Sistemas Operativos Avanzados

2016/17

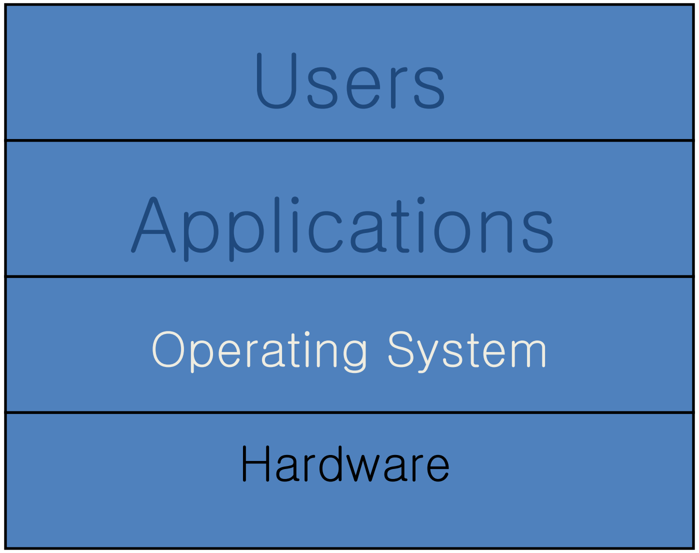
vpuente@unican.es

What is an Operating System?

Operating System (OS):

Software that converts hardware into a useful form for applications

Not easy to define precisely…



Users

Applications

Operating System

Hardware

What DOES OS Provide?

- Role #1: Abstraction - Provide standard library for resources

- What is a resource?
  - Anything valuable (e.g., CPU, memory, disk, I/O device)

- What abstraction does modern OS typically provide for each resource?
  - CPU:
    - process and/or thread
  - Memory:
    - address space
  - Disk:
    - files

- Advantages of OS providing abstraction?
  - Allow applications to reuse common facilities
  - Make different devices look the same
  - Provide higher-level or more useful functionality

- Challenges
  - What are the correct abstractions?
  - How much of hardware should be exposed?

What DOES OS Provide?

- Role #2: Resource management – Share resources well

- Advantages of OS providing resource management?
  - Protect applications from one another
  - Provide efficient access to resources (cost, time, energy)
  - Provide fair access to resources

- Challenges
  - What are the correct mechanisms?
  - What are the correct policies?

OS Organization

- How to cover all the topics relevant to operating systems?

Three PIECES

- Virtualization:
  - Make each application believe it has each resource to itself
- Concurrency:
  - Events are occurring simultaneously and may interact with one another
- Persistence: Access information permanently
  - Lifetime of information is longer than lifetime of any one process
  - Machine may be rebooted, machine may lose power or crash unexpectedly

Advanced Topics (beyond our reach)

- Current systems
  - Multiprocessors
  - Networked and distributed systems
  - Virtual machines
  - Containers
  - …

- Many of the pushed by the explosive demand (a.k.a. Massive complexity under constrained cost)

- This is the support of the world: it will keep changing …

- Some of them covered in SVS (M1679)

Why study Operating Systems?

- Build, modify, or administer an operating system
- Understand system performance
  - Behavior of OS impacts entire machine
  - Tune workload performance
  - Apply knowledge across many layers
    - Computer architecture, programming languages, data structures and algorithms, and performance modeling
- Fun and challenging to understand large, complex systems

Approach

- We will follow “Operating System: Three Easy Pieces” (OSTEP) style
  - From the bottom concepts to state-of-the-art approaches
  - Eminently practical style: all supported by ”simulators” and simple coding examples
  - Assumes some basic knowledge in architecture, C, assembler and system administration
  - More than just a text book…

- Structure
  - The three parts are split in small *pieces* (~40 in the book)
  - Each chapter is build over the previous one (can’t miss the beat)
  - Each chapter has attached a ”Homework” to reinforce the : from using python simulators to write small pieces of code (C)
  - 5+1 Labs, developed on top of xv6

Lecture/Lab structure

- We mix dynamically both
  - The real thing is that there is no separation between “theory” and “practice”

- Blocks of:
  - 1st hour: Introduction to the topic
  - 2nd hour: Introduce/develop of Labs
  - Personal work (out the lab): 6 hours (labs and homework)
  - 10 hours/week
  - Strict schedule

- Although the original course/book is designed for 15 week semester (150h work), we will need to drop some details or advanced topics (and half of the labs)

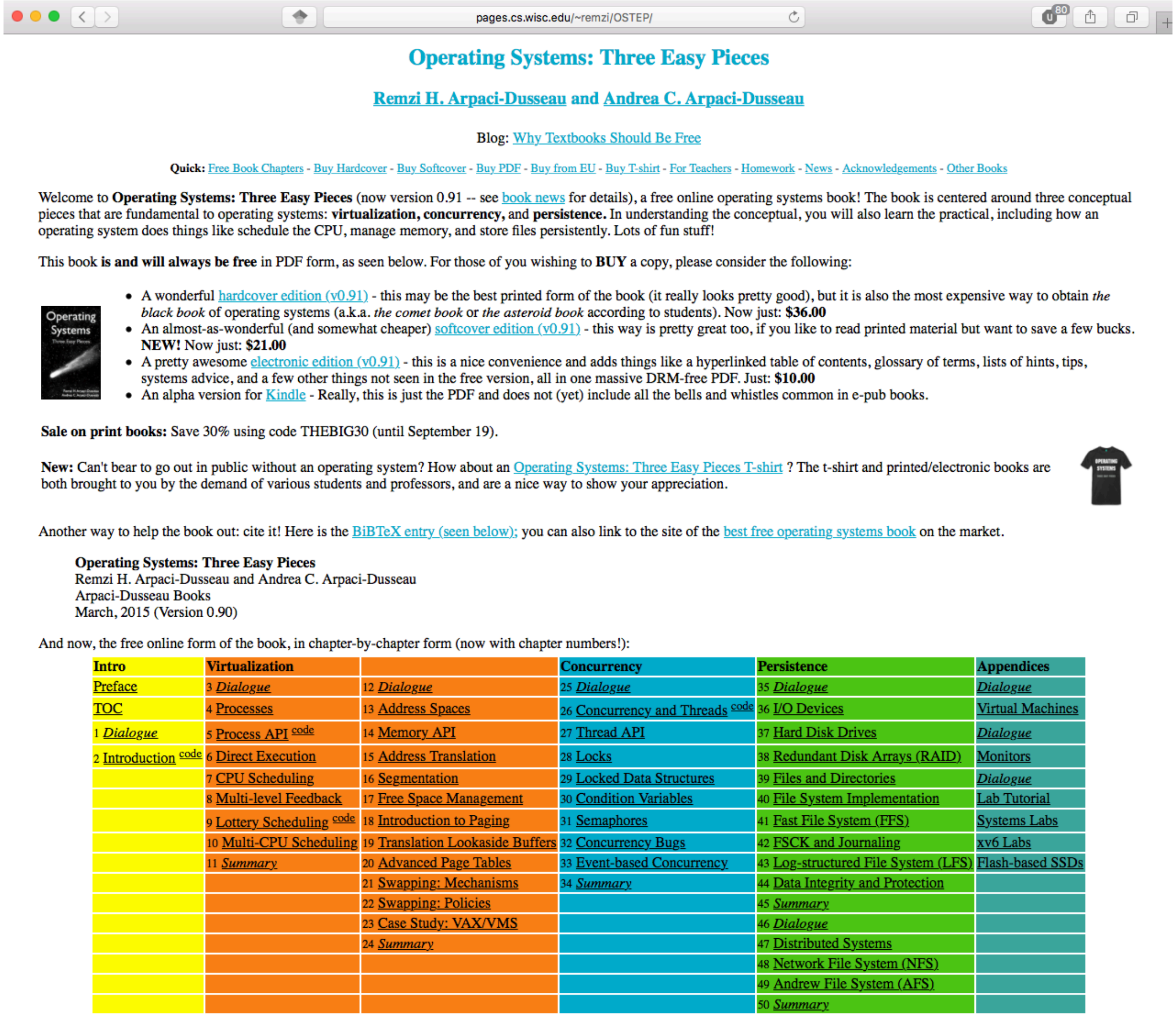
Schedule

Start	Chapter	Lab	Homework
19-sep.	1 Intro	P0 Lab Intro and C refresh	
20-sep.	4. The Abstraction: The Process/ 5. Interlude: Process API		Process Intro / Process API
26-sep.	6. Mechanism: Limited Direct Execution		Direct Execution
27-sep.	7. Scheduling: Introduction	P1 System Calls	Scheduler
3-oct.	8: Scheduling: The Multi-Level Feedback Queue		MLFQ Scheduling
4-oct.	9: Scheduling: Proportional Share		Lottery Scheduling
10-oct.	10. Multiprocessor Scheduling (Advanced)	P2 Scheduling	
11-oct.	13. The Abstraction: Address Space / 14. Memory API		VM API
17-Oct	15. Address Translation		Relocation
18-Oct	16. Segmentation		Segmentation
24-Oct	17. Free-Space Management		Free Space
25-Oct	18. Paging: Introduction		Paging
31-Oct	19. Translation Lookaside Buffers		TLBs
7-Nov	20. Paging: Smaller Tables		Multi-level Paging
8-Nov	21. Swapping: Mechanisms	P3 Memory	Paging Mechanism
14-Nov	22. Swaping: Policies		Paging Policy
18-Nov	Mid Term Exam		
15-Nov	26. Concurrency: An Introduction / 27. Interlude: Thread API		Threads (Intro)/Threads (API)
21-Nov	28. Locks		Threads (Locks)
22-Nov	29. Lock-based Concurrent Data Structures		
28-Nov	30. Condition Variables		Threads (CVs)
29-Nov	31. Semaphore		
5-Dic	32. Common Concurrency Problems.	P4 Threads	Threads (Bugs)
12-Dic	33. Event-based Concurrency (Advanced)		
13-Dic	36. I/O Devices		
19-Dic	37. Hard Disk Drives		Disks
20-Dic	39. File and Directories		FS Intro
9-Ene	40. File system Implementation.		FS Implement
10-Ene	42. Crash Consistency: FSCK and Journaling		FFS
16-Ene	42. Crash Consistency: FSCK and Journaling	P5 File systems	
17-Ene	43. Log-structured File Systems		
18-Ene	Mid Term Exam		

Material

- All written material will be in “English”
  - Lecture notes, Homework/Lab guides, etc…
- *Git* as communication “device”: all material will be delivered via www.gitlab.com
  - An e-mail inviting to join the course project will be sent to unican account
  - Slides, labs, other reference material is there
  - It uses “git” to have a “time-track”
    - Lecture notes updates
    - Additional material
- Use git to allow me “track” your personal work

Book (ostep.org)



Operating Systems: Three Easy Pieces

Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

Blog: Why Textbooks Should Be Free

Quick: Free Book Chapters - Buy Hardcover - Buy Softcover - Buy PDF - Buy from EU - Buy T-shirt - For Teachers - Homework - News - Acknowledgements - Other Books

Welcome to **Operating Systems: Three Easy Pieces** (now version 0.91 -- see book news for details), a free online operating systems book! The book is centered around three conceptual pieces that are fundamental to operating systems: **virtualization, concurrency,** and **persistence.** In understanding the conceptual, you will also learn the practical, including how an operating system does things like schedule the CPU, manage memory, and store files persistently. Lots of fun stuff!

This book **is and will always be free** in PDF form, as seen below. For those of you wishing to **BUY** a copy, please consider the following:

- A wonderful hardcover edition (v0.91) - this may be the best printed form of the book (it really looks pretty good), but it is also the most expensive way to obtain *the black book* of operating systems (a.k.a. *the comet book* or *the asteroid book* according to students). Now just: **$36.00**
- An almost-as-wonderful (and somewhat cheaper) softcover edition (v0.91) - this way is pretty great too, if you like to read printed material but want to save a few bucks. **NEW!** Now just: **$21.00**
- A pretty awesome electronic edition (v0.91) - this is a nice convenience and adds things like a hyperlinked table of contents, glossary of terms, lists of hints, tips, systems advice, and a few other things not seen in the free version, all in one massive DRM-free PDF. Just: **$10.00**
- An alpha version for Kindle - Really, this is just the PDF and does not (yet) include all the bells and whistles common in e-pub books.

Sale on print books: Save 30% using code THEBIG30 (until September 19).

New: Can't bear to go out in public without an operating system? How about an Operating Systems: Three Easy Pieces T-shirt ? The t-shirt and printed/electronic books are both brought to you by the demand of various students and professors, and are a nice way to show your appreciation.

Another way to help the book out: cite it! Here is the BiBTeX entry (seen below); you can also link to the site of the best free operating systems book on the market.

Operating Systems: Three Easy Pieces
Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
Arpaci-Dusseau Books
March, 2015 (Version 0.90)

And now, the free online form of the book, in chapter-by-chapter form (now with chapter numbers!):

Intro	**Virtualization**		**Concurrency**	**Persistence**	**Appendices**
Preface	3 *Dialogue*	12 *Dialogue*	25 *Dialogue*	35 *Dialogue*	*Dialogue*
TOC	4 Processes	13 Address Spaces	26 Concurrency and Threads ^{code}	36 I/O Devices	Virtual Machines
1 *Dialogue*	5 Process API ^{code}	14 Memory API	27 Thread API	37 Hard Disk Drives	*Dialogue*
2 Introduction ^{code}	6 Direct Execution	15 Address Translation	28 Locks	38 Redundant Disk Arrays (RAID)	Monitors
	7 CPU Scheduling	16 Segmentation	29 Locked Data Structures	39 Files and Directories	*Dialogue*
	8 Multi-level Feedback	17 Free Space Management	30 Condition Variables	40 File System Implementation	Lab Tutorial
	9 Lottery Scheduling ^{code}	18 Introduction to Paging	31 Semaphores	41 Fast File System (FFS)	Systems Labs
	10 Multi-CPU Scheduling	19 Translation Lookaside Buffers	32 Concurrency Bugs	42 FSCK and Journaling	xv6 Labs
	11 *Summary*	20 Advanced Page Tables	33 Event-based Concurrency	43 Log-structured File System (LFS)	Flash-based SSDs
		21 Swapping: Mechanisms	34 *Summary*	44 Data Integrity and Protection	
		22 Swapping: Policies		45 *Summary*	
		23 Case Study: VAX/VMS		46 *Dialogue*	
		24 *Summary*		47 Distributed Systems	
				48 Network File System (NFS)	
				49 Andrew File System (AFS)	
				50 *Summary*	

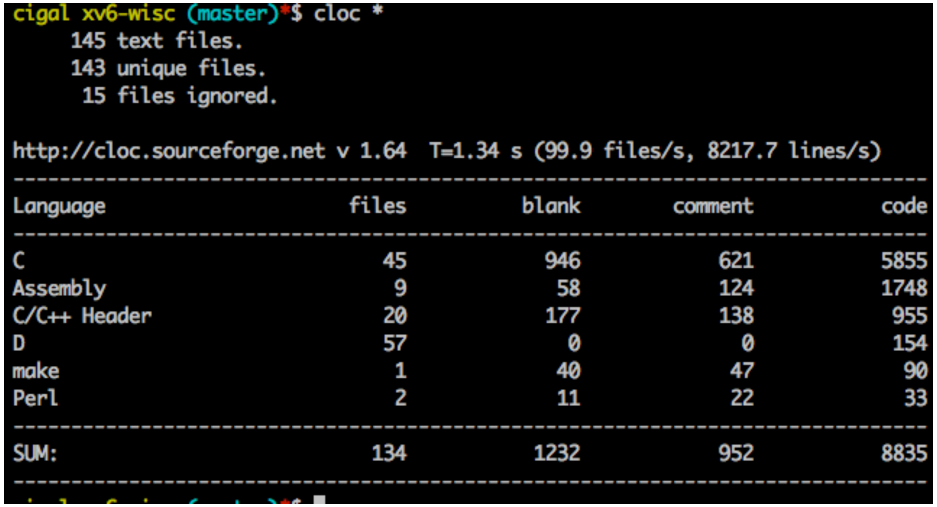
Homeworks

- Some chapter (most) include homework
  - Homeworks can be used to solidify your knowledge of the material in each of the chapters
  - Most homeworks are based on running little **simulators,** which mimic some aspect of an operating system: For example, a disk scheduling simulator could be useful in understanding how different disk scheduling algorithms work:
    - Most of them provides the solution
  - Some home-works are just short programming exercises, allowing you to explore how real systems work and complement Lab work.

- Homework are done in personal-time

Labs: C and xv6

- Refresh C knowledge
- Use a “toy” kernel to dig into implementation details
  - It is a clean and beautiful little kernel, and thus a perfect object for our study and usage.
  - It was developed by OS Eng. In MIT as a port of K&R original Unix R6/PDP11
  - Use al real kernel (such as linux) will be certainly overkill



```
cigal xv6-wisc (master)*$ cloc *
     145 text files.
     143 unique files.
      15 files ignored.

http://cloc.sourceforge.net v 1.64  T=1.34 s (99.9 files/s, 8217.7 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                               45            946            621           5855
Assembly                         9             58            124           1748
C/C++ Header                    20            177            138            955
D                               57              0              0            154
make                             1             40             47             90
Perl                             2             11             22             33
-------------------------------------------------------------------------------
SUM:                           134           1232            952           8835
-------------------------------------------------------------------------------
```

Prerequisites

- All OS and architecture previous subjects(ugh!)

Evaluation

- 40% Final exam
- 60% 2 Mid-term exams
  - Virtualization
  - Concurrency & Persistence

- Mid-term
  - Includes all: Theory and Lab (practical)
  - If average > 6 ➔ Course will be passed