DPST1092/CPTG1392 26T2 — Course Introduction

https://www.cse.unsw.edu.au/~dp1092/26T2/

DPST1092/CPTG1392 Staff

Lecturer	Sasa Maric
Tutors	Justin, Keyvan, Frank, Zovin, Osvaldo, Jeounghi, Michael, Tet, Luis, Cen, Ernest and Tenuun
Course Admin	Stephen

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 (for HD students)

Not assumed knowledge —

- *bit operations*, *file operations*

Reflecting on DPST1091/CPTG1391

What would you do differently if you went back and did DPST1091/CPTG1391 again?

What advice would you give to someone doing DPST1091/CPTG1391 this term?

Will you follow that advice this term duing DPST1092/CPTG1392 ?

You should try and answer these questions in your own time. Reflection is a very important part of learning

Textbook

DPST1091/CPTG1391

DPST1092/CPTG1392

Course Goals

DPST1091/CPTG1391 - you learned how to program.

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

DPST1092/CPTG1392 - you will learn how to use your programming skills

- 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 DPST1092/CPTG1392 students

We expect DPST1092/CPTG1392 students to become more independent with their programming:

- further develop linux skills from DPST1091/CPTG1391
- further develop debugging skills from DPST1091/CPTG1391
- practice reading manuals and documentation during DPST1092/CPTG1392

Previous results! AI warning

Everyone is an expert until they have to do it!

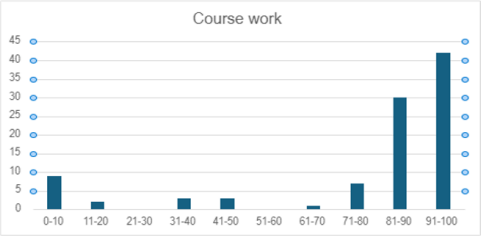

Figure 1: Previous results

Previous results! AI warning

Don't make the same mistakes!!

Last term 51 per cent of students failed this course!!!!

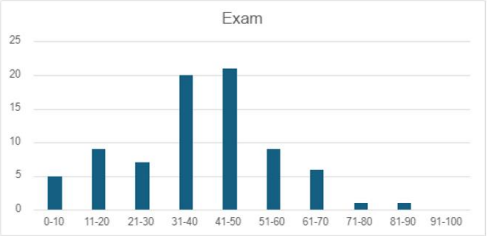

Figure 2: Previous results

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 DPST1092/CPTG1392.
Recommended reference ...
Computer Systems: A Programmer's Perspective,
Bryant and O'Hallaron

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

Available at the LSU

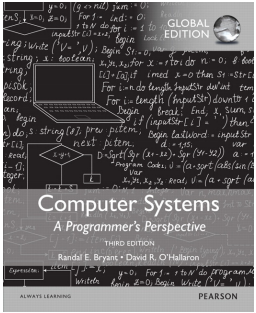

Figure 3: Computer Systems A Programmer's Perspective

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`, `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 on campus starting week 1, and every week after (except week 7)
  - please ask and answer questions and engage
  - lectures recorded via echo but not live streamed.
- 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

- **BE LOUD** talk discuss and learn.
- 2-hours of tutorials a week, starting week 1, and every week after (except week 7)
- face to face tutorials held on campus To get the best out of tutorials ...
- Attendance will be taken
- attempt the problems yourself beforehand
  - not marked, and no submission
  - but you will learn more if you try the problems yourself
- **MAKE MISTAKES**
- **YOU NEED PRACTISE CODING, SO COME TO TUTORIALS**

Labs

Each week you will have a two-hour lab class (except week 7).

- Labs will include an group activity
- Total 2 marks per week (Mark capped at 15)
- 30% mark for **individual** submission and 70% for group activity in class
- Submitted via `give`, before Monday 9am
- **MUST ATTEND AT LEAST 5 GROUP ACTIVITIES PER TERM**
- Build skills needed for assignments and exam
- Automarked (with partial marks) — 15% of final mark
- Labs may include challenge exercises ...
  - Some are very hard and intended for HD students
  - almost full marks (95+%) possible
    without completing any challenge exercises

Lab Schedule

Week	Activity
Week 1	Group Activity
Week 2	Group Activity
Week 3	Group Activity
Week 4	Group Activity
Week 5	Group Activity
Week 6	Standard Lab/Assignment Consultation
Week 7	Flex Week, No LABS
Week 8	Group Activity
Week 9	Group Activity
Week 10	Group Activity
Week 11	Standard Lab/Assignment Consultation
Week 12	Standard Lab/Final Exam Consultation

Assignments

- Ass1: Assembly (MIPS) Programming, weeks 5–8, 20%
- Ass2: C Systems Programming, weeks 9–12, 20%
- 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.
- Please don't use ChatGPT, final exam questions are taken from assignments.
- If you do the assignments yourself you will have no problems passing the exam.

Plagiarism

- Labs, Tests, and Assignments must be entirely your own work.
- Everything must be submitted individually
- 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
- Closed book exam, you will not have your notes.
- most questions will ask you to write C or assembler.
- Just because autotests are passed doesnt mean you passed the question
- Every question is checked manually
- **HURDLE: You must score 18+/45 (40%) on the final exam to pass course**
- We will talk about this more...

Assessment

- 15% Labs
- 20% Assignment 1 — due week 8 (usually)
- 20% Assignment 2 — due week 12 (usually)
- 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 **(HURDLE)**
- must come to at least 5 group activity labs **(HURDLE)**

For example:

55/100 overall, 17/45 on final exam $\Rightarrow$ **55 FAIL** not 55 Pass

How to Pass this Course

- Coding is a *skill* that improves with practice
- The more you practice, the easier you will find assignments/exams
- Its ok to use ChatGPT to help you a bit
- Do the lab exercises
- Do the assignments *yourself*
- Practice programming outside classes
- Treat extra tutorial questions like a mini prac exam
- No special secret **Come to class**