Advanced Operating Systems and Virtualization

Alessandro Pellegrini

A.Y. 2018/2019

SAPIENZA
UNIVERSITÀ DI ROMA

Basic Information

- Lecture Schedule:
  - Course begins today! ☺
  - Course ends on May 31st
  - Lecture slots:
    - Wednesday, 17.00–19.00 (Room B2, Via Ariosto);
    - Friday, 08.00–11.00 (Room B2, Via Ariosto).
- Office Hours:
  - See on my webpage for the schedule
- Contact: pellegrini@diag.uniroma1.it

Exam Rules

- A written test (3/5 of the final mark)
- A code project (2/5 of the final mark)
  - Implementation of facilities within the Linux Kernel
  - Instructions will be given during the course

- We will see internals from Linux 2.4/2.6/3.0/4.0
  - Pick your preferred version!
  - Best if you are compatible with more than one!

Course Outline

- A Primer on Modern Hardware Architectures
- x86 Initial Boot Sequence.
- Linux Kernel Boot
- Memory Management.
- System Calls Management
- Interrupt Management
- Building the Kernel
- Kernel Data Structures

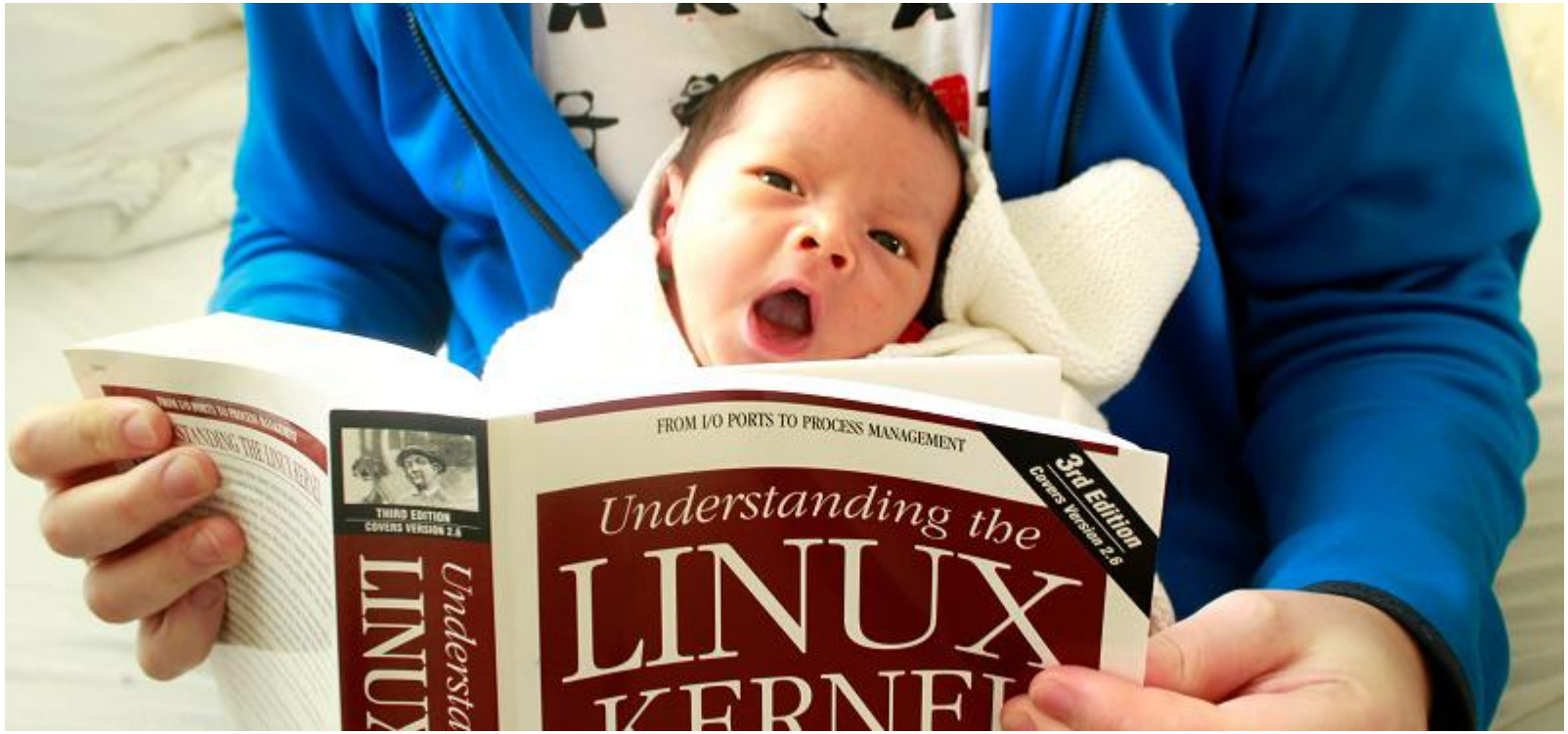
Course Outline

- Virtual File System and Devices
- Userspace Initialization
- Process Startup and Management
- Scheduling Processes
- Loadable Kernel Modules
- Kernel Messaging
- Security Aspects
- Hot Patching

Reference Material

- Daniel P. Bovet, Marco Cesati, *Understanding the Linux Kernel*. O'Reilly.
- Mel Gorman, *Understanding the Linux Virtual Memory Manager*. Prentice Hall.
- Alessandro Rubini, Jonathan Corbet, *Linux Device Drivers*, O'Reilly.
- David A. Rusling, *The Linux Kernel*.

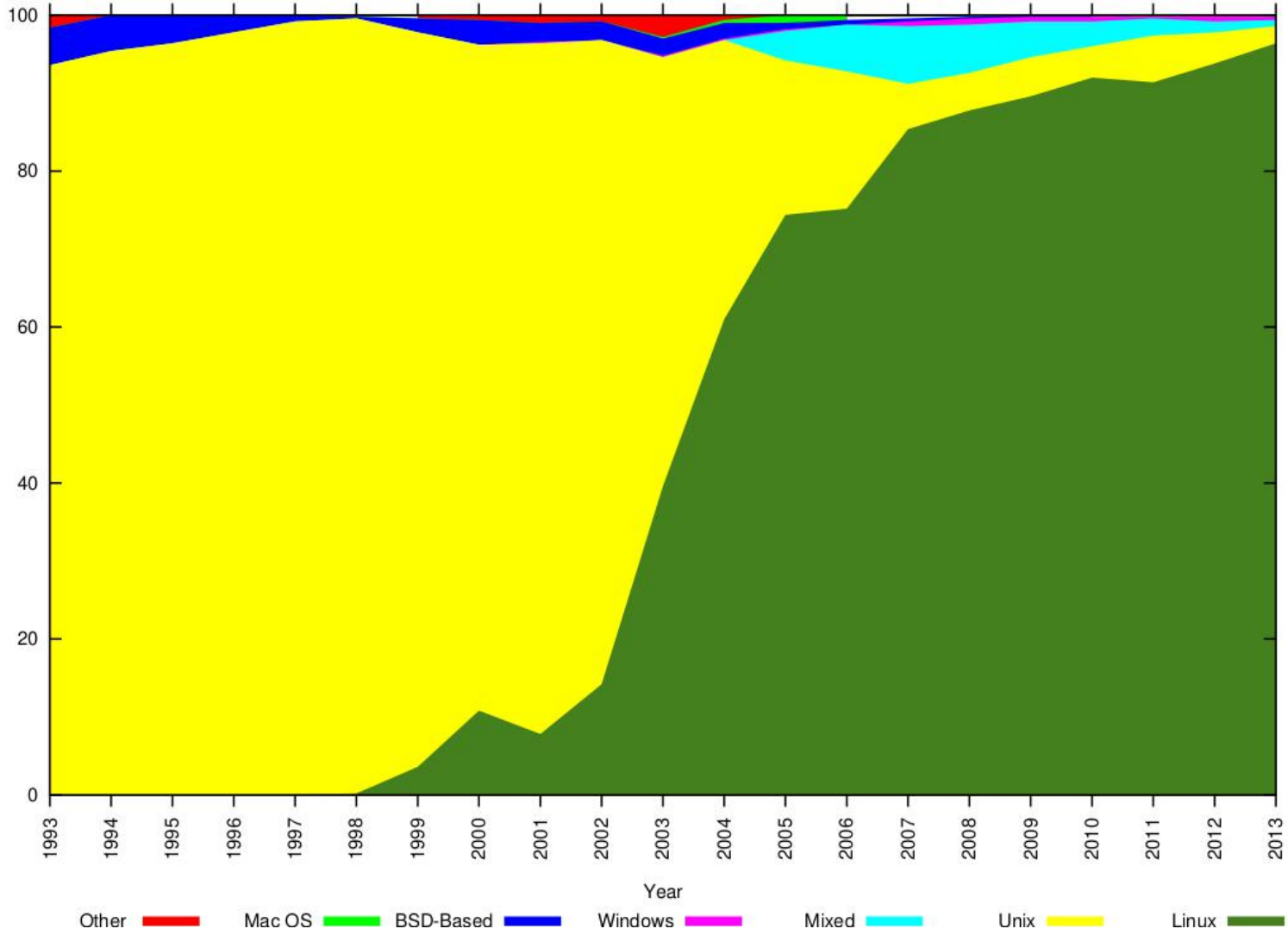
Reference Material



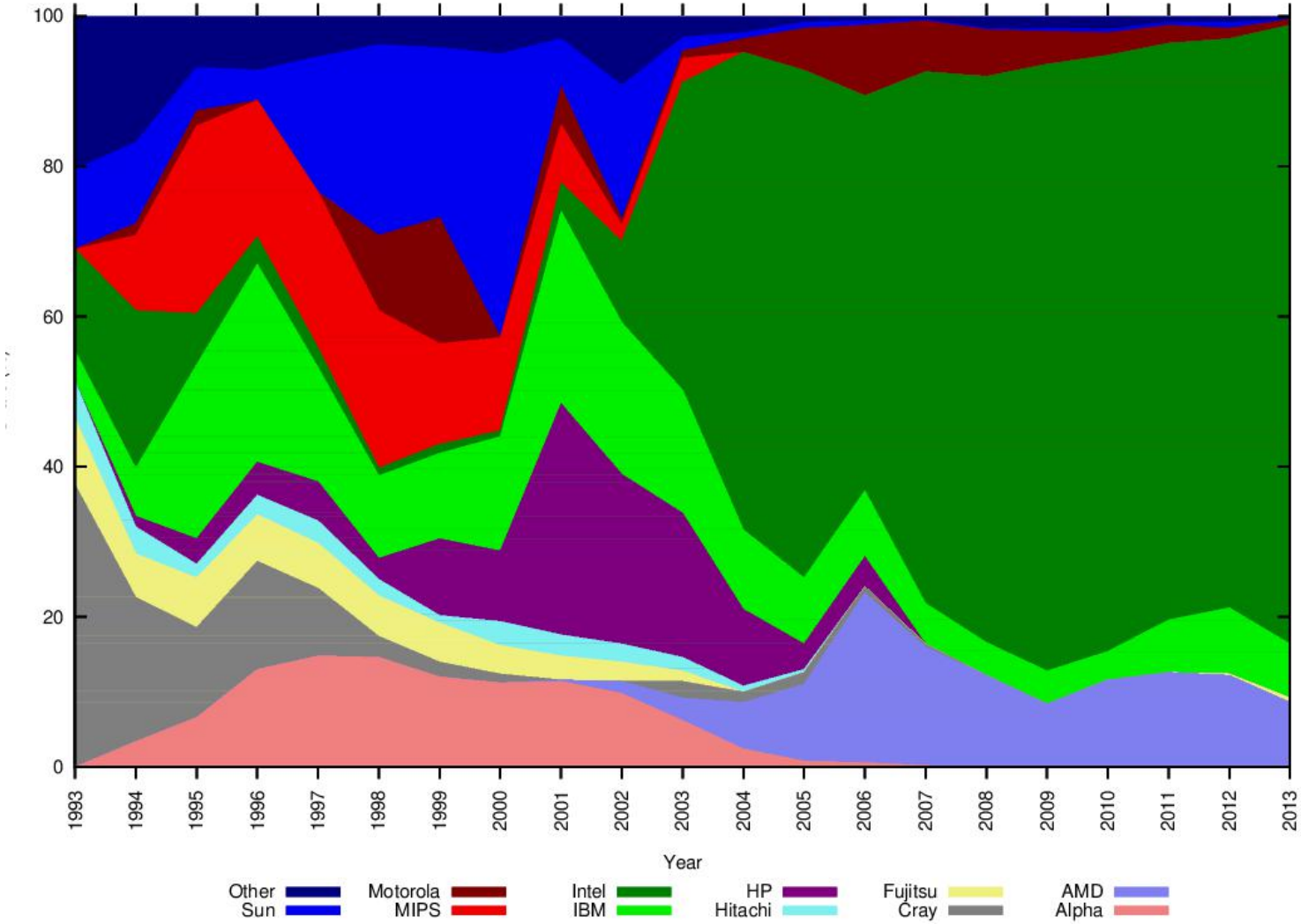
What you should know already

- Computing Architectures
  - Registers, I/O, Interrupts principles, flat memory model, ...
  - Numerical Representations
- Basic x86 assembly notation
- Operating Systems Principles
  - Threads and Processes
  - System Calls
- Algorithms and Data Structures
- Some notions on Concurrency
  - Synchronization, race conditions, critical sections, locks, ...

Why Linux?


100
80
60
40
20
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
Year
Other
Mac OS
BSD-Based
Windows
Mixed
Unix
Linux

Why x86?


100
80
60
40
20
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
Year
Other
Sun
Motorola
MIPS
Intel
IBM
HP
Hitachi
Fujitsu
Cray
AMD
Alpha

Boot Sequence

BIOS/UEFI	The actual Hardware Startup
Bootloader Stage 1	Executes the Stage 2 bootloader (skipped in case of UEFI)
Bootloader Stage 2	Loads and starts the Kernel
Kernel Startup	The Kernel takes control of and initializes the machine (machine-dependent operations)
Init	First process: basic environment initialization (e.g., SystemV Init, systemd)
Runlevels/Targets	Initializes the user environment (e.g., single-user mode, multiuser, graphical, ...)

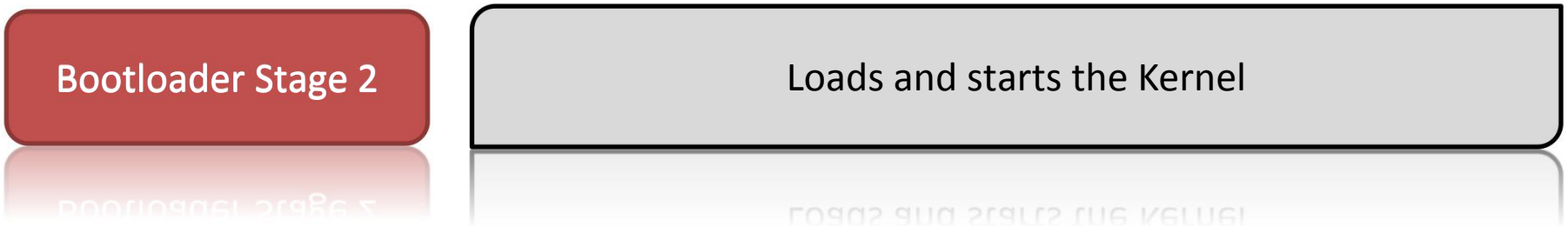
Boot Sequence

- BIOS: Basic Input/Output System
  - Performs some system integrity checks
  - Searches, loads, and executes the Stage 1 boot loader program.
- UEFI: Unified Extensible Firmware Interface
  - More standardized than BIOS
  - Gives much more versatility

Boot Sequence

- Stored in the Master Boot Record (MBR)
- Less than 512 bytes in size
  - primary boot loader info in 1st 446 bytes
  - partition table info in next 64 bytes
  - mbr validation check in last 2 bytes.
- Not enough space to load the kernel: activates Bootloader Stage 2

Boot Sequence



- Typical software: LILO or GRUB
- Allows kernel selection
- Loads from disk the actual kernel startup image and gives control to it

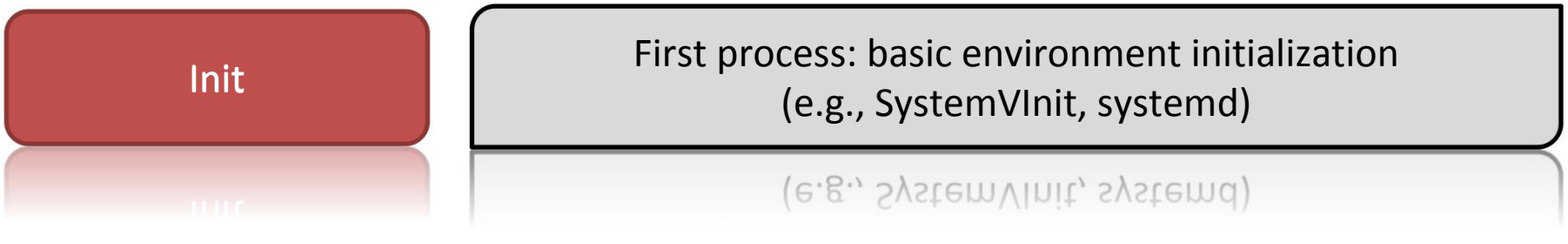
Boot Sequence

Kernel Startup

The Kernel takes control of and initializes the machine (machine-dependent operations)

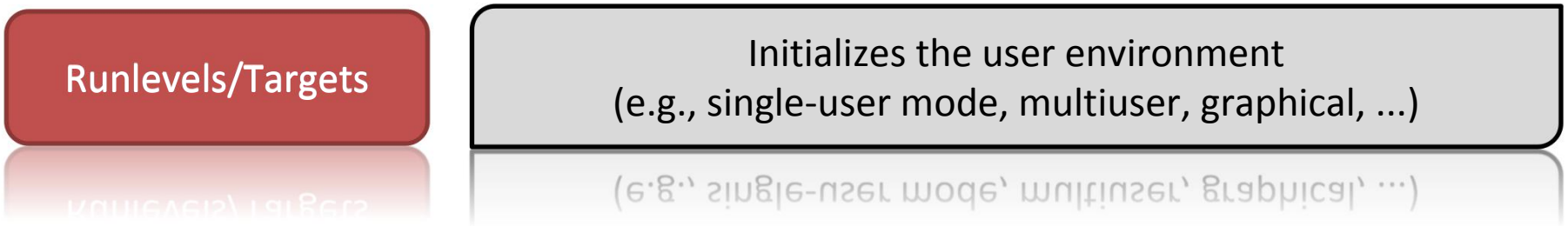
- Configures the hardware environment
  - On x86 this requires multiple memory image initializations
- Mounts the root file system
- Configures internal data structures
- Spawns the first process (init)

Boot Sequence



- Configures the software environment
- Loads the default runlevel
- Spawns other (interactive) processes

Boot Sequence



- They represent the state of a machine
  - running processes and services offered
- On UNIX, they are traditionally six
  - 0: halts the machine
  - 1: single-user mode
  - 2-5: multi-user with different services/facilities
  - 6: reboots the machine