Lab 1

Computer Architecture I ShanghaiTech University
Lab 1 Lab 2

Goals

- Setup your Linux.
- Exercise number representations.
- Learn some simple C programming.

Reading

- P&H: 2.4

Policies and Partners

You are **REQUIRED** to have a partner for lab checkoffs. This will reduce the number of check-offs we have to perform (allowing us to answer more of your questions) as well as give you someone to discuss class material with. **BOTH** partners will need to be present at check-off to receive credit and both partners will be asked to participate during the check-off. Try your best to find someone in your lab section with similar work habits as yourself.

How Checkoffs Work - on Campus

You'll notice that at the end of (almost) every exercise, there is a section labelled "Check-off." The items in this section are what you must successfully demonstrate to your TA in order to receive credit for completing the lab. Once you and your partner finish **ALL of the exercises**, you should put your names AND ShanghaiTech email address on the checkoff list on the board, and a TA will come and check you off.

How Checkoffs Work - online

You will make an appointment with your Lab TA and your lab partner to be checked off online. This appointment should be during (or close to) the actual lab time. You will then make a zoom call with all 3 persons. The students share their screen and then discuss and check off all items in the lab.

Labs are graded out of x points, which will evaluate to 100%. A lab is considered on-time if it is turned in within a week of the lab session in which it is initially assigned to you. For example, the lab assigned to you in this weeks lab is this document, lab 1. Thus, the latest you may get lab 1 checked off is the beginning of your lab next week.

Exercises

Exercise 1: Have a 64bit Linux installed on your Laptop

As CS students you should have a Linux (e.g. Ubuntu) installed on your Laptop. It is strongly suggested to have it as dual boot, but using a VM is also ok. Using macOS will not work for all of the homeworks! The autolab runs Linux (Ubuntu).

Show your TA the working Linux with gcc installed.

Exercise 2: Autolab

Show the TA your autolab account and that you finished HW1 (or working on it at least). Setup your Pinyin name in Autolab.

Show your TA your autolab account and that you set your name there to your Pinyin name.

Exercise 3: Binary Alphabet

At this point, you may begin to work with a partner.

Let's take 4-bit numbers. If we stack five 4-bit numbers on top of each other in binary, we can make patterns and pictures! To help visualize this, you can think of 0's and 1's as white and black instead. For example, what does the following bit pattern look like?

```
0  1  1  0      □ ■ ■ □
1  0  0  1      ■ □ □ ■
1  1  1  1 -->  ■ ■ ■ ■
1  0  0  1      ■ □ □ ■
1  0  0  1      ■ □ □ ■
```

Checkoff:

- What five decimal digits produce the pattern above? What five hexadecimal digits?

- What letter is drawn with 1,1,9,9,6? 0xF8F88?
- What is the hexadecimal representation you would use to spell the letter 'B'? 'N' (you probably won't want to use 5 hex digits for this one)?

Exercise 4: 1,000 $1 Bills

I hand you a thousand $1billsandtenenvelopes.Yourjobistofindawaytoputvariousnumbersofdollarbillsinthosetenenvelopessothatnomatterwhata$
1000), you can simply hand me some combination of envelopes and always be assured of giving me the correct amount of cash.

Checkoff:

- Explain to your TA how to distribute the dollar bills in the ten envelopes.

Exercise 5: Sizes

Write a C program that displays the sizes (in byte) of different data types. You are only allowed to have one `printf` in the program - it has to be used inside a precompiler macro. This macro can only have one argument. Each type should be output in a new line similar to this (with correct values of course):

```
Size of short: 3
Size of int: 5
```

The sizes of the following types should be printed:

- char
- short
- short int
- int
- long int
- unsigned int
- void *
- size_t
- float
- double
- int8_t
- int16_t
- int32_t
- int64_t
- time_t
- clock_t
- struct tm
- NULL

Compile using:

```
gcc -Wpedantic -Wall -Wextra -Werror -std=c89
```

Use `-m32` to compile it for 32bit and `-m64` to compile it for 64bit.

Checkoff:

- Show and explain your source code to the TA.
- Show the compilation of the 32bit and 64bit version to the TA.
- Run both programs and explain the sizes and differences.

Schwertfeger, Sören <`soerensch` *AT* `shanghaitech.edu.cn`>
Chundong Wang <`wangchd` *AT* `shanghaitech.edu.cn`>
Modeled after UC Berkeley's CS61C.
Last modified: 2020-02-27