COMP1521 24T3 — Course Introduction

https://www.cse.unsw.edu.au/~cs1521/24T3/

COMP1521 Teaching Team

Convenor & most Lectures: Andrew Taylor

Admin & Guest Lectures: Xavier Cooney, Dylan Brotherston, Abiram Nadarajah

Tutors: Abby Lin, Abiram Nadarajah, Agnes Tjokrosetio, Alexander Knight-Viale, Alexander Blackmore, Amelie Robinson, Amy Hodder, Amy Tian, Anna Brew, Ashley Saipaia, Brodie Hales, Callum Reilly, Cassandra Eliot, Cyril Vivek Subramanian, Daniel Chen, Dong Huang, Dylan Brotherston, Ethan Haffenden, Flynn Lambrechts, Henry Guo, Isabel Wee, JJ Roberts-White, Jack Robbers, James Appleton, Jessica Xu, Jimmy Kirkpatrick, Joanna Wong, Jonah Hopkin, Larry Tang, Meredith Zhang, Michael He, Olivia Ding, Philip Tokareff, Raymond Li, Samuel Schreyer, Selina Wu, Shane Kadish, Siyu Qiu, Sophie Fox, Tasfia Ahmed, Tiger Liu, William Feng, Xavier Cooney,

COMP1521 Student Background

Most students in this course have completed

- COMP1511 or COMP1911

COMP1511 and COMP1911 covers *fundamental* C programming.

COMP1511 also covers topics, e.g. *linked lists*, *ADTs* not needed for COMP1521.

For this week's tuts and labs:

- review/strengthen assumed C knowledge
  - cover some small things not covered in COMP1511

Assumed Knowledge C knowledge

Assumed knowledge —

- design an algorithmic solution
- describe your solution in C code, using ...
  - variables, assignment, tests (`==`, `!`, `<=`, `&&`, etc)
  - `if`, `while`, `scanf()`, `printf()`
  - functions, `return`, prototypes, `*.h`, `*.c`
  - arrays, structs, pointers, `malloc()`, `free()`

Not assumed knowledge —

- linked structures, ADTs, sorting,
- *recursion*, *bit operations*, *file operations*
  - recursion and `for` loops will be mentioned in Week 1 tutorials

Course Goals

COMP1511/1911 ...

- gets you thinking like a *programmer*
- solving problems by developing programs
- expressing your solution in the C language

COMP1521 ...

- gets you thinking like a *systems programmer*
- with a deep understanding of run-time behaviour
- and better able to reason about your C programs

COMP1511/1911 vs COMP1521

COMP1511/1911 ...

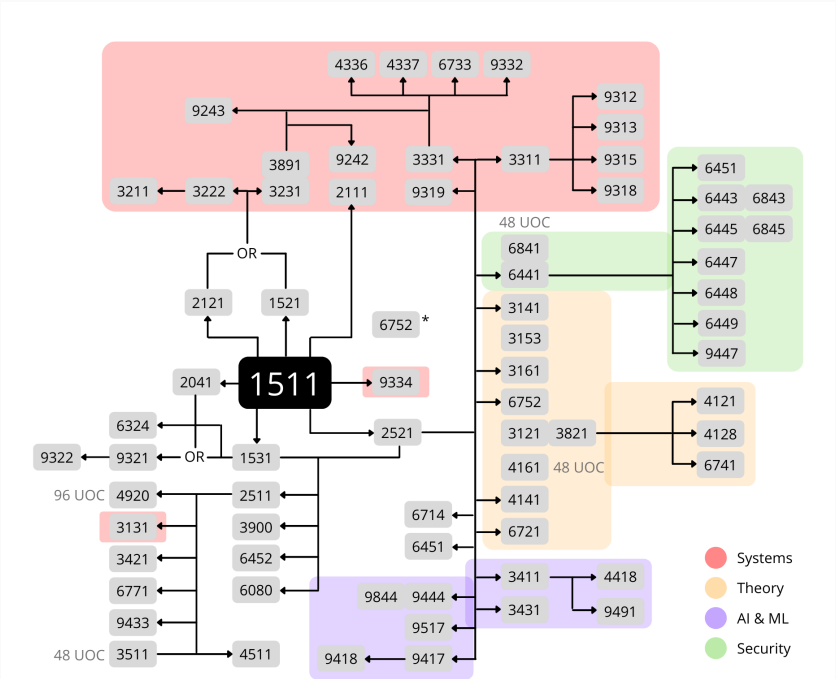
Figure 1: COMP1511/1911

COMP1521 ...

Figure 2: COMP1521

or maybe ...

Figure 3: COMP1521



Themes

Major themes ...

1. software components of modern computer systems
2. how computer represent data including integers & floats & emoji 🤔🤔🤔🤔
3. how C programs execute (at the machine level)
4. how to write (MIPS) assembly language
5. how operating systems are structured
6. Unix/Linux system-level programming particularly file operations
7. introduction to processes, thread and concurrency

Goal: you are able to understand execution of software in detail.

Textbook

There is no prescribed textbook for COMP1521.
Recommended reference ...
Computer Systems: A Programmer's Perspective,
Bryant and O'Hallaron

- covers most topics, and quite well
- but uses a different machine code

Available in UNSW Bookshop

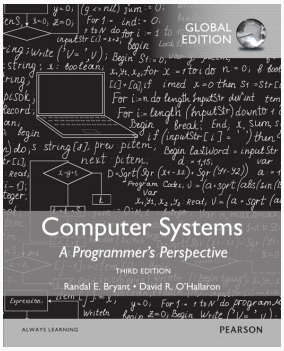


Figure 5: Computer Systems A Programmer's Perspective

Textbook

Course Material has been drawn from

- *Introduction to Computing Systems:*
  from bits and gates to C and beyond,
  Patt and Patel
- *The Elements of Computer Systems:*
  Building a modern computer system from first principles,
  Nisan and Schocken
- COMP2121 Course Web Site, Parameswaran and Guo
- Past COMP1521 lecturers, admin, and tutors

Always give credit to your sources!

Systems and Tools

Prac work based on *Linux* tools

- all tools available on the *CSE lab machines* (Debian Linux)
- can use *VLAB* or *SSH* to connect to CSE from home

Compilers: `dcc` on CSE machines (`clang` or `gcc` elsewhere)

Assembly language: `mipsy` (`mipsy_web` online, vscode extension)

Use your own favourite text editor: `ed`, `vim`, `emacs`, `nano`, `gedit`, `vscode`, etc.

Other tools: `make`, `man`, `bc -ql`, `python3`, etc.

Learn to love the *shell* and *command-line* ... very useful!

Lectures

- Monday, 11:00—13:00; Wednesday, 16:00—18:00;
  - Monday in-person, Wednesday online
  - lectures recorded and linked from course home page.
- present a brief overview of theory
- focus on practical demonstrations of coding
- demonstrate problem-solving (testing, debugging)
- lecture slides available on the web before lecture.
- guest lectures potentially from Xavier, Abiram & Dylan
- Labour Day means we miss a lecture
  - We may hold the lecture at a different time, or pre-record it (TBA)

Tut-labs

- COMP1521 has 3-hour tut-labs, starting week 1, and every week after (except week 6)
  - Our first tut-labs are immediately after this lecture
  - Labour Day means some people miss tutes on Monday week 5
    - An alternative time for your class will be arranged by your tutor
- 6 of our tut-labs are online
- delivered via Blackboard Collaborate (accessed via Moodle)
- the remaining timeslots are *face-to-face* classes
- please follow UNSW policy: https://www.covid-19.unsw.edu.au/

Tutorials

To get the best out of tutorials ...

- attempt the problems yourself beforehand
  - not marked, and no submission
  - but you will learn more if you try the problems yourself
- ask if you don't understand a question or how to solve it
- Do *not* keep quiet in tutorials: talk, discuss, ask question
- Your tutor may ask for your attempt to start a discussion.

Labs

Each tutorial is followed by a two-hour lab class.

- Several exercises, mostly small coding tasks
- Build skills needed for assignments, exam
- Done *individually*
- Submitted via `give`, before Monday 12:00 (midday)
- Automarked (with partial marks) — 15% of final mark
- Labs may include challenge exercises ...
  - may be silly, confusing, or impossibly difficult
  - almost full marks (95+%) possible
    without completing any challenge exercises

Tests

From week 3, and every week after (including week 6):

- released on Thursday 3pm (after the lecture)
- due exactly one week later
  - Submitted via `give`
- immediate reality-check on your progress.
- done in your own time under self-enforced *exam conditions*.
- time limit of 1 hour
  - can keep working after hour for 50% of mark
- automarked (with partial marks)
- best 6 of 8 tests contribute 10% of final mark
- any violation of test conditions
  $\Rightarrow$ zero for whole component

Assignments

- Ass1: Assembly (MIPS) Programming, weeks 3–5, 15%
- Ass2: C Systems Programming, weeks 7–9, 15%
- Assignments give you experience with larger programming problems than lab exercises
- Assignments will be carried out *individually*.
- They *always* take longer than you expect.
- Don't leave them to the last minute.
- Standard UNSW late penalties apply, 5% per day for 5 days, computed hourly.

Code of Conduct

CSE offers an inclusive learning environment for all students.

In anything connected to UNSW, including social media,
these things are student misconduct and will not be tolerated:

- racist/sexist/offensive language or images
- sexually inappropriate behaviour
- bullying, harassing or aggressive behaviour
- invasion of privacy

Show respect to your fellow students and the course staff

Plagiarism

Cheating of any kind constitutes academic misconduct and carries a range of penalties.
Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Plagiarism

- Labs, Tests, and Assignments must be entirely your own work.
- You can not work on labs, tests, or assignments as a pair or in a group.
- Plagiarism will be checked for and *penalized*.
- Plagiarism may result in suspension from UNSW.
- Scholarship students may lose scholarship.
- International students may lose visa.
- Supplying your work to any another person may result in loss of all your marks for the lab/assignment.

Use of Generative AI Tools

- Generative AI tools, e.g. GitHub Copilot, ChatGPT have great potential to assist coders
- Code they generate often has subtle errors & security vulnerabilities
  - also often generate poor code
- expert coders (hopefully) can spot these problems
- need a deep understanding of language/system to make good use of these tools
- COMP1521 students don't yet have this understanding
- Use of tools such as Copilot, ChatGPT may slow you getting this understanding
- Use of generative AI tools including GitHub Copilot, ChatGPT not permitted in COMP1521
  - later course will likely allow use of these tools
- dcc-help, autotest-help are specialized generative AI tools designed for CSE students
  - use of dcc-help and autotest-help is permitted in COMP1521

Plagiarism

- Labs, Tests, and Assignments must be entirely your own work.
- You can not work on labs, tests, or assignments as a pair or in a group.
- Plagiarism will be checked for and *penalized*.
- Plagiarism may result in suspension from UNSW.
- Scholarship students may lose scholarship.
- International students may lose visa.
- Supplying your work to any another person may result in loss of all your marks for the lab/assignment.

Final Exam

- **in-person** 3-hour practical exam: in CSE labs, on CSE lab computers
  - You must be in Sydney to sit the exam during the exam period
- limited environment: you get the tools and software of a lab computer, not your own computer
  - You don't get access to your normal CSE account, so no custom configuration files
  - no dcc-help or autotest-help
- may be some multiple-choice/short-answer questions, similar to tut questions.
- most questions will ask you to read C or assembler
- most marks for questions which ask you to write C or assembler
- also may ask you to answer written questions
- hurdle: you must score 18+/45 (40%) on the final exam to pass course

Assessment

- 15% Labs
- 10% Tests
- 15% Assignment 1 — due end of week 5
- 15% Assignment 2 — due start of week 10
- 45% Final Exam

Above marks may be scaled to ensure an appropriate distribution

To pass, you must:

- score 50/100 overall
- score 18/45 on final exam

For example:

55/100 overall, 17/45 on final exam $\Rightarrow$ **55 UF** not 55 PS

How to Pass this Course

- coding is a *skill* that improves with practice
- the more you practice, the easier you will find assignments/exams
- do the lab exercises
- do the assignments *yourself*
- practice programming outside classes
- treat extra tutorial questions like a mini prac exam