COMP 1002 — Introduction to Tools and Techniques in Computer Science 1

Course Description

Calendar entry

This is a lab-based course. Every Computer Scientist needs to make use of an expansive set of modern computing tools and techniques. This course provides a hands-on experiential introduction to working with the tools and techniques we use every day to design, develop, analyze, and maintain software.

Prerequisite: One of any 40S Mathematics (50%), MATH 1018, or MSKL 0100.

General Course Description

One of the tools that we use as computer scientists is a computer (of course!). Computer scientists and software developers use their computer to get their jobs done every day. Students (in Computer Science and other disciplines) also need to use their computers to do research and complete their course work.

Sometimes we (as students, computer scientists, and software developers) have to independently figure out how to use new tools or independently learn to use new tools, techniques, or strategies to get our work done, communicate with other Computer Scientists, or do research.

COMP 1002 is a course designed to help you (as a student) get up to speed with tools, techniques, and strategies that you may be expected to use in courses, research, or in your workplace. COMP 1002 is intended to be an environment that provides you with support and credit for the work you're doing.

Detailed Prerequisites

Before entering this course, a student should be able to:

- Use a computer for basic tasks (e.g., web browsing, typing documents).

Course Goals

By the end of this course students will:

- Install software on their computer and write structured text.
- Connect to other computers and transfer data between computers.
- Use command-line tools to manage files.
- Compile and run programs written in different programming languages.

- Use version control software.

Learning Outcomes

The Mechanics of Programming

Setup, writing structured text, and file management

Students should be able to:

1. Install software on their personal computer (e.g., `pandoc` and VS Code).
2. Write plain text documents (including code and formulas) using structured markup formats (e.g., Markdown and LaTeX).
3. Convert plain text documents to different formats (e.g., `docx`, `pdf`).
4. Define their own strategy for organizing files in folders.
5. Build a directory structure with a visual file explorer for the purpose of organizing their files.

Remote connections and command-line file management

Students should be able to:

1. Connect to a remote computer using `ssh`.
2. Create, rename, delete, and inspect folder structures on the command-line.
3. Navigate folder structures on the command-line.
4. Use a command-line text editor to open, inspect, edit, and save a file.
5. Complete commands and file names with tab completion.
6. Transfer files between their computer and a remote computer.
7. Use the `handin` command in the U of M CS UNIX environment.

Compiling and running code, introducing version control

Students should be able to:

1. Download a file from the internet on the command-line.
   Compile and run programs of various programming languages on the command-line.
2. Use version control software on their own code to keep track of changes over time (e.g., initializing and adding changes to a repository).
3. Use version control software on their own code to make changes and quickly be able to revert back to a working state.
4. Identify files that should not be version controlled and make sure that they aren't included in a repository's history (e.g., `.gitignore`).

Advanced version control, managing personal projects

Students should be able to:

1. Use branching and merging with a version control repository.
2. Synchronize repositories between computers for a personal project (e.g., `git push` and `git pull`).
3. Manage personal project tasks as issues using web-based issue tracking software.

Advanced shell usage

Students should be able to:

1. Find files on the command line (e.g., by name using patterns) and execute commands on those files.
2. Filter lines from large text files using patterns.
3. Read and change permissions on files.
4. Compare plain text files.

Getting help and digital notebooks

Students should be able to:

1. Get help about a command by running the command.
2. Get help about a command by reading manual pages.
3. Find and assess the quality of help online.
4. Determine appropriate ways to get help while maintaining academic integrity in the context of Computer Science.
5. Use a web-based interactive notebook to document progress in a lab notebook.