COMP2100

Systems Programming

Session 2, Special circumstances 2021

School of Computing

Contents

Session 2 Learning and Teaching Update

The decision has been made to conduct study online for the remainder of Session 2 for all units WITHOUT mandatory on-campus learning activities. Exams for Session 2 will also be online where possible to do so.

This is due to the extension of the lockdown orders and to provide certainty around arrangements for the remainder of Session 2. We hope to return to campus beyond Session 2 as soon as it is safe and appropriate to do so.

Some classes/teaching activities cannot be moved online and must be taught on campus. You should already know if you are in one of these classes/teaching activities and your unit convenor will provide you with more information via iLearn. If you want to confirm, see the list of units with mandatory on-campus classes/teaching activities.

Visit the MQ COVID-19 information page for more detail.

General Information

Unit convenor and teaching staff
Convenor, Lecturer
Richard Han
richard.han@mq.edu.au

Lecturer
Young Lee
young.lee@mq.edu.au

Senior Tutor
Kym Haines
kym.haines@mq.edu.au

Tutor
Kate Stefanov
kate.stefanov@mq.edu.au

Tutor
Alex Taylor
alexander.taylor@mq.edu.au

Tutor
Mahmood Yousefiazar
mahmood.yousefiazar@mq.edu.au

Credit points
10

Prerequisites
COMP1010 or COMP125

Corequisites

Co-badged status
COMP6100

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:

ULO1: Use a command line interface to control the computer.

ULO2: Produce code that utilises system software to provide controlled access to system resources.

ULO3: Analyze machine representations of data, such as integer and floating point.

ULO4: Analyze assembly code fragments. Trace execution and re-express with higher level descriptions.

ULO5: Identify security flaws that can arise from program execution.

ULO6: Explain how the hardware impacts on software performance and how software can interact directly with the hardware.

General Assessment Information

Grading Requirements

The assessment has three components: the programming tasks, workshop participation 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.

Programming Tasks

The two programming tasks together constitute a hurdle assessment task (see assessment policy for more information on hurdle assessment tasks)

Throughout the semester you will engage in two programming tasks that are assessed individual

work. These tasks develop your skills and assess your progress. These programming tasks are to be done in your own time, but you may also use workshop time to work on them and to seek assistance from your workshop tutor. They are due as specified on iLearn for each task.

The programming tasks are designed to be completed over several weeks. Do not leave them until the last week. To encourage you to work throughout the period when the tasks are available, there are **progress marks** associated with each task. You gain the progress marks by achieving milestones specified for the task by their due dates. Each milestone consists of a specified achievement level in a specified stage of the programming task. If you do not achieve the milestone for a particular progress mark by the due time, then you lose the corresponding progress mark and your total progress mark score is reduced. If you subsequently achieve the missed milestone by the due time of a later progress milestone, then you will be awarded that later progress mark. You can only be awarded each progress mark once. You can earn the progress marks in advance by achieving the milestones early, but you cannot recover lost progress marks by completing the work later.

The two programming tasks are a **hurdle** in this unit. You must achieve at least 10 marks out of 25 in one of the two of the programming tasks in order to pass the unit. In practice, you are unlikely to pass the unit unless you achieve at least pass marks (12.5 out of 25) in both of the programming tasks.

Programming Tasks are Individual Work

The programming tasks 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 that are publicly available. You may **not** become involved in any way in helping a fellow student to find the solution to their particular task, nor may you share with them any aspect of the solution of your particular task. If you decide to develop or modify a tool (including software tools, procedures or methods) to assist you in solving your programming task, you may not provide that tool to your fellow students, nor may you publish it. Each practical task will include additional specific instructions to help you understand what is acceptable and what is not.

Each programming task 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 programming tasks 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 programming task files.
- *Sharing solutions, quizzes or exams*: Looking at, copying, or supplying a programming

task solution, quiz or exam.

- *Sharing procedures, methods or tools* that have been developed to solve challenges of a programming task. If you wish, you may develop your own procedures, methods or tools to assist you in the tasks, but you may not share them with others, publish them or store them in a public repository such as github. If you find someone else's procedures, methods or tools, you may not use them -- if in doubt, ask your tutor or the unit convenor. For example, you may not provide or use somebody else's sequence of steps that can be followed to solve (or partially solve) a programming task. Similarly, you may not provide or use somebody else's program that is designed to solve (or partially solve) a stage of a programming task.
- *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-COMP2100 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 COMP2100 iLearn 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.

Programming Task Submissions

Submission instructions for programming tasks will be provided on iLearn. The two tasks contain differing forms of automatic assessment and feedback tools that will assist you during the task. The automated feedback will help you identify your mistakes, but does not replace your own analysis of the problem, debugging and testing. The automated assessment contributes strongly to our understanding of your performance in the task.

Each programming task will specify a particular due date. The due time will be 5:00pm on the due date unless otherwise specified.

Late submissions

Submissions in Data File Lab are made through the lab command; submissions in Binary Bomb Lab are made automatically as you progress through the task of analysing and executing the "binary bomb" program. 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** for due dates in the programming tasks. 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 use one or more of your free days 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 programming 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 the closing date of the last programming task.

Free extensions can be applied to any due date - task 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 task 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.

Special Consideration

If you experience serious and unavoidable difficulties that affect your ability to meet the due dates for progress or the closing date of a programming task, you may apply for special consideration as explained at https://students.mq.edu.au/study/my-study-program/special-consideration. If the request is accepted, the action may be to grant an extension of the relevant due date(s), or it may be to require you to submit an alternative assessment item. Extensions, if granted, are managed through the automated assessment system that you access via the lab command.

If you apply for special consideration, please note:

- Apply promptly. Late applications may make it impossible to sensibly offer an extension, and you may risk having to complete a different assessment task which would mean starting from scratch. For example, if you are ill for two days just before the due date, an extension of two days would be reasonable, but that extension cannot be granted more than two days after the due date since the extension end date would have already passed!
- Email the convenor and unit lecturer to let us know what is happening. This will make it easier for us to respond in a timely manner.
    - During weeks 1-6, email young.lee@mq.edu.au and also the convenor len.hamey@mq.edu.au
    - During weeks 7-13, email len.hamey@mq.edu.au
- Use free extension days initially. If you have free extension days available, you can use them to claim a short free extension to enable you to continue working on the programming task and achieve your goal. At the same time, you can lodge a formal special consideration request asking for the consideration that you need. Send an email as above, letting us know that you have used your free extension days for special consideration. If the special consideration is granted as an extension, we will convert your free extension into a formal special consideration extension (the lab command calls this an 'authorised' extension) and your free extension days will be returned to you. However, if the special consideration does not result in an extension, your free extension days will have been used.

Combining Special Consideration and Free Extensions

You can claim a free extension on top of a formal special consideration extension - the two extensions will be added together. If that is not what you want, you should email as discussed above so that if special consideration is granted, your free extension can be converted to an authorised extension.

Workshop Participation

Workshop tutors will record participation in the workshops, with 1 mark allocated to each week's participation. To obtain full marks, you should participate in at least ten (10) workshops. You can request special consideration if serious and unavoidable difficulties prevent you from participating in workshops. An alternative participation assessment is a likely outcome; for example, participating in online discussion and/or submitting your solutions to the workshop questions.

Weekly Online Quizzes

Short online quizzes will be provided in iLearn relevant to the content of each of weeks 1-12. The quizzes are intended to help you assess your progress in learning and highlight areas that

you need to study further. Each quiz will be available for two weeks. You may attempt a quiz multiple times.

The quizzes contribute up to 10% of your final mark, based on a maximum value of 1% for each quiz. You can achieve full marks for the quizzes by earning a total of at least 10 marks in the quizzes.

Assessment Tasks

Name	Weighting	Hurdle	Due
Final examination	30%	No	Exam period
Binary Bomb Lab	25%	Yes	Week 13
Data File Lab	25%	Yes	Week 7
Workshop participation	10%	No	Every week
Weekly online quiz	10%	No	Every week

Final examination

Assessment Type [1]: Examination
Indicative Time on Task [2]: 2 hours
Due: **Exam period**
Weighting: **30%**

Final examination.

On successful completion you will be able to:

- Use a command line interface to control the computer.
- Produce code that utilises system software to provide controlled access to system resources.
- Analyze machine representations of data, such as integer and floating point.
- Analyze assembly code fragments. Trace execution and re-express with higher level descriptions.
- Identify security flaws that can arise from program execution.
- Explain how the hardware impacts on software performance and how software can interact directly with the hardware.

Binary Bomb Lab

Assessment Type [1]: Programming Task
Indicative Time on Task [2]: 35 hours
Due: **Week 13**
Weighting: **25%**
This is a hurdle assessment task (see assessment policy for more information on hurdle assessment tasks)

Use debugging tools to understand the behaviour of a precompiled program. Find input strings that will be acceptable to the program (i.e. defuse the binary bomb). Marks are awarded for defusing each of several phase puzzles, and a limited number are marks are deducted for exploding the bomb. Students receive immediate feedback on their attempts, and can monitor their final mark for this assignment in real time. Each student has an individualised task.

On successful completion you will be able to:

- Use a command line interface to control the computer.
- Analyze machine representations of data, such as integer and floating point.
- Analyze assembly code fragments. Trace execution and re-express with higher level descriptions.
- Identify security flaws that can arise from program execution.

Data File Lab

Assessment Type [1]: Programming Task
Indicative Time on Task [2]: 35 hours
Due: **Week 7**
Weighting: **25%**
This is a hurdle assessment task (see assessment policy for more information on hurdle assessment tasks)

Develop C programs to manipulate data files. Submissions are automatically marked with immediate feedback to assist learning. Multiple submissions are permitted. Most marking is automatic and immediate; however, a portion of marks (up to 5 marks out of 20) is awarded after the assignment submissions are closed, based on instructor assessment of program writing style compared to published guidelines. Each student has an individualised task.

On successful completion you will be able to:

- Use a command line interface to control the computer.
- Produce code that utilises system software to provide controlled access to system resources.
- Analyze machine representations of data, such as integer and floating point.

Workshop participation

Assessment Type [1]: Participatory task
Indicative Time on Task [2]: 0 hours
Due: **Every week**
Weighting: **10%**

Students will participate in online workshop classes.

On successful completion you will be able to:

- Use a command line interface to control the computer.
- Produce code that utilises system software to provide controlled access to system resources.
- Analyze machine representations of data, such as integer and floating point.
- Analyze assembly code fragments. Trace execution and re-express with higher level descriptions.
- Identify security flaws that can arise from program execution.
- Explain how the hardware impacts on software performance and how software can interact directly with the hardware.

Weekly online quiz

Assessment Type [1]: Quiz/Test
Indicative Time on Task [2]: 3 hours
Due: **Every week**
Weighting: **10%**

Students will complete a short quiz online each week. Multiple attempts are permitted.

On successful completion you will be able to:

- Use a command line interface to control the computer.
- Produce code that utilises system software to provide controlled access to system

resources.

- Analyze machine representations of data, such as integer and floating point.
- Analyze assembly code fragments. Trace execution and re-express with higher level descriptions.
- Identify security flaws that can arise from program execution.
- Explain how the hardware impacts on software performance and how software can interact directly with the hardware.

---

[1] If you need help with your assignment, please contact:

- the academic teaching staff in your unit for guidance in understanding or completing this type of assessment
- the Writing Centre for academic skills support.

[2] Indicative time-on-task is an estimate of the time required for completion of the assessment task and is subject to individual variation

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 read set sections of the text book each week. The text book is available in electronic form in the library, or you can purchase a printed copy from a book seller of your choice.

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, programming 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 programming tasks will be discussed

from time to time in lectures.

Lectures will be provided online and may consist of pre-recorded lectures and online live sessions. Participation in lectures is not required but is highly recommended. Lecture recordings will be provided on echo360.

Workshops

Each week you should prepare your solutions to the set tutorial questions and attend your enrolled workshop. Workshops may be on campus or online. Workshops provide an opportunity to discuss specific questions related to any aspect of the unit: the lecture content, the programming 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. Workshops combine tutorial-style discussion with practical programming experience. Your workshop tutor will provide you with individual assistance and may also from time to time address the entire class to provide useful information to assist with programming tasks.

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 above).

Don't post your code, or anyone else's code, on the forums!

If you see a post from another student that appears to be cheating, please notify the "senior workshop tutor" as soon as possible and we will remove the offending post.

Senior Workshop Tutor

An experienced tutor will be given the role of senior workshop tutor (see the staff list at the start of this document). This tutor is your primary contact for email enquiries related to unit content, and is the most likely person to respond to posts on the forum. For enquiries about the administration of the unit, please contact the unit convenor.

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, Command line interface, 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.

Policies and Procedures

Macquarie University policies and procedures are accessible from Policy Central (https://policies.mq.edu.au). 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

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

To find other policies relating to Teaching and Learning, visit Policy Central (https://policies.mq.edu.au) and use the search tool.

Student Code of Conduct

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

Results

Results published on platform other than eStudent, (eg. iLearn, Coursera etc.) 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 or if you are a Global MBA student contact globalmba.support@mq.edu.au

Student Support

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

Learning Skills

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

- Getting help with your assignment
- Workshops
- StudyWise
- Academic Integrity Module

The Library provides online and face to face support to help you find and use relevant information resources.

- Subject and Research Guides
- Ask a Librarian

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

If you are a Global MBA student contact globalmba.support@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.

Changes from Previous Offering

Lectures are online as a result of the pandemic.