Lab 1

Computer Architecture I @ ShanghaiTech University

Goals

- Setup your Linux.
- Setup Gradescope.
- Learn some simple C programming.
- Learn basic linux commands.

How Checkoffs Work

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 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. 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

As CS students you should have a **REAL** Linux (e.g. Ubuntu) installed. In CS110 and CS110P, we recommend you installing Ubuntu 24.04 LTS (64bit) on your computer. If you are using an X86 computer (Intel or AMD), you should install it as dual boot. If you are using an ARM-based computer (e.g. Apple Silicon), contact us immediately. Note that some part of the lab may not work on macOS!

Show your TA the working Linux with gcc installed.

Exercise 2: Gradescope

Show the TA your gradescope account and that you finished HW1.

Exercise 3: Linux command

The following commands/programs are very useful in Linux.

- cd
- ls
- pwd
- mkdir
- rm
- cp
- mv
- cat
- man
- touch

Explain the usage of each command to the TA.

Exercise 3: Binary Alphabet

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 7,8,16,8,7? 0xF8F88?
- What is the hexadecimal representation you would use to spell the letter 'H'? 'N' (you probably won't want to use 5 hex digits for this one)?

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
- struct { int a; short b; }

Compile using:

```
gcc -Wpedantic -Wall -Wextra -Wvla -Werror -std=c11
```

Use `-m32` to compile it for 32bit and `-m64` to compile it for 64bit. Note that you may need to install a package named `gcc-multilib` to compile with `-m32`.

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.
- Explain why size of `struct { int a; short b; }` is not the sum of sizes of `int` and `short`.

The following TA(s) are responsible for this lab:
Luyue Tian <tianly2022 AT shanghaitech.edu.cn>