Lab 1 Report

Yilin Zhang 23020036094. Lab group: None.

Lab date: 11th Nov 2025. Report date: 7th November 2025.

1 Install Ubuntu 20.04

I own a x86_64 intel CPU PC, with two hard disks in it. Sadly, non of them runs a Ubuntu desktop, because beside a Windows 11 system, I chose Fedora Workstation as my second PC system. Since Fedora and Ubuntu are both GNU-Linux system, I desided to install Ubuntu as a vertual machine within Fedora.

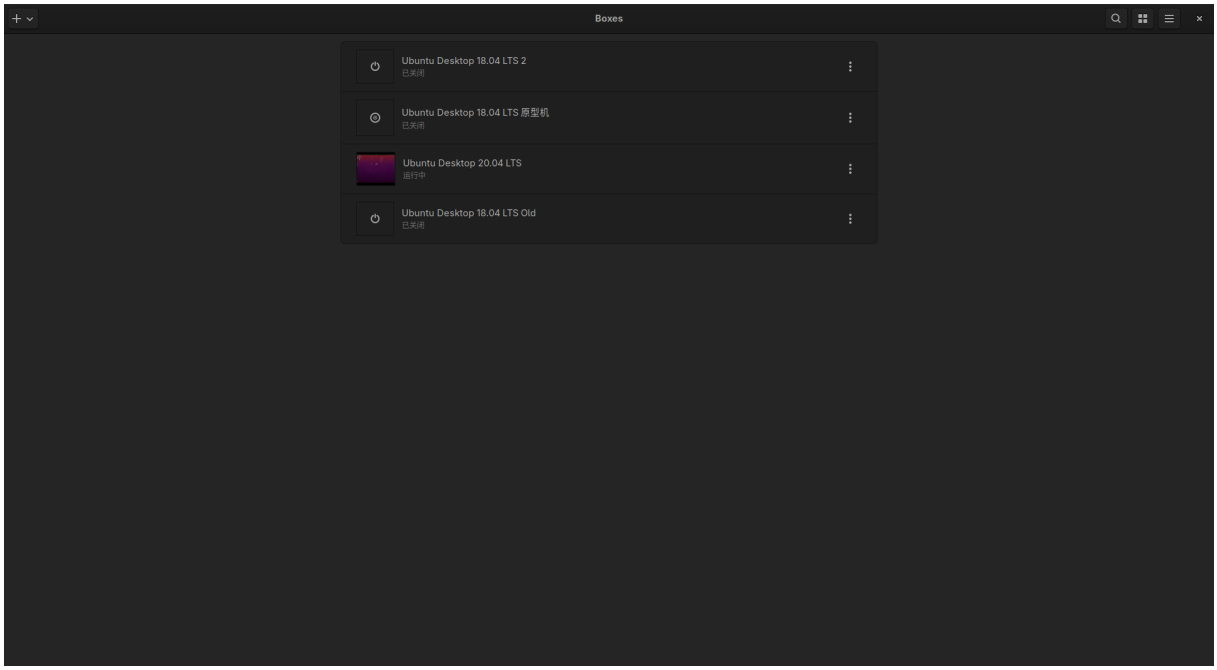

Figure 1: Box app in Fedora 43 system contains several Ubuntu virtual machines.

I didn't chose tsinghua mirror, but ustc mirror instead [1]. Because using ustc mirror is much more convenient: Only a few commands needs to be copied.

```
sudo sed -i 's@//.*archive.ubuntu.com@//mirrors.ustc.edu.cn@g'
↪  /etc/apt/sources.list
sudo sed -i 's/security.ubuntu.com/mirrors.ustc.edu.cn/g'
↪  /etc/apt/sources.list
sudo sed -i 's/http:/https:/g' /etc/apt/sources.list
sudo apt-get update
```

2 Install packages

Run the command below, nothing special. By the way, the are already installed in a new build system, “Nothing to do”.

```
sudo apt install build-essential cmake git
```

3 Git and C++

- **Git** usage are showed in section 4.
- **C++**: Just simply follow the guild. The building experience are showed below.

```
(base) ros@ros:~/vnav-codes-yilin/lab1/lab1_demo/build$ cmake ..
-- COnfiguring done
-- Generating done
-- Build files have been written to:
↪  /home/ros/vnav-codes-yilin/lab1/lab1_demo/build
(base) ros@ros:-/vnav-codes-yilin/lab1/lab1_demo/build$ make
Scanning dependencies of target mylib
[ 25%] Building CXX object CMakeFiles/mylib.dir/mylib.cpp.o
[ 50%] Linking CXX static library libmylib.a
[ 50%] Built target mylib
Scanning dependencies of target main
[ 75%] Building CXX object CMakeFiles/main.dir/main.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
```

Then just run the ``main'' command.

```
(base) ros@ros:-/vnav-codes-yilin/lab1/lab1_demo/build$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  libmylib.a  main
↪  Makefile
(base) ros@ros:-/vnav-codes-yilin/lab1/lab1_demo/build$ ./main
Hello world!
In library
```

4 Exercises

Git

I forked a copy under my own GitHub account here from this OUC vlab cource.

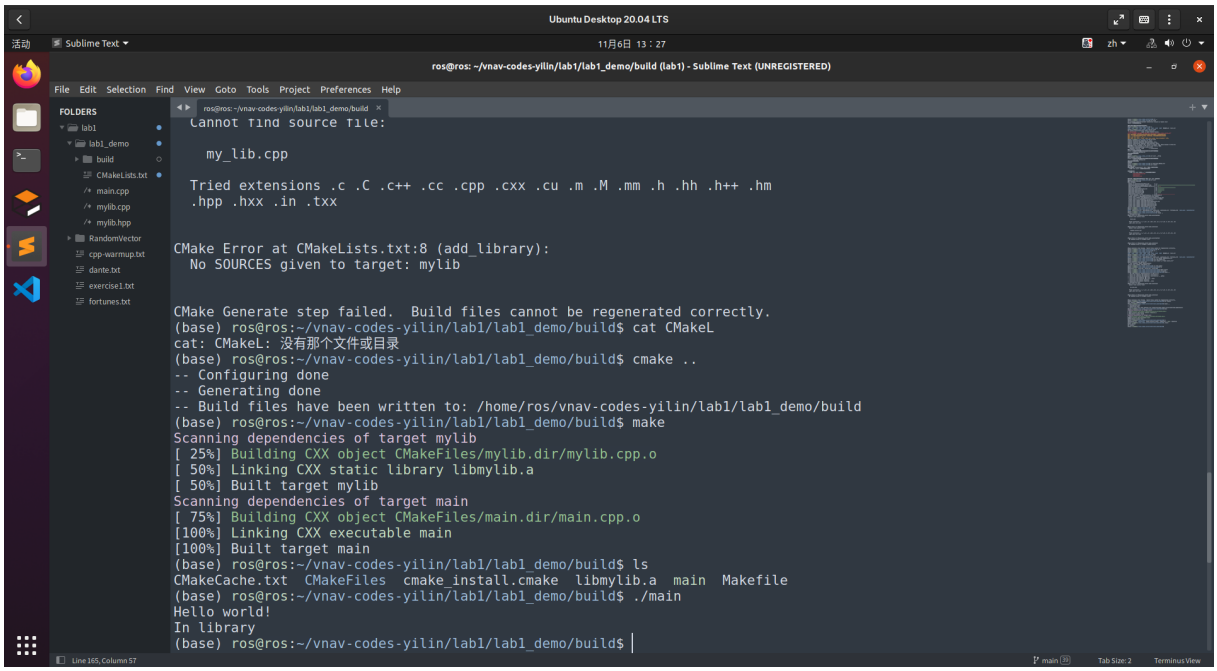

Figure 2: A screenshot of cmake demo's result

Shell

1. Answer to the following questions

- I installed a proxy tool on my ubuntu, therefore, I could ``wget'' dante.txt smoothly.
- This question needs to use command ``wc''.
  a. It contains **19567** lines.
  b. It contains **97676** words.
  c. **14338** lines are not blank.

2. Output redirecting

I did ``fortune >> fortunes.txt'' in this question. The result can be seen using ``cat'':

```
If you sow your wild oats, hope for a crop failure.
You tread upon my patience.
^^I^^I-- William Shakespeare, "Henry IV"
Q:^^IHow did you get into artificial intelligence?
A:^^ISeemed logical -- I didn't have any real intelligence.
You will be called upon to help a friend in trouble.
We should be careful to get out of an experience only the wisdom that is
in it - and stay there, lest we be like the cat that sits down on a hot
stove-lid. She will never sit down on a hot stove-lid again - and that
is well; but also she will never sit down on a cold one any more.
^^I^^I-- Mark Twain
```

C++: Warm-up Exercises

Operators

1. The values of ``i'' is 2, and the value of ``j'' is 1.
2. The following code prints ``b''.

References and Pointers

```
10 10

1764

0

0
```

Numbers

1. The differences are: ‘int‘ is always 4 bytes. In 64-bit systems, we need a longer integer, so ‘long‘ is 8 bytes, refers to the longest integer type in the system. However, the longest length in 32-bit system is 4 bytes, so ‘long‘ here is 4 bytes, which is a bit confusing. To solve this problem, we invent ‘long long‘ which is 8 bytes in both 32-bit and 64-bit systems. Btw, ‘short‘ is 2 bytes.
2. The difference, simply, would be ‘float‘ is 8 bytes, and ’double’ is 16 bytes. ‘i‘ here is 3.
3. The difference is that ‘unsigned‘ moves the negative part all the way to the right, which makes the integer a larger, non-negative number. The value of ‘c‘ is 0xff, 255 I guess.
4. The value of ‘i‘ is 0.

C++: RandomVector

All compliments are posted in the file ``random_vector.cpp'' below. [2, 3] I used an extra private method ``void mySort(std::vector<double> & vect)'' here to help print histogram. Any sort method can be used here, and I chose bubble sort.

```cpp
#include "random_vector.h"
#include <cfloat>
#include <iostream>
#include <random>
#include <vector>
// #include <algorithm>
```

```cpp
// #include <numeric>

RandomVector::RandomVector(int size, double max_val) {
    std::random_device rd;
    std::mt19937 eng(rd()); // Mersenne Twister 19937
    std::uniform_real_distribution<double> distr(0.0, max_val);
    for (int i = 0; i < size; i++) vect.push_back(distr(eng));
}

void RandomVector::print() {
    for (double v : vect) std::cout << v << " ";
    std::cout << std::endl;
}

double RandomVector::mean() {
    double mean_to_return = 0.0;
    for (double v : vect) mean_to_return += v;
    mean_to_return /= vect.size();
    return mean_to_return;
    // return std::accumulate(vect.begin(), vect.end(), 0);
}

double RandomVector::max(){
    double max_to_return = 0.0;
    for (double v : vect) {
        if (max_to_return < v) max_to_return = v;
    }
    return max_to_return;
    // return *std::max_element(vect.begin(), vect.end()); // just like
    ↪  *void in C
}

double RandomVector::min(){
    double min_to_return = DBL_MAX;
    for (double v : vect) {
        if (v < min_to_return) min_to_return = v;
    }
    return min_to_return;
    // return *std::min_element(vect.begin(), vect.end());
}

void RandomVector::printHistogram(int bins) {
    if (bins <= 0) return;

    double min_value = min();
    double bin_length = (max() - min_value) / bins;
    int max_volume = 0, j = 0;
    std::vector<double> separators, sorted_vect = vect;
    std::vector<int> volumes;
```

```
    // init volumes, separators
    for (int i = 0; i < bins; i++) {
        volumes.push_back(0);
        min_value += bin_length;
        separators.push_back(min_value);
    }

    mySort(sorted_vect);
    for (double v : sorted_vect) {
        while (j < bins - 1 && separators[j] < v) j++;
        if (j < bins) volumes[j]++;
    }

    for (int volume : volumes) {
        if (max_volume < volume) max_volume = volume;
    }

    for (int i = 0; i < max_volume; i++) {
        for (j = 0; j < bins; j++) {
            if (max_volume - i <= volumes[j]) {
                std::cout << "*** ";
                continue;
            }
            std::cout << "    ";
        }
        std::cout << std::endl;
    }
}

void RandomVector::mySort(std::vector<double> & vect) {
    // bubble sort
    int len = vect.size();
    double temp;
    for (int i = len - 1; i > 0; --i) {
        for (int j = 0; j < i; ++j) {
            if (vect[j] > vect[j + 1]) {
                temp = vect[j];
                vect[j] = vect[j + 1];
                vect[j + 1] = temp;
            }
        }
    }
}
```

``void printHistogram(int bins)'' here is the one needs to explain, the others are too easy. It can be seperated into 4 steps:

1. **Inicialize volumes and separaters**: The counting result can be seen as several

volumes with its height of the numbers. And there are bins numbers of volumes. I used the same number of separaters to devide them, with each separater is the largest number of the linear ``min()'' value to the ``max()'' value. Take ``bins == 5'' for example, these two viriables are set to ``0, 0, 0, 0, 0'' and ``0.2, 0.4, 0.6, 0.8, 1.0''.

2. **Fill volumes**: One easy way to count how many numbers are in a particular volumn is to sort the whole vector first, therefore, a ``sort()'' method may be used. Then we simply put a printer to the first value of the separaters, compare all numbers in the vector, and see whether they are smaller then this value. Thus, the first volumn ended adds up. Then the second, the third...

3. **Find the highest volume**: This step prepares the maximum count for the next step.

4. **Print the result**: I used one for loop in another to make it a squared space. In the inner loop, each height of the volumes are compared to the maximum number in step below. On the top there's usually only one or two volumes that needs to be printed, then line by line to the bottom.

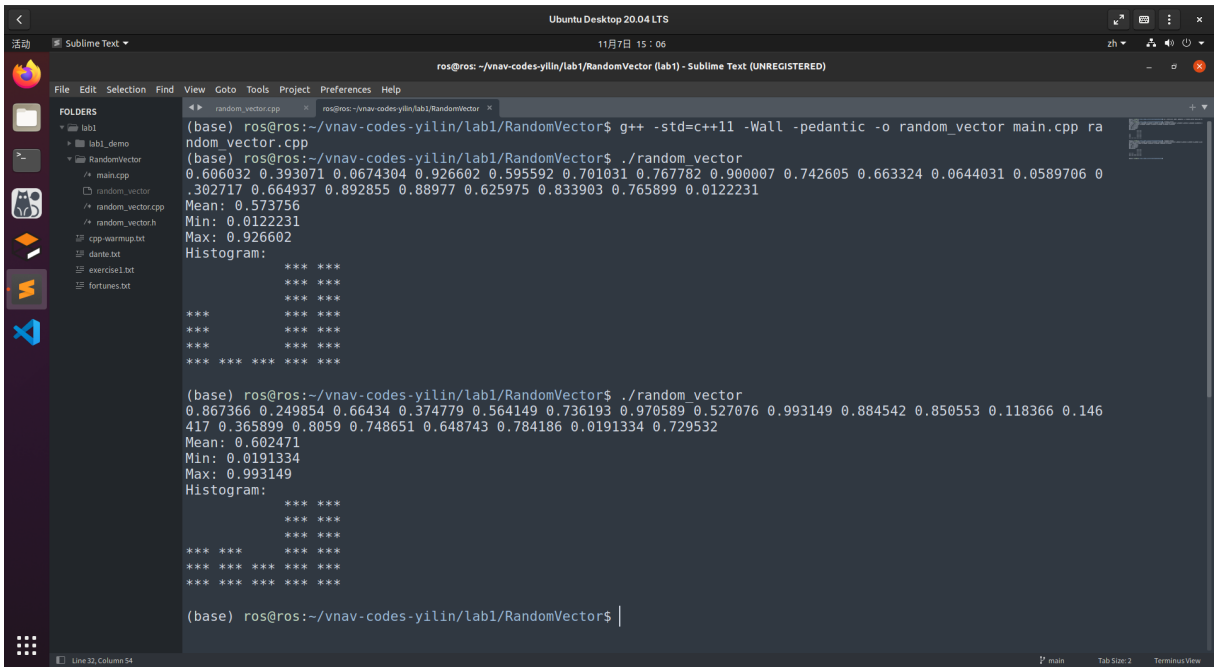

Figure 3: The random results of C++ exercise.

Optional

There are some comments in my code, they are the easy ways with the functions in the header ``<algorithm>''.

5 Some words

I own a Proxmox VE machine, and have already installed Linux systems over a hundred times, I suppose. Configuring Ubuntu is not a big deal for me. Nevertheless, **C++** is an entirely new language to me. The last exercise took me more than 3 days! I can't forgive that because it shouldn't have been such a problem.

But I have **something else** to say. I wrote this down, feeling somewhat aggrieved, and I apologize for that first. The first time I heard we would use an MIT course in our robotics curriculum, I was quite happy to see OUC adopting state-of-the-art tutorials, though also a little scared of its difficulty. I finished lab1 with great enthusiasm and went to submit my homework. Then, I found a "LaTeX" report requirement on the website. I thought, okay, maybe the OUC teachers want to improve our ability to write lab reports. That's fine.

The point is, I later heard from classmates that we are not only required to complete the exercises MIT students do but also **include the entire process of Lab1**! That is ridiculous! Documenting everything about installing a new Ubuntu system and configuring its environment is utterly **meaningless**! Even MIT students don't have to submit all that, so why should we? These are just silly, step-by-step instructions that my grandma could follow! **THERE IS NO NEED TO DOCUMENT EVERYTHING!**

In my opinion, we should focus on what we truly need to learn. The MIT website for this course offers a clear, complete guide with a straightforward grading standard. Perhaps just replicating their experience would be better than this. Please forward this to our teacher; we need to make a change.

References

[1] University of Science and Technology of China. Ustc mirror help: Ubuntu, 2025. URL `https://mirrors.ustc.edu.cn/help/ubuntu.html`.

[2] RUNOOB.com. C++ vector container, 2025. URL `https://www.runoob.com/cplusplus/cpp-vector.html`.

[3] RUNOOB.com. C++ algorithm lib <algorithm>, 2025. URL `https://www.runoob.com/cplusplus/cpp-libs-algorithm.html`.