Introduction

COMP3231/9201/3891/9283
(Extended) Operating Systems
Dr. Kevin Elphinstone
Dr. Leonid Ryzhyk
Operating Systems
@
UNSW
John Lions  

• Played a leading role in bringing UNIX to Australia
  – Founding president of Australia UNIX Users Group
• Based his OS course on understanding the UNIX V6 source code
  – Forward thinking at the time.
  – Authored a source code commentary to aid understanding

  “After 20 years, this is still the best exposition of the workings of a "real" operating system”
  — Ken Thompson, co-author of Unix

  – Publication was suppressed by AT&T, and the commentary was widely photocopied "underground".
  – Finally officially published in 1996.
• Lions Garden dedicated in 2002
• 2006 Alumni established John Lions Chair of Operating Systems
  – 2009 Gernot Heiser became the inaugural chair.
1990s

• 1991 DiSy (Distributed Systems) group started
  – Gernot Heiser (and others) and two PhD students: Jerry Vochteloo and myself.
• 1995 Established collaboration with Jochen Liedtke, original architect of L4 microkernel
  – Developed L4mips microkernel
    • Featured fastest interprocess communication at the time
    • Still fastest on single issue processor
• 1997 COMP9242 Advanced Operating Systems was born.
  – Designed and built U4600:
    • 64-bit MIPS computer
  – Software based on L4mips
2000s

- 2002 UNSW/ANU wins bin to establish NICTA
- Two parallel streams began
  - Commercialisation of L4
  - ERTOS research group and agenda established; Gernot Heiser leader.
UNSW/NICTA startup
OK Labs Timeline

- 1994: Begin of microkernel research at UNSW
- 1997–2003: multiple open-source releases
- 2004: First consulting engagement with Qualcomm
- 2006: Open Kernel Labs founded, first L4 phone ships in Japan
- Today: Customer base of blue-chip multinationals
  - Qualcomm, ST-Ericsson, Motorola, …
- Total deployment to date 1.1 billion devices!
  - Present shipping rate: > 20 million per month
We report on the formal, machine-checked verification of the seL4 microkernel from an abstract specification down to its C implementation. We assume correctness of compiler, assembly code, hardware, and boot code.
The ultimate way to keep your computer safe from harm

FLAWS in the code, or "kernel", that sits at the heart of modern computers leave them prone to occasional malfunction and vulnerable to attack by worms and viruses. So the development of a secure general-purpose microkernel could pave the way for a safer world. Computer scientists involved in this field believe that a kernel designed for that purpose could provide a much safer computing environment.

"The kernel is the software component that sits at the heart of a computer," explains John Klein, a computer scientist at the University of Edinburgh. "It manages the resources of the computer, such as memory, and provides interfaces for the user to interact with the computer."

According to Klein, current kernels are too complex and too closely tied to the underlying hardware, making them difficult to secure. "They're like a Russian doll, with many different layers. Each layer adds complexity and makes it harder to understand what's going on inside," he says.

"A microkernel is a much simpler approach. It reduces the attack surface, making it easier to find and fix security vulnerabilities," adds Klein.

"However, microkernels are much more complex to implement and require a lot more effort," says Klein. "But they offer the potential for much greater security than traditional kernels."

Klein and his team are working on a new microkernel, called "Forsyth", named after the city where the University of Edinburgh is located. Forsyth is designed to be secure and efficient, and is currently being tested on a number of different computer systems.

"We believe that Forsyth could be a significant step forward in the development of secure computing," says Klein. "It's not perfect, but it's a step in the right direction."

The ultimate way to keep your computer safe from harm is to use a secure microkernel, such as Forsyth, and to be vigilant about potential security threats. By doing so, you can help to create a safer computing environment for everyone.
Welcome to OS @ UNSW
Course Outline

• Prerequisites
  – COMPXXXX Data structures and algorithms
    • Stacks, queues, hash tables, lists, trees, heaps,....
  – COMPXXXX Microprocessor and Interfacing
    • Assembly programming
    • Mapping of high-level procedural language to assembly language
    • Interrupts
  – You are expected to be competent programmers!!!!
    • We will be using the C programming language
      – The dominant language for OS implementation.
      – Need to understand pointers, pointer arithmetic, explicit memory allocation.
Why does this fail?

```c
void set(int *x, int *y) {
    *x = 1; *y = 2;
}

void thingy() {
    int *a, *b;
    set(a, b);
    printf("%d %d\n", *a, *b);
}
```
Lectures

• Common for all courses (3231/3891/9201/9283)
• Tue, 3-5pm, Biomedical Theatre D (K-E27-D)
• Thu, 3-4pm, Chemical Sc M18
  • (ex Applied Sc (K-F10-M18) Webster Theatre B
    • Extended OS Thu 4-5pm, Webster 251 (K-G14-251)
      • starts in week 2
    • The lecture notes will be available on the course web site
      • Available prior to lectures, when possible.
      • Slide numbers for note taking, when not.
    • The lecture notes and textbook are NOT a substitute for attending lectures.
Extended OS Comp3891/9283

• A combination of:
  – Examination of topics in more depth
  – Looking at research in area (past/present)
  – OS/161 internals in more depth

• Assumes the tutorials are too easy
  – Effectively replaces the tutorial with extra interactive lecture.
Tutorials

• Start in week 2
• A tutorial participation mark will contribute to your final assessment.
  – Participation means participation, NOT attendance.
  – Comp3891/9283 students excluded
  – Comp9201 optional
• You will only get participation marks in your enrolled tutorial.
Assignments

• Assignments form a substantial component of your assessment.
• They are challenging!!!!
  – Because operating systems are challenging
• We will be using OS/161,
  – an educational operating system
  – developed by the Systems Group At Harvard
  – It contains roughly 20,000 lines of code and comments
Assignments

• Don’t under estimate the time needed to do the assignments.
  – 80% is understanding
  – 20% programming

• If you start a couple days before they are due, you will be late.

• To encourage you to start early,
  – Bonus 10% of awarded mark of the assignment for finishing a week early
  – See course handout for exact details
    • Read the fine print!!!!
Assignments

• Assignments are in pairs
  – except warm-up Asst0
  – Info on how to pair up available soon

• We usually offer advanced versions of the assignments
  – Available bonus marks are small compared to amount of effort required.
  – Student should do it for the challenge, not the marks.
  – Attempting the advanced component is not a valid excuse for failure to complete the normal component of the assignment
    • consider it a different optional assignment

• Extended OS students (COMP3891/9283) are encouraged to attempt the advanced assignments
Assignments

• Three assignments
  – due roughly week 6, 9, 13
• Also warm up bonus assignment due in week 4
  – It’s a warm up to have you familiarize yourself with the environment and easy marks.
  – Do not use it as a gauge for judging the difficulty of the following assignments.
Assignments

• Late penalty
  – 4% of total assignment value per day
    • Assignment is worth 20%
    • You get 18, and are 2 days late
    • Final mark = 18 – (20*0.04*2) = 16 (16.4)

• Assignments are only accepted up to one week late. 8+ days = 0
Assignments

• To help you with the assignments
  – We dedicate a tutorial per-assignment to discuss issues related to the assignment
  – Prepare for them!!!!!
Plagiarism

• We take cheating seriously!!!
• We systematically check for plagiarised code
  – Penalties are generally sufficient to make it difficult to pass
# Sample Cheating Statistics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>enrolment</td>
<td>178</td>
<td>410</td>
<td>320</td>
<td>300</td>
<td>107</td>
<td>298</td>
<td>156</td>
<td>333</td>
<td>133</td>
</tr>
<tr>
<td>suspected cheaters</td>
<td>10(6%)</td>
<td>26(6%)</td>
<td>22(7%)</td>
<td>26(9%)</td>
<td>20(19%)</td>
<td>15(5%)</td>
<td>???(?%)</td>
<td>13 (4%)</td>
<td>???(?%)</td>
</tr>
<tr>
<td>full penalties</td>
<td>2*</td>
<td>6*</td>
<td>9*</td>
<td>14*</td>
<td>10</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>reduced penalties cheaters</td>
<td>7</td>
<td>15</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>failed cheaters</td>
<td>4</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>10</td>
<td>12</td>
<td>5</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>suspended</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Note: Full penalty 0 FL not applied prior to 2001/S1*
Exams

• There is NO mid-session
• The final written exam is 2 hours
• Supplementary exams are oral.
  – Supplementaries are available according to UNSW & school policy, not as a second chance.
Assessment

- Exam Mark Component
  - Max mark of 100
- Based solely on the final exam

- Class Mark Component
  - Max mark of 100
- 10% tutorial participation
- 90% Assignments
3891/9283

- No tutorial participation component
- Assignment marks scaled to 100
9201

- Optional tutorial participation, we’ll award the better mark of
  - Tutorial participation included as for comp3231
  - Class marked based solely on the assignments
The final assessment is the harmonic mean of the exam and class component.

If \( E \geq 40 \),

\[
M = \frac{2EC}{E+C}
\]
Postgrads (9201/9283)

- Maximum of a 50/50 weighted harmonic mean and a 20/80 harmonic mean
  - Can weight final mark heavily on exam if you can’t commit the time to the assignments
  - You are rewarded for seriously attempting the assignments

- if $E \geq 40$,

$$M = \max \left( \frac{2E}{E + C} ; \frac{5E}{E + 4C} \right)$$
Assessment

- If $E < 40$

$$M = \min\left(44, \frac{2EC}{E + C}\right)$$
Assessment

• You need to perform reasonably consistently in both exam and class components.

• Harmonic mean only has significant effect with significant variation.

• Reserve the right to scale, and scale courses individually if required.
  – Warning: We have not scaled in the past.
Textbook

References

- McKusick et al., *The Design and Implementation of the 4.4 BSD Operating System*, Addison Wesley, 1996
Forum and Wiki

• Forum for Q/A about assignments and course
  – Ask questions there for the benefit of everybody
  – Look there before asking

• Wiki
  – Look here before asking on the forum
  – Contains
    • Tips for setting up
      – Note: we only support CSE machines, you’re on your own at home
      – Most students get a workable home environment going
    • Tips for the assignments
      – Only as good as feedback or your contributions
Consultations/Questions

• Questions should be directed to the forum.
• Admin related queries to Aaron Carroll
  aaronc@cse.unsw.edu.au
• Personal queries can be directed to me
  kevine@cse.unsw.edu.au
• We reserve the right to ignore email sent directly to
  us (including tutors) if it should have been directed to
  the forum.
• Consultation Times
  – TBA
Course Outline

• “the course aims to educate students in the basic concepts and components of operating systems, the relevant characteristics of hardware, and the tradeoffs between conflicting objectives faced by operating systems in efficiently supporting a wide range of applications.”
Course Outline

- Processes and threads
- Concurrency control
- Memory Management
- File Systems
- I/O and Devices
- Scheduling
- Security (maybe)