## 1 Introduction

The purpose of this tutorial to provide an introduction to the some of the software tools we use in the humanoid robotics lab @ GT. The original version of this document was written by Can Erdogan. It is organized as a series of questions which are supposed to cover a variety of topics, let the reader practice them and further their experience. This document covers the following topics:

• Operating system - Linux, grep, apt-get

• Text editors - Vim, Emacs

• Version control - Git

• Compiling - CMake

If you are already familiar with these topics, you can probably finish this project in a day or so. Otherwise, 1~1.5 weeks should be sufficient to get used to these tools.

## 2 Operating system

All the machines in the lab use Linux as the operating system, specifically Ubuntu, version 12.04 (Precise Pangolin), or Debian, version 7.0 (Wheezy). If you were to choose Windows as your operating system, you are welcome to do so but most of your lab partners will have much more experience in Linux and be able to help you more easily if you use Linux. Regardless, in the following of this document, we will assume that the reader uses Ubuntu 12.04. Note that as of this time of reading, 12.04 might not be the latest but is the long-term support release.

One of the most common operations with a Linux operating system is installing libraries and programs either from Ubuntu repositories or di□erent 3rd party sources. A library can be installed in several different ways but almost all of them require you to first use the Linux terminal. The most common commands one uses in the terminal can be listed as: cd, ls, mv, mkdir, rm, grep, cat, sudo, man. The last command man opens the manual pages for these commands (all except cd). The cd command let's you "change directory". To enter a folder named "Temp", just like opening it in windows, you'd say "cd Temp". To go back, "cd ..". You can find out where you are in the file system with the pwd command. You will at least have to use cd to install the libraries. Also, grep will be your best friend sometime soon. Lastly, sudo allows you to run commands with security privileges.

First, you can use apt-get, synaptic or aptitute to search for the file in your list of repositories and install it from there. Second, a .deb file can be downloaded and installed with the Debian package manager dpkg. Lastly, a library can be installed from the source code.

Finally, one command you should definitely know is "locate". The locate command finds all the files in your system that contain the given word. This is useful when you want to find out if a library is installed in your system or where it is. After installing a library, you might have to run "sudo updatedb" for the database file to update itself and include the new library.

Checklist

• Acquire Ubuntu 12.04 (or the latest version of Ubuntu)

• Open a terminal and use the man command to briefly find out what the other commands do.

• Install the following libraries with apt-get: build-essential, git, gdb, cmake, libboost-dev-all. We will be using these tools and libraries in the following sections.

• Install Google Chrome using a .deb file. This is the only example I could come up with where .deb installation approach is the default but in general, one should definitely be aware of this tool.

## 3 Text editors

There are various ways to edit files and write code in Linux environment. To edit files, there are a lot of options: gedit, vim, emacs, nano and etc. People who are unfamiliar to the Linux environment and prefer a more notepad-like user interface, usually prefer gedit. Most people in the lab use vim and I, Can Erdogan, personally prefer it as well. However, emacs is also a very popular editor choice (see “Editor war” in wikipedia). Choose an editor.

Assuming you chose vim, I’d like to make a note on how to use it. Vim has a few modes: normal, insert, visual, etc. At first, you’d be more interested in the insert and normal modes. When you open vim, it starts with the normal mode where you can type in commands to save (w), quit (q), edit other files (e) by first typing a colon symbol (:). Now, if you want to type text into a file (i.e. get into the insert mode), press i and if you want to go back to the normal mode press escape. If you want to undo, go to normal mode and press u, and for redo, press ctrl-r.

As a side note, vim has a let of options one can configure to make the environment more friendly. One option I definitely suggest is set number which places line numbers at the left of the screen, making it easier for you the location of a bug and etc. Another option is the tabsize, how large a “tab” is in the code. Because we tend to have a lot of embedded functions and loops, we tend to choose a tabsize of two with the command: set tabstop=2. (Tutorial assumes you have set this.) One last command is set hlsearch which highlights the words you search for with the \ command. Create a file called .vimrc (note the dot in front) in your main folder /home/username/.

### Checklist

• Get the cheat sheet from the web for your choice of editor.

• Learn how to open a file, write some text to it, save it.

• Rename the file, copy it, make a new folder and move it there with the Linux commands from the last section. One reason why we want you to be able to do these operations from the terminal, i.e. the command line, is that sometimes you can not use the user interface and your mouse, and might have to use the keyboard throughout your work.

## 4 Version control

Version control is a method to share files between people while preserving the history of changes to the files. One of the most common version control programs is git. We will be using git in the rest of the project to share code for compiling and debugging.

Checklist

• "Clone" the repository from the following address: https://github.com/cerdogan/tutorial.git. This is similar to making a copy of the repository on the server to your computer.

• Checkout a branch called "learningGit" and follow the instructions in the README file. There are a number of steps that will help you learn the basics of learning git.

## 5 Compiling

The dominant coding language in the lab is C++. All the robots and the simulation software is based on C++. The g++ is the compiler for the C++ language. Although it is often recommended that one takes a course in basic C/C++ languages to learn the basics, it is quite easy to learn the language by experience over time, mostly by googling the error messages from the compiler. Just to clarify, the compiler, g++, is a program that turns the C++ code, mostly written in .cpp/.h files, to executables that we can run.

If you have no experience with C++, I'd strongly recommend following the small tutorials in http://www.cplusplus.com/doc/tutorial/ until and including "Classes (II)". To follow the tutorials, you will have to compile the code. The main idea is for a file "bla.cpp" that has the code, you can run "g++ bla.cpp" to compile it and run the output file "a.out" with the command "./a.out".

Next, I will briefly describe how to link to libraries and include files while compiling an executable in the command line (with the g++). The basic idea is if your code calls a function or uses a variable that is not defined in that file, you'd get an "error: was not declared in this scope". Then, you would like to include the file that contains this definition with the "-I" flag. Secondly, there is a difference between declaring a function and defining it. When an executable wants to use a function and it can find its declaration but can not find its definition, an "undefined reference" error would be received. To fix this problem, one should link to the library that contains the definitions with the "-l" flag and indicate its location with the "-L" flag.

Lastly, we will cover CMake. You probably have used CMake to install Eigen and FCL by now but the goal is for you to be able to write CMake scripts easily. CMake is a tool that helps users organize their compilation scheme when their code is distributed in a lot of files and it uses a large number of libraries. The idea is a CMake script creates a Makefile which includes the g++ commands with the correct flags we just discussed. And the most important tool in learning how to use a CMake script is when you "make" a program, that is compile it, use the "VERBOSE=1" flag to actually see the g++ command. This would let you associate the keywords that you have used in the script with their effects in the compilation.

CMake mostly uses the following keywords: add executable, include directories, link directories, link libraries. The add executable keyword describes which files (i.e. bla.cpp) have the main code that you are compiling. The other three keywords respectively stand for the "-I", "-l", and "-L" flags.

In the exercises below, all the code is correctly written, that is there are no bugs and with the right flags, they should all compile and execute successfully. This is one of the cornerstones of your daily process, being able to compile and execute your code, and I'd strongly suggest taking the time to understand these basics.

Checklist

• Checkout the branch “compiling”

• Compile the three examples from the command line using the g++ compiler

• Fill the empty spaces in the first two CMake scripts.

• Write a CMake scripts for the third example from scratch. To use the two helper .cpp files, use the add library and the file command with GLOB parameter.

• Take a screenshot of the output of the compiled executables when compiled from command line.

Send the screenshots and the CMake files.