Introduction to Tools for Computer Science and Software Development

Ian Ooi, Amelia Peterson

October 6, 2012

Contents

Chapter 1

Setting up your System

Your development environment is a personal choice. Some factors to consider include target platform, your familiarity with different software and environments, and the tools available to you in each environment. We're proponents of Linux development, but there are situations where a different system may be ideal.

1.1 Linux

1.1.1 Choosing a Distro

There are numerous different distributions, and it really (again) comes down to preference and what you want from your system. Debian and Debian based distributions (e.g. Ubuntu and Linux Mint) are very beginner friendly and come pre-packaged with most of the tools you'll need for some of those basic operations you take for granted (displaying GUIs, adjusting volume and brightness, and connecting to the internet for instance). They are somewhat less flexible as a result, or require more work to configure if you want a different significantly different setup than one of the pre-packaged ones. Debian based distributions are excellent, however, and generally you don't need much beyond the pre-packaged configuration.

If you're feeling adventurous or just want to try something different, there are numerous other distributions, such as Arch Linux, which is a bit less beginner-friendly and more difficult to configure.

1.2 Windows

Setting up Windows should be a simple matter of either:

A) Buying a computer with Windows on it, or...

B) Buying a license for Windows, popping the disc in (or whatever other installation media you are using) and following the directions given

From here, you can choose what you want to develop with. If you want to use Microsoft Visual Studio, simply purchase a license (or get one through school, work etc.) and install. Other IDEs (discussed later) are also available for C/C++ development as well as most other languages. Some C/C++ IDEs include:

- Eclipse
-

If you don't want a full IDE, there are also numerous editors, compilers and debugging tools available as individual programs.

1.2.1 Editors

Choosing an editor is truly a matter of personal preference, depending on the features you're looking for. Some, like Vim or Emacs are very feature rich and extensible, with the tradeoff being a high learning curve. Others, such as using Notepad, are very simple, but can be difficult because they lack useful features like syntax highlighting. In general, you want something with (at the very least) syntax highlighting and often an option for soft tabs (replacing tabs with a number of spaces, usually 4). C++ doesn't need soft tabs per se, but it's useful for preserving your code formatting across systems and programs. Soft tabs becomes more necessary in situations where whitespace (spaces, tabs, newlines) become meaningful, such as in Python or Haskell. Some will also include extra features, such as built in debugging, compiling or running programs, and editing extras such as macros.

- Notepad++
- SciTE

- Vim
- Emacs
- Code::Blocks
- Notepad2

1.2.2 Compilers and Interpreters

You will need a suitable compiler or interpreter for whatever language you are developing in. Notable compilers for C++ are GCC (g++ is the C++ compiler) and MSVCC, which comes with Microsoft Visual Studio.

GCC can be used by installing MinGW, Cygwin, or another tool which either mimics or

1.3 Mac

Coding on a Mac can be much the same as coding on a Linux system. Both are (since Mac OS X) Unix based, and as such, most of the tools available. In general, you can set up your system the same way a Linux user would, though some additional tools and software may be available.

Chapter 2

Basic Terminal Usage

Alright then, wow that was lots of words. Instead, have lists of commands you'll need.

2.1 Linux/Mac

When using the terminal, help text and man pages are available for most tools, commands and software, as well as extensive information on the internet.

Manual pages can be accessed by running `man <command>`, so, for example, to find help on the command `grep`, you would run `man grep`. This also works for built in C++ and C functions, e.g. `man printf`.

Command line tools usually include help text as well, accessed either through the `--help` flag (e.g. `ls --help`) or from the `-h` flag (e.g. `sudo -h`).

2.1.1 Moving Around/Browsing your files

- `ls`: List the files in a directory. Common flags are `-a` to list hidden files and `-l` to print extra information.

  ```
  ls -a
  ```

  ```
  ls -al
  ```

- `cd`: Move to a directory. Note `..` is the directory above and `.` is the current directory.

- `mv` Move file or folder to a different directory. (Will automatically recurse for folders!)
- `cp` Copy a file or folder to a different directory. (Will not automatically recurse, must specify with `cp -r`.)

2.1.2 Permissions

- `sudo`
- `su`
- `chmod`
- `chown`
- `gksudo` GTK frontend for sudo, creates a window which prompts for the password instead of through command line. Useful for executing from the run dialog (Alt+F2) or things like dmenu.

2.1.3 Other useful commands

- `grep` Regular expressions
- `cat`
- `less`
- `more`
- `alias`
- `man`
- `apt-get`
- `aptitude`
- `dpkg`
- `screen` Terminal multiplexing. Lets you have multiple terminals in one, as well as split screens, move between them and detach to save sessions for later. *Screen will also allow you to connect to a serial device.*

- `tmux` A different terminal multiplexor with some additional features for multiplexing, but will not connect to serial devices.
- `pacman`

2.1.4 Configuring

If you use bash as your shell (if you are on Debian/Ubuntu/Linux Mint bash is the default), you can configure it using your `.bashrc` file, found in your home directory (`/home/yourusername/.bashrc`, equivalent to `~/.bashrc`).

Any valid bash shell commands can be written and they will execute. For example, `alias cp='cp -v'` will cause cp to always run with the verbose (`-v`) flag.

2.2 Windows

2.2.1 Moving Around/Browsing Files

-

Chapter 3

Text Editors and IDEs

3.1 Command Line Editors

3.2 GUI editors/IDEs

Chapter 4

Compiler Usage

4.1 Command Line Compilers

4.2 GUI based compilers

4.3 Linking Libraries

4.4 Cross-Compiling

Chapter 5

Debugging

5.1 Types of Errors

5.1.1 Compile-time

-

5.1.2 Runtime

- Segmentation Fault (Seg Fault): Your program tried to access memory that it couldn't, or otherwise did something with memory it shouldn't have. This is usually caused by something wrong with pointers, your other logic, or a stack overflow, where your program ran infinitely or used too much memory (infinite/large sized arrays/vectors etc.).

-

5.2 Debugging Tools

5.2.1 GDB

5.2.2 Visual Studio Debugger

5.3 Debugging Memory

5.3.1 Valgrind

5.3.2 Dr. Memory

5.4 Debugging Performance

5.5 Understanding Compiler Errors

Chapter 6

LaTeX

There comes a time in most science, math and engineering students' lives when they need to write pages of equations. LaTeXis made for typesetting documents, including equations, tables, pictures and text, and is perfectly suited for pages of proofs, lab reports, and general documents. It may look intimidating at first, because you're writing markup to create your document, and it seems a bit like code, but it makes formatting much simpler and creates neater, more professional documents.

The basic structure of a LaTeXdocument looks like this:

```
% this is a comment, it won't be in your document
% include documentclass at the top of every LaTeX document
% 12pt is an argument telling it to make the general font 12pt
% article is just a basic type of document for most purposes
\documentclass[12pt]{article}

% specify a title
\title{My TAs Love Me Because They Can Read My Work}

% specify an author
\author{Chuck Finley}

% specify the date, \today just inserts the current date
\date{\today}
% \date{October 5, 2012} %specifies a specific date instead
```

```
% start the body of the document
\begin{document}
% tell LaTeX to actually make a title, it will change the size
% and center it for you
\maketitle
% these are still comments
You can write whatever you want here, just like you would
in Word, LibreOffice or OpenOffice.  One major difference
is that LaTeX removes extraneous (more than 1) spaces    such
        as   these.

A single empty line will start a new paragraph and indent
it for you.  The first paragraph is not indented by
default, don't worry about it.  It bothered me at first,
but it's really going to be ok.

LaTeX (\LaTeX) commands start with a backslash (\).
\emph{This text is emphasized, similar to italics.}
\textbf{This text is bold.}  You have to escape
backslashes, such as \\.  The same works to type \%.
\end{document}
```

Chapter 7

Build Systems

7.1 Makefiles

7.2 CMake

Chapter 8

Source Control