Introduction to Linux

IAAS Lectures

University of Stuttgart

Brian Setz, M. Sc.

brian.setz@iaas.uni-stuttgart.de

Institute of Architecture of Application Systems

Contents

- Introduction
- Brief History
- Overview of Linux
- Debian
- Environment Setup
- Exercises

Introduction - Lab Sessions

- Lab Sessions
  - Focus on Linux, understand how Linux functions
  - Interactive examples and demonstrations
  - Assignments at the end of every lab

- Grading: pass / resubmit / fail
  - Max. 2 fails to pass the lab → requirement for exam
  - Not handing in on time → fail
  - Not handing in as PDF → fail

- Material from labs can (will) be covered on the exam

Format

- 7 exercises (Lab sessions on Friday at 14:00-15:30 **according to lab schedule slide**) → Exercises in **<u>groups of two</u>**
  - Requires (root) access to Linux (supported: GNU/Linux Debian)
  - No laptop / PC? Find a partner (possibly use AWS Free Tier)
  - Apply theory from lectures in a Linux environment
  - Copying exercises is forbidden, your responsibility to keep your answers private
    - Cite your sources
- **Deadline: see the deadline slide for the exact time and date**
  - Submission: ILIAS → Operating Systems (neue PO) → 0 Exercises → Submission
  - Write a short, coherent report, include lab number, names + student numbers, submit as <u>PDF</u> before the deadline
  - Not only results, include all the steps you took to get to the results
- **Before Monday 28.10.2019 23:59:59, send email to Brian with the <u>two names + student numbers</u> of your group members** → brian.setz@iaas.uni-stuttgart.de

IAAS Lectures

Introduction – Preliminary Schedule Labs @ 14:00-15:30

Date Lab	Lab Topic
18.10.2019 @ 14:00	*No lab*
25.10.2019 @ 14:00	**1: Introduction, Linux, Virtualization**
08.11.2019 @ 14:00	**2: Process Management**
15.11.2019 @ 14:00	*No Lab*
22.11.2019 @ 14:00	*No Lab*
29.11.2019 @ 14:00	**3: Process Scheduling**
06.12.2019 @ 14:00	Q&A
13.12.2019 @ 14:00	**4: System Calls + Interrupts**
20.12.2019 @ 14:00	*Q&A*
10.01.2020 @ 14:00	**5: Synchronization + Time**
17.01.2020 @ 14:00	**6: Memory Management**
24.01.2020 @ 14:00	Q&A
31.01.2020 @ 14:00	**7: Virtual File System & Block I/ O & I/O Systems**
07.02.2020 @ 14:00	Q&A

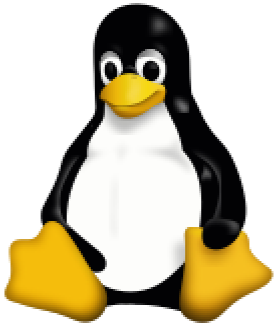
Introduction – Preliminary Schedule Deadlines

Date Lab	Lab Topic
07.11.2019 @ 23:55:00	Deadline Lab 1
21.11.2019 @ 23:55:00	Deadline Lab 2
12.12.2019 @ 23:55:00	Deadline Lab 3
09.01.2019 @ 23:55:00	Deadline Lab 4
23.01.2019 @ 23:55:00	Deadline Lab 5
30.01.2019 @ 23:55:00	Deadline Lab 6
13.02.2019 @ 23:55:00	Deadline Lab 7

Linux

What is Linux?

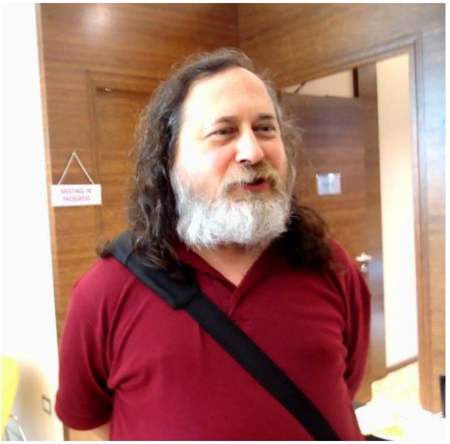
- Linux, free and open-source operating system built around the Linux kernel
- Linux kernel found in:
  - Linux OS's
  - Android
  - Google's Chrome OS



- Pre-emptive multitasking operating system
- Supports symmetrical multiprocessing
- Asynchronous interrupts
- Portable (24+ different processor architectures)

Linux – Brief History

- 1970 – **Ken Thompson**, **Dennis Ritchie** (AT&T Bell Lab) release Unix
  - Portable, Modular → Shell scripting + command line
- 1977 – Development of Berkeley Software Distribution (BSD)
- 1983 – **Richard Stallman** starts GNU (GNU is Not Unix) project
  - Goal: free UNIX-like operating system
- 1985 – Intel releases 80386 (i386)
  - First x86 microprocessor with 32 bit instructions + memory management with paging
- 1987 – **Andrew S. Tanenbaum** releases MINIX → academic, sourcecode restricted
- 1992 – BSD vs. Unix lawsuit → limits adaption and development
- 1990-1992 → GNU kernel (GNU Hurd) fails to attract development effort
- 1990's → <u>Lack of a free and widely adopted kernel</u>



Richard Stallman

Linux Kernel

- 1991 – **Linus Torvalds** develops first version of Linux
  - Usenet posting in comp.os.minix

Hello everybody out there using minix -

*I'm doing a (free) operating system (**just a hobby, won't be big and professional** like gnu) **[…]** Any suggestions are welcome, but I won't promise I'll implement them :-)*

Linus (torvalds@kruuna.helsinki.fi)

*PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it **probably never will support anything other than AT-harddisks**, as that's all I have :-(.*

— Linus Torvalds

Linux vs. GNU/Linux

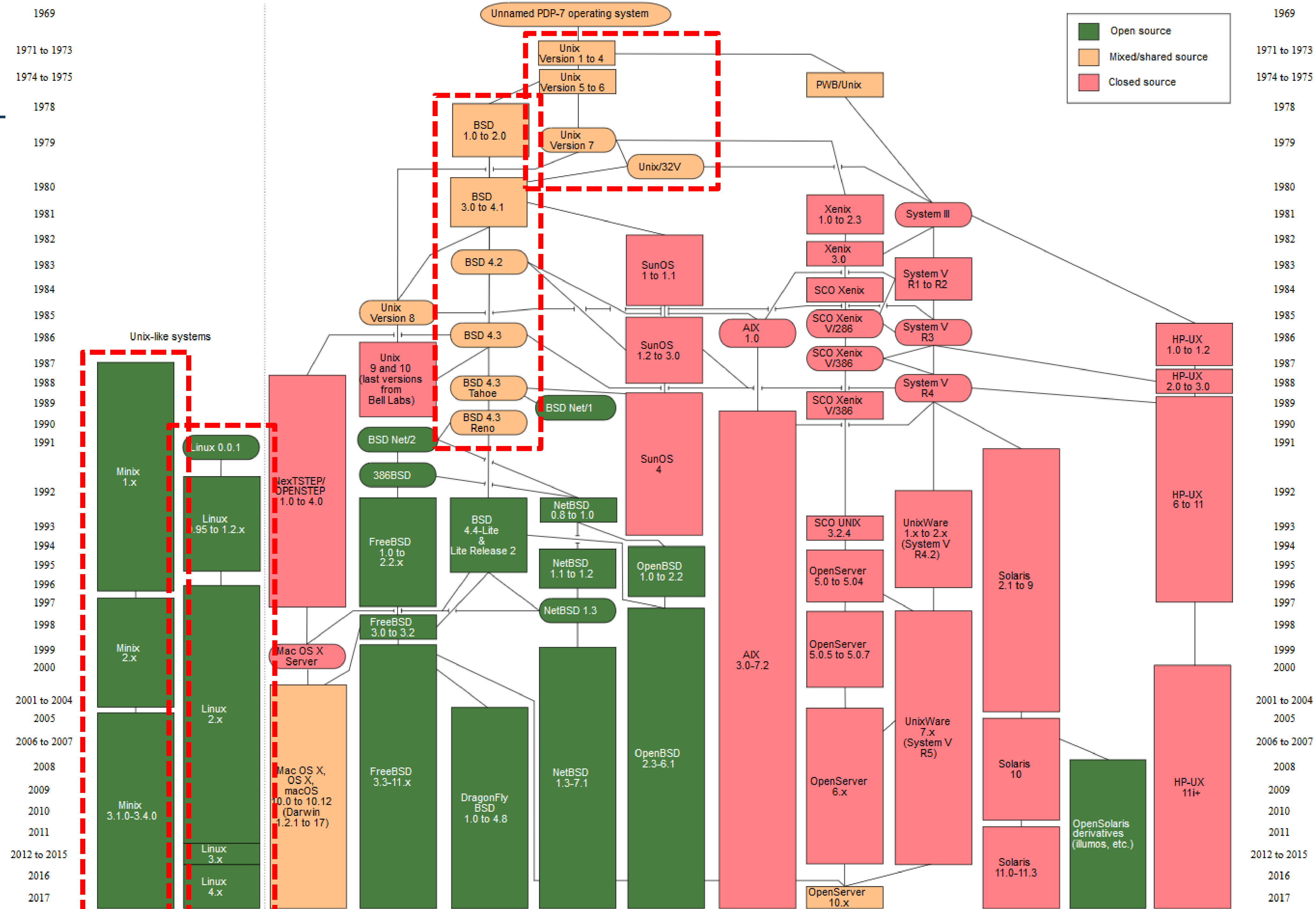
- Linux vs. GNU vs. GNU/Linux?
- Linux → kernel, mediate access to resources such as CPU, RAM, IO (GNU C)
- GNU → tooling, e.g. `bash`, `cat`, `cp`, `mv`, `screen`, `tar`, and many more
- GNU/Linux → Unix-like Operating System
  - Uses Linux kernel and GNU tooling to create a fully functional system

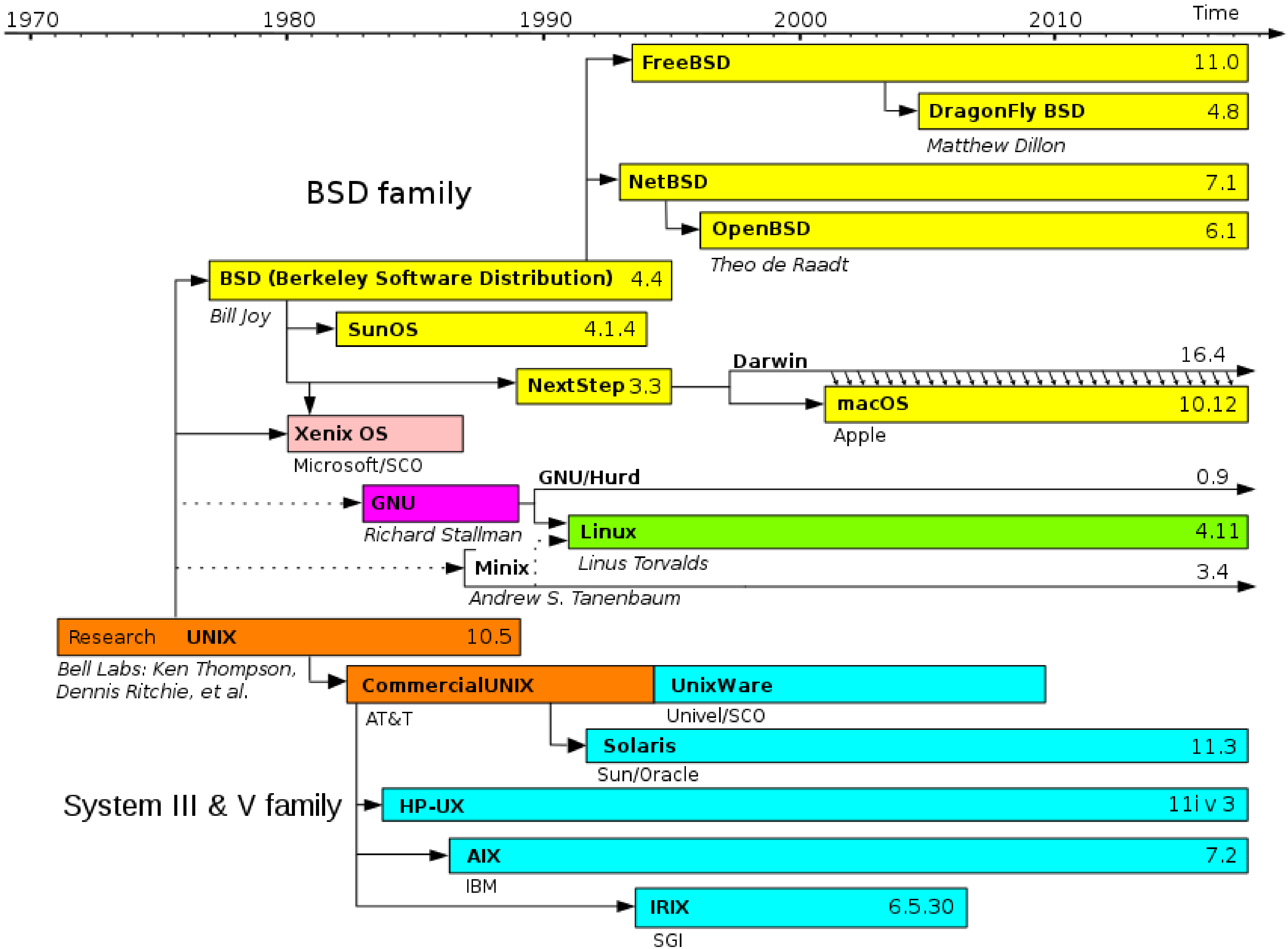
- 1994 – Linux 1.0 → Support for other processor architectures
- 1996 – Linux 2.0 → Symmetric Multiprocessing (SMP)
- 2011 – Linux 3.0 → Drivers and hardware support, 20th anniversary
- 2015 – Linux 4.0 → Live kernel patching, file system improvements
- 2019 – Linux 5.0 → Spectre and Meltdown security patches, hardware support
- 15 Sept 2019 – Linux 5.3 → Latest Stable Kernel

OS Standardization

- POSIX → Portable Operating System Interface
  - Initiative of Richard Stallman
  - Standard Application Programming Interfaces (API's) for Unix-like systems
  - Enables creation of tools, applications and platforms that work on a range of OS's
    - Portable code
  - Compliant → macOS, Solaris …
    - Mostly compliant → Android, FreeBSD, OpenBSD, Linux …

- LSB → Linux Standard Base
  - Extends POSIX, specifically for Linux distributions
  - Standard file system layout
  - Standard common packages (applications)


Open source
Mixed/shared source
Closed source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
SunOS 1 to 1.1
BSD 4.2
System V R1 to R2
SCO Xenix
Unix Version 8
AIX 1.0
SCO Xenix V/286
System V R3
HP-UX 1.0 to 1.2
Unix-like systems
BSD 4.3
SunOS 1.2 to 3.0
SCO Xenix V/386
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
System V R4
HP-UX 2.0 to 3.0
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
BSD Net/2
SunOS 4
NeXTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
HP-UX 6 to 11
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
FreeBSD 1.0 to 2.2.x
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
Minix 2.x
Mac OS X Server
Linux 2.x
AIX 3.0-7.2
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x (System V R5)
Solaris 10
OpenSolaris derivatives (illumos, etc.)
OpenBSD 2.3-6.1
FreeBSD 3.3-11.x
NetBSD 1.3-7.1
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
DragonFly BSD 1.0 to 4.8
OpenServer 6.x
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x


1970
1980
1990
2000
2010
Time
FreeBSD 11.0
DragonFly BSD 4.8
Matthew Dillon
BSD family
NetBSD 7.1
OpenBSD 6.1
Theo de Raadt
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
Darwin 16.4
NextStep 3.3
macOS 10.12
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd 0.9
GNU
Richard Stallman
Linux 4.11
Linus Torvalds
Minix 3.4
Andrew S. Tanenbaum
Research UNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
CommercialUNIX
AT&T
UnixWare
Univel/SCO
Solaris 11.3
Sun/Oracle
System III & V family
HP-UX 11i v3
AIX 7.2
IBM
IRIX 6.5.30
SGI

Tanenbaum-Torvalds debate (MINIX vs Linux)

Andrew S. Tanenbaum

Linus Torvalds

Tanenbaum-Torvalds debate (MINIX vs Linux)

- Torvalds is inspired by Tanenbaum's MINIX
  - Tanenbaum's book, Operating Systems: Design and Implementation
- In 1992, Tanenbaum begins a Usenet discussion in `comp.os.minix`
  - Argues micro kernels (MINIX) are superior to monolithic kernels (Linux)
  - Poor portability, too closely tied to Intel's (80)386
  - Linux not suitable for students
- Torvalds replies
  - MINIX has design flaws (no multithreading)
  - Linux is free and developed in spare time
- Controversy about Linux copying MINIX source code
  - Proven false

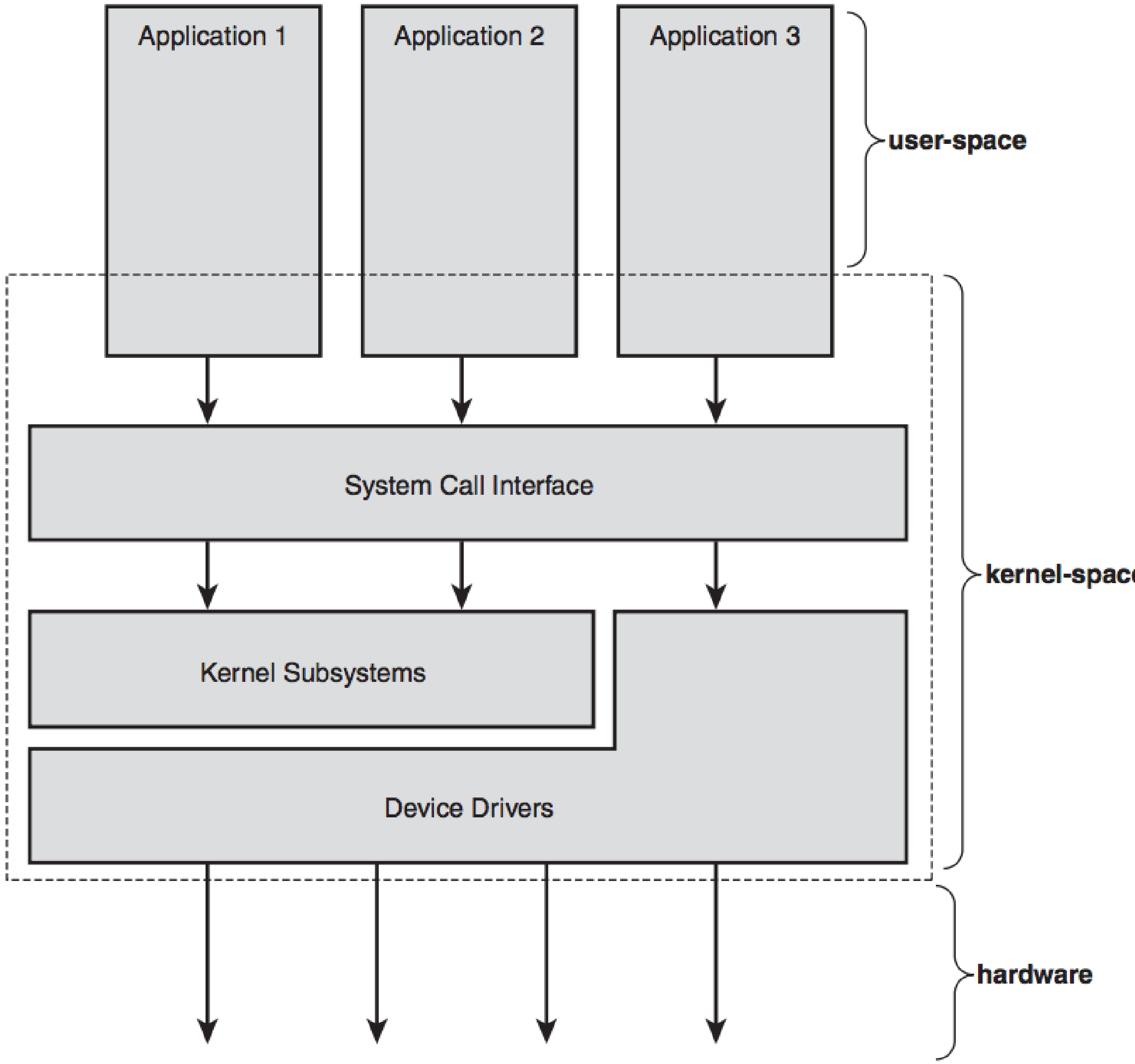
Linux Kernel

- Kernel includes
  - Interrupt handler → service interrupt requests
  - Process scheduler → share processor time among processes
  - Memory manager → process address spaces
  - System services → networking, inter-process communication
- Kernel executed in elevated system state called **kernel-space**
  - Protected memory space
  - Full access to hardware
  - Kernel threads
- User applications executed in **user-space**

User-space Kernel-space

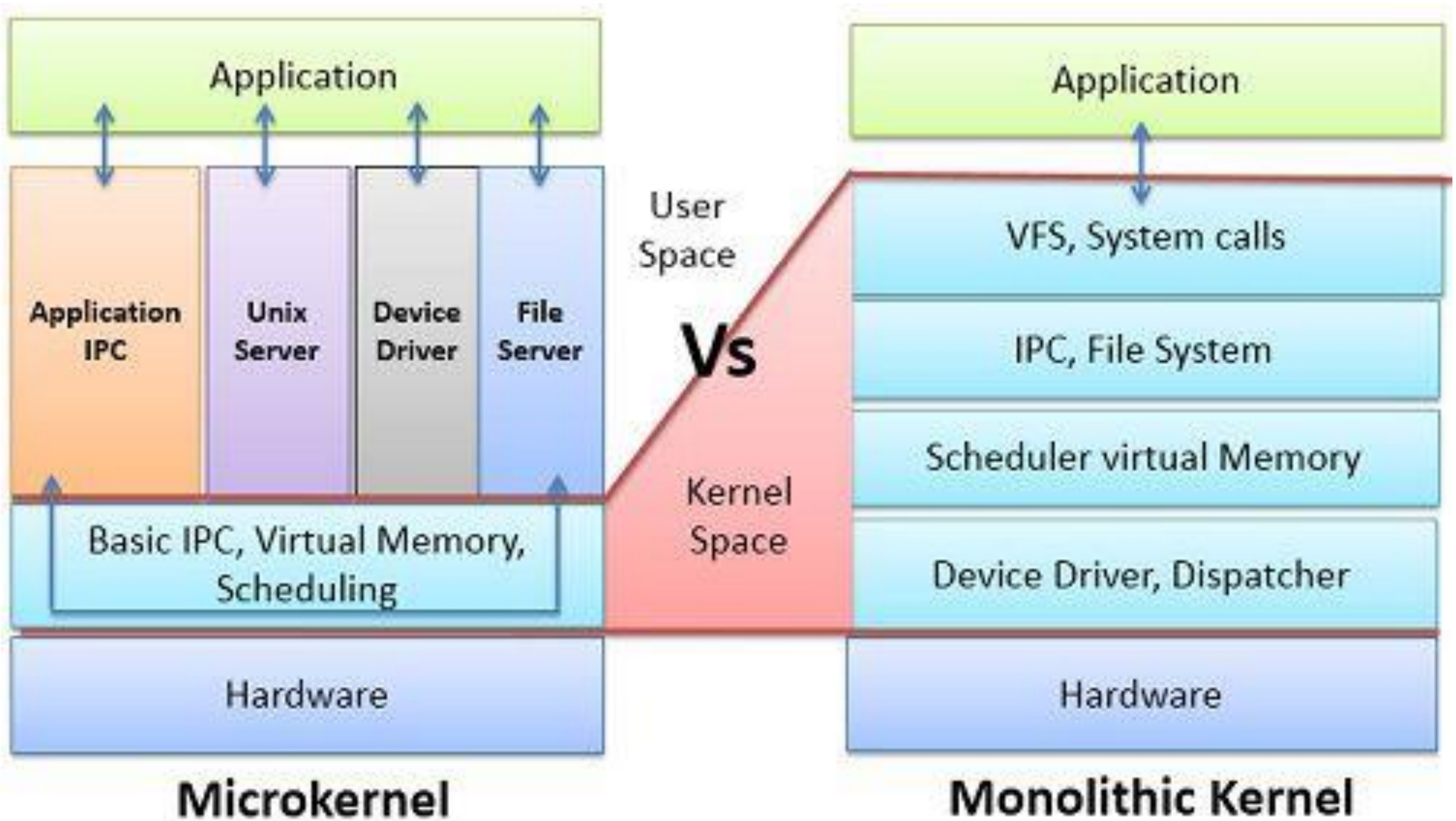
- Applications communicate with kernel via system calls
  - E.g. `accept()`, `chmod()`, `exit()`, `fork()`, `ioctl()`, `recv()`, `write()`
- Kernel communicates with hardware via interrupts
  - Hardware wants to interact → sends interrupt to processor → caught by kernel
  - For synchronization kernel can disable interrupts (all/one)
  - Run in interrupt context, not process context
- In Linux kernels, a processor does one of the following:
  - Execute user code in a process (user-space)
  - Execute system calls on behalf of a process (kernel-space)
  - Handle interrupts not associated with a process (kernel-space)

User Space vs. Kernel Space



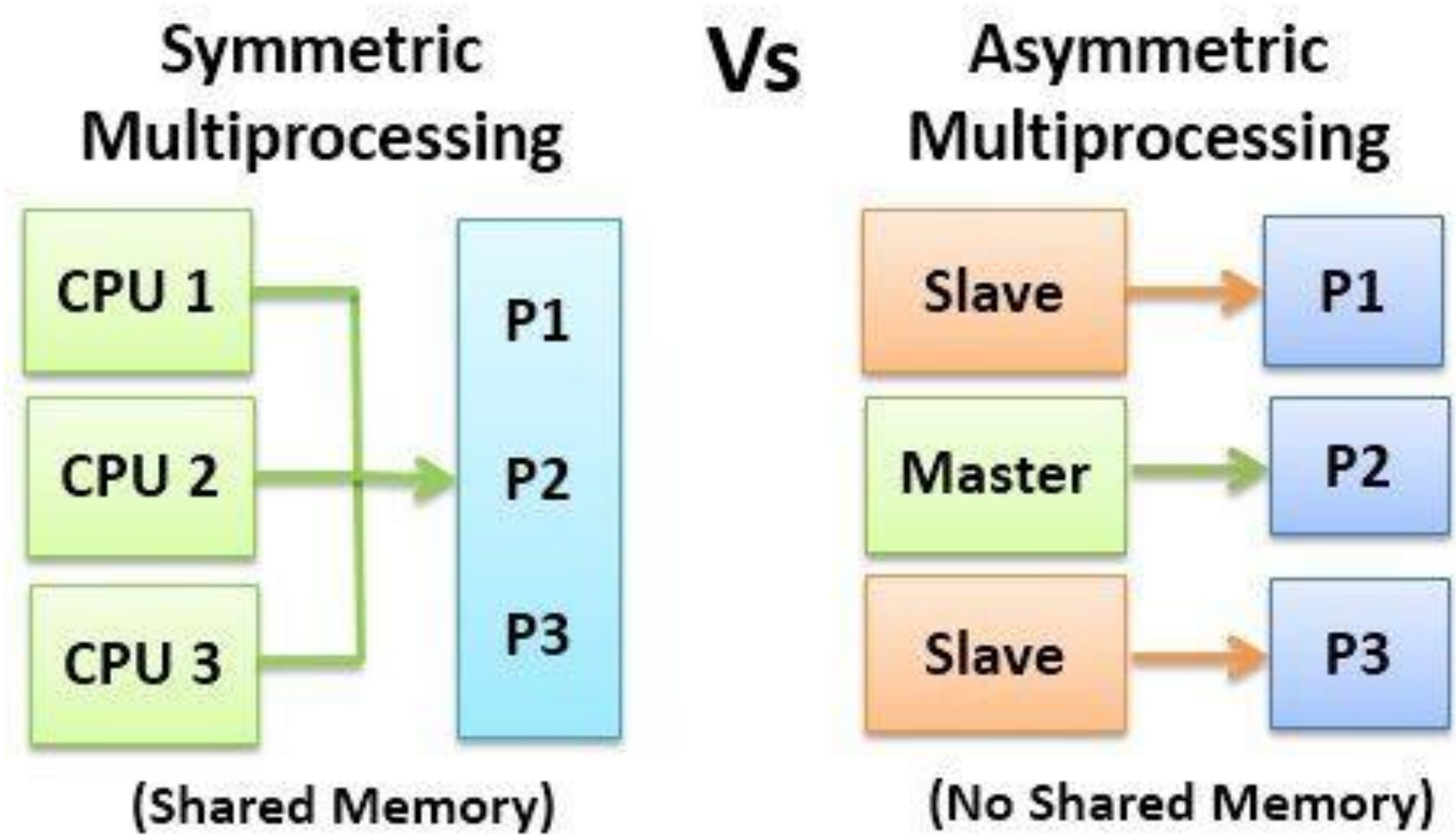
Linux Kernel characteristics

- Monolithic (as opposed to micro-kernel) → module support



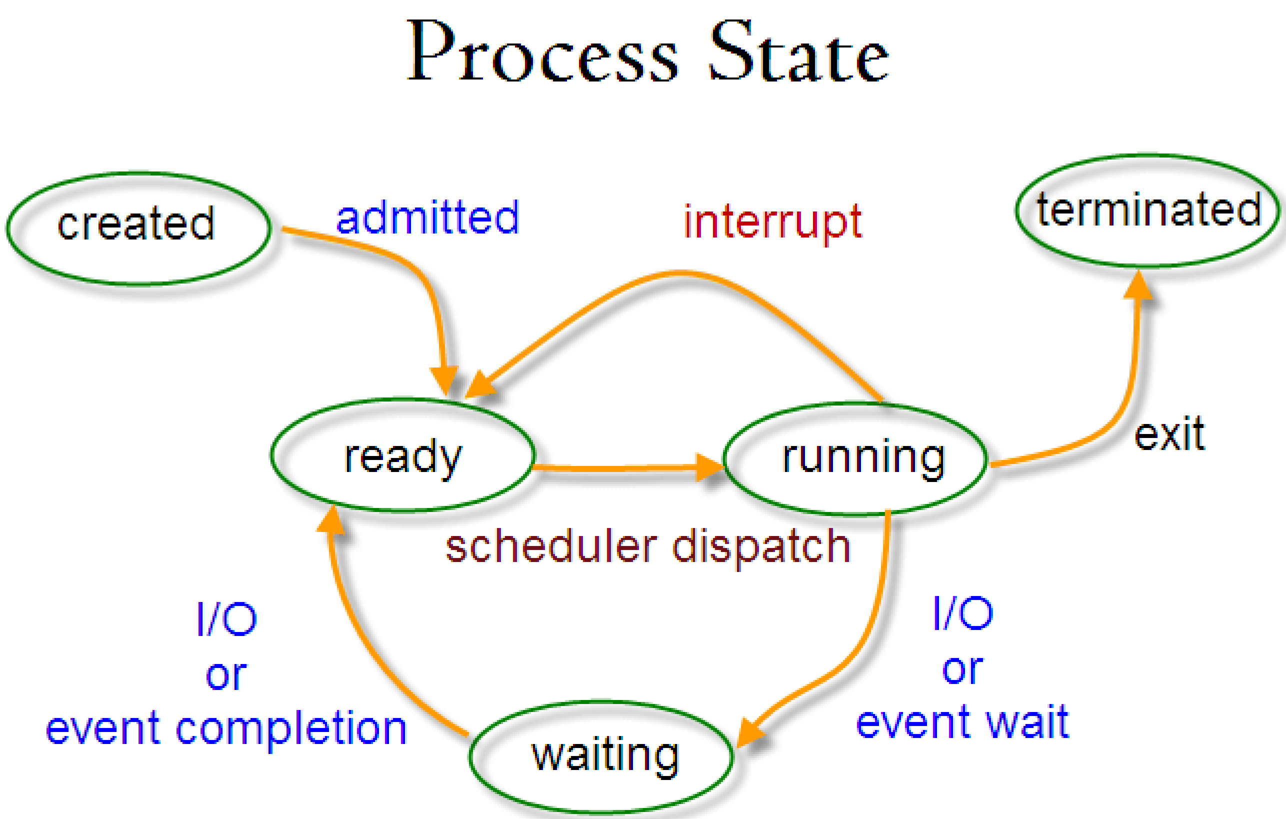
Linux Kernel characteristics

- Monolithic (as opposed to micro-kernel) → module support
- Symmetric Multiprocessor (SMP) support



Linux Kernel characteristics

- Monolithic (as opposed to micro-kernel) → module support
- Symmetric Multiprocessor (SMP) support
- No differentiation between threads and normal processes, both are tasks



Filesystem Hierarchy Standard (FHS)

- / → root of the file system
- /bin/ → Essential binaries (required to boot / rescue) for all users (e.g. bash)
- /boot/ → boot loader, kernels, initrd
- /dev/ → devices files
- /etc/ → system-wide configuration files
- /home/ → user home directories
- /lib/ → libraries for binaries in /bin/ and /sbin/
- /mnt/ → mounted file systems
- /media/ → removable media
- /opt/{bin,include,lib,sbin,share} → application software packages (non-packaged)
- /proc/ → kernel and process status files

Filesystem Hierarchy Standard (FHS)

- /tmp/ → temporary files
- /sbin/ → Essential system binaries for system administrators (e.g. iptables)
- /sys/ → exporting kernel objects
- /usr/{bin,include,lib,sbin,share,src,X11R6,local,local/bin,local/src} →
  - shareable read-only data (not for boot / rescue), e.g. `man` files
  - <u>U</u>NIX <u>s</u>ystem <u>r</u>esources
- /var/ → variable data (logs: /var/log)
- **Non-official**
  - /lost+found/ → recovered files / fragments
  - /selinux/ → SE-Linux settings
  - /srv/ → data served by the system
- https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf

Lectures

Linux Boot Process

- 6 Stages, BIOS-based example

1. BIOS (Basic Input Output System)
   - Tests hardware during POST (Power-On Self Test)
   - Looks for bootloader in bootable media, loads it into memory, and passes control
2. MBR (Master Boot Record)
   - First sector on bootable media (512 bytes)
   - Loads and executes GRUB
3. GRUB (Grand Unified Bootloader)
   - Read config: /boot/grub/grub.conf (BIOS), UEFI slightly different
   - Loads and executes kernel (vmlinux/vmlinuz) and initrd images

Linux Boot Process

4. Kernel
   - Kernel mounts the temporary root file system from initrd image
   - Initrd is used until real root file system is mounted, also contains drivers
     - Newer versions of Linux: initramfs
   - Configure RAM and hardware, start process /sbin/init

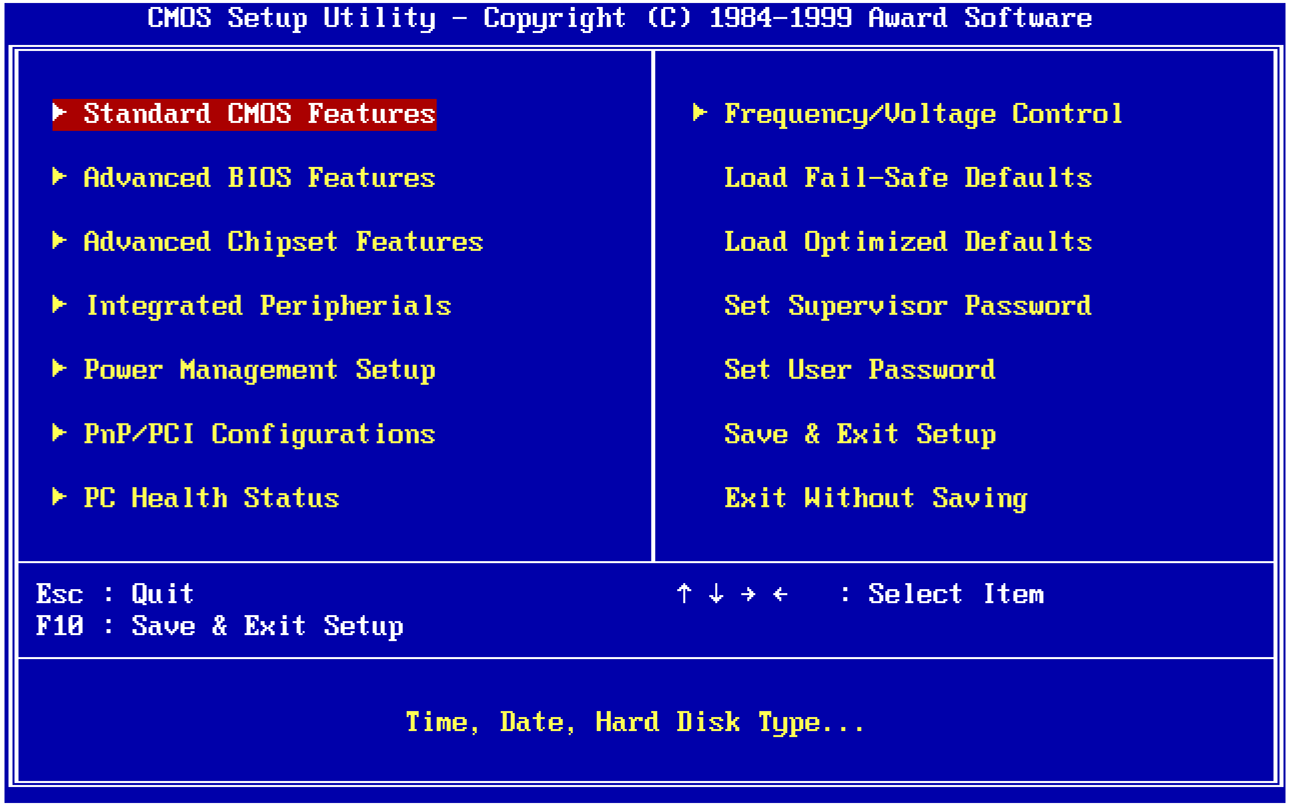
5. Init process
   - Locate and mount root file system
   - Run initialisation scripts (located in /etc/init.d), based on runlevel
   - Debian: First, run level S → run level 2

Linux Boot Process

6. Runlevel applications
   - Start services and applications depending on run level

- Levels in Debian
  - 0 - Halt
  - S - Single-user on boot
  - 1 - Single-user
  - 2,3,4,5 - Multi-user
  - 6 – Reboot
  - 7,8,9 - Unused
- Scripts in /etc/rc<run level>.d/ are executed
  - Symlinks to /etc/init.d/

BIOS


CMOS Setup Utility - Copyright (C) 1984-1999 Award Software
Standard CMOS Features
Advanced BIOS Features
Advanced Chipset Features
Integrated Peripherials
Power Management Setup
PnP/PCI Configurations
PC Health Status
Frequency/Voltage Control
Load Fail-Safe Defaults
Load Optimized Defaults
Set Supervisor Password
Set User Password
Save & Exit Setup
Exit Without Saving
Esc : Quit
F10 : Save & Exit Setup
↑ ↓ → ← : Select Item
Time, Date, Hard Disk Type...

BIOS vs. UEFI

- UEFI
  - Unified Extensible Firmware Interface
  - Faster (fast boot) → caches files for faster start up
  - Allows larger hard drives
    - MBR vs. GUID Partition Table (GPT), MBR supports up to 2.1 TB
  - Processor and driver independent → ARM
  - 32bit and 64bit mode → addressable space
  - Secure boot → malware

- Once booted → GUI Login or Login Shell

UEFI

Intel® Visual BIOS
intel EXTREME BOARD inside
Home
Main
Devices & Peripherals
Cooling
Performance
Security
Power
Boot
Load Defaults
Exit
Processor Information
Processor Type Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
L2 Cache RAM 4 x 256 KB
L3 Cache RAM 8 MB
Processor Signature 000206A7
Intel® Management Engine Firmware 8.0.1.1399
Memory Information
L2 Cache RAM 4 x 256 KB
Total Memory 8 GB
Total Memory TBD
DIMM 3 (Memory Channel A Slot 0) 4 GB
DIMM 1 (Memory Channel A Slot 1) Not Installed
DIMM 4 (Memory Channel B Slot 0) 4 GB
DIMM 2 (Memory Channel B Slot 1) Not Installed
System Date & Time
Wednesday
11
Apr. 2012
09:14:23PM
Date: 04 / 11 / 2012
Time: 09 : 13 : 18 PM
Set
Event Logging
Clear Event Log No
Event Logging Enable
ECC Event Logging Enable
Default BIOS Start Page
Advanced - Main
Here you can look at your basic system information, set the BIOS time, select what page you'd like to see when Visual BIOS is opened, and view important system events.
Tab - Next option
Enter - Accept change
Alt - Reveal shortcut keys
Esc - Discard/exit
F9 - Load defaults
F10 - Save and exit
Search
0.9.4700

Terminals, Emulators, and Shells

- Terminal
- Virtual Terminal
- Pseudo Terminal
- Shell

Terminal

- Text input/output environment
- Teletypewriter (TTY)
- Hardware-based
- 1960-1980

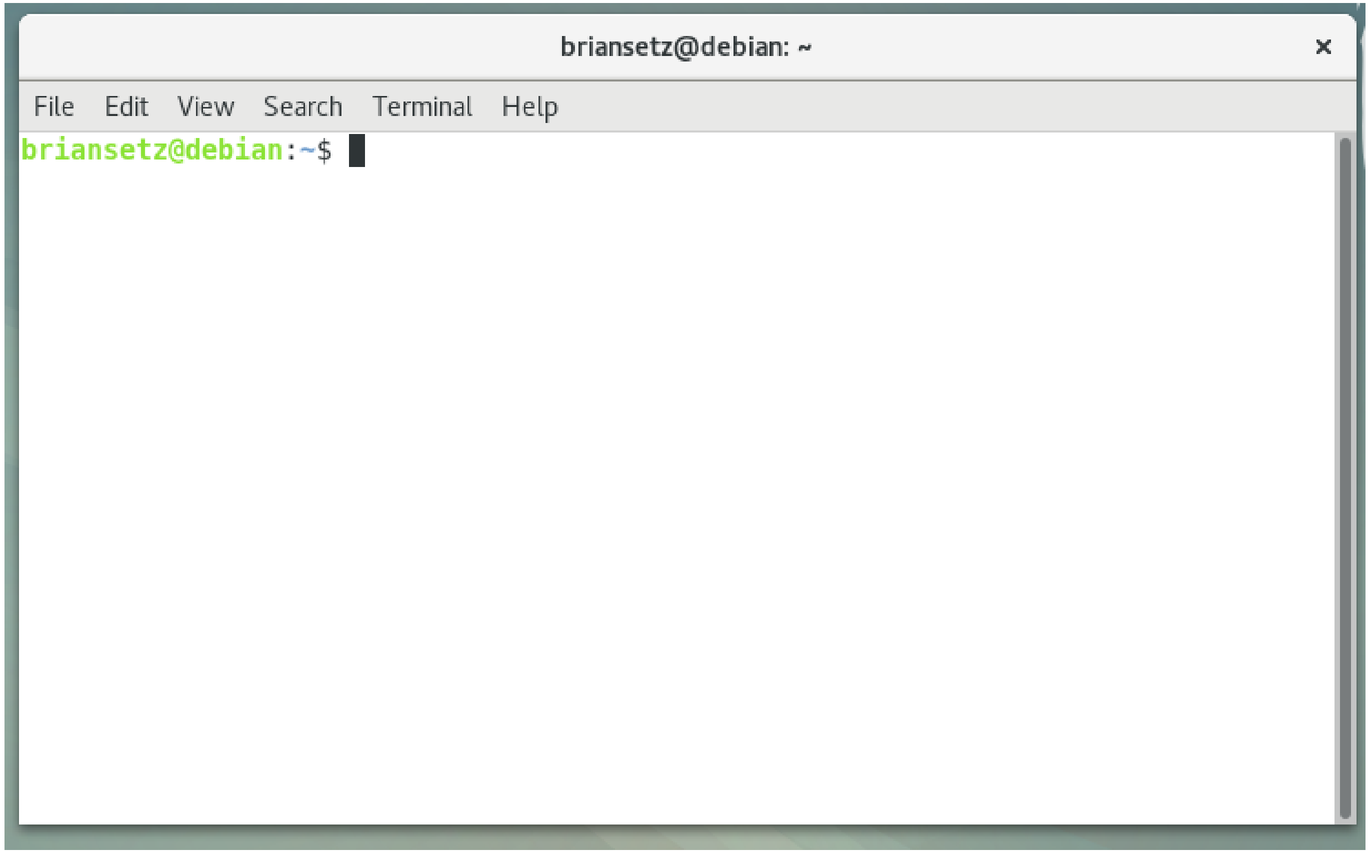
Virtual Terminal

- Emulates a hardware terminal
  - CTRL+ALT+F3

- Device files /dev/tty

Psuedo Terminal

- Terminal Emulator
  - GNOME Terminal
  - `screen`
  - `ssh`
- Device files /dev/pts/



Shell

- Command-line interpreter

- Default login shell: `bash`

- Alternatives, different features
  - `sh`
  - `zsh`
  - `csh`
  - `ksh`
  - …

SH

- Bourne Shell (sh)
  - Developed by Stephen Bourne (1977)

```
$ 
```

Bash

- Bourne Again Shell (bash)
  - Developed by Brian Fox (1989)

```
briansetz@debian:~$ 
```

Shell Scripts

- Shell scripts are interpreted by the shell
  - Different shells offer different functionality
- How to know which shell should execute a shell script?
- Shebang
  - `#!/bin/sh` → this script will be interpreted by `sh`
  - `#!/bin/bash` → this script will be interpreted by `bash`

(bash) Shell Commands

- `man` (manual) command
  - Describes how to use commands → documentation
  - How to use `man`? `man man`
- Pipes and redirects
  - Pipe `x | y` → (standard) output of x is used as (standard) input for y
    - `ps aux | grep sh`
  - Redirect `x [1/2/&]> [file descriptor]` → redirect output of x to file descriptor (1 = std. out, 2 = std. err, & = both)
    - `cat /var/log/syslog > /home/briansetz/outlog`
    - `>` → truncate, `>>` → append
- Execute in background → `x &`
- Multiple commands → `x; y; z` or `x && y && z`

Linux Distributions

- List of all GNU/Linux distributions
  - https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

- Majority (80-90%) of all distributions are based on / derived from:
  - Debian (1993)
  - Slackware (1993)
  - RedHat (1995)

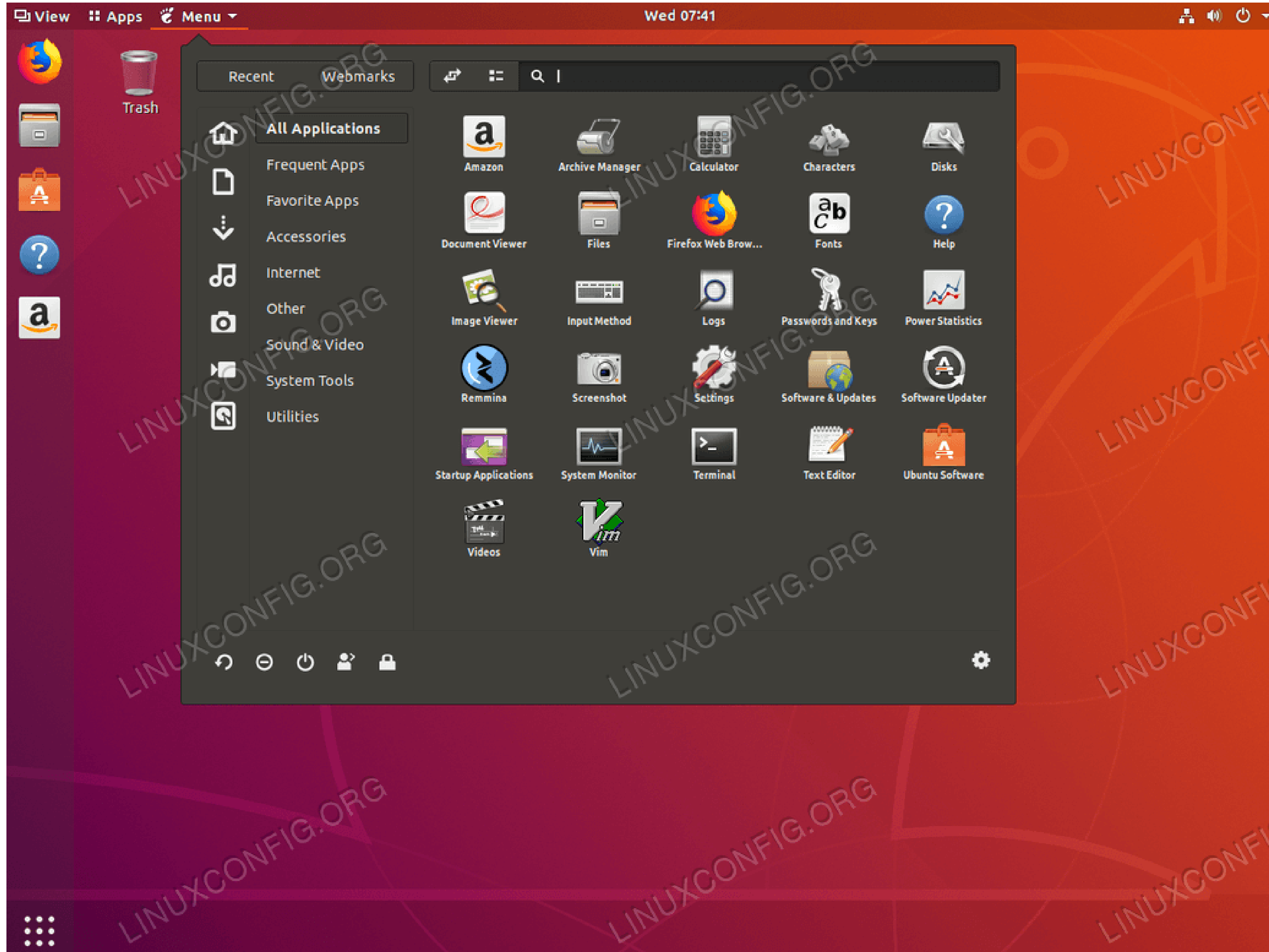
- Some examples…

Linux Distributions

- Debian

Linux Distributions

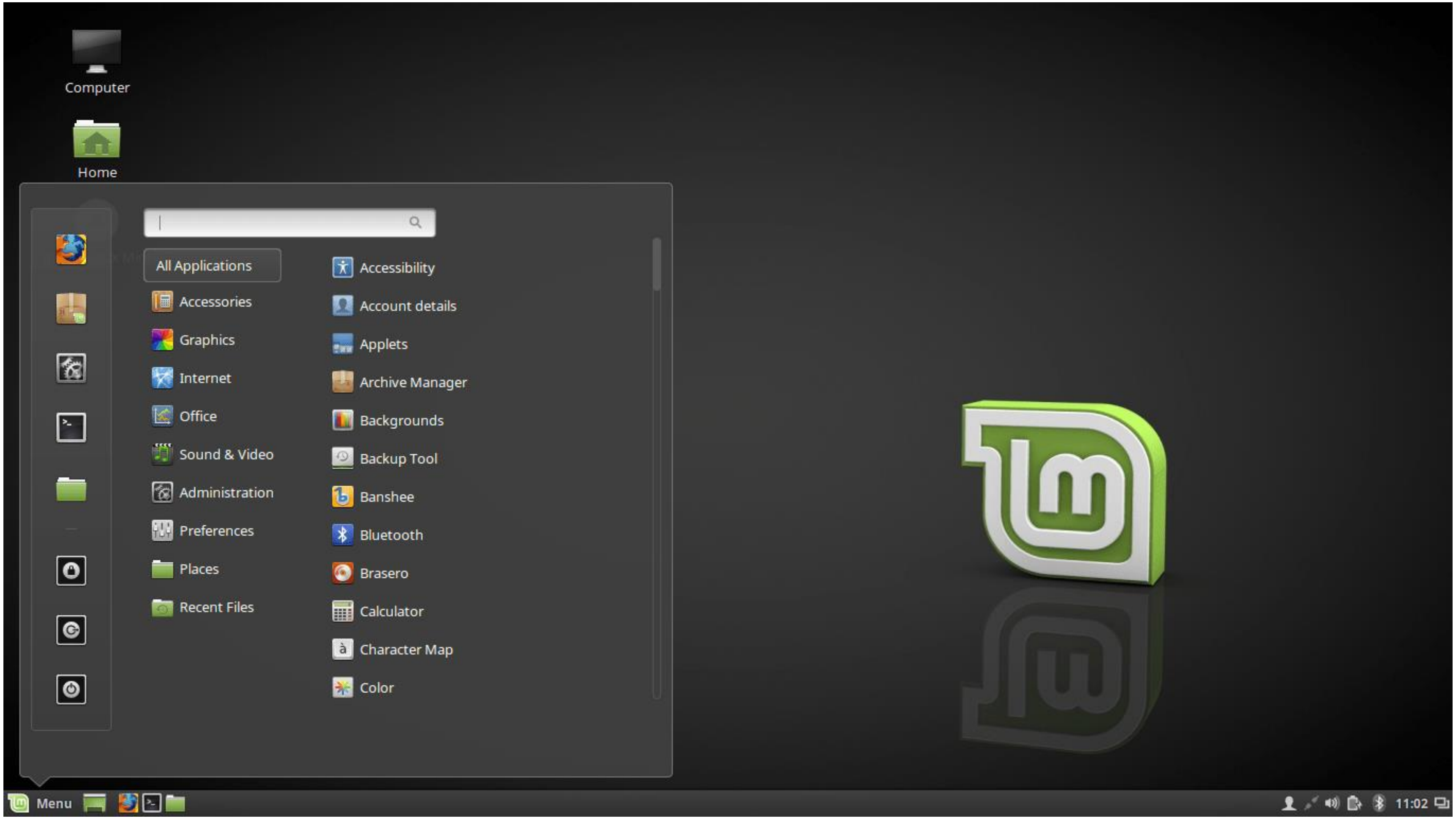
- Ubuntu



Lectures

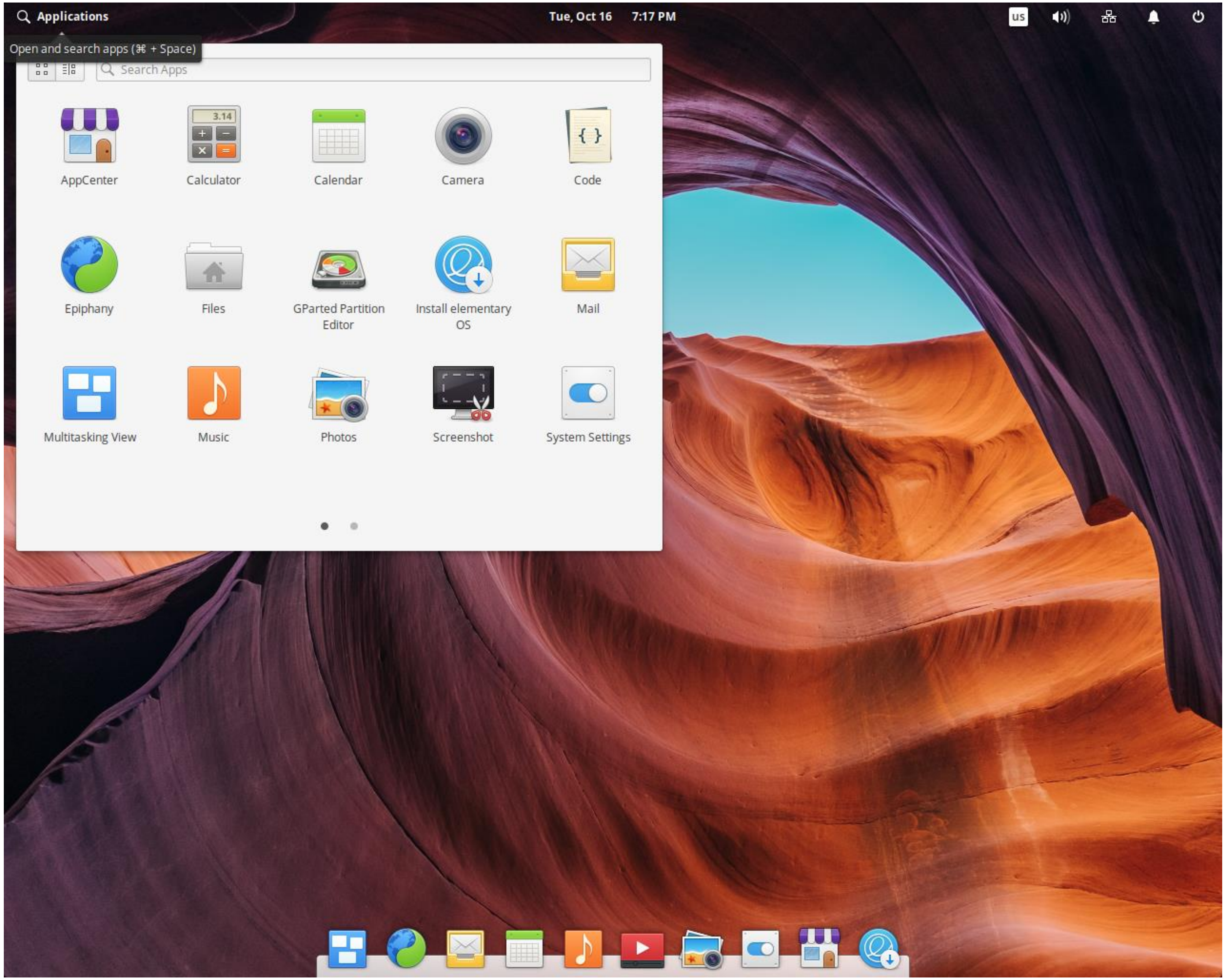
Linux Distributions

- Linux Mint



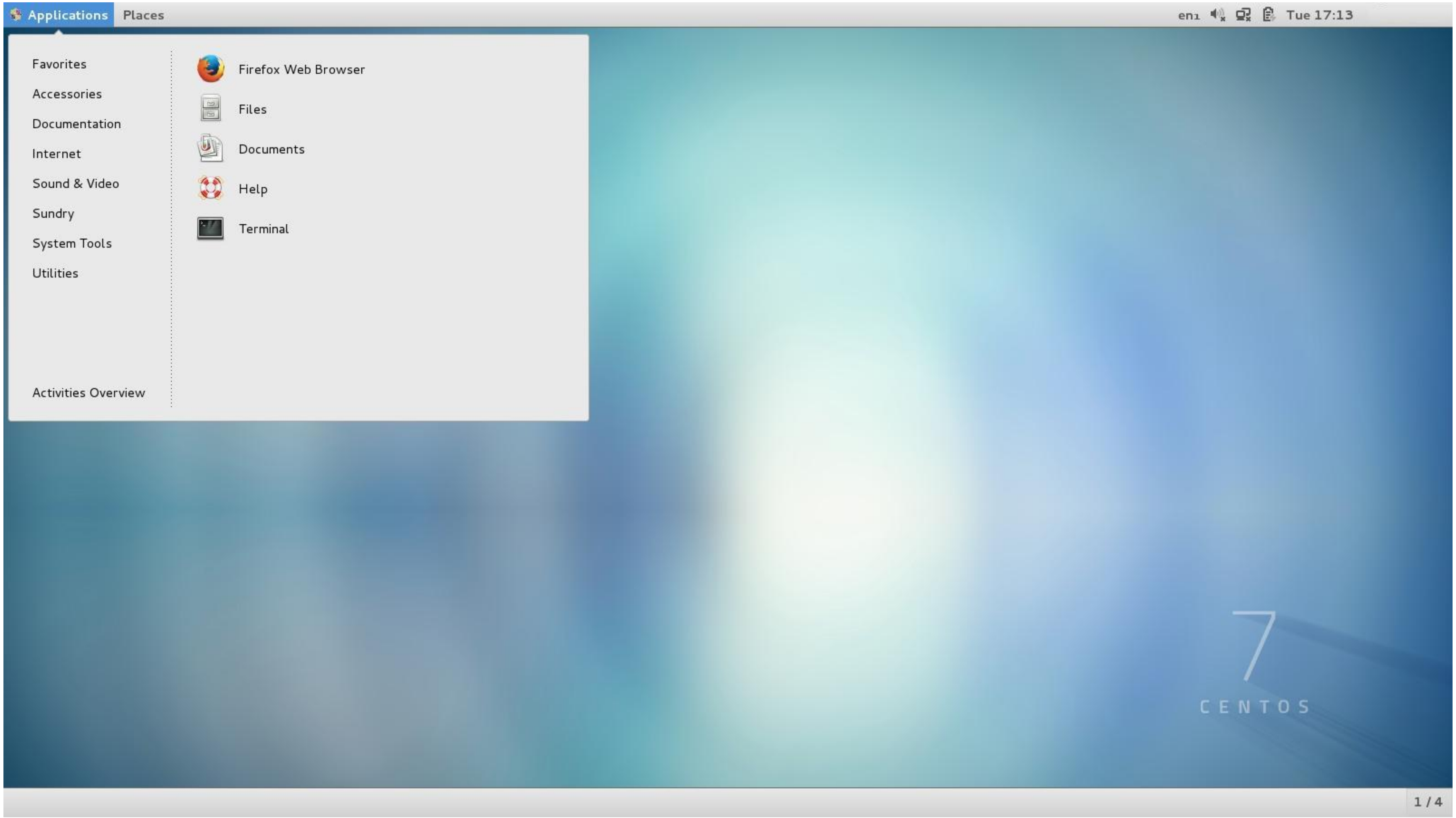
Linux Distributions

- Elementary OS



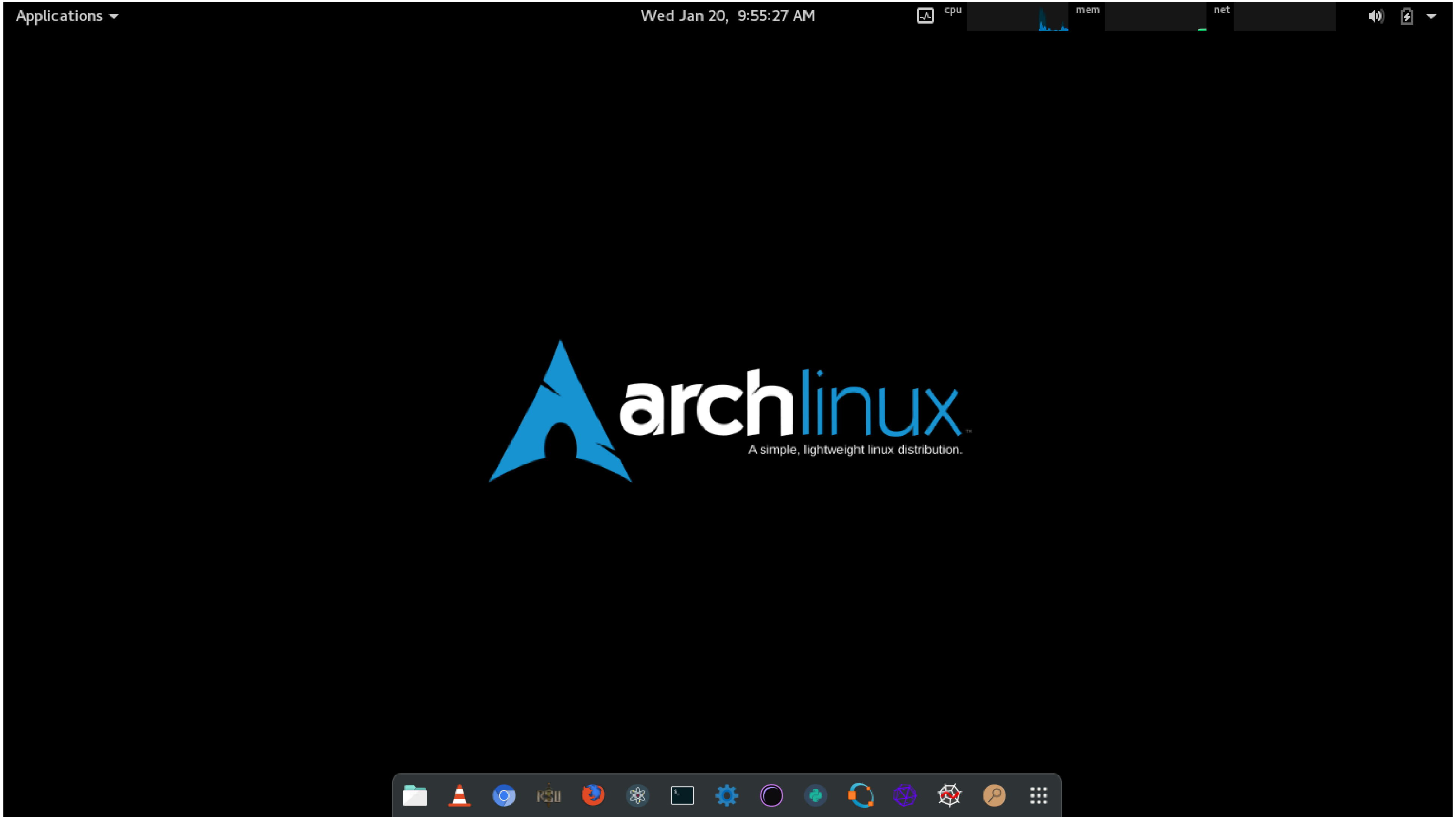
Linux Distributions

- CentOS



Linux Distributions

- Arch



Lectures

Exercises Environment

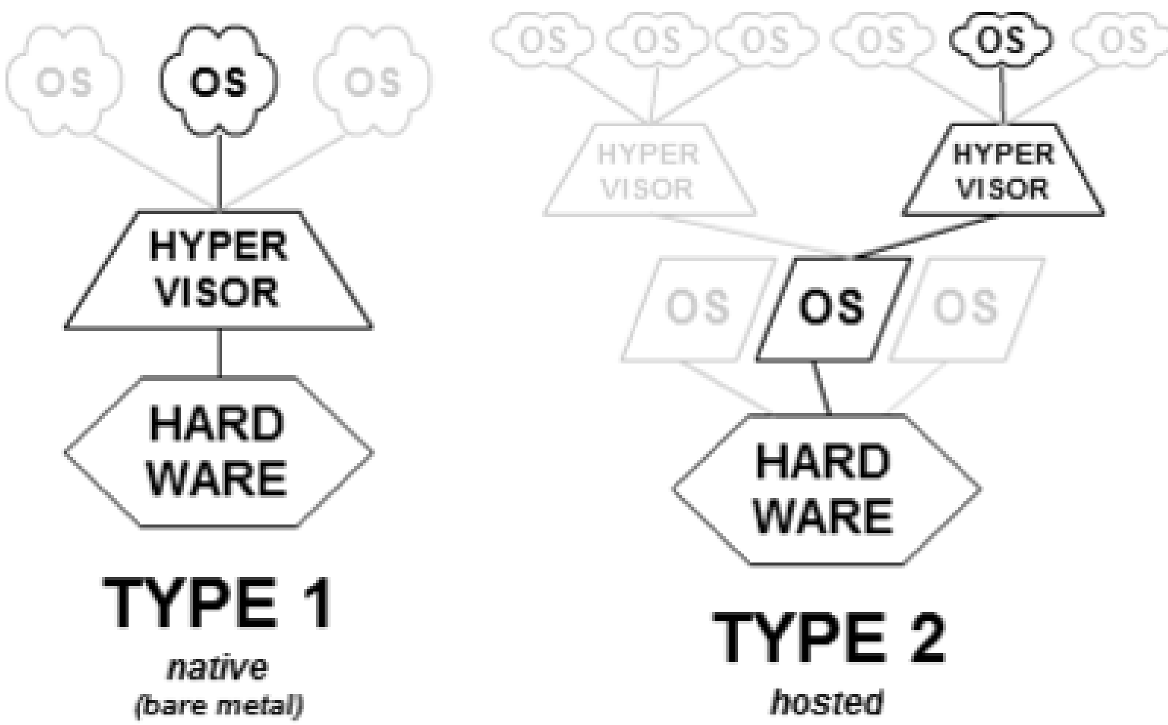
Debian

- Unix-like operating system
- Supports multiple kernels: Linux, BSD-based, GNU Hurd
- GNU Tooling (`coreutils`)

- Download: https://cdimage.debian.org/mirror/cdimage/archive/9.5.0/i386/iso-dvd/
  - Only download `debian-9.5.0-i386-DVD-1.iso`
  - During labs only Debian 9.5.0 i386 is supported

Virtualbox

- VirtualBox → Hypervisor, creates and runs virtual machines
  - Type 2 hypervisor
- Download: https://www.virtualbox.org/wiki/Downloads
  - Enable VT-d, optionally VT-x also (BIOS)
  - Disable Hyper-V (MS Window's type 1 hypervisor)



Steps (1)

1. Install VirtualBox
2. Create new Virtual Machine
    1. Type: Debian 32bit
    2. Memory: 1GB+
    3. Create a virtual hard disk now → VDI → Dynamically Allocated: 32GB
3. Change Virtual Machine Settings
    1. System → Processor → Increase processors to 50%
    2. Display → Video Memory: Max → Enable 3D Acceleration (optional)
    3. Storage → CD → Choose Virtual Optical Disk File → Select Debian ISO
4. Start the Virtual Machine

Steps (2)

5. Install Debian, important settings:
   - <u>Use your full name as user name</u>
   - Partition disks: **Guided – use entire disk and set up LVM**
   - Partitions disks: **Separate /home, /var, and /tmp partitions**
   - Configure the package manager: **Scan another CD or DVD? → NO**
   - Configure the package manager: **Use a network mirror? → YES**
   - Software selection: **GNOME, print server SSH server, standard system utilities**
   - Install the GRUB boot loader to the master boot record? → YES

Steps (3)

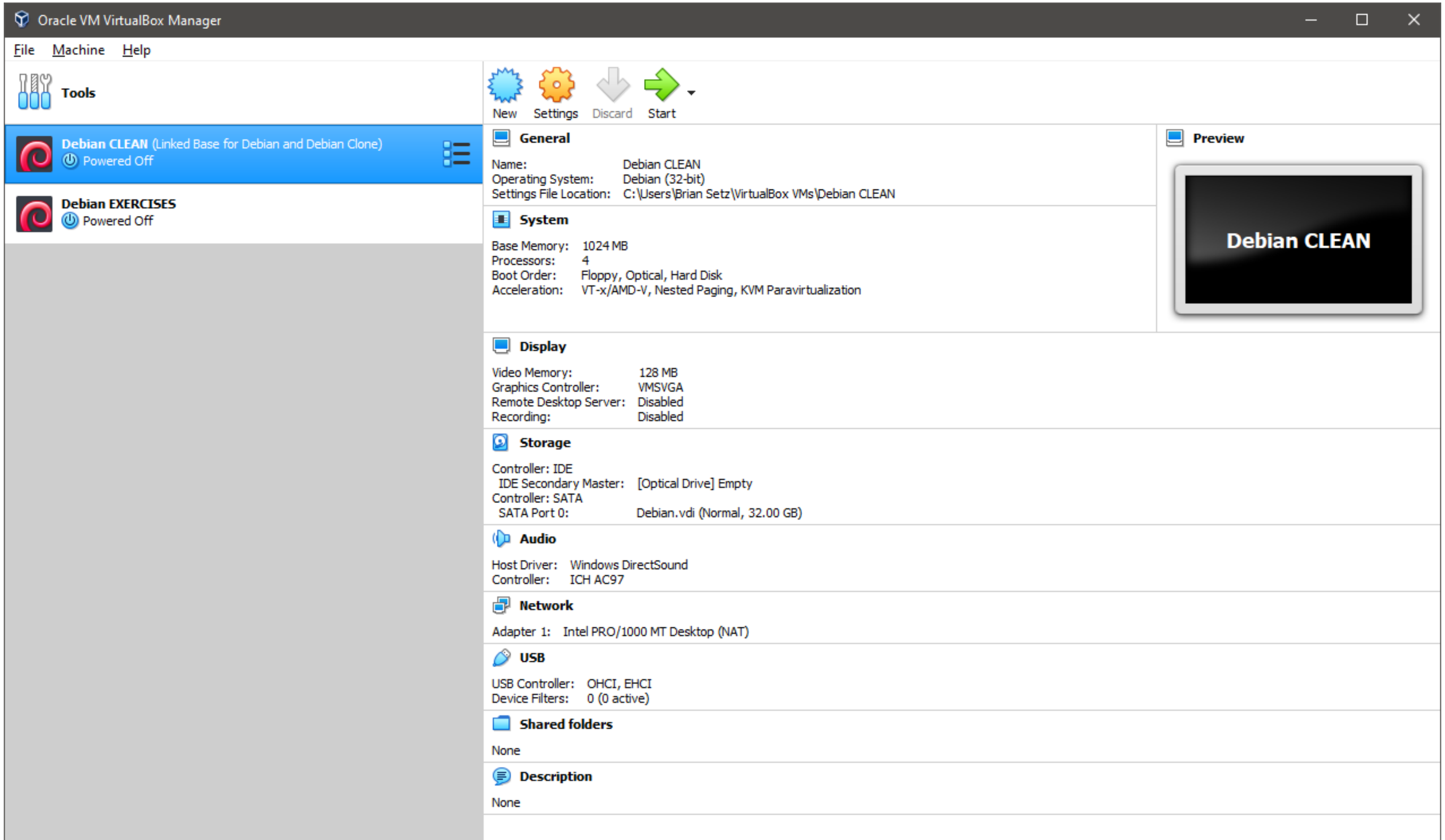
6. Update Debian
   - Open Terminal
   - Switch to root user: `su`
   - Edit sources.list: `nano /etc/apt/sources.list` and put a #-sign in front of the line starting with `deb cdrom`
   - Update the system: `apt update` followed by `apt upgrade`
   - Reboot
   - Open Terminal, switch to root user, and run: `uname -r` the output should be version `4.9.0-11-686` or greater
7. Install dependencies
   - As root, run: `m-a prepare` and install the dependencies

Steps (4)

8. Install the VirtualBox guest additions
   - In VirtualBox, click on Devices → Insert Guest Additions CD image
   - In Debian, click cancel and open the File Manager
   - Click on the CD, and right click in the window that opened and select Open Terminal
   - As root, run: `sh VBoxLinuxAdditions.run`
   - Wait for the process to finish
9. Configure VirtualBox
   - In VirtualBox click Devices → Shared Clipboard → Bidirectional
   - And, Devices → Drag and Drop → Bidirectional
   - Shutdown Debian
10. **In VirtualBox, right click the VM and select Clone → Linked Clone!**
    - **Rename original to: Debian CLEAN, rename clone to: Debian EXERCISES**

Video Instructions

- Video instructions are available on YouTube: https://youtu.be/S3D_2ytNqEs
  - Short version, actual installation takes 30-45 minutes depending on hardware



Exercises

Exercises

Always include a full description (e.g. input / output) of the performed tasks, in addition to the answers. Submit a short, coherent report as PDF.

1. Install VirtualBox, setup Debian within VirtualBox, **(a)** use the `lsb_release` command to print <u>all</u> the distribution details, and **(b)** use the `uname` command to print the kernel *release* version. Read the `man` pages of these commands.

2. Determine **(a)** the shell that is used by default by using the `echo` command and the `$SHELL` variable, and **(b)** list the directories found in the `$PATH` variable.

3. List the number of scripts that run **(a)** at run level S, **(b)** run level 2, and **(c)** run level 5. Use the `ls, wc` and pipe commands. Hint: check the man pages to list each file on a new line and 'pipe' the results from one command to the other.

4. Research the difference between systemd and init, **(a)** describe in your own words the difference between these systems, and **(b)** determine which of the two is used by the operating system you have installed in VirtualBox. How can you tell?

Lectures