COMP2129: OPERATING SYSTEMS AND MACHINE PRINCIPLES

Semester 1, 2017 | 6 Credit Points | Mode: Normal-Day
Coordinator(s): John Stavrakakis

WARNING: This unit is an archived version! See Overview tab for delivered versions.

1. INTRODUCTION

In this unit of study, elementary methods for developing robust, efficient, and re-usable software will be covered. The unit is taught in C, in a Unix environment. Specific coding topics include memory management, the pragmatic aspects of implementing data structures such as lists and hash tables and managing concurrent threads. Debugging tools and techniques are discussed and common programming errors are considered along with defensive programming techniques to avoid such errors. Emphasis is placed on using common Unix tools to manage aspects of the software construction process, such as version control and regression testing. The subject is taught from a practical viewpoint and it includes a considerable amount of programming practice.

2. LEARNING OUTCOMES

Learning outcomes are the key abilities and knowledge that will be assessed in this unit. See assessment summary table below for details of which outcomes are assessed where. Outcomes are listed according to the course goals that they support.

Engineering/IT Specialisation (Level 2)

1. Ability to learn to use Unix commands and system calls (including usage of flags etc) from online manual system.
2. Understanding of common memory-related errors (such as memory leaks, dangling pointers) and how to avoid these.Higher performance could involve detecting errors in example code, and fixing them using debuggers.
3. Ability to correctly implement standard linked list data structuresHigher performance could involve slightly more complicated structures such as binary search trees.
4. Ability to read and write correct, clean code in C that allocates, deallocates and manages memory.
5. Ability to read and write code that correctly uses the main standard library functions, especially for I/O, file handling, and string handling.Higher performance could involve elegant use of these functions, particularly avoiding idioms which are extremely inefficient.
6. Experience in using debugging tools.
7. Understanding of the approach and concepts of Unix, including its tools philosophy, processes (including pipes and redirection), the file system, and the shell.
8. Experience of following a thorough automated testing regime using tools such as make, diff, scripts to present the outcomes, and a tool to manage regression testing..Higher performance could involve ability to construct such a regime.

Professional Conduct (Level 2)

9. Ability to use code quality strategies appropriate for C, including preprocessor techniques, and use of common idioms.

For further details of course goals related to these learning outcomes, see online unit outline at http://cusp.eng.usyd.edu.au/students/view-unit-page/alpha/COMP2129 .

3. ASSESSMENT TASKS

ASSESSMENT SUMMARY

Assessment name	Team-based?	Weight	Due	Outcomes Assessed
Tutorial participation	No	10%	Multiple Weeks	2, 3, 4, 5, 6, 7, 8, 9
Final Exam	No	40%	Exam Period	1, 2, 3, 4, 5, 6, 7, 8, 9
*Assignment 1	No	4%	Week 3 (As specified by your unit coordinator)	1, 5, 7, 9
*Assignment 2	No	8%	Week 6 (As specified by your unit coordinator)	1, 2, 4, 5, 6, 7, 8, 9
*Assignment 3	No	10%	Week 11 (As specified by your unit coordinator)	1, 2, 3, 4, 5, 6, 8, 9
Assignment 4	No	10%	Week 13 (Friday, 9 am)	2, 3, 4, 5, 6, 7, 8, 9
Quiz 1	No	6%	Week 4 (As specified by your unit coordinator)	1, 5, 7, 8, 9
Quiz 2	No	6%	Week 9 (As specified by your unit coordinator)	1, 2, 3, 4, 5, 6, 7, 8, 9
Quiz 3	No	6%	Week 13 (As specified by your unit coordinator)	1, 2, 3, 4, 5, 6, 7, 8, 9

ASSESSMENT DESCRIPTION

* indicates an assessment task which must be repeated if a student misses it due to special consideration. A repeated assessment`s questions/tasks may vary from the original.

Quiz: Test knowledge and understanding of concepts and systems programming. Format is pen and paper. Attendance required.

Tutorial participation: attendance, contribution to tutorial discussion and performance in online challenges. The challenges are a series of programming exercises to be completed throughout the semester. Challenges are released regularly and have varying submission dates. Late enrolments (up to week 4) will be able to submit prior weeks.

Assignments 1-3

- Demonstrating programming ability from specification

- To be completed on campus using standard computer environment

Assignment 4: Assessed by electronic submission. Attendance to tutorial for that week is required as further assessment may be required as per the assignment specification. Late submissions within 3 days only, no re-weighting is possible with this assignment.

Final Exam: The final exam covers all aspects of the course and may involve answering questions about the C language and Unix system, reading short programs and writing short programs, and understanding concurrent and parallel programming.

Late assignment submission of challenges is 50% per day.

Late assignment submission of assignment is 25% per day.

Non-attendance for in lab assessment tasks, such as manual marking or quiz, will receive zero marks, unless special consideration is granted.

ASSESSMENT GRADING

Final grades in this unit are awarded at levels of HD for High Distinction, DI (previously D) for Distinction, CR for Credit, PS (previously P) for Pass and FA (previously F) for Fail as defined by University of Sydney Assessment Policy. Details of the Assessment Policy are available on the Policies website at http://sydney.edu.au/policies . Standards for grades in individual assessment tasks and the summative method for obtaining a final mark in the unit will be set out in a marking guide supplied by the unit coordinator.
It is a policy of the School of Information Technologies that in order to pass this unit, a student must achieve at least 40% in the written examination. For subjects without a final exam, the 40% minimum requirement applies to the corresponding major assessment component specified by the lecturer. A student must also achieve an overall final mark of 50 or more. Any student not meeting these requirements may be given a maximum final mark of no more than 45 regardless of their average.

4. ATTRIBUTES DEVELOPED

Attributes listed here represent the course goals designated for this unit. The list below describes how these attributes are developed through practice in the unit. See Learning Outcomes and Assessment sections above for details of how these attributes are assessed.

Attribute	Method
Design (Level 2)	design and implementation of a program to solve a specified problem
Engineering/IT Specialisation (Level 2)	fundamental skills in the Unix operating system at user level
Maths/Science Methods and Tools (Level 2)	fundamental skills in programming and a conceptual understanding of the link between the low level machine and the high level language
Professional Conduct (Level 2)	writing quality code following a systematic process

For further details of course goals and professional attribute standards, see the online version of this outline at http://cusp.eng.usyd.edu.au/students/view-unit-page/alpha/COMP2129 .

5. STUDY COMMITMENT

Laboratory: Laboratory classes help the student consolidate the material presented in lectures. Activities include development of small programs, practice quiz questions, and open discussion of topics in systems programming.

Activity	Hours per Week	Sessions per Week	Weeks per Semester
Lecture	2.00	1	13
Laboratory	2.00	1	12
Independent Study	12.00	1	13

Standard unit of study workload at this university should be from 1.5 to 2 hours per credit point which means 9-12 hours for a normal 6 credit point unit of study. For units that are based on research or practical experience, hours may vary. For lecture and tutorial timetable, see University timetable site at: web.timetable.usyd.edu.au/calendar.jsp

6. TEACHING STAFF AND CONTACT DETAILS

COORDINATOR(S)

Name	Room	Phone	Email	Contact note
Dr Stavrakakis, John			john.stavrakakis@sydney.edu.au	

LECTURERS

Name	Room	Phone	Email	Contact note
Dr Stavrakakis, John			john.stavrakakis@sydney.edu.au	

TUTORS

Tyson Thomas

Alan Robertson

Elie Moreau

Neill Foweraker

Yixing Zheng

Kosta Dunn

Simon Koch

Abdul Zreika

Greg McAllen

Scott Maxwell

William Wang

7. RESOURCES

PRESCRIBED TEXTBOOK(S)

Randal E. Bryant and David R. O`Hallaron, *Computer Systems: A Programmer`s Perspective* (3). Boston, Pearson Education, 2016. 9781292101767.

RECOMMENDED REFERENCES

Lin and Snyder, *Principles of Parallel Programming*. Pearson Education, 2008.

Jeri R. Hanly, Elliot B. Koffman, *Problem Solving and Program Design in C* (6th). Addison Wesley, 2010. 0321198034.

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*. Prentice Hall, 1988. 0-13-110362-8.

Paul Davies, *The Indispensable Guide to C*.

COURSE WEBSITE(S)

You will need to have UniKey credentials to access course material via elearning website

https://elearning.sydney.edu.au/

8. ENROLMENT REQUIREMENTS

ASSUMED KNOWLEDGE

INFO1105 OR INFO1905.

PREREQUISITES

INFO1103 OR INFO1903.

9. POLICIES

ACADEMIC HONESTY

While the University is aware that the vast majority of students and staff act ethically and honestly, it is opposed to and will not tolerate academic dishonesty or plagiarism and will treat all allegations of dishonesty seriously.

All students are expected to be familiar and act in compliance with the relevant University policies, procedures and codes, which include:

- *Academic Honesty in Coursework Policy 2015*
- *Academic Honesty Procedures 2016*
- *Code of Conduct for Students*
- *Research Code of Conduct 2013* (for honours and postgraduate dissertation units)

They can be accessed via the University"s Policy Register: http://sydney.edu.au/policies (enter "Academic Honesty" in the search field).

Students should never use document-sharing sites and should be extremely wary of using online “tutor” services. Further information on academic honesty and the resources available to all students can be found on the Academic Integrity page of the University website:

http://sydney.edu.au/elearning/student/EI/index.shtml

Academic Dishonesty and Plagiarism

Academic dishonesty involves seeking unfair academic advantage or helping another student to do so.

You may be found to have engaged in academic dishonesty if you:

- Resubmit (or "recycle") work that you have already submitted for assessment in the same unit or in a different unit or previous attempt;
- Use assignment answers hosted on the internet, including those uploaded to document sharing websites by other students.
- Have someone else complete part or all of an assignment for you, or do this for another student.
- Except for legitimate group work purposes, providing assignment questions and answers to other students directly or through social media platforms or document ("notes") sharing websites, including essays and written reports.
- Engage in examination misconduct, including using cheat notes or unapproved electronic devices (e.g., smartphones), copying from other students, discussing an exam with another person while it is in progress, or removing confidential examination papers from the examination venue.
- Engage in dishonest plagiarism.

Plagiarism means presenting another person's work as if it is your own without properly or adequately referencing the original source of the work.

Plagiarism is using someone else's ideas, words, formulas, methods, evidence, programming code, images, artworks, or musical creations without proper acknowledgement. If you use someone's actual words you must use quotation marks as well as an appropriate reference. If you use someone's ideas, formulas, methods, evidence, tables or images you must use a reference. You must not present someone's artistic work, musical creation, programming code or any other form of intellectual property as your own. If referring to any of these, you must always present them as the work of their creator and reference in an appropriate way.

Plagiarism is always unacceptable, regardless of whether it is done intentionally or not. It is considered dishonest if done knowingly, with intent to deceive or if a reasonable person can see that the assignment contains more work copied from other sources than the student's original work. The University understands that not all plagiarism is dishonest and provides students with opportunities to improve their academic writing, including their understanding of scholarly citation and referencing practices.

USE OF SIMILARITY DETECTION SOFTWARE

All written assignments submitted in this unit of study will be submitted to the similarity detecting software program known as **Turnitin**. Turnitin searches for matches between text in your written assessment task and text sourced from the Internet, published works and assignments that have previously been submitted to Turnitin for analysis.

There will always be some degree of text-matching when using Turnitin. Text-matching may occur in use of direct quotations, technical terms and phrases, or the listing of bibliographic material. This does not mean you will automatically be accused of academic dishonesty or plagiarism, although Turnitin reports may be used as evidence in academic dishonesty and plagiarism decision-making processes.

Computer programming assignments may also be checked by specialist code similarity detection software. The Faculty of Engineering & IT currently uses the MOSS similarity detection engine (see http://theory.stanford.edu/~aiken/moss/) . These programs work in a similar way to TII in that they check for similarity against a database of previously submitted assignments and code available on the internet, but they have added functionality to detect cases of similarity of holistic code structure in cases such as global search and replace of variable names, reordering of lines, changing of comment lines, and the use of white space.

IMPORTANT: School policy relating to Academic Dishonesty and Plagiarism.

In assessing a piece of submitted work, the School of IT may reproduce it entirely, may provide a copy to another member of faculty, and/or to an external plagiarism checking service or in-house computer program and may also maintain a copy of the assignment for future checking purposes and/or allow an external service to do so.

Other policies

See the policies page of the faculty website at http://sydney.edu.au/engineering/student-policies/ for information regarding university policies and local provisions and procedures within the Faculty of Engineering and Information Technologies.

10. WEEKLY SCHEDULE

Note that the "Weeks" referred to in this Schedule are those of the official university semester calendar https://web.timetable.usyd.edu.au/calendar.jsp

Week	Topics/Activities
Week 1	Lecture: Admin/Introduction to UNIX and C
Week 2	Lecture: Addressable memory, string and arrays
	Lab: Introduction to Unix and C
Week 3	Lecture: Memory management, structures and files
	Lab: Unix text processing and C pointer basics
	Assessment Due: *Assignment 1
Week 4	Lecture: Linked lists and debugging
	Lab: Unix shells scripts, C pointers, structs, unions and files
	Assessment Due: Quiz 1
Week 5	Lecture: Function pointers, common C functions, Unit testing
	Lab: Dynamic memory and debugging
Week 6	Lecture: Public holiday
	Lab: Program structure, common C errors
	Assessment Due: *Assignment 2
Week 7	Lecture: Parallelism and concurrency
	Lab: Compiler stages and linked lists
Week 8	Lecture: Thread synchronisation, POSIX threads
	Lab: Signals and IPC
Week 9	Lecture: Thread safety: Testing and Debugging
	Lab: Parallelism and concurrency
	Assessment Due: Quiz 2
Week 10	Lecture: Scalable algorithm templates
	Lab: Shared memory
Week 11	Lecture: Performance - memory and measure
	Lab: Thread constructs & reliability
	Assessment Due: *Assignment 3
Week 12	Lecture: TBA
	Lab: Recursion and advanced topics in C
Week 13	Lecture: Revision and Examination overview
	Lab: Revision
	Assessment Due: Assignment 4
	Assessment Due: Quiz 3
Exam Period	Assessment Due: Final Exam