

# Lecture 1

Introduction

School of Information Technology and Electrical Engineering  
The University of Queensland

2

## What's This Course All About?

- Exposure to UNIX operating system
- Systems Programming in C
- Underlying Principles of
  - Operating Systems
  - Computer Networks
- You will become more effective programmers and system designers by having knowledge of the underlying systems

3

## Course Profile

- Describes
  - The course in detail
  - What you can expect
  - What we expect of you
- **You should obtain and read the course profile**
- Now for *some* of the details ...

5

## Welcome

- **COMP2303 / COMP7306**
  - Network and Operating Systems Principles
- Teaching Staff
  - Dr Joel Fenwick
  - A/Prof Peter Sutton
  - Tutors
    - Richard Holland
    - Luke Johnston
    - Thomas King
    - Patrick Laub

## Today...

- Course overview
- Intro to Operating Systems
- If you have questions – ask them at any time

4

## Course Content Overview

Week 1	Intro to Course, Operating Systems, UNIX
Weeks 2-3	C Programming
Week 4	UNIX shell, Shell Scripts
Weeks 5-9	Operating Systems Concepts
Week 7	<b>Mid-semester exam</b>
Weeks 9-13	Networking Concepts
Week 13	Review

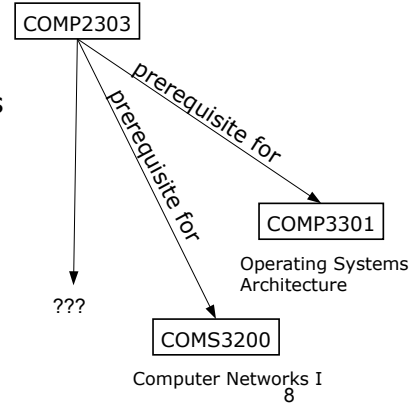
6

# Assumed Background

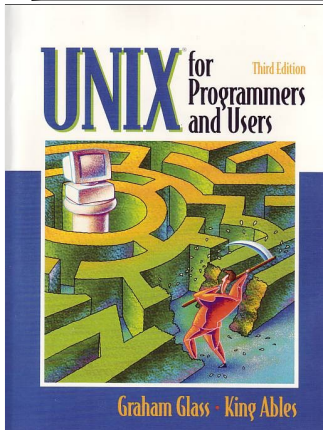
- You must know something about programming
  - e.g., CSSE1001 or CSSE7030
- The more comfortable you are with programming in general, the easier you will find this course
- You should also have ...
  - ... Some knowledge of computer systems
  - ... Knowledge of binary representations (2's complement etc)
  - ... Knowledge of binary operations (AND, OR, XOR, ...)
  - ... Ideally, some prior exposure to C
    - CSSE1000 or CSSE7035

# Other Courses

- COMP2303 leads to more detailed courses in the area of computer systems and networks

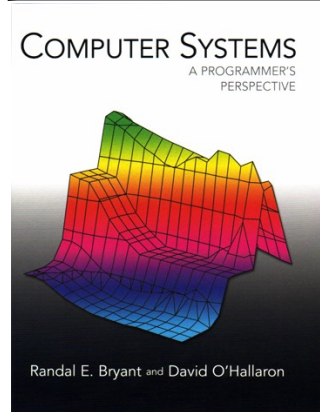


# Textbooks



- Glass & Ables
  - UNIX for Programmers and Users
- Covers most aspects of the course

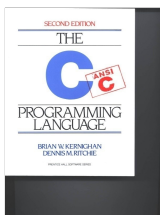
# Textbooks (cont.)



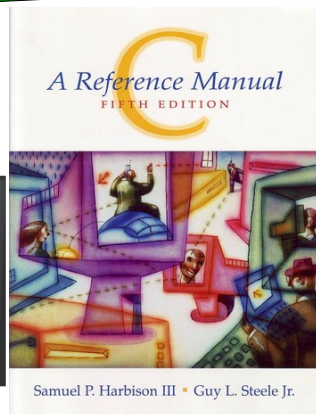
- Bryant & O'Halloran
  - Computer Systems: A Programmer's Perspective
- Previous recommended text for this course (2005,2006)
- Course is loosely based on this book, but it doesn't cover all aspects and goes into more detail than we will

# Textbooks (cont.)

- Harbison and Steele -
  - C: A Reference Manual (5th edition)
  - Highly recommended as a **reference** on C
- Kernighan and Ritchie
  - The C Programming Language (2nd ed, 1988)
  - Does not cover C99

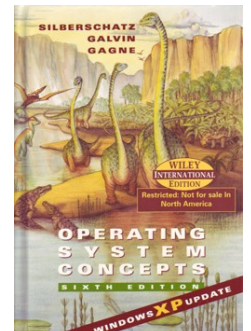
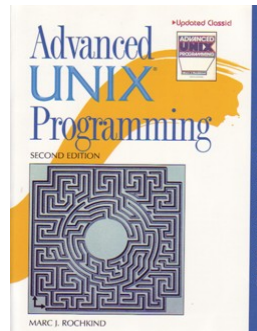


OR



# Textbooks (cont.)

- Useful references



## Assessment

- **Assignments** (100 marks total)
  - Four Assignments
    - Equal weight (25 marks) but not equal difficulty
    - A1 – Simple C Programming
    - A2 – debugging
    - A3 and A4 – UNIX systems programming in C
- **Exams** (100 marks total)
  - Mid-semester exam (in Friday lecture, week 7)
    - Multiple choice, open book
  - Final exam
    - Written answers, open book
  - Overall exam mark is better of
    - 30% mid-semester + 70% final
    - 15% mid-semester + 85% final
  - Exams cover theory and programming

13

## Grade determination

- Final mark (out of 100) determined as **geometric mean** of assignment and exam marks (and then rounded to nearest integer)

$$Final_{mark} = \sqrt{Assignment_{mark} \times Exam_{mark}}$$

- Grade determined from final mark
  - **7** = 85 to 100    □ **4** = 50 to 64
  - **6** = 75 to 84    □ **3** = 45 to 49
  - **5** = 65 to 74    □ **2** = 20 to 44
  - Comp7306 has different cutoffs
- No minimum requirements on exam or assignment marks

14

## Late Submissions

- Assignments are all due electronically
  - 1,3,4 using subversion
  - Submission by **11pm** on due date
  - 20% per 24hrs (or part thereof) late penalty
  - No submissions accepted more than 96 hours after the deadline under any circumstances.
- **Read course profile for the fine print!**

15

## Plagiarism and Collusion

- All assignments are individual
  - All submitted code must be your work
  - Using code provided on COMP2303 website is acceptable
  - Use of any other published code is unacceptable
- **ALL submitted code will be subject to plagiarism and collusion detection**
- Don't copy or look at code from other students or allow your code to be copied or seen – this is cheating
  - Misconduct proceedings will be initiated if plagiarism and/or collusion is found
- You are encouraged to discuss assignments but this should not include sharing code

16

## Plagiarism and Collusion (cont.)

- Assessment can serve (at least) two purposes:
  - Feedback to you on your learning
  - Measuring your performance for the purpose of generating a grade
- Plagiarism and/or collusion compromises both of these

17

## Teaching Modes

- **Lectures** – three hours per week
  - Tuesday 2-4pm and Friday 1-2pm
- **Pracs** – two hours per week
  - Sign-up to two one-hour sessions
    - one "C" and one "P" session
  - Pracs start this week (week one)
    - ("P" sessions only this week, no "C" sessions)
- **Consultation Times**
  - To be advised, or make an appointment

18

## What to expect in pracs

- Exercises early in weeks.
  - Eg intro to unix
- Teaching in some weeks.
  - Eg make
- Work on assignments, get help.

19

## What to expect in lectures

- Stand and stretch breaks half way through each hour (approximately)
- 10 minute break in the middle of Tuesday lecture
- Stories!
- Some practical examples, tool demos, explanations
- Take notes!
  - Lecture slides don't capture everything
- Lectures will generally cover higher level concepts (except for weeks 2-4)
- Mid-semester exam in Friday lecture slot in week 7

21

## Things I may do during lectures

### Employ comical exaggeration

- Concepts are abstract
- Computers are fast
- Hard to differentiate between good and bad solutions

23

## Short Break

Stand up and stretch

## Things I may do during lectures

### Ask questions

- "Why?" - you may need to justify answers.
- I don't expect people to be able to answer all questions immediately.
- May need to move quickly to give someone else a chance.
  - Dealing with some answers may require material we haven't covered.

22

## What we expect from you...

- **Attendance** at lectures
  - You may be disadvantaged if you don't attend
- **Seek help if you're having trouble**
  - Don't leave it too late
- Hard work - **12 hours per week**
  - Average time commitment for average student seeking an average grade (4 or 5)
  - Assignments will take at least 14 hours each

24

## What we expect from you...

### •Feedback and ideas

- What can we improve?
  - Especially if some aspect of the course is causing you distress.
- What do you want to learn about?
  - Course is pretty full so no major changes.

### •How?

- In person
- Email me
- On the newsgroups
- Anonymous email from the subject webpage
  - Remember very little is truly anonymous.
  - If I don't know who you are I can't respond.

25

## Facilities

- Pracs in **78-110**
  - PC lab, from which you can remotely access UNIX server
  - After hours access available
    - You'll need a swipe card – see the ITEE office
  - Login using UQ password
- Server: **moss.eait.uq.edu.au**
  - Runs Linux
  - Access from lab PCs possible, via
    - ssh (command line)
    - X-window (graphical)
  - Remote access possible
    - ssh to **moss.labs.eait.uq.edu.au**
    - See <http://studenthelp.itee.uq.edu.au/remotel/>

27

## Linux at home

- If you haven't done so already. This is a good opportunity to try linux on your own hardware.
  - Lots of people to answer questions.
  - Can work without connecting to moss.
    - Always test on moss.
- While we can answer questions we do not provide support for install problems.
  - We probably won't debug on your hardware.
  - **If it eats your pets and destroys your computer – not our fault!**

29

## Resources

- Course website
  - <http://www.itee.uq.edu.au/~comp2303>
  - Lecture Notes
    - Usually posted just in advance of lectures
  - Pracs
    - Programming problems and exercises
- Newsgroup – get a good reader
  - [uq.itee.comp2303](http://uq.itee.comp2303)
- Notices
  - Distributed via email, newsgroup, web

26

## Using your own hardware (optional!)

- Connect to moss via ssh (putty)
- Work on your own computer. At your own risk. **Always test on moss! If it does not work on moss it does not work!**
- If your computer is running:
  - Linux – Make sure you have gcc, make and svn installed.
  - MacOSX – You will need to install the development tools from your OS cd.
  - Windows – consider installing linux.

28

## Install options

- Virtual machine: A program simulates a whole computer on which you can install and run an OS.
  - VirtualBox, vmware, parallels
- Dual boot: Choose between a number of OS at boot time. (Need to reboot to switch).
  - Wubi – windows installer for Ubuntu
  - Debian, Ubuntu, many others
- Use your isp's mirrors where possible

## Suggested software

- A newsgroup reader
  - Eg thunderbird
- (Optionally) A chat client that can talk XMPP
  - Eg pidgin, jabber, google chat

31

## Lecture Outline

- What is an operating system?
  - Different views of operating systems
  - History of operating systems
  - Example operating systems
  - Hardware support
  - OS organisation
- 
- Slide Credits
    - E.N. Elnozahy, U Texas
    - R. Chandra, Cornell University

33

35

## Any Questions?

32

## What is an Operating System?

- Write down what you think an operating system does.
- 
- [Discussion to take place in class]

34

36

## Break

37

## The Hardware View

- The operating system is the layer of software that interacts directly with the hardware, concerns revolve around:
  - The boot process
  - Devices and how the OS can use them
  - The interactions between H/W and OS

39

## The Application Programmer's View

- The OS is like a library with a well defined set of API's
  - What abstractions are available from the OS?
  - How well is the API structured? Not too low-level, or high-level.
  - How portable is the interface?
  - Protection of the intellectual investment-- don't want to keep rewriting the same program for each new OS release.
    - Explains why Windows has been so successful!

41

## The 5 Views of OS

- Your view of an OS depends on who you are:
  - The **hardware** view
  - The **operating system designer's** view
  - The **application programmer's** view
  - The **end-user's** view
  - The **system administrator's** view

38

## The OS Designer's View

- The interest revolves mainly about the OS itself, its internal structure, its efficiency, performance, data structures, etc..
  - How can we make the OS more efficient
  - How can we add more functionality?
  - How do we debug the OS? Make it more reliable, scalable, etc..

40

## The End-User's View

- The OS is just a program that happens to be pre-installed
  - Must not crash or externalize the ugly aspects of the machine
  - Must protect investment in existing software & applications
  - Users care about applications, not the OS
  - A good OS is the one that is most transparent
- Contrast Windows, MacOS & UNIX

42

# The System Administrator View

- An OS is a program that allows the efficient and equitable usage of resources:
  - How can it track usage for accounting?
  - How easy is it to install new software?
  - Security
  - Fairness
- Contrast Windows, MacOS, UNIX, and mainframe systems

43

# Multiprogramming Systems

- Multiprogramming systems increased utilization
  - Developed in the 1960s
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Optimizes for throughput at the cost of response time



45

# Personal Operating Systems

- PC OS's, 1974+
  - Apple II, and others
- MSDOS 1980+
  - The PC revolution
- Windowing OS 1983+
  - Apple (MacOS) & Xerox (Pilot OS)
  - Windows 3.1, OS/2
- Computers are cheap □ everyone has a computer
- Initially, the OS was a library
- Advanced features were added back
  - Multiprogramming, memory protection, etc



47

# History of Operating Systems

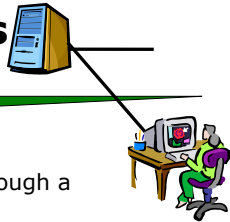
- Earliest computers had no OS
  - programmed directly
- Initially, OS was just a run-time library
  - You linked your application with the OS, loaded the whole program into memory, and ran it
  - How do you get it into the computer? Through the control panel!
- Simple batch systems (mid 1950s – mid 1960s)
  - Permanently resident OS in primary memory
  - Loaded a single job from card reader, ran it, loaded next job...
  - Control cards in the input file told the OS what to do
  - Spooling allowed jobs to be read in advance onto tape/disk



44

# Time Sharing Systems

- Timesharing (1970s) allows interactive computer use
  - Users connect to a central machine through a terminal
  - User feels as if they have the entire machine
  - Based on time-slicing: divides CPU equally among the users
  - Allows active viewing, editing, debugging, executing process
  - Security mechanisms needed to isolate users
  - Requires memory protection hardware for isolation
  - Optimizes for response time at the cost of throughput



46

# Distributed Operating Systems

- Cluster of individual machines
  - Over a LAN or WAN or fast interconnect
  - No shared memory or clock
- Asymmetric vs. symmetric clustering
- Sharing of distributed resources, hardware and software
  - Resource utilization, high availability
- Permits some parallelism, but speedup is not the issue
- SANs, Oracle Parallel Server

48

## Parallel Operating Systems

- Multiprocessor or tightly coupled systems
- Many advantages:
  - Increased throughput
  - Cheaper
  - More reliable
- Asymmetric vs. symmetric multiprocessing
  - Master/slave vs. peer relationships



49

## OS Types & Examples

- **Desktop:** MSDOS, Windows 95/98/ME/NT/2000/XP/Vista/7, MacOS, Linux
- **Workstation / Server:** HPUX, AIX, Solaris, Linux, BSD (many variants), Windows Server, Novell Netware
- **Minicomputers:** OS/400, VMS
- **Mainframes:** CMS/MVS (now z/OS)
- **Embedded:** OS-9, VxWorks, Lynx, PalmOS, Windows CE, Symbian OS, uCLinux, IOS
  - Some of these are real-time

51

## Hardware Support

- Operating system needs to:
  - control I/O devices
  - control access to the hardware
 all while denying these privileges to user programs:
  - for protection
  - for abstraction/ease of use
- Hardware supports two modes of operation (or more):
  - access to hardware & I/O devices is done through privileged instructions, these are only available in "**supervisor**" mode
  - privileged instructions cannot be executed in "**user**" mode

53

## Real Time Operating Systems

- Goal: To cope with rigid time constraints
- Hard real-time
  - OS guarantees that applications will meet their deadlines
  - Examples: TCAS, health monitors, factory control
- Soft real-time
  - OS provides prioritization, on a best-effort basis
  - No deadline guarantees, but bounded delays
  - Examples: most electronic appliances
- Real-time means "predictable"
  - NOT fast



## Short Break

Stand up and stretch

## Implementation

- Using a bit in the processor (i.e. 0 or 1)
- Operating system code runs in supervisor mode, while user program code runs in user mode
- Switching from user to supervisor mode occurs on:
  - **interrupts:** hardware devices needing service
  - **exceptions:** user program acts silly (divide by 0, bus error, etc)
  - **trap instructions:** user program requires OS service (**system call**)
- Switching back occurs by an **RTI** instruction

54

## On Interrupts

- Hardware calls the operating system at a pre-specified location
- Operating system saves state of the user program
- Operating system identifies the device and cause of interrupt
- Responds to the interrupt (possibly killing program, <CTRL-C>)
- Operating system restores state of the user program (if applicable) or some other user program
- Execute an RTI instruction to return to the user program
- User program continues exactly at the same point it was interrupted.

**Key Fact: None of this is visible to the user program**  
55

## On System Calls

- User program executes a trap instruction (system call)
- Hardware calls the operating system at a pre-specified location
- Operating system identifies the required service and parameters, e.g. `open(filename, O_RDONLY);`
- Operating system executes the required service
- Operating system sets a register to contain the result of call
- Execute an RTI instruction to return to the user program
- User program receives the result and continues

**Key Fact: To the user program, it appears as a function call executed under program control**

57

## System Calls

- Programming interface to services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs using APIs
- Three common APIs:
  - Win32 API for Windows
  - **POSIX** API for POSIX-based systems (UNIX, Linux, Mac OS X)
    - **This is the emphasis for COMP2303**
  - Java API for the Java virtual machine (JVM)

59

## On Exceptions

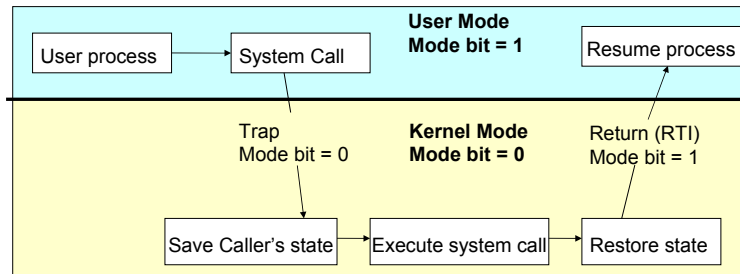
- Hardware calls the operating system at a pre-specified location
- Operating system identifies the cause of the exception (e.g. divide by 0)
- If user program has exception handling specified, then OS adjust the user program state so that it calls its handler
- Execute an RTI instruction to return to the user program
- If user program did not have a specified handler, then OS kills it and runs some other user program, as available

**Key Fact: Effects of exceptions are visible to user programs and cause abnormal execution flow**

56

## Crossing Protection Boundaries

- User calls OS procedure for "privileged" operations
- Calling a kernel mode service from user mode program:
  - Using System Calls
  - System Calls switches execution to kernel mode



## Reducing System Call Overhead

- Problem: The user-kernel mode distinction poses a performance barrier
  - Crossing this hardware barrier is costly.
  - System calls take 10x-1000x more time than a procedure call
- Solution: Perform some system functionality in user mode
  - Libraries (DLLs) can reduce number of system calls,
    - by caching results (getpid) or
    - buffering (open/read/write vs. fopen/fread/fwrite).

60



## Summary

- Operating Systems
  - make computer hardware easier to use
  - manage resources
    - CPU time, disk space, memory, I/O devices
- Hardware (CPU) support is needed
  - "user" mode and "supervisor" mode
- Programs access operating system services via **system calls**

68

## Things To Do This Week

- Enrol (if you haven't already)
- Read course profile
- Sign-up for prac sessions
- Attend prac ("P" sessions only this week, no "C" sessions)
  - Intro to UNIX (see course website)
- Consider buying textbook(s)
- Attend Friday lecture
  - More UNIX intro

67

69

71

70

72

Comp<sup>2303</sup><sub>7306</sub>



Comp<sup>2303</sup><sub>7306</sub>

