
LAB 7

CS 361: Systems Programming / Spring 2023

Description

In this lab session, you will:

- Explore signals
- Use function `strstr` to break strings into substrings
- Use the `scanf` family of functions to parse text

Please read this document carefully and follow the instructions on the last section to complete this lab session. When you answered all the questions, please show your work to the TA.

Guide

1. Accept the invitation for Lab 7 on Github classroom: <https://classroom.github.com/a/vYw-lOcE>
2. Import the Github repository created to your machine using vscode, as explained in Assignment 0
3. Make sure that you can launch a terminal inside vscode via menus: Terminal > New Terminal
4. Read this guide and answer the questions as they appear. You should answer a total of 12 questions.

Running each program

In this lab, you will use the command `make` to compile all programs, just as in earlier labs. You can run each program directly (e.g., using command `./lab7-1`)

Signals

Sending signals to other processes

In Linux, you can use command `kill` to send signals to any process you own. Command `kill` takes the following arguments:

```
kill -s <SIGNAL> <PID>
```

You can list all the signals supported with command `kill -l`

Question 1: Read the description about command `kill`. What are all the signals that `kill` supports?

Question 2: Inspect program `lab7-1`. Compile and run program it in a terminal. Open a second terminal. How can you use command `kill` so send signal `INT` to the running process? Make sure the command works by opening the original terminal and making sure the process is not running.

Question 3: Inspect program `lab7-1`. Compile and run program it in a terminal. Open a second terminal. How can you use command `kill` so send signal `KILL` to the running process? Make sure the command works by opening the original terminal and making sure the process is not running.

Handling signals

You can handle signals using function `signal`, with the signature defined below:

```
typedef void (*sighandler_t)(int);  
sighandler_t signal(int signum, sighandler_t handler);
```

Function `signal` takes two arguments:

- `signum`: the number of the signal to handle. This number is defined with `SIG` macros, with the same names as you answered in Question 1. For instance, for signal `SEGV` there is a macro `SIGSEGV`.
- `handler`: a function that will execute when the signal is delivered. The function returns void and takes a single integer as the argument.

Question 4. Read the description about handling signals. Use the information in this document to do the following tasks. Inspect the contents of file `lab7-2`. Write a handler function to break the loop. Use function `signal` to register that function with signal `INT`. Compile and run `lab7-2`. On another terminal, use the command `kill` to send the signal `INT`. What happens?

Question 5. Compile and run `lab7-2`. Press `CTRL+C`. Can you explain what happened?

Question 6. Read the description about handling signals. Inspect the contents of file `lab7-2`. Use function `signal` to register that function with signal `KILL`. Compile and run `lab7-2`. On another terminal, use the command `kill` to send the signal `KILL`. What happens?

strstr

Function `strstr` finds the first occurrence of a substring inside another NULL-terminated string, and returns a pointer to that substring. The signature of function `strstr` is as follows:

```
char *strstr(const char *haystack, const char *needle);
```

The arguments are:

- `haystack`: the string where you want to find the substring
- `needle`: the substring you are looking for

Function `strstr` returns a pointer to inside `haystack` where `needle` starts, or NULL if it cannot find `needle`.

Question 7: Read the description about function `strstr`. Use the information in this document to do the following tasks. Inspect file `lab7-3.c`. Fill the array with the character 'a'. Note that this string is not NULL-terminated. What is the result of using `strstr` to find the substring "bb"?

Question 8: Change file `lab7-3.c` so that the string is NULL-terminated. What is the result of using `strstr` to find the substring "bb" ?

Question 9: Change file `lab7-3.c` so there is a "bb" substring in the middle. What is the result of using `strstr` to find the substring "bb" ?

scanf

Function `scanf` can parse a string from the standard input into its constituent parts. It takes a format string, similar to `printf`; and pointers to memory where to store the results

Function `sscanf` is similar to `scanf`, but parses the results on a string in memory.

Both functions return the number of items matched. If nothing is matched, they return zero.

For instance, the following program:

```
char string[] = "This is CS 361";
int number;
int matched = sscanf(string, "This is CS %d", &number);
int notmatched = sscanf(string, "This is not CS %d", &number);
```

Stores the value 361 in variable `number`, saves the value 1 in `matched`, and the value 0 on `notmatched`.

The format string `%s` allows you to recognize a single unbroken word, and takes a buffer to save that word.

Note that `sscanf` does not terminate the string with a NULL character, so it's a good idea to always set the contents of the buffer to zero before using `sscanf`:

```
char string[] = "This is CS THREE_SIX_ONE";
char buffer[20];
int number;
memset(buffer, 0, 20);
int matched = sscanf(string, "This is CS %s", &buffer);
printf("%s\n", buffer);
```

Question 10: Use the information in this document to do the following tasks. Inspect file `lab7-4`. Use function `sscanf` instead of `strcmp` so that the program outputs "CS <NUMBER>" for any number provided.

Question 11: What happens if you pass a name (not a number) to `lab7-4`?

Question 12: Change `lab7-4` so that it accepts any single word and outputs "CS <WORD>" (e.g., "CS THREE_SIX_ONE")

Extra / Optional

- Notice the following code from Assignment 2:

```
char * sep;

if((sep=strstr(c,"<"))){
    *sep=0;
    char * filein = sep+1;
    ...
}
```

Can you explain what it does?

- What happens after question 12 if you input a really long string (i.e., larger than 256 bytes)? Can you fix this issue?

Grading

Show your UIC card to the TA when you enter the lab, or type your UIN on the chat when joining remotely. Stay in the session until you show your work, or until the TA announces that the lab is over.

- You have to remain present for the whole lab to get attendance, which you can then use to resubmit Assignment 4.
- You can leave early after showing your work to the TA (answers to all questions). In this case, you will get a 5% bonus in Assignment 4.