COMP202

Systems Programming

S2 Day 2018

Dept of Computing

Contents

General Information

Unit convenor and teaching staff
Unit Convenor
Len Hamey
len.hamey@mq.edu.au
Contact via len.hamey@mq.edu.au
E6A (9WW) 327
Consultation after lectures or request an appointment by email

Lecturer
Young Lee
young.lee@mq.edu.au
Contact via young.lee@mq.edu.au
E6A (9WW) 376

Workshop tutor
Kym Haines
kym.haines@mq.edu.au

Workshop tutor
Damian Jurd
damian.jurd@mq.edu.au

Workshop tutor
Mahmood Yousefiazar
mahmood.yousefiazar@mq.edu.au

Workshop tutor
Daniel Sutantyo
daniel.sutantyo@mq.edu.au

Credit points
3

Prerequisites
COMP125

Corequisites

Co-badged status
ITEC692

Unit description
This unit studies the boundary between software and the systems on which software executes. It considers the impact of constraints imposed by operating systems and hardware systems on the design and performance of computer software. Students will learn how to operate within these constraints to build effective systems software. The unit studies these issues by looking below the abstractions provided by high-level programming languages. The unit is an introduction to systems programming and related issues. It provides an entry point into more advanced study of computer systems in the following areas: hardware implementation, operating systems, network design and programming, and programming language design and implementation.

Important Academic Dates

Information about important academic dates including deadlines for withdrawing from units are available at https://www.mq.edu.au/study/calendar-of-dates

Learning Outcomes

On successful completion of this unit, you will be able to:

Be able to produce code that utilises system software to provide controlled access to system resources.

Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.

Be able to produce software that meets performance expectations on given hardware.

Be able to analyze machine representations of software and assemble and disassemble simple code fragments.

Be able to identify security flaws that can arise from program execution.

General Assessment Information

Grading Requirements

The assessment has two components: the practical tasks and the exam components (including quizzes and the final examination). The final mark for the unit will be calculated by combining the marks for all assessment tasks according to the percentage weightings shown in the assessment summary.

Practical Assignment Submissions

Submission instructions for practical assignments will be provided on iLearn. The assiggments contain differing forms of automatic assessment and feedback tools that will assist you during the assignment and contribute to our understanding of your performance in the assignment.

Each practical assignment will specify a particular due date. The due time will be 11:59pm on the

due date.

Late submissions

Submissions are made through the lab command (unless specified otherwise in the individual practical task specification). The lab command automatically enforces submission deadlines. However, there is provision for extensions in appropriate circumstances.

Free Extension Days

Throughout the unit, you can claim up to a total of **three free extension days**. Free extension days can be used for any purpose, ranging from a single sick day to a sudden realisation of how to improve your solution. A free extension does not require any formal documentation, and you can apply it instantly through the lab command - see the lab command documentation in the Lab Notes section on iLearn.

Free extensions are valuable and you are encouraged to meet the deadlines of the practical tasks whenever possible and conserve your free extension days in case you need them later in the semester. If you have free extension days remaining at the end of the semester, you may apply them to your last assignment due date.

Free extensions can be applied to any due date - assignment closing dates or progress mark due dates. In the opinion of the convenor, using a free extension for a progress mark due date is probably a waste of the free extension because the remaining progress marks and the final due date for the assignment will not be changed by the extension of a single progress mark deadline. However, if you believe that extending the progress mark deadline is in your best interests, then please see the lab command documentation for details on how to do this.

Extensions Due to Disruption

You may apply for special consideration under the Disruption to Studies Policy http://www.mq.edu.au/policy/docs/disruption_studies/policy.html. If the disruption request is accepted, the action may be to grant an extension of the due date of the assignment, or it may be to require you to submit an alternative assessment item.

If you are applying under the Disruption to Studies policy, please apply promptly and also email the convenor at len.hamey@mq.edu.au at the same time. If an application is not received soon after the deadline, it may not be possible to sensibly grant an extension to the existing practical assignment. For example, if you are ill for two days just before the closing date of an assignment, an extension of two days is likely to be appropriate, but it would not be possible to grant an extension of two days if the application is received by the convenor more than two days after the due date for the assignment! If your disruption application is late, you risk being required to complete a different assignment - this would mean starting from scratch.

If you have free extension days available, you can claim a short free extension to enable you to complete the assignment, and at the same time lodge a formal disruption request asking for the extension that you need. You should email len.hamey@mq.edu.au to say that you want the formal disruption extension to replace your free extension. If a disruption extension is granted, Len will then manually convert your free extension to a disruption extension (as appropriate) and your free extension days will again be available. If the disruption extension is not granted, your

free extension days will have been used.

Combining Disruption and Free Extensions

You can claim a free extension on top of a formal disruption extension - the two extensions will be added together. If that is not what you want, you should email len.hamey@mq.edu.au as discussed above.

Assessment Tasks

Name	Weighting	Hurdle	Due
Practical	40%	Yes	As specified in each task
Quizzes	10%	No	Weeks 7, 12
Examination	50%	No	Final examination

Practical

Due: **As specified in each task**
Weighting: **40%**
This is a hurdle assessment task (see assessment policy for more information on hurdle assessment tasks)

Throughout the semester you will engage in two practical tasks that are assessed individual work. These tasks develop your skills and assess your progress. These practical tasks are to be done as a major part of your weekly lab sessions, but also in your own time. They are due as specified in each task.

There are progress marks associated with each practical. You gain the progress mark each week by achieving a milestone in the practical work. If you do not achieve the milestone for a particular progress mark, then you lose the progress mark and your total number of progress marks is reduced. If you achieve that milestone by the due time of a later progress mark, then you will be awarded the later progress mark. You can earn the progress marks in advance by achieving the milestones early, but you cannot gain the progress marks by completing the work late in a hurry.

The two practical assignments are a hurdle in this unit. You must achieve at least 8 marks out of 20 in one of the two of the assignments in order to pass the unit.

On successful completion you will be able to:

- Be able to produce code that utilises system software to provide controlled access to system resources.

- Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.
- Be able to produce software that meets performance expectations on given hardware.
- Be able to analyze machine representations of software and assemble and disassemble simple code fragments.
- Be able to identify security flaws that can arise from program execution.

Quizzes

Due: **Weeks 7, 12**
Weighting: **10%**

Two in-class quizzes will be held to help you monitor your progress in the unit. Quizzes will be held in tutorial classes. Details will be announced on iLearn.

On successful completion you will be able to:

- Be able to produce code that utilises system software to provide controlled access to system resources.
- Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.
- Be able to produce software that meets performance expectations on given hardware.
- Be able to analyze machine representations of software and assemble and disassemble simple code fragments.
- Be able to identify security flaws that can arise from program execution.

Examination

Due: **Final examination**
Weighting: **50%**

The final examination will assess your understanding of the unit content and your ability to integrate concepts learned throughout the unit to solve problems.

On successful completion you will be able to:

- Be able to produce code that utilises system software to provide controlled access to system resources.
- Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.
- Be able to produce software that meets performance expectations on given hardware.
- Be able to analyze machine representations of software and assemble and disassemble simple code fragments.

- Be able to identify security flaws that can arise from program execution.

Delivery and Resources

Text Book

"Computer Systems: A Programmer's Perspective" 3rd edition, R.E. Bryant and D.R. O'Hallaron, Pearson 2015.

You are required to purchase a copy of the unit text and read the set sections each week.

Recommended Text

"The C Programming Language" 2nd edition, Brian W Kernighan and Dennis M Ritchie, Prentice-Hall 1988.

This small book is the classic reference on C programming.

iLearn Web Site

All learning materials will be published on iLearn including lecture slides, practical tasks and tutorial questions.

You are required to check the iLearn website at least once a week to ensure that you are aware of the latest materials available there.

Lectures

Lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Key ideas for the practical assignments will be discussed from time to time in lectures. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording.

Workshops

Each week you should prepare your solutions to the set tutorial questions and attend your enrolled workshop. Workshops provide an opportunity to ask specific questions related to any aspect of the unit: the lecture content, the practical tasks and the set tutorial questions.

Workshops provide an opportunity for you to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set assignment tasks, each of approximately 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical assignment will be explained in the asignment specification along with the due date. The assignments will be posted on iLearn.

Your practical demonstrator will provide you with individual assistance and may also from time to time address the entire class to provide useful information to assist with the practical assignment.

The practical assignments are set as individual work and each one is personalised for a particular student. The University's academic honesty policy will be enforced. You may assist your fellow students with general concepts, pointers to resources and useful tools or commands.

You may **not** become involved in any way in helping a fellow student to find the solution to their particular assignment, nor may you share with them any aspect of the solution of your particular assignment. Each practical assignment will include specific instructions to help you understand what is acceptable and what is not.

Each practical assignment must be the sole work of the student turning it in. Any cheating will be handled under the University's Academic Honesty Policy.

The following are guidelines on what collaboration is allowed for laboratory exercises and what is not [adapted from CS:APP website]

What is Cheating?

- *Sharing code or other electronic files*: either by copying, retyping, looking at, or supplying a copy of a file from this or a previous semester. Be sure to store your work in protected directories, and log off when you leave any lab computer, to prevent others from copying your work without your explicit assistance. Don't use a public repository (such as github) for your assignment files.
- *Sharing written assignments, quizzes or exams*: Looking at, copying, or supplying an assignment, quiz or exam.
- *Using other's code:* Using code that you did not write yourself. You may not use code from courses at other institutions, or from any other non-202 source (e.g., software found on the Internet). You may, however, use Internet and other sources to learn useful concepts, ideas, and programming idioms. If you find helpful ideas on the Internet or in other sources, you should acknowledge them in comments in your code. For Internet sources, a URL is sufficient acknowledgement.
- *Looking at other's code.* Although mentioned above, it bears repeating. Looking at other students' code or allowing others to look at yours is cheating. There is no notion of looking "too much," since no looking is allowed at all.

What is NOT Cheating?

- Clarifying ambiguities or vague points in class handouts or textbooks.
- Helping others use the computer systems, networks, compilers, debuggers, profilers, or other system facilities.
- Helping others with high-level design issues.
- Helping others with high-level (not code-based) debugging techniques.
- Using code from the CS:APP website or from the class Web pages.
- Learning programming idioms and techniques from examples.
- Reading Unix manual pages, forums, etc in order to find out how to perform particular tasks (e.g. set a breakpoint in a debugger) or use programming language/library features

(such as printf).

- Asking for help from the practical demonstrator, tutor or lecturer.

Be sure to store your work in protected directories, and log off when you leave any lab computer, to prevent others from copying your work without your explicit assistance.

Unit Forum

A forum for unit discussions is provided on iLearn. Students are free to post questions, comments or hints in relation to any aspect of the unit, except that you should avoid posting any questions, hints, comments or solutions that could be interpreted as cheating (see Workshops, above).

If you see a post fom another student that appears to be cheating, please notify your demonstrator, tutor or the lecturer as soon as possible and we will remove the offending post.

Unit Schedule

The detailed unit schedule will be available on iLearn. The unit is organised into two 6-week periods, with topics approximately as follows.

Week 1-6: C programming, Data representations (integer and float), some system API.

Weeks 7-12: Assembly code and how C programs appear in the machine, Operating System features and implementation.

Learning and Teaching Activities

Lecture

Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.

Workshop

Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Self-Test Quiz

Weekly short quizzes are provided with solutions on iLearn. Use these quizzes to assess your own learning and as additional practice in preparation for the in-class quizzes and examination.

Policies and Procedures

Macquarie University policies and procedures are accessible from Policy Central (https://staff.mq.edu.au/work/strategy-planning-and-governance/university-policies-and-procedures/policy-central). Students should be aware of the following policies in particular with regard to Learning and Teaching:

- Academic Appeals Policy
- Academic Integrity Policy
- Academic Progression Policy
- Assessment Policy
- Fitness to Practice Procedure
- Grade Appeal Policy
- Complaint Management Procedure for Students and Members of the Public
- Special Consideration Policy *(**Note:** The Special Consideration Policy is effective from 4 December 2017 and replaces the Disruption to Studies Policy.)*

Undergraduate students seeking more policy resources can visit the Student Policy Gateway (https://students.mq.edu.au/support/study/student-policy-gateway). It is your one-stop-shop for the key policies you need to know about throughout your undergraduate student journey.

If you would like to see all the policies relevant to Learning and Teaching visit Policy Central (https://staff.mq.edu.au/work/strategy-planning-and-governance/university-policies-and-procedures/policy-central).

Student Code of Conduct

Macquarie University students have a responsibility to be familiar with the Student Code of Conduct: https://students.mq.edu.au/study/getting-started/student-conduct

Results

Results shown in *iLearn*, or released directly by your Unit Convenor, are not confirmed as they are subject to final approval by the University. Once approved, final results will be sent to your student email address and will be made available in eStudent. For more information visit ask.mq.edu.au.

Practicals

The practical tasks in this unit are used at other Universities internationally. You must not publish your solutions on the Internet in any form.

You must carefully follow the guidelines laid out above and in each practical task concerning what are and are not acceptable ways to interact with other students.

Student Support

Macquarie University provides a range of support services for students. For details, visit http://stu

dents.mq.edu.au/support/

Learning Skills

Learning Skills (mq.edu.au/learningskills) provides academic writing resources and study strategies to improve your marks and take control of your study.

- Workshops
- StudyWise
- Academic Integrity Module for Students
- Ask a Learning Adviser

Student Services and Support

Students with a disability are encouraged to contact the Disability Service who can provide appropriate help with any issues that arise during their studies.

Student Enquiries

For all student enquiries, visit Student Connect at ask.mq.edu.au

IT Help

For help with University computer systems and technology, visit http://www.mq.edu.au/about_us/offices_and_units/information_technology/help/.

When using the University's IT, you must adhere to the Acceptable Use of IT Resources Policy. The policy applies to all who connect to the MQ network including students.

Graduate Capabilities

Creative and Innovative

Our graduates will also be capable of creative thinking and of creating knowledge. They will be imaginative and open to experience and capable of innovation at work and in the community. We want them to be engaged in applying their critical, creative thinking.

This graduate capability is supported by:

Learning outcome

- Be able to produce software that meets performance expectations on given hardware.

Assessment tasks

- Practical
- Quizzes
- Examination

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical

underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.

- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Capable of Professional and Personal Judgement and Initiative

We want our graduates to have emotional intelligence and sound interpersonal skills and to demonstrate discernment and common sense in their professional and personal judgement. They will exercise initiative as needed. They will be capable of risk assessment, and be able to handle ambiguity and complexity, enabling them to be adaptable in diverse and changing environments.

This graduate capability is supported by:

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.
- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Commitment to Continuous Learning

Our graduates will have enquiring minds and a literate curiosity which will lead them to pursue knowledge for its own sake. They will continue to pursue learning in their careers and as they participate in the world. They will be capable of reflecting on their experiences and relationships with others and the environment, learning from them, and growing - personally, professionally and socially.

This graduate capability is supported by:

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.
- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Discipline Specific Knowledge and Skills

Our graduates will take with them the intellectual development, depth and breadth of knowledge, scholarly understanding, and specific subject content in their chosen fields to make them competent and confident in their subject or profession. They will be able to demonstrate, where relevant, professional technical competence and meet professional standards. They will be able to articulate the structure of knowledge of their discipline, be able to adapt discipline-specific knowledge to novel situations, and be able to contribute from their discipline to inter-disciplinary solutions to problems.

This graduate capability is supported by:

Learning outcomes

- Be able to produce code that utilises system software to provide controlled access to system resources.
- Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.
- Be able to produce software that meets performance expectations on given hardware.
- Be able to analyze machine representations of software and assemble and disassemble simple code fragments.
- Be able to identify security flaws that can arise from program execution.

Assessment tasks

- Practical
- Quizzes
- Examination

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.
- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.
- Weekly short quizzes are provided with solutions on iLearn. Use these quizzes to assess your own learning and as additional practice in preparation for the in-class quizzes and examination.

Critical, Analytical and Integrative Thinking

We want our graduates to be capable of reasoning, questioning and analysing, and to integrate and synthesise learning and knowledge from a range of sources and environments; to be able to critique constraints, assumptions and limitations; to be able to think independently and systemically in relation to scholarly activity, in the workplace, and in the world. We want them to have a level of scientific and information technology literacy.

This graduate capability is supported by:

Learning outcomes

- Be able to explain how software can interact directly with the hardware without the mediating influence of a complex operating system.
- Be able to identify security flaws that can arise from program execution.

Assessment tasks

- Practical
- Quizzes
- Examination

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some

lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.

- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Problem Solving and Research Capability

Our graduates should be capable of researching; of analysing, and interpreting and assessing data and information in various forms; of drawing connections across fields of knowledge; and they should be able to relate their knowledge to complex situations at work or in the world, in order to diagnose and solve problems. We want them to have the confidence to take the initiative in doing so, within an awareness of their own limitations.

This graduate capability is supported by:

Learning outcomes

- Be able to produce code that utilises system software to provide controlled access to system resources.
- Be able to produce software that meets performance expectations on given hardware.
- Be able to analyze machine representations of software and assemble and disassemble simple code fragments.
- Be able to identify security flaws that can arise from program execution.

Assessment tasks

- Practical
- Quizzes
- Examination

Learning and teaching activities

- Weekly lectures are a core learning experience where we will discuss the theoretical underpinnings and concepts that are essential to this unit. Attendance at lectures is not required but is highly recommended. Lectures will be recorded on echo360 but some lectures will include interactive components that may not be adequately captured by a recording. Attendance may be taken at lectures.
- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts

of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

- Weekly short quizzes are provided with solutions on iLearn. Use these quizzes to assess your own learning and as additional practice in preparation for the in-class quizzes and examination.

Effective Communication

We want to develop in our students the ability to communicate and convey their views in forms effective with different audiences. We want our graduates to take with them the capability to read, listen, question, gather and evaluate information resources in a variety of formats, assess, write clearly, speak effectively, and to use visual communication and communication technologies as appropriate.

This graduate capability is supported by:

Learning and teaching activities

- Weekly workshop sessions provide an opportunity for you to ask questions and to develop your skills in systems programming and your understanding of the key concepts of the unit. Practical work will consist of set tasks, each of 4 weeks duration, which are assessed and contribute to your final mark. The assessment criteria for each practical task will be explained in the task specification along with the due date. The tasks will be posted on iLearn.

Changes from Previous Offering

Reduced from three assignments to two.

Restructure lecture content into two 6-week modules.

Grading

See also the section General Assessment Information, above.

This unit will be graded Fail, Pass, Credit, Distinction or High Distinction as defined in the Macquarie University Grading Policy.

Detailed achievement levels for each learning outcome are described as follows.

Criteria	Developing	Functional	Proficient	Advanced
LO#1. Use of system/library calls in program code	Inappropriate or inadequate use of system/library calls in program code	Uses system/library calls where required	Takes opportunities to use system/library calls in preference to writing their own code	Uses system/library calls consistently in preference to writing their own code

Knowledge of system/library interface	Unaware of appropriate system/ library facilities	Can identify relevant system/ library facilities	Can describe application of system/library facilities	Can describe operation/ implementation of system/library facilities
LO#2. Explain software directly interacting with hardware	Inability to explain or inaccurate explanation	Explanation at a superficial level	Explanation demonstrates understanding of hardware capabilities used by software	Explanation demonstrates deep understanding of hardware/software interface
LO#3. Software meeting performance expectations	Unable to identifty performance-related software design issues	Identification of performance-related software design issues	Modifies software design to improve performance	Can discuss performance issues in depth
LO#4. Analysis of machine code	Unable to identify key features of machine code	Explains/reverse engineers key structures in machine code but some identification is incorrect or partial	Accurately identifies structures in machine code and able to explain/reverse engineer short pieces of code	Able to interpret/ reverse engineer significant pieces of machine code
LO#5. Identify security flaws	Unable to recognise common security flaws; incorrectly identifies security flaws	Able to identify common security flaws such as buffer overflow opportunity	Able to identify security flaws and explain problem	Able to identify security flaws, explain problem and offer solution

Changes since First Published

Date	Description
30/ 07/ 2018	Add Daniel Sutantyo as staff. Correct hurdle requirement for assignments: 8 out of 20 (was 6 out of 15 which is equivalent but assignments are now marked out of 20).