# COMP202

## Systems Programming

S2 Day 2014

*Computing*

## Contents


| | |
|---|---|
| General Information | 2 |
| Learning Outcomes | 2 |
| General Assessment Information | 3 |
| Assessment Tasks | 4 |
| Delivery and Resources | 5 |
| Unit Schedule | 7 |
| Policies and Procedures | 8 |
| Graduate Capabilities | 9 |
| Changes since First Published | 12 |

# General Information

Unit convenor and teaching staff
Unit Convenor
Len Hamey
len.hamey@mq.edu.au
Contact via len.hamey@mq.edu.au
E6A327
Consultation after lectures or request an appointment by email

Credit points
3

Prerequisites
COMP125

Corequisites

Co-badged status

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). In order to pass this unit, you must perform satisfactorily in each component separately and also achieve a total mark that represents a pass grade.

You must make a significant attempt at each of the five practical tasks. If you do not meet this requirement, we may award require you to perform additional work in order to complete the unit.

## Practical Submissions

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

### Late submissions

We will determine the date of submission of your assignment by the last time that you submitted your solution. This means that if you submit the assignment, then discover that you have made a mistake and resubmit it after the due time, your submission will be late.

Each task will specify a particular due date which will be a Wednesday. The due time will be 1:59:59pm on the Wednesday of the due week. If the time of your submission shown on the system is after 1:59:59pm then you will be considered late. In order to allow time for marking, we will not accept any submission that is more than two days late except in severe circumstances. A disruption of one or two days' duration, even if it falls at the due date, is not severe because we offer you free late days that you can use in such circumstances.

The penalty for late submission is 20% of your mark per day for up to two days. A submission that is 5 minutes late will be considered to be 1 day late and incur a penalty of 20% unless you have free late days remaining (see below).

If your performance in a practical task is severely affected by disruption, please (1) notify us as soon as possible by email, (2) submit whatever work you have accomplished as soon as possible after the due date and (3) submit a formal disruption request.

### Free late days

Throughout the semester we offer you three free late days. This means that we will remove the penalty that would normally apply to each of the first three days of late practical submission. For example, if you submit each of the first three practicals two days late, then we would apply two

free late days to the first practical so there would be no late penalty, one free late day to the second practical so the penalty would be 20% and you would have no free late days for the third practical so the penalty would be 40%. We encourage you to preserve your free late days (by submitting on time) in case you need them later in the semester.

# Assessment Tasks

| Name | Weighting | Due |
|---|---|---|
| Practical | 50% | As specified in each task |
| Quizzes | 10% | Weeks 7, 12 |
| Examination | 40% | Final examination |

## Practical

Due: **As specified in each task**
Weighting: **50%**

Throughout the semester you will engage in five practical tasks that are assessed individual work. These tasks develop your skills and assess your progress. They are due as specified in each task.

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. 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: **40%**

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" 2nd edition, R.E. Bryant and D.R. O'Hallaron, Pearson 2011.

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. 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.

## Tutorials

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

## Practicals

Practical sessions 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 tasks, each of 2-4 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.

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 task.

The practical 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. 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. Each practical task will include specific instructions to help you understand what is acceptable and what is not.

The practical tasks in this unit are used at other Universities internationally. You will be required to sign an agreement for acceptable use of these materials before you will be given access. This agreement is to ensure that you do not post any questions, hints or solutions to the practical exercises to any Internet website or forum (other than the iLearn forum for this unit). Any violation of your agreement will be considered a serious offence.

Each lab 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.
- *Sharing written assignments or exams*: Looking at, copying, or supplying an assignment 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).
- *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.
- 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 Practicals, 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 following is an approximate schedule and is subject to change. In all cases, refer to iLearn for up-to-date information.

| Week | Topic | Practical | Quizz |
|---|---|---|---|
| 1 | Unit outline, C Programming introduction | intro | |
| 2 | Bits, ints, float | 1: Data | |
| 3 | Machine level programs - basics, control, procedures | | |
| 4 | Machine level programs - data, advanced | 2: Bomb | |
| 5 | Memory hierarchies and caches | | |
| 6 | Linking, exceptions | | |
| 7 | Signals, I/O | 3: Buffer | Q1 |
| break 1 | | | |
| break 2 | | | |
| 8 | Virtual memory - concepts, systems | 4: Processes | |
| 9 | Memory allocation | | |
| 10 | Hardware level I/O and microcontroller concepts | | |
| 11 | Network programming, concurrent programming | 5: TBD | |
| 12 | Multithread and synchronization | | Q2 |
| 13 | Revision and examination preparation | | |

# Policies and Procedures

Macquarie University policies and procedures are accessible from Policy Central. Students should be aware of the following policies in particular with regard to Learning and Teaching:

Academic Honesty Policy http://mq.edu.au/policy/docs/academic_honesty/policy.html

Assessment Policy http://mq.edu.au/policy/docs/assessment/policy.html

Grading Policy http://mq.edu.au/policy/docs/grading/policy.html

Grade Appeal Policy http://mq.edu.au/policy/docs/gradeappeal/policy.html

Grievance Management Policy http://mq.edu.au/policy/docs/grievance_management/policy.html

Disruption to Studies Policy http://www.mq.edu.au/policy/docs/disruption_studies/policy.html *The Disruption to Studies Policy is effective from March 3 2014 and replaces the Special Consideration Policy.*

In addition, a number of other policies can be found in the Learning and Teaching Category of 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/support/student_conduct/

The practical tasks in this unit are used at other Universities internationally. You will be required to sign an agreement for acceptable use of these materials before you will be given access. Any violation of your agreement will be considered a serious offence.

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://students.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://informatics.mq.edu.au/help/.

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

# Graduate Capabilities

## 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

## 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

## 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

## 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

# Changes since First Published

| Date | Description |
| --- | --- |
| 04/08/2014 | Add statement of requirement to perform satisfactorily in exams and pabs as separate components, and to complete the five labs. |
| 28/02/2014 | The Description was updated. |