COMP2041/9041 - Software Construction - Course Outline

Course Staff

Staff Name	Role	Email	Extension
Andrew Taylor	Lecturer & Admin	andrewt@cse.unsw.edu.au	55525

Class Details

Day	Time	Room
Tuesday	13:00-15:00	Rex Vowels (EE LG1)
Thursday	14:00-15:00	Rex Vowels (EE LG1)

You will have chosen a 3 hour tut-lab slot when you enrolled.

Consultations times vary through session and are listed on the course home page.

Distance Stream

A distance (WEB) stream is available. Students in this stream will need to rely on lecture recordings and the material placed on the web. Student should consider carefully whether this is sufficient for them to successfully complete the course.

Communication

Sometimes urgent information may be sent to you by email. Make sure you pay careful attention to any email you receive.
All official email will be sent to your CSE email address which is by dfefault forwarded to your UNSW address. If you redirect your mail please do so carefully & check that your redirection works.

Additional information will be provided in the Course Forum (linked to the class home page). You should read it regularly. The forums is the best place to ask questions about the course.

Course Summary

The following is a summary of the topics that will be covered in this course.

1. Tools for software construction
    - Programming languages (C)
    - Scripting languages (Perl, Shell, brief intro to python)
    - Filters (sort, sed, grep, tr, ...)

    - Analysis tools (debuggers, profilers)
    - development tools (make, git, ...)
2. Techniques for software construction
    - Analysis, design, coding, testing, debugging, tuning
    - Interface design, documentation, configuration
3. Qualities of software systems
    - Correctness, clarity, reliability, efficiency, portability, ...

The focus for the practical work will be on C and Perl. However, you would be well advised to acquaint yourselves with the facilities provided by the Unix shell.

Course Aims

This course is designed for students who have mastered the basics of programming. It aims to broaden your knowledge of techniques and tools for software construction.

Learning Outcomes

By the end of the course, you should have these attributes which will be useful to you for the remainder of your studies and after graduation:

- have practical experience in programming the scripting languages Perl, the Unix shell and optionally some Python.
- have a broader & deeper knowledge of building software systems
- more appreciation of the use of specific technologies and strategies during software development
- exposure to tools for version control, performance improvement, configuration and debugging,
- improvement of your ability to articulate & communicate concepts related to programming & systems

Assumed Knowledge:

COMP2041/9041 assumes that you have a sound understanding of a procedural programming language such as C and can:

- produce a correct procedural program from a spec
- understand fundamental data structures + algorithms (char, int, float, array, struct, pointers, sorting, searching)
- appreciate use of abstraction in computing

For undergraduate (COMP2041) students, the above material will have been covered in first year courses such as COMP1917 Higher Computing 1 and COMP1927 Higher Data Str. & Algos.

For postgraduate (COMP9041) students, the above material will have been covered in COMP9021 Principles of Programming and COMP9024 Data Structures/Algorithms, or similar material will have been covered in their undergraduate degree.

A limited amount of specific knowledge of the C programming language may be assumed during the course.

Students who are not competant C programmers should discuss with the lecturer at the first lecture the impact this might have. Typically students who are competant in a similar languages such as C++ and Java only need to do some extra reading.

Teaching Strategies

Lectures: Lectures will be used to present the theory and practice of the techniques and tools in this course. There will be extensive use of case studies and practical demonstrations during lectures. Lecture notes will be available on the course web pages before each lecture. **Tutorials** From week 2 you will also be expected to attend a one-hour tutorial session to clarify ideas from lectures and work through exercises based on the lecture material. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tute, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tute. There are no marks for tutorial attendance. **Laboratory Classes**: following the tute class each week, there will be a two-hour lab class, during which you will work on a variety of small practical problems involving the tools introduced in lectures. Because this course is practical in nature, lab class are a very important component, and you should make every effort to attend the labs and complete the exercises diligently. In particular, **keep up-to-date** with the Lab work; if you fall behind it affects your ability to understand later material in the course.
To obtain a mark for a lab exercise you must both demonstrate the completed lab exercise to your tutor during a lab class and submit it using give.

If you don't complete a lab exercise during the scheduled class, you can still obtain the mark if you both submit the completed exercises before midnight Sunday and you demonstrate it to you tutor in the follow week's lab.

COMP9041 students are recommended to attend tutorials and labs, but may opt to complete the work in their own time and not have it formally assessed. To do this, they must advise the lecturer by email by the end of week 2.

Assignments

In the assignment work, you will work through the process of building and/or modifying software systems, using the tools and techniques described in lectures. The assignment work will focus on Perl and Perl+CGI.
There will be two assignments the first will be a Perl application due week 7/8, the second due will be Perl/CGI due week 12. They will be of roughly equal weight.

Exam

There will be a three-hour primarily practical exam, to be held in the CSE labs during the exam period. It consists of five small implementation tasks and one written section. During this exam you will be able to execute, debug and test your answers. The implementation tasks will be similar to those encountered in lab exercises
You will not be expected to remember the details of programming languages used in the course; reference information will be provided along with the exam paper, giving a summary of any language that we expect you to use.

It is a hurdle requirement for this course that you pass the exam.

It also is a hurdle requirement for this course that you perform satisfactorily on the implementation tasks in the exam. This is defined as successfully at least two of the five implementation tasks.

Teaching Rationale

This course has a heavy practical orientation. Lectures will revolve around live demonstrations of programming and use of tools. Labs & assignments form a key part.

Assessment

Component	Value
Lab Work	10%
Assignments	30%
Exam	60%

These assessment weights might be varied by a few percent when the assignments have been chosen.

If your final exam mark is less than 50% then your overall mark will not be allowed to exceed your exam mark. In other words you must pass the final exam to pass the course.

As mentioned above, your performance on the practical component of the final exam must also be satisfactory to pass the course.

The lecturer may scale overall marks, or individual components, up or down to obtain a desired mark distribution.

You may be excluded from the prac exam if you have < 10/40 for assignments+labs.

```
ass    = mark for assignments     (out of 30)
labs   = mark for assessed labs   (out of 10)
exam   = mark for exam            (out of 60)

okExam =  two implementation tasks solved on exam

mark = ass + labs + pexam + texam
grade    = HD|DN|CR|PS  if mark >= 50 && okExam
         = FL           if mark < 50 && okExam
         = UF           if !okExam
```

Academic honesty and plagiarism

What is Plagiarism?

Plagiarism is the presentation of the thoughts or work of another as one's own.* Examples include:

- direct duplication of the thoughts or work of another, including by copying material, ideas or concepts from a book, article, report or other written document (whether published or unpublished), composition, artwork, design, drawing, circuitry, computer program or software, web site, Internet, other electronic resource, or another person's assignment without appropriate acknowledgement;
- paraphrasing another person's work with very minor changes keeping the meaning, form and/or progression of ideas of the original;
- piecing together sections of the work of others into a new whole;
- presenting an assessment item as independent work when it has been produced in whole or part in collusion with other people, for example, another student or a tutor; and
- claiming credit for a proportion a work contributed to a group assessment item that is greater than that actually contributed.

For the purposes of this policy, submitting an assessment item that has already been submitted for academic credit elsewhere may be considered plagiarism. Knowingly permitting your work to be copied by another student may also be considered to be plagiarism.

Note that an assessment item produced in oral, not written, form, or involving live presentation, may similarly contain plagiarised material.

The inclusion of the thoughts or work of another with attribution appropriate to the academic discipline does not amount to plagiarism.

The Learning Centre website is main repository for resources for staff and students on plagiarism and academic honesty. These resources can be located via:www.lc.unsw.edu.au/plagiarism

The Learning Centre also provides substantial educational written materials, workshops, and tutorials to aid students, for example, in:

- correct referencing practices;
- paraphrasing, summarising, essay writing, and time management;
- appropriate use of, and attribution for, a range of materials including text, images, formulae and concepts.

Individual assistance is available on request from The Learning Centre. Students are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting, and the proper referencing of sources in preparing all assessment items.

All work submitted for assessment must be your own work. Lab exercises and assignments must be completed *individually*. We regard copying of assignments or lab exercises, in whole or part, as a very serious offence. We use plagiarism detection software to search for multiply-submitted work.

1. Submitting part or all of other students' work, with or without acknowledgement, is not acceptable.
2. Submitting solutions written by other persons is also not acceptable.
3. Building on ideas and partial solutions obtained from public sources, such as web resources, may be acceptable, provided full acknowledgement is made. However, the final mark will take into account the starting point and how much development work would have been required. Failing to acknowledge web or other resources is unacceptable.

4. Discussing approaches to solutions with other students is quite appropriate, but any discussions should remain at the design level, and must not include program text. Comparison tools will detect any common code across the student body.
5. The safest approach is to work diligently on your own, seeking help from the forum or course staff.

- Submission of work derived from another person, or jointly written with someone else will, may result in automatic failure for COMP2041/COMP9041 with a mark of zero.
- Allowing another student to copy from you will may result in a mark of zero for your own assignment or lab exercises. Do not provide your work to any other person, even people who not UNSW students. You will be held responsible for the actions of anyone you provide your work to.
- Severe or second offences will result in automatic failure, exclusion from the university, and possibly other academic discipline.

Refer also to the Yellow Form material on plagiarism and the Learning Centre website

Course schedule

The anticipated course sequence is: shell scripting (weeks 1-3), Perl (weeks 3-6), web applications (6-9), programming tools(9-12) We may need to vary this to some degree as we update material in the courses.
The lectures are timetabled for weeks 1-12. It is possible the week 13 lecture slots will be used for a remedial revision lecture or other optional presentations so please keep them free.

Resources for Students

There is no required textbook for the course. Useful reference books include the following:

- Kernighan & Pike, The Practice of Programming,
  Addison-Wesley, 1998.
  (Inspiration for 2041 - philosophy and some tool details)
- McConnell, Code Complete (2ed),
  Microsoft Press, 2004.
  (Many interesting case studies and practical ideas)
- Wall, Christiansen & Orwant , Programming Perl (3ed),
  O'Reilly, 2000. (Original & best Perl reference manual)

- Schwartz, Phoenix & Foy, Learning Perl (5ed),
  O'Reilly, 2008. (gentle & careful introduction to Perl)
- Christiansen & Torkington, Perl Cookbook (2ed),
  O'Reilly, 2003. (Lots and lots of interesting Perl examples)
- Schwartz & Phoenix, Learning Perl Objects, References, and Modules (2ed),
  O'Reilly, 2003. (gentle & careful introduction to parts of Perl mostly not covered in this course)
- Schwartz, Phoenix & Foy, Intermediate Perl (2ed),
  O'Reilly, 2008. (good book to read after 2041 - starts where this course finishes)
- Sebesta, A Little Book on Perl,
  Prentice Hall, 1999. (Modern, concise introduction to Perl)
- Orwant, Hietaniemi, MacDonald, Mastering Algorithms with Perl,
  O'Reilly, 1999. (Algorithms and data structures via Perl)
- Kochgan & Wood 2003, UnixÂ® Shell Programming,
  Sams Publishing 2003 (Careful intoduction to Shell Programming)
- Peek, O'Reilly, Loukides, Bash Cookbook,
  O'Reilly, 2007. (Recipe(example) based intro to Shell programming)
- Powers, Peek, O'Reilly, Loukides, Unix Power Tools (3ed),
  O'Reilly, 2003. (Comprehensive guide to common Unix tools)
- Loukides & Oram, Programming with GNU Software,
  O'Reilly, 1997. (Tutorial on the GNU programming tools (gcc,gdb,...))
- Robbins, Unix in a Nutshell (4ed),
  O'Reilly, 2006. (Concise guide to Unix and its toolset)
- Kernighan & Pike, The Unix Programming Environment,
  Prentice Hall, 1984. (Pre-cursor to the textbook, intro to Unix tools)

For pointers to other useful reading material, including documentation for all of the tools used in the practical work, see the course Web pages.

Course evaluation and development

Student feedback on this course will be obtained via electronic survey at the end of session, and will be used to make continual improvements to the course. Students are also encouraged to provide informal feedback during the session, and to let the lecturer in charge know of any problems, as soon as they arise. Suggestions will be listened to very openly, positively, constructively and thankfully, and every reasonable effort will be made to address them.
This feedback is used to improve the course materials & their delivery. In the most recent session feedback was very favourable probably as results of changes based on previous session's feedback. Some lab exercises and lecture topics will be updated to better reflect current practice.

Other matters

- Occupational Health and Safety policies
- Information for students with disabilities Contact the lecturer ASAP if you have any disabilities that may affect this course.