output:
```

Part 4 - Difficulties & Solutions

In order to prevent some kinds of invalid information, I chose to use `getchar()` instead of `fget()` or `fscanf()`.

> `fscanf()` cannot regard **start(several spaces)wrong** as a invalid input.

In order to have a quick restart of the program, I use `goto` to simplify my codes.

> As the longest command "restart" has 7 letters, when the program read the 8th letter, we can regard this command as a invalid command and prompt the user to reput.

For legibility, I associate a symbol to each index and use these codes.

```
char *commands[] = {"start", "stop", "restart", "status", "exit" };
switch(){

}
#define START_CMD    0
```