OS - Retrospection

THE UNIVERSITY OF
NEW SOUTH WALES

Tid-bits from course outline

This course is oriented towards exposing students to the essential concepts and issues that underly operating systems and their design.

- **Technical**
  - Make students understand the key concepts and mechanisms of modern operating systems:
    - processes and process management,
    - memory management techniques,
    - on-line storage methods (file systems),
    - security and protection,
    - concurrency issues,
- **Educational**
  - Make students understand the reasons why operating systems are built the way they are, and what the implications and lessons are for other software systems. Specific learning objectives are:
    - appreciation of design trade-offs and design decisions and their dependence on the target environment;
    - appreciation of the distinction between mechanisms and policies, and why this is important;
    - exposure to low-level code;
    - exposure to current trends in operating systems research and development.
- **Professional**
  - The tutorial formats will give students practice in the presentation of solutions to an audience of peers, and will challenge them to critique peer technical presentations. Furthermore, the whole course encourages critical examination and analysis of "standard" solutions.
  - The assignments give students an opportunity to develop skills required to work as a team on a technical project, and the opportunity to work with a substantial body of code created by a third-party.

Operating Systems
@
CSE.UNSW

THE UNIVERSITY OF
NEW SOUTH WALES

Systems Courses

- COMP9242 Advanced Operating Systems
  - In-depth coverage of OS implementation issues
  - Learn what makes OS fast and what makes them slow
  - Learn how the OS deals with multiprocessors, caches, ...
  - Write your own OS
- In Session 2 taught by Prof. Gernot Heiser and Dr. Kevin Elphinstone

- Real- time systems COMP3241/9245
- General
  - Time analysis and scheduling
  - Software engineering for real-time systems
  - Real-time systems and programming languages

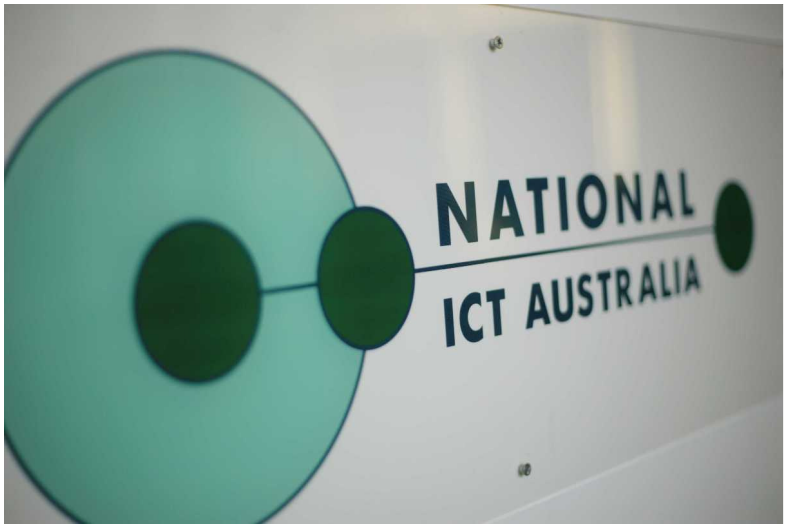
- Distributed systems COMP9243
  - Examines issues in building distributed systems and infrastructure
  - Peer-to-peer, web services, network file systems, name services, ……

OS Research
DiSy Group
ERTOS Program - NICTA

- 6 FTE researchers (PhDs)
- 4 FTE research engineers / research assistants
- 12 PhD students

NICTA Background

- National Research Laboratory
- Established in 2002
- Funded at least until 2011



Operating Pillars

- Established on:
  - Research – Built on exceptional research talent
  - Education – Built on enhancing ICT education
  - Commercialisation – Built on consideration of use
  - Collaboration – Built on exceptional partnerships

RESEARCH

NICTA is focusing its ICT research talent toward advances in technology which will produce significant social, environmental, and economic benefits for Australia

THE UNIVERSITY OF NEW SOUTH WALES

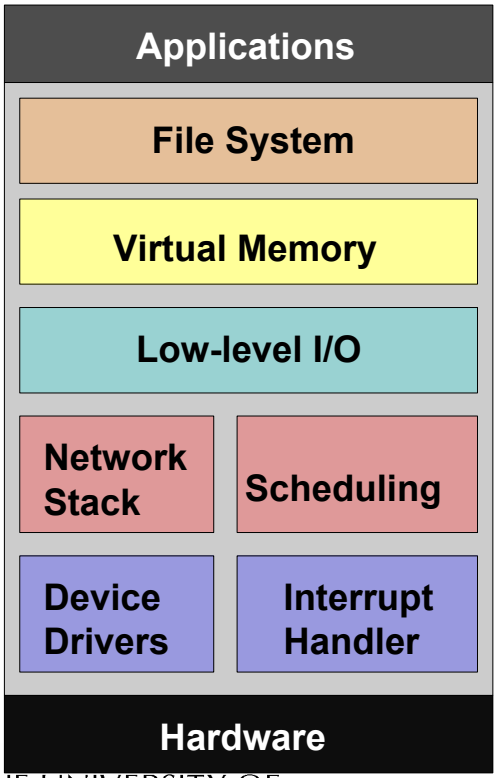
Embedded System Are Ubiquitous

But are they *Secure?*

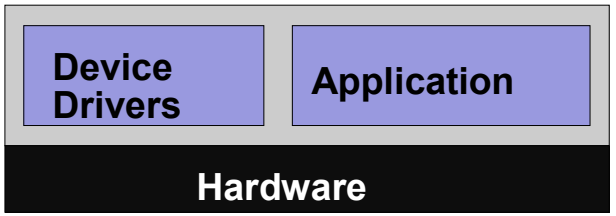
General-Purpose Vs. Embedded

- Traditional View:

General Purpose System



Embedded System



- minimal
- no OS at all or small "real-time executive"
- no protection

Security Challenges

- Growing functionality
  - increasing software complexity
  - increased number of faults
  - increased likelihood of security faults
- Wireless connectivity
  - subject to attacks from outside (crackers)
- Downloaded content (entertainment)
  - subject to attacks from inside (viruses, worms)
- Increasing dependence on embedded systems
  - increased exposure to embedded-systems security weaknesses

Embedded Systems Software

Present Approaches 1: Real-time Executives

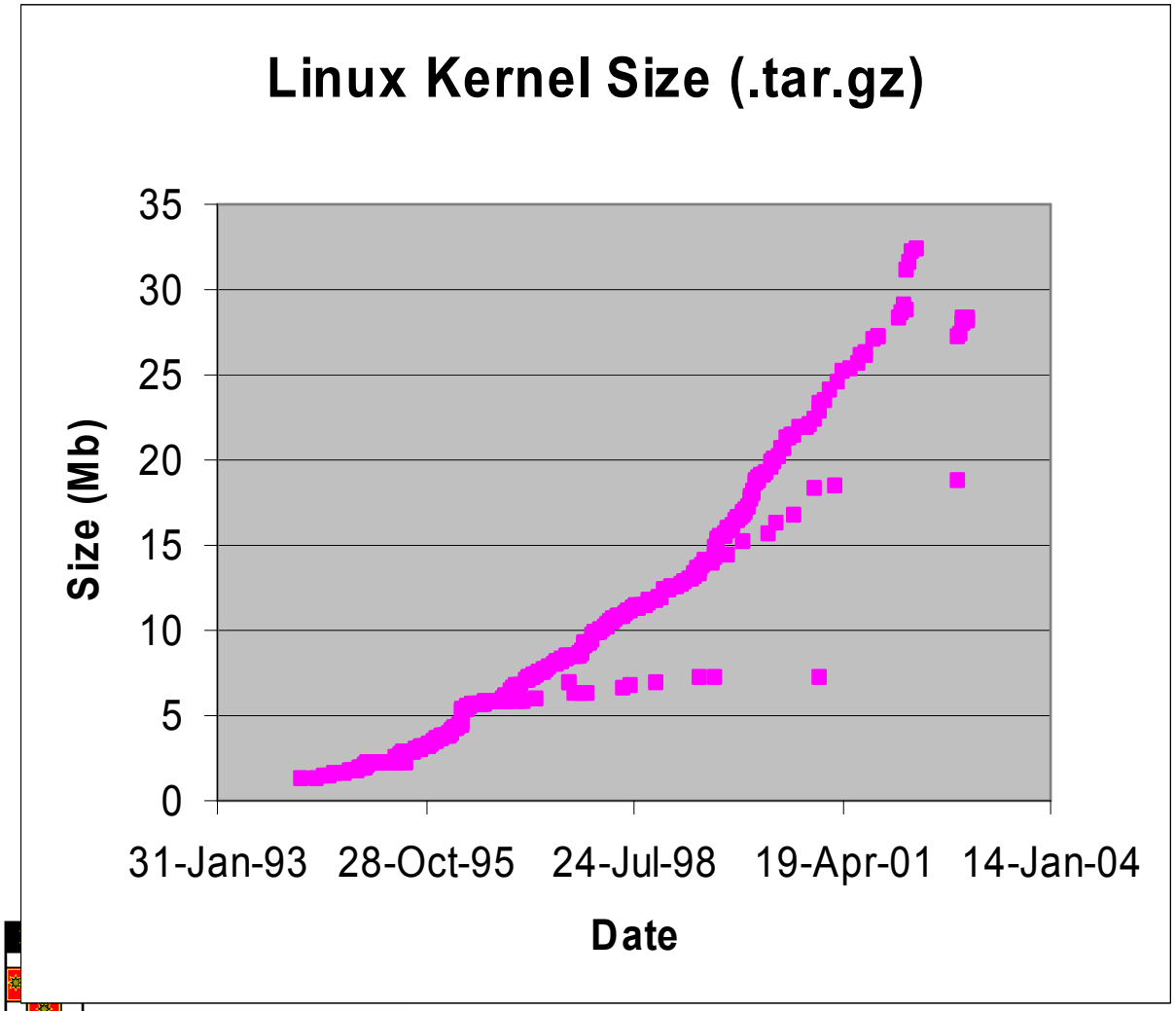
- Small, simple operating system
  - optimised for fast real-time response
  - suitable for systems with very limited functionality
- No internal protection
  - every small bug/failure is fatal
  - no defence against viruses, limited defence against crackers

Embedded Systems Software

Present Approaches 2: Linux, Windows Embedded

- Scaled-down version of desktop operating system
  - operating system protected from application misbehaviour
  - excessive code base for small embedded system
  - too much code on which security of system is dependent
- Dubious or non-existent real-time capabilities
  - unsuitable for hard real-time systems

Linux Kernel Evolution



For reference:

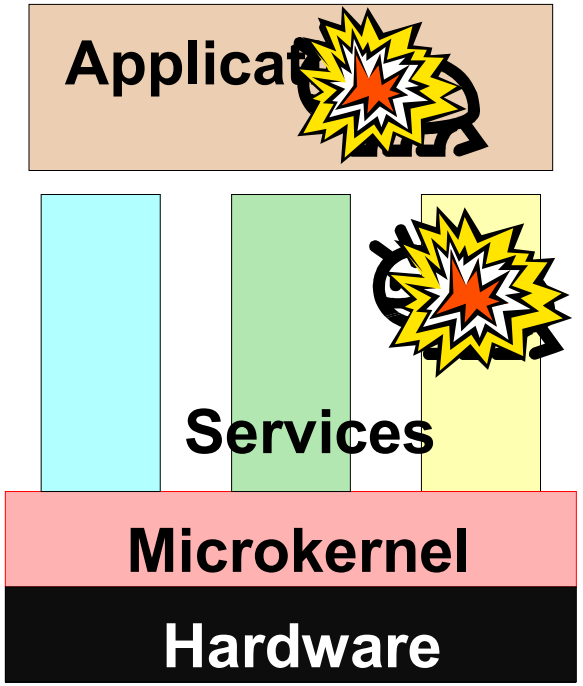
Linux 2.4.18 = 2.7 million lines of code

Embedded Systems Software

Our Approach: Microkernels

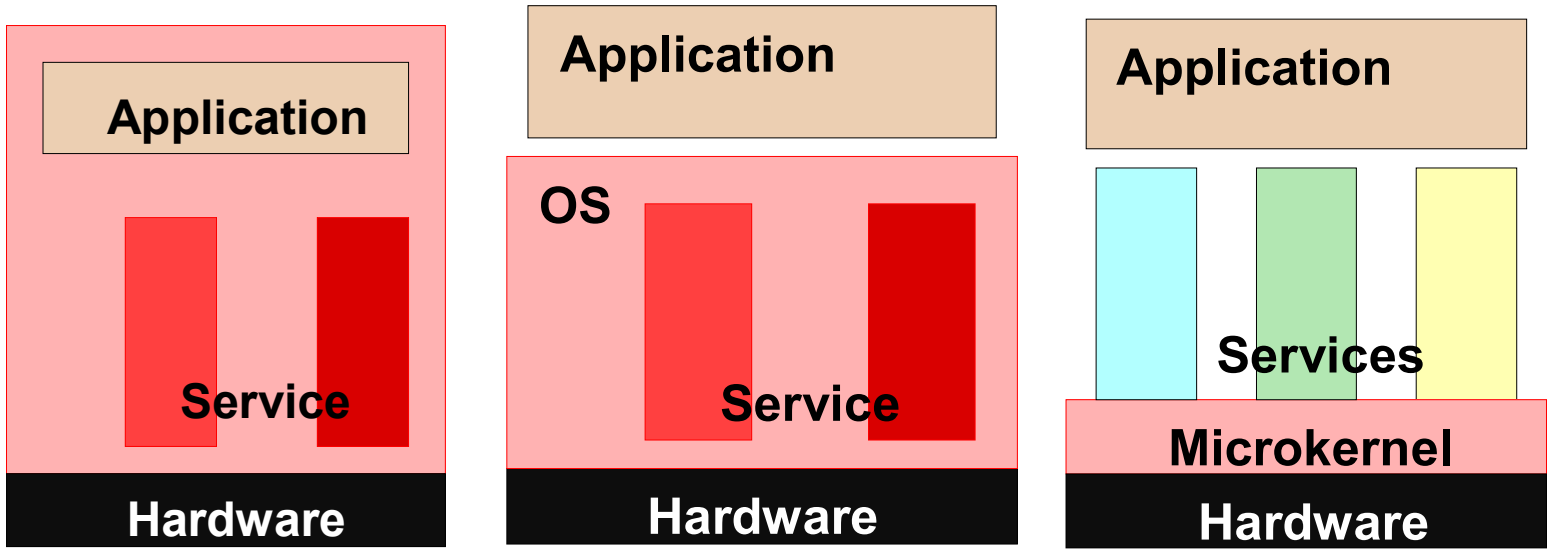
- Extremely small kernel
  - microkernel only contains code that must run in privileged mode
  - all other “systems” code runs as unprivileged servers
  - microkernel protected from application and other systems code
  - microkernel provides protection of all components from each other

- What’s the difference?



Trusted Computing Base

System:	Traditional Embedded	Linux/ Windows	Microkernel-based



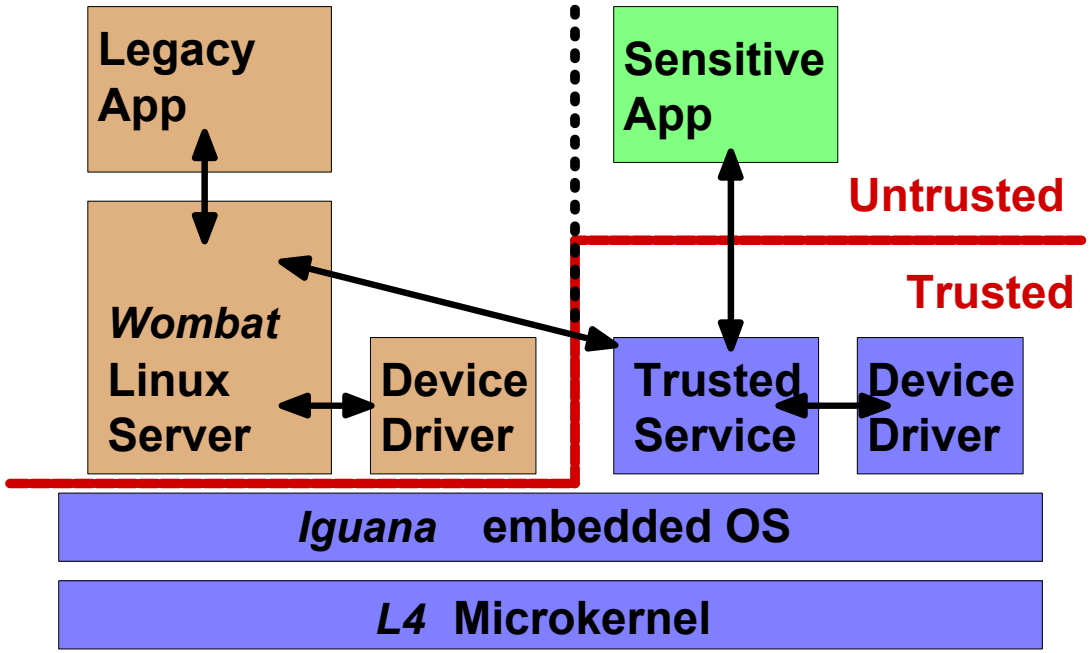
TCB:	**All** Code	100,000's loc	10,000's loc

Small is beautiful:

- Small kernel ⇒ potentially small TCB
- Small TCB ⇒ more trustworthy TCB!

Challenge: Can we guarantee the trustworthiness of the TCB?

A Sample System



- Sensitive part of system has small TCB
- Compromised legacy system cannot interfere with trusted part
- Linux server is optional
- Commercial Deployed (Qualcomm)

Why am I telling you this?

Does the following Interest you?

- Gaining in-depth experience in OS research
- Working on a very challenging projects
- Collaborating closely with active researchers
- Getting a high thesis mark
- International travel
- Fame and fortune

Prerequisites

- Keen interest in OS
- Demonstrable background/ability in OS
- Sharp Intellect
- Committed to working on a project

Still Interested?

- Check out

http://www.disy.cse.unsw.edu.au/

http://www.ertos.nicta.com.au/

On-line Course Survey???

- The on-line course survey will be available
- Please make time to do it
- Awarded 2 bonus marks to everyone who completes the survey.

- See the class web site for the URL

Final Exam

- Monday, 18th June, 8:45 – 11:00
- Two Hours
- No examination materials allowed
  - Uni calculators will be provided
- Don’t trust me – check the timetable yourself

Exam Format

- 6 questions
  - 4 should be answered in separate books
  - 1 must be ***answered on the exam paper*** itself.
  - 1 must be answered on the multiple choice answer sheet provided
  - 96 Marks in total

Exam Format

- Q1 is multiple choice (25% marks)

  You will receive one mark for each correct classification, and lose one mark for each incorrect classification. You gain zero marks for each answer left unclassified. The overall mark for this question will not be negative, i.e. the minimum mark is zero.

Exam Format

- Q2..Q6, roughly:
  - half working out a solution to a problem
  - half written answers to a question

For written answers

- Be clear and concise (get to the point quickly)
  - Long, rambling answers will be penalised

Sample Question

- Name four disk arm scheduling algorithms and describe an advantage or disadvantage of each of them.

- Sample Marking Scheme (out of 8)
  - 2 Marks for each algorithm (1 for the name, 1 for the pro/con)

Reasonable answer

- FCFS, SSTF, SCAN, C-SCAN
- FCFS does not take into account head position, may move head excessively, especially in the case of concurrent applications accessing disk (deteriorates to random)
- SSTF reduces head movement by choosing request with shortest seek time first, but may result in starvation of distant requests (e.g if a request is always available nearby)
- SCAN/Elevator better than FIFO, and avoids starvation, but does not take advantage of sequential locality on the down scan
- C-SCAN like SCAN, except avoids disk access on the down-scan and hence improves support for sequential locality

Dumb answers

- FIFO, Clock, EDF, and Two-level scheduling
  - Don’t just as add acronyms you can remember

Dumb answers

- Disk arm scheduling algorithms are used to move the head backward and forward on the disk. We can use many different algorithms to decide and some are better than others. One algorithm include first-come first served. It moves the arm to the location on disk in the order the request arrive in, it is bad cause it has overheads. Sometimes requests will be to inside of disk and outside of disk and arm will move far making disk slow. Moving the disk arm is bad.
- SSTF is where disk scheduler chooses block that is closest to disk head and goes there. It is better as is does not move the arm a long way, but has overheads too but not as many as FCFS. It is slow because we must search list of disk requests find the closest one. May cause CPU starvation if we spend to much time searching list and no other programs can run

Answer the question!!!

- Don’t repeat the question, we set the exam, we know what it is!!!!
- Don’t just write what you know (or don’t know) about the topic area
  - You make us have to search for the real answer.
  - You may be correct, but say a lot of unrelated incorrect stuff.
- Don’t contradict yourself
  - X is better/faster/more efficient than Y, and later Y is better than X
- Marks are awarded for stating WHY an answer is correct.
  - Demonstrates understanding

Exam Content

- For structure and style, look at the sample exam from past years.
- For content, the tutorial questions are a reasonable *guide*.

The questions attempt to examine understanding rather than particular implementations

- Don't expect
  - "Describe OS/161's exception handling on a timer interrupt"
- But you may get
  - "Describe (in general) a feasible sequence of steps that occur in response to a timer interrupt that results in the current process being pre-empted and a new task running"

Examinable Content

- All Lectures, Tutorials, Assignments.

Consultations

- **To be announced**

Practice Exam Questions Available RSN