“Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship in Middle-earth. Go in Peace! I will not say: do not weep; for not all tears are an evil.”
Final exam details

- Closed book, closed notes
- No computers, phones, calculators, etc.
- 2 hr. exam – start at 7pm sharp
- Focus is on virtual memory to dist. systems
- But first-half topics may be needed
- Includes projects 3 and 4
Nuts and bolts

- 7-9pm on Monday, 04/29
- Go to your assigned room, find your exam:

<table>
<thead>
<tr>
<th>Start uniqname</th>
<th>End uniqname</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ERICHSEB</td>
<td>DOW 1013</td>
</tr>
<tr>
<td>ESATTLER</td>
<td>JONTOTO</td>
<td>BBB 1670</td>
</tr>
<tr>
<td>JOONJAE</td>
<td>LOUISMIN</td>
<td>DOW 1017</td>
</tr>
<tr>
<td>LYECE</td>
<td>PDAK</td>
<td>DOW 1014</td>
</tr>
<tr>
<td>PETERJH</td>
<td>SHIZHOU</td>
<td>DOW 1010</td>
</tr>
<tr>
<td>SKOLERST</td>
<td>WANGGY</td>
<td>DOW 1005</td>
</tr>
<tr>
<td>WENCTSA</td>
<td>ZHONGHS</td>
<td>BBB 1690</td>
</tr>
<tr>
<td>ZHOUFENG</td>
<td>ZZZ</td>
<td>DOW 1006</td>
</tr>
</tbody>
</table>
Different from midterm?

- Same basic exam style, but...
  - 1\textsuperscript{st} half of class has more programming
  - 2\textsuperscript{nd} half of class has more concepts
  - Practice exams are a representative sample
How to study

- Do sample exams, time yourself
- Reflect on midterm strengths/weaknesses
- Redo all the discussion questions
- Review **all** of project 3 and project 4
- A lot of lecture material **not** in the projects
- Study groups: ask each other questions
  » Textbook a good source of questions
Exam-taking tips

- Skim problems – answer easiest first
- Read coding questions **carefully**
  - Think and design before writing code
- Don’t get bogged down by any 1 question
  - Can get partial credit even on tough questions
- Stuck? Answer part of the question well
- Familiarity helps you avoid time pressure
Topics – Address spaces

- Abstractions provided:
  - Address independence
  - Larger (virtual) memory
  - Protection/controlled sharing
- Uniprogramming vs. multiprogramming
- Static vs. dynamic translation
Topics – Address spaces (2)

- Translation mechanisms:
  - Base & bound
  - Segmentation
  - Paging
  - Multi-level paging
  - TLB

- Replacement
  - Random, FIFO, OPT, LRU, clock
Topics – Address spaces (3)

- Deferring and avoiding work:
  - Writebacks to disk
  - Zero-filling
  - Copy-on-write

- Page table bits
  - Protection, valid, resident, reference, dirty
  - How to eliminate or emulate (dirty, reference)

- Multi-process issues:
  - Global vs. local replacement, thrashing, working set
Topics – Address spaces (4)

- MMU performs lookup
  - if needed: triggers fault, retries operation
- Creating a process
  - Fork, exec, copy-on-write, mmap’ed files
  - Shell code
- Kernel address space
  - Differences from user address space
  - Protecting translation data, mode bit
Topics – File systems (1)

- Storage devices
  - Hard drives vs. solid state drives (flash storage)
- Disk scheduling
  - FIFO, SSTF, SCAN
- File system design
Topics – File systems (2)

- File system structure
  - Files, directories, metadata, inodes
  - Organization within a file:
    » Contiguous, indexed files, multi-level indexing
  - Organization of files:
    » Hierarchical directories, traversing pathnames
- File caching
Topics – File systems (3)

- Transactions
  - Limits of careful ordering of updates
  - Shadowing
  - Logging
- Log-structured file systems
- RAID
Topics – networking

- UNIX API
- Sockets
  - Like a virtual network card
  - Connect to (hostname, port) pair
- TCP
  - Reliability
  - Ordering
  - Byte stream abstraction
Topics – distributed systems

- Client-server
  - Server structure
  - RPC and stub generation
- Happens-before and lamport clocks
- Distributed file systems
  - Caching
Topics – dist. systems (2)

- Fault tolerance
  - Types of faults (crash vs Byzantine)
  - Primary-backup replication
Questions?
482 – The Big Ideas

- Abstraction – virtualizing a resource
- Concurrency and consistency
- Caching and exploiting locality
- Indirection
  - Gains power, hurts performance
  - Recover performance via caching
- Ordering, atomicity, and transactions
- Tolerating faults through redundancy