Advanced Operating Systems and Virtualization

Alessandro Pellegrini

A.Y. 2017/2018

Basic Information

- Lecture Schedule:
  - Course begins today! ☺
  - Course ends on June 1st
  - Lecture slots:
    - Tuesday, 08.00 am –10.00 am (Room A3)
    - Friday, 08.00 am –11.00 am (Room A3).
- Office Hours:
  - 1st and 3rd Wednesday of each month, at 3.00 pm
- Contact: pellegrini@dis.uniroma1.it

Exam Rules

- A written test (2/5 of the final mark)
- A code project (3/5 of the final mark)
  - Implementation of facilities within the Linux Kernel
  - Specifications 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

- Booting on an x86 System
  - Memory Management
  - Virtual File System
  - Process/Thread Management
  - Kernel API (e.g., System Calls)
  - Interrupt Management
  - Kernel Data Structures

- How to make a portable Kernel

Course Outline

- Additional Kernel Facilities
  - Kernel Loadable Modules
  - Kernel Debugging
  - Hot Patching
- Security
  - Rootkits
  - Operating systems security aspects
  - Authentication and abilitation
  - Protection domains and secure operating systems
  - System internal attacks and countermeasures
  - IDS and Reference Monitor architectures

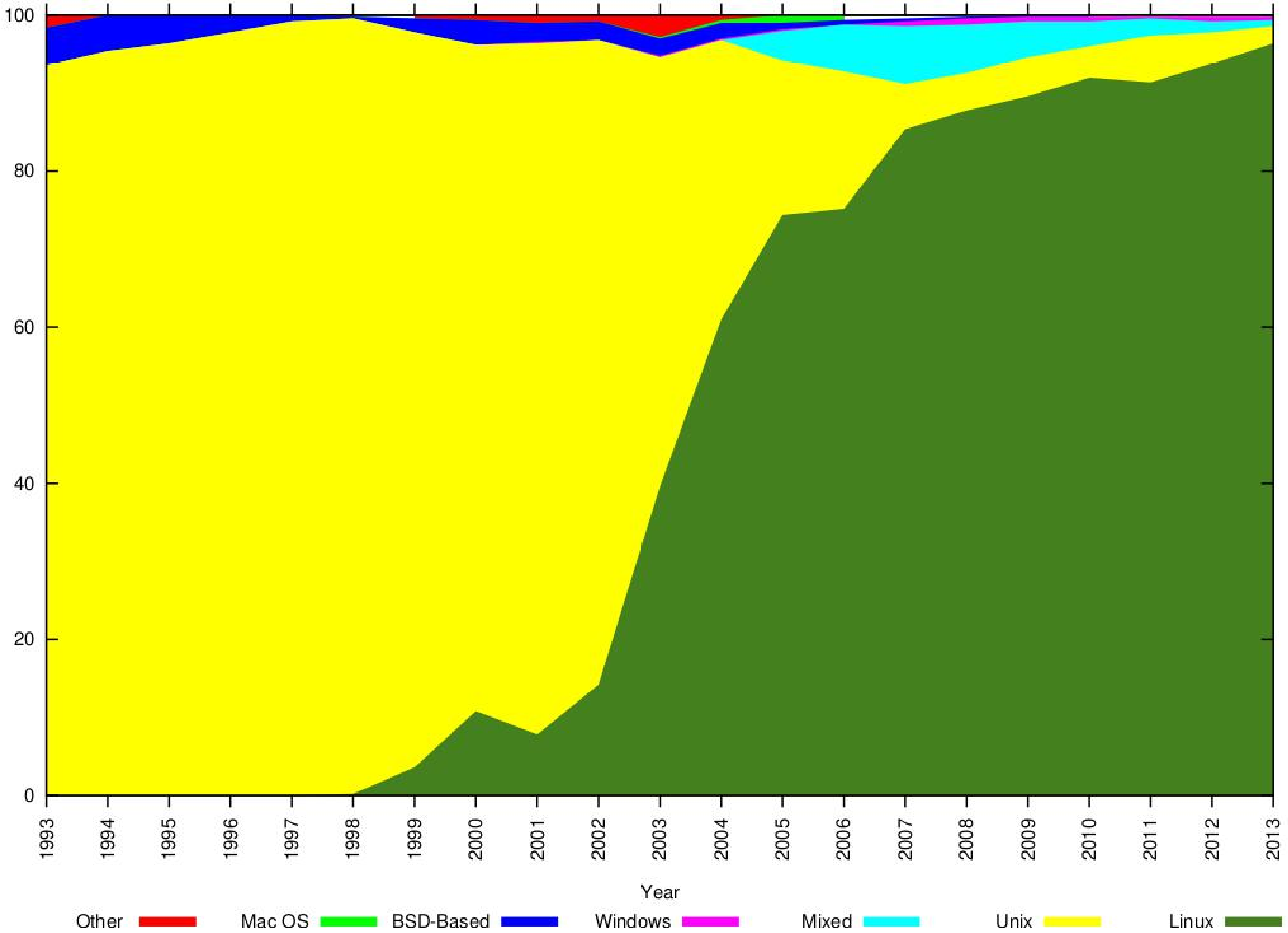
Course Outline

- System Virtualization
  - Basic techniques for system virtualization
  - Support for the guest system execution flow

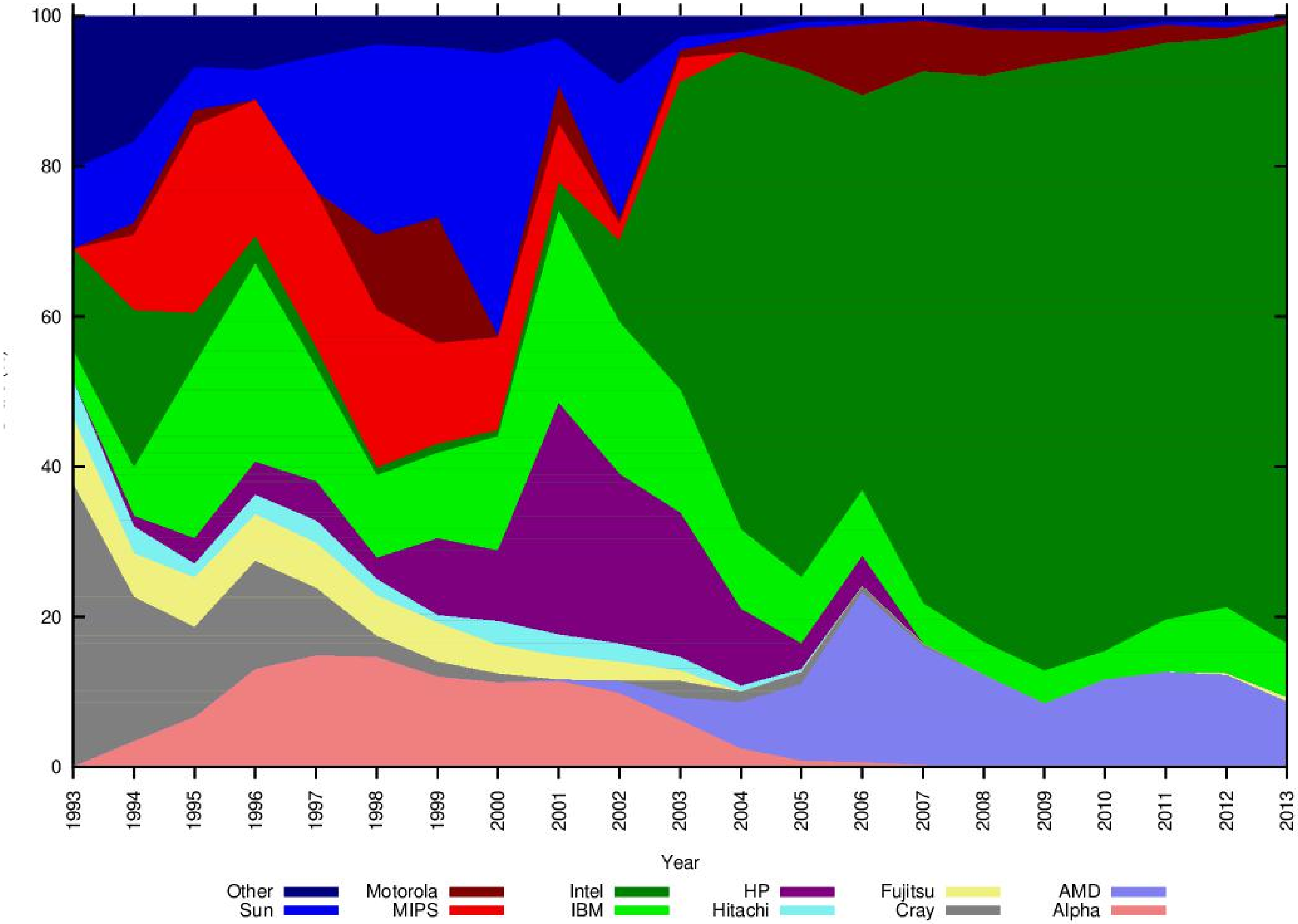
What you should know already

- 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

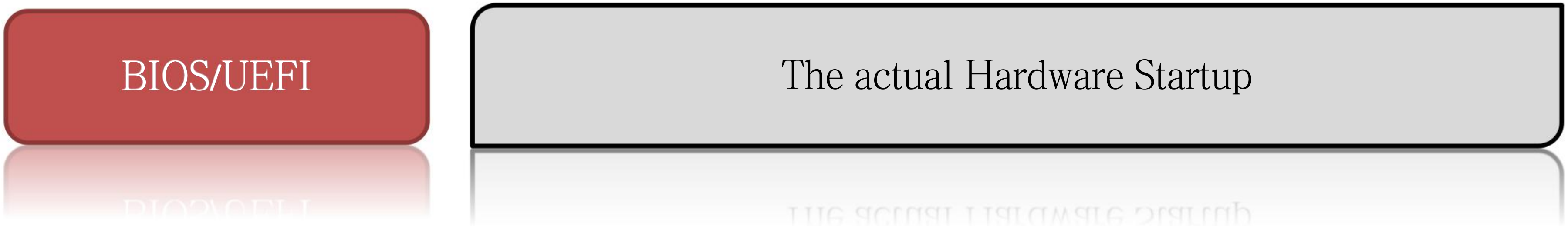
Why x86?


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

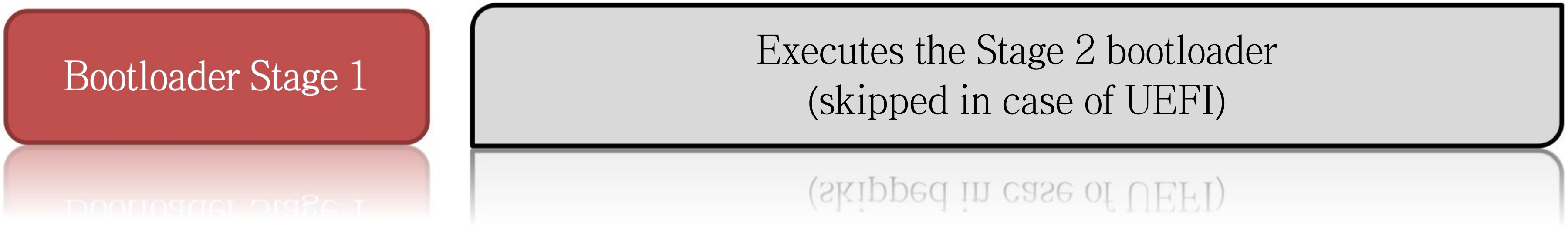
BIOS/UEFI	The actual Hardware Startup
Bootloader Stage 1	Executes the Stage 2 bootloader (skipped in case of UEFI)
Bootloader Stage 2	Loads and starts the Kernel
Kernel Startup	The Kernel takes control of and initializes the machine (machine–dependent operations)
Init	First process: basic environment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

Boot Sequence



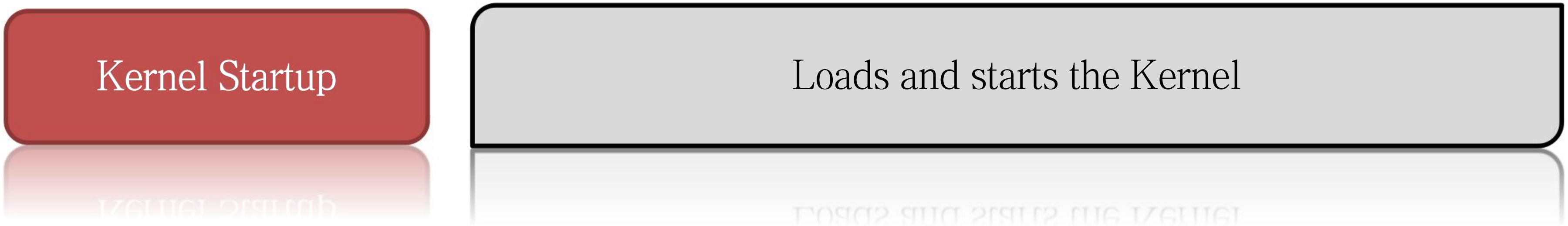
- BIOS: Basic Input/Output System
  - Performs some system integrity checks
  - Searches, loads, and executes the Stage 1 boot loader program.
- UEFI: Unified Extensible Firmware Interface
  - More standardized than BIOS
  - Gives much more versatility

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

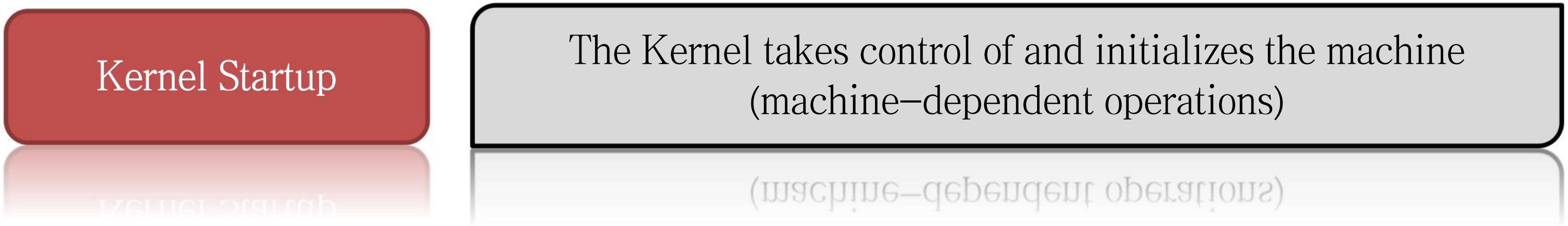


Kernel Startup

Loads and starts the Kernel

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

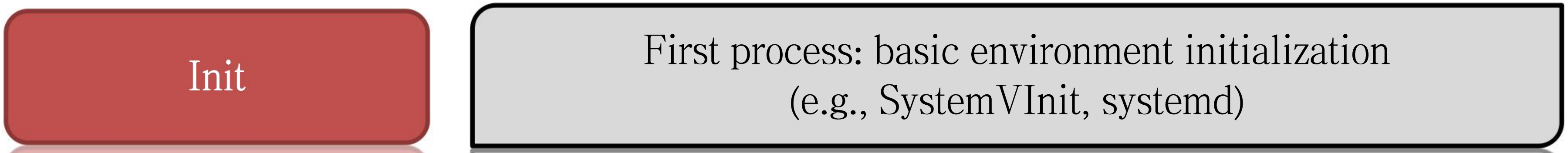
Boot Sequence


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



Init

First process: basic environment initialization
(e.g., SystemVInit, systemd)

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

Boot Sequence

Runlevels/Targets

Initializes the user environment
(e.g., single–user mode, multiuser, graphical, ...)

- 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