DPST1092 23T2 — Course Introduction

https://www.cse.unsw.edu.au/~dp1092/23T2/

DPST1092 Staff

Lecturer	Angela Finlayson
Tutors	Harshana Randeni, Bridget McCarthy, Stephen Vinall, Michelle Wong, Liam Liu

Assumed Knowledge C knowledge

Assumed knowledge —

- design an algorithmic solution
- describe your solution in C code, using ...
  - variables, assignment, tests (==, !, <=, &&, etc)
  - `if`, `while`, `for`, `break`, `scanf()`, `printf()`
  - functions, `return`, prototypes, `*.h`, `*.c`
  - arrays, structs, pointers, `malloc()`, `free()`
  - recursion (we know you might still struggle with this)

Not assumed knowledge —

- *bit operations*, *file operations*

Reflecting on CP1511

What would you do differently if you went back and did CP1511 again?

What advice would you give to someone doing CP1511 this term?

Will you follow that advice this term duing CP1521?

Course Goals

CP1511 ...

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

CP1521 ...

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

Expectations of CP1521 students

We expect CP1521 students to become more independent with their programming:

- further develop linux skills from CP1511
- further develop debugging skills from CP1511
- practice reading manuals and documentation during CP1521

Themes

Major themes ...

1. software components of modern computer systems
2. how computer represent data including integers & floats
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 CP1521.
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 at the LSU

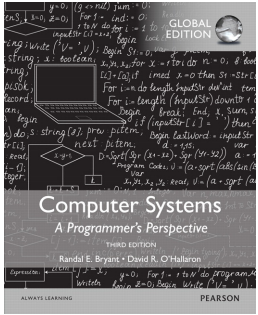

Figure 1: Computer Systems A Programmer's Perspective

Acknowledgements

Course Material has been drawn from

- COMP1521 UNSW

Always give credit to your sources!

Systems and Tools

Prac work based on *Linux* tools

- all tools available on the *CSE lab machines*
- 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` for GUI)

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!

The Linux Manual

man

The linux manual (man) is divided into the following sections

- Executable programs or shell commands eg. ls, cp
- System calls (we will be looking at many of these in the coming weeks)
- Library calls eg. strcpy, scanf

And other sections we won't be using much

You can find the full table by using the command man man which shows the manual page about the manual.

You can get more information about individual sections by using man 1 intro, man 2 intro etc.

Advice: man will be available in the exam. Get used to using it!

Lectures

- Delivered via collaborate, starting week 1, and every week after (except week 7)
  - feel free to ask questions via chat or turn on your microphone and ask
  - lectures recorded.
- 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.

Tutorials

- 2-hours of tutorials a week, starting week 1, and every week after (except week 7)
- face to face tutorials held on campus, online tutorials are held on collaborate

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
- Volunteer to do a code review for a bonus mark

Labs

Each week you will have a two-hour lab class (week 7 will be a catch up lab).

- Several exercises, mostly small coding tasks
- Build skills needed for assignments, exam
- Done *individually*
- Submitted via `give`, before Tuesday 11:59pm (just before midnight)
- 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, till week 11 (no test released in week 7):

- released on Thursday 9am 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 9 tests contribute 10% of final mark
- any violation of test conditions
  $\Rightarrow$ zero for whole component

In week 12 we will do a mini sample exam in a lecture timeslot under exam conditions which will count as the 9th weekly test

Assignments

- Ass1: Assembly (MIPS) Programming, weeks 5–8, 15%
- Ass2: C Systems Programming, weeks 9–12, 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.

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 using plagiarism detection software and *penalized*.
- 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
- 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
- you must score 18+/45 (40%) on the final exam to pass course

Assessment

- 15% Labs
- 10% Tests
- 15% Assignment 1 — due week 8
- 15% Assignment 2 — due week 12
- 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

Note: the 1 bonus mark from doing a code review counts towards your ClassMark of 55 during the term (also including assignments, labs and tests). You can not get more than 55 for your ClassMark.

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