Estonian Information Technology College

Operating system booting

Operating systems I800

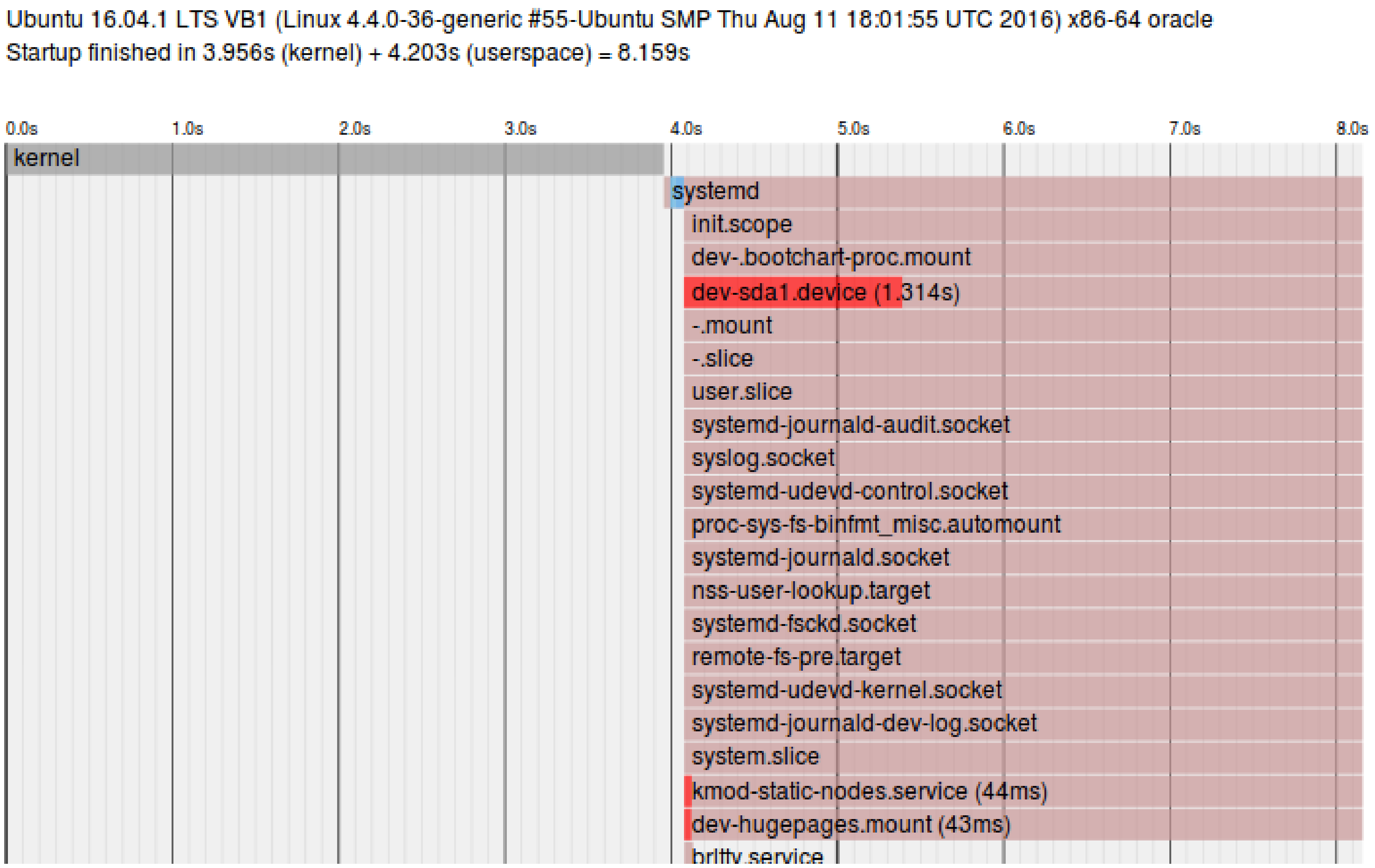
Edmund Laugasson
edmund.laugasson@itcollege.ee

There has been used materials from Margus Ernits, Katrin Loodus when creating current slides.

Ubuntu booting via *systemd*

- *systemd-analyze* – analysis of boot performance
  - *man systemd-analyze*
  - *time* – spent for boot process
  - *blame* – start the service time (q to exit)
  - *critical-chain* – displays a time-critical service tree
  - *plot* – SVG graphic image from the entire boot process
  - *dump* – comprehensive human readable overview of the system status
- image creation command: *systemd-analyze plot > boot.svg*

systemd-analyze plot > boot.svg



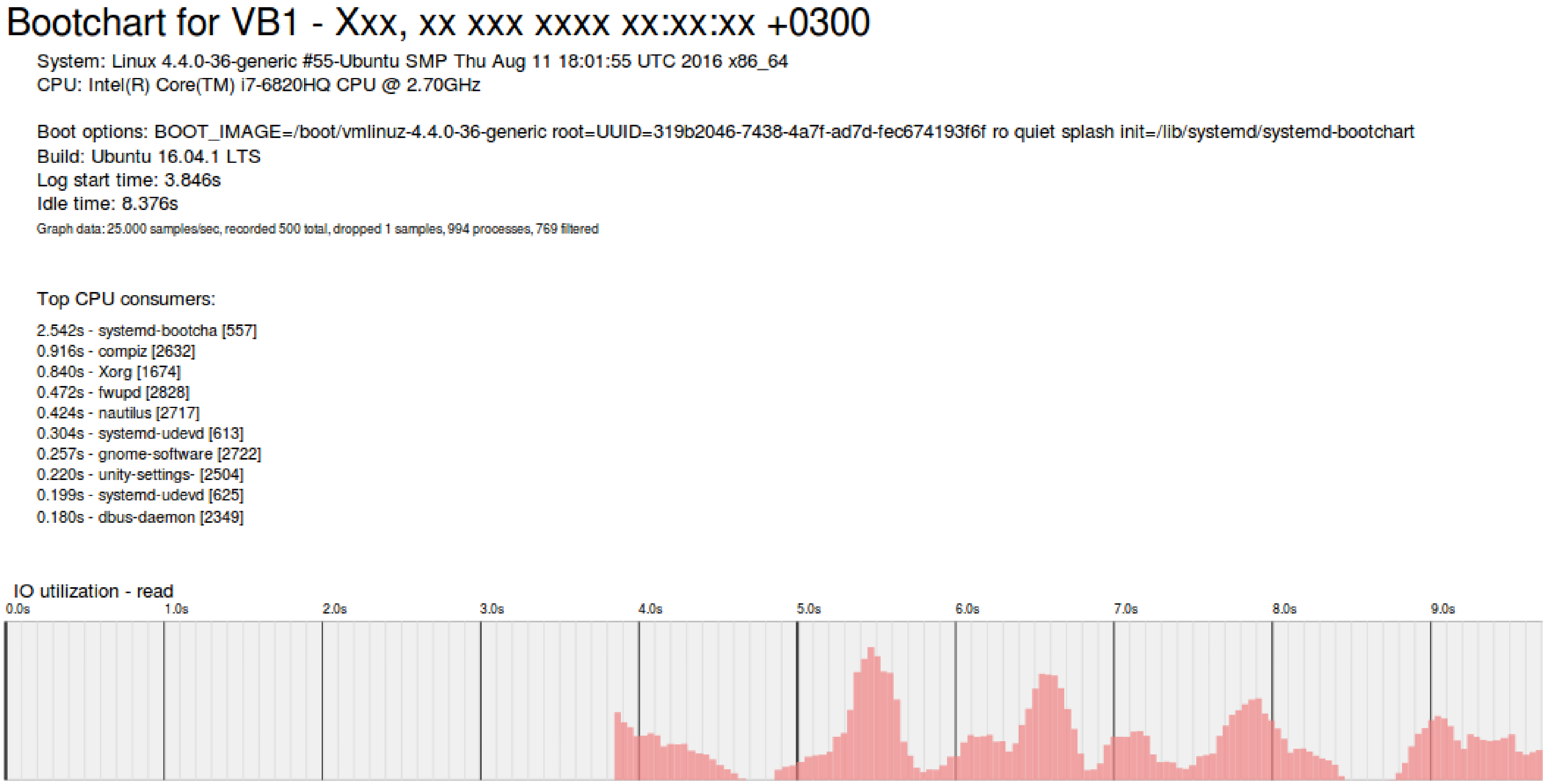
Ubuntu 16.04.1 LTS VB1 (Linux 4.4.0-36-generic #55-Ubuntu SMP Thu Aug 11 18:01:55 UTC 2016) x86-64 oracle
Startup finished in 3.956s (kernel) + 4.203s (userspace) = 8.159s

image in whole size (.svg)

Bootchart

- since 15.04 the Ubuntu is using *systemd* https://wiki.ubuntu.com/SystemdForUpstartUsers
- *sudo nano /etc/default/grub*
  - `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash init=/lib/systemd/systemd-bootchart"`
- sudo update-grub
- *sudo nano /etc/systemd/bootchart.conf* (nano text editor: F3 save, F2 exit)
  - `[Bootchart]`
  - `Samples=500`
  - `Frequency=25`
  - `Relative=no`
  - `Filter=no`
  - `#Output=<folder name, defaults to /run/log>`
  - `#Init=/path/to/init-binary`
  - `PlotMemoryUsage=no`
  - `PlotEntropyGraph=no`
  - `ScaleX=100`
  - `ScaleY=20`
  - `ControlGroup=yes`
  - `PerCPU=no`
- *sudo reboot*

/run/log/bootchart-xxxxxxxx-xxxx.svg



Bootchart for VB1 - Xxx, xx xxx xxxx xx:xx:xx +0300

System: Linux 4.4.0-36-generic #55-Ubuntu SMP Thu Aug 11 18:01:55 UTC 2016 x86_64
CPU: Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz

Boot options: BOOT_IMAGE=/boot/vmlinuz-4.4.0-36-generic root=UUID=319b2046-7438-4a7f-ad7d-fec674193f6f ro quiet splash init=/lib/systemd/systemd-bootchart
Build: Ubuntu 16.04.1 LTS
Log start time: 3.846s
Idle time: 8.376s
Graph data: 25.000 samples/sec, recorded 500 total, dropped 1 samples, 994 processes, 769 filtered

Top CPU consumers:

2.542s - systemd-bootcha [557]
0.916s - compiz [2632]
0.840s - Xorg [1674]
0.472s - fwupd [2828]
0.424s - nautilus [2717]
0.304s - systemd-udevd [613]
0.257s - gnome-software [2722]
0.220s - unity-settings- [2504]
0.199s - systemd-udevd [625]
0.180s - dbus-daemon [2349]

IO utilization - read

pilt kogusuuruses (.svg)

dmesg

- log about boot process
- filtering: *dmesg | grep <string>*
  - *dmesg | grep usb*
  - *dmesg -e*
  - *dmesg -H*
  - *man dmesg*

```
[    0.000000] Linux version 4.4.0-36-generic (buildd@lcy01-01) (gcc
version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.2) ) #55-Ubuntu SMP Thu
Aug 11 18:01:55 UTC 2016 (Ubuntu 4.4.0-36.55-generic 4.4.16)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.4.0-36-generic
root=UUID=319b2046-7438-4a7f-ad7d-fec674193f6f ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
```

dmesg log in full size (.txt)

Ubuntu boot process in short

- BIOS – recognizing hardware
- boot loader – locates at storage’s MBR (first sector)
  a) on local hard drive
  b) on external storage (USB, DVD, CD jne)
  c) in network – from network interface card (NIC) read-only memory (ROM) there will be run PXE (*Pre-Execution Environment*)
- kernel – access to hardware, runs the *init* process
- *init* processes (*systemd, upstart* etc)

Understanding the Linux Boot Process - CompTIA Linux+, LPIC-1

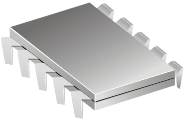
https://www.youtube.com/watch?v=mHB0Z-HUauo (9m 6s)

For successful boot

- BIOS must find the boot loader – depends on hardware
- boot loader must find the kernel and initrd – depends on BIOS setup
- kernel will run and with help of initrd have to find the / partition
- to fix */initrd.img*:
  - *man update-initramfs*
    - *sudo update-initramfs -u (updates the newest installed kernel initrd)*
    - *sudo update-initramfs -c -k 4.4.0-34-generic (precise kernel initrd)*
  - *man mkinitramfs*

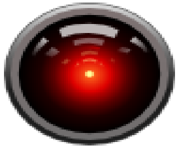
Boot phases

- read-only memory (ROM) phase



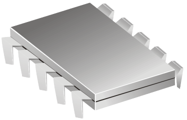
- boot block phase

- kernel phase



- process phase

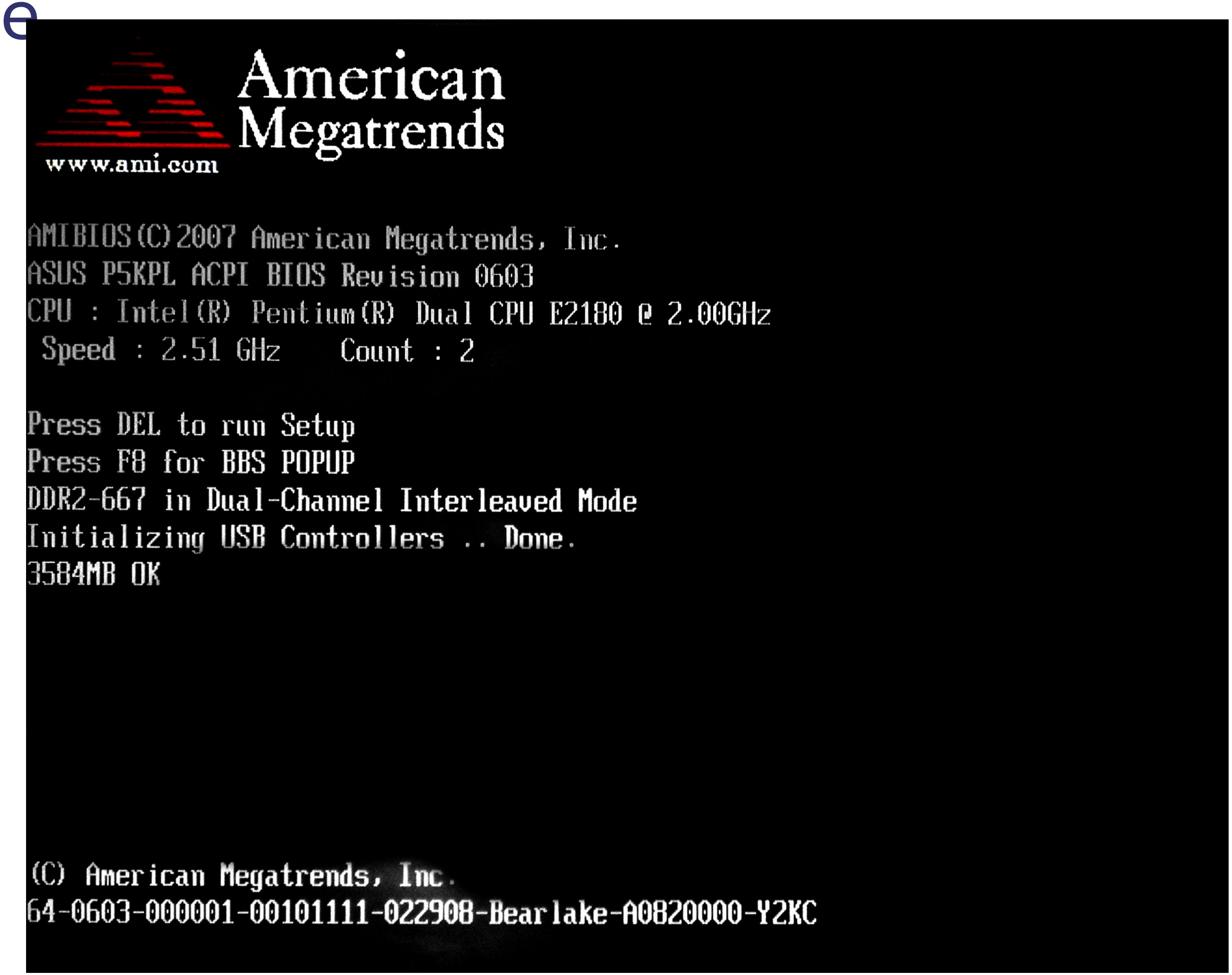
Read-only memory (ROM) phase



- ...will be fulfilled by turning PC on
- In IBM PC there will be BIOS (*Basic Input/Output System*) started from ROM memory in first place
- POST – *Power-on Self Test*
  - devices like disks, memory, processor(s) etc will be detected
  - error code(s) in case of problem(s)
- Newer BIOS alternatives
  - Extensible Firmware Interface (EFI)
  - CoreBoot (LinuxBIOS)
  - Libreboot

•**read-only memory phase**
•boot block phase
•kernel phase
•process phase

BIOS, 1st phase

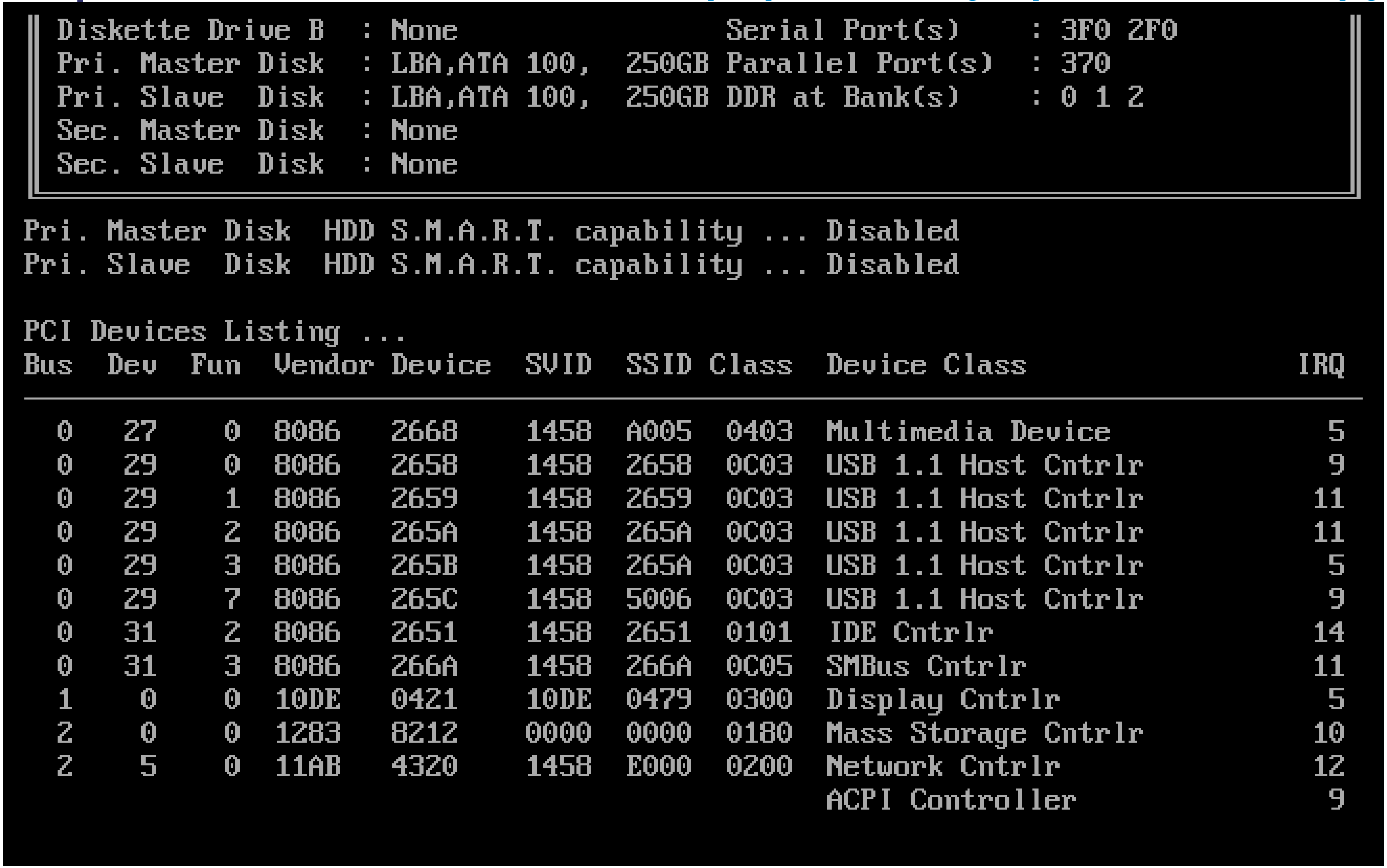


```
AMIBIOS(C)2007 American Megatrends, Inc.
ASUS P5KPL ACPI BIOS Revision 0603
CPU : Intel(R) Pentium(R) Dual CPU E2180 @ 2.00GHz
 Speed : 2.51 GHz     Count : 2

Press DEL to run Setup
Press F8 for BBS POPUP
DDR2-667 in Dual-Channel Interleaved Mode
Initializing USB Controllers .. Done.
3584MB OK


(C) American Megatrends, Inc.
64-0603-000001-00101111-022908-Bearlake-A0820000-Y2KC
```

https://upload.wikimedia.org/wikipedia/commons/9/92/POST_P5KPL.jpg

BIOS, 2nd phase

https://upload.wikimedia.org/wikipedia/commons/1/1a/POST2.png



```
  Diskette Drive B  : None               Serial Port(s)    : 3F0 2F0
  Pri. Master Disk  : LBA,ATA 100,  250GB Parallel Port(s)  : 370
  Pri. Slave  Disk  : LBA,ATA 100,  250GB DDR at Bank(s)    : 0 1 2
  Sec. Master Disk  : None
  Sec. Slave  Disk  : None

Pri. Master Disk  HDD S.M.A.R.T. capability ... Disabled
Pri. Slave  Disk  HDD S.M.A.R.T. capability ... Disabled

PCI Devices Listing ...
Bus  Dev  Fun  Vendor Device  SVID  SSID Class  Device Class               IRQ
------------------------------------------------------------------------------
  0   27    0  8086   2668    1458  A005  0403  Multimedia Device            5
  0   29    0  8086   2658    1458  2658  0C03  USB 1.1 Host Cntrlr          9
  0   29    1  8086   2659    1458  2659  0C03  USB 1.1 Host Cntrlr         11
  0   29    2  8086   265A    1458  265A  0C03  USB 1.1 Host Cntrlr         11
  0   29    3  8086   265B    1458  265A  0C03  USB 1.1 Host Cntrlr          5
  0   29    7  8086   265C    1458  5006  0C03  USB 1.1 Host Cntrlr          9
  0   31    2  8086   2651    1458  2651  0101  IDE Cntrlr                  14
  0   31    3  8086   266A    1458  266A  0C05  SMBus Cntrlr                11
  1    0    0  10DE   0421    10DE  0479  0300  Display Cntrlr               5
  2    0    0  1283   8212    0000  0000  0180  Mass Storage Cntrlr         10
  2    5    0  11AB   4320    1458  E000  0200  Network Cntrlr              12
                                                ACPI Controller              9
```

S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology)

https://en.wikipedia.org/wiki/Comparison_of_S.M.A.R.T._tools

Read-only memory phase

- After the initialization of devices there will be run bootstrap *loader* program, that reads into RAM the boot sector MBR - *Master Boot Record* (512 bytes) from boot device according BIOS boot order setting
- program located in boot sector will be executed and with that the boot block phase starts

- **read-only memory phase**
- boot block phase
- kernel phase
- process phase

Boot block phase

- in boot block phase the program loaded from MBR, will load the kernel with boot parameters defined in boot loader configuration files (e.g. */etc/default/grub* ja */etc/grub.d/**) into RAM
- quite often the kernel loader in case of nowadays OS’es does not fit into MBR
  - there is also keeped primary partition table
- to solve the mentioned issue – boot block will be divided into two part:
  - first one is located in MBR and it reads in second part
  - first part with second one forms the boot block

Boot block phase, MBR

- MBR – Master Boot Record first 512 bytes
  - First 446 bytes is first part of boot block (stage 1)
  - next 64 bytes is primary partition table
  - 2 bytes – 0xAA55 is magic number to ensure that the block is really the MBR block

https://en.wikipedia.org/wiki/Master_boot_record

https://en.wikipedia.org/wiki/Partition_table

Boot block phase

- The boot block program task is to load into memory the operating system kernel and run it
- Therefore the program must know, how to load the kernel
  - should understand the file system in order to load the kernel
- Common boot block programs (boot loaders)
  - GRUB *Grand Unified Boot loader*
  - LiLo *Linux Loader*
  - Ntldr *Windows kernel loader*

- read-only memory phase
- **boot block phase**
- kernel phase
- process phase

Boot block phase

- When the program in boot block can not load the kernel of operating system then the *chain loading* will be used
- Boot block will load the loader into memory, which is operating system specific and will run it
- There is a choice of 1...n operating systems that can be located on different storage
- There is also option to change boot parameters

- read-only memory phase
- **boot block phase**
- kernel phase
- process phase

https://help.ubuntu.com/community/BootOptions

https://wiki.ubuntu.com/Kernel/KernelBootParameters

https://wiki.ubuntu.com/RecoveryMode

http://askubuntu.com/questions/100232/how-do-i-change-the-grub-boot-order

Kernel phase

- read-only memory phase
- boot block phase
- **kernel phase**
- process phase

- In case of kernel phase
  - */boot/vmlinuz*
- /initrd.img - “*initial ram drive*”
  - early *user space*
  - temporary root file system
  - loads the network card (rtc hardware) support before the OS will run
  - */initrd.img* will be unmounted
  - in kernel phase Linux will unpack the kernel and initialize memory structures
  - after running the kernel there will be loaded the init program and run it, please see also https://www.youtube.com/watch?v=LTFLEXYY6jY
  - hardware support will be loaded

kernel information in Ubuntu and also others

- *man uname*
- version
  - *uname -r*
- 32-bit or 64-bit
  - *arch*
  - *uname -i* (hardware platform)
  - *uname -m* (hardware name)
  - *uname -p* (processor type)
- kernel name
  - *uname -s*
- operating system name
  - uname -o

Boot block phase in MS Windows

- In case of MS Windows XP and Server 2003
  - NTDETECT.COM will be loaded and run
  - the kernel and HAL (Hardware Abstraction Layer) files will be loaded (ntoskrnl.exe ja hal.exe)
  - the kernel memory structure and drivers will be loaded
  - the kernel will be loaded

Kernel phase in MS Windows system

- the structures read from registry will be initialized
- the process ***idle*** will be created
- the process ***System*** will be created
- the hardware abstraction layer process ***hal*** will be created
- drivers will be loaded
- the session manager ***smss.exe*** (*Session Manager SubSystem*) will be run

MS Windows Vista and Server 2008

- the boot block ***bootmgr*** or ***Windows Boot Manager*** will read ***BCD*** or ***Boot Configuration Data*** database, e.g. ***\Boot\Bcd*** (earlier time there has been ***boot.ini*** file used)
- then the kernel loader ***winload.exe*** (or ***winresume.exe***) will be loaded and run

Some references about Windows boot process

- https://technet.microsoft.com/en-us/library/ee221031(v=ws.10).aspx Boot Process and BCDEdit
- https://jon.glass/looks-at-the-win10-boot-process/ - Windows 10 boot specifics

Process phase

- process phase depends on operating sustem
- the multiuser environment will be created
- the graphical user interface processes will be created (in case of workstation)

- read-only memory phase
- boot block phase
- kernel phase
- **process phase**

Lilo

- Lilo - **LI**nux **Lo**ader
- was widely used earlier time
- the configuration is located at
  - ***/etc/lilo.conf***
- Cons
  - after changing configuration the MBR has to be always rewritten
- Pros
  - tested and working

GRUB

- **GRUB** - Grand Unified Bootloader
- **GRUB 2**
  - new version that has been created from scratch
  - nowadays widely used
- **GRUB Legacy**
  - widely used in earlier distributions
  - not developed anymore

GRUB2

- Options
  - scriptable
  - internationalization support (different codepages through gettext and translations)
  - more supported file systems, e.g. ext4
  - the framework supports further developments (was also the reason why GRUB was rewritten almost from scratch)

GRUB 2

- installing instead of GRUB Legacy
  - ***apt-get install grub2***
  - allow to use the *chainloading*
  - when everything works then ***upgrade-from-grub-legacy*** command would be used
- configuration is located at (do not edit manually!)

  /boot/grub/grub.cfg
- for changing the configuration, there should be edited the following files: ***/etc/default/grub*** and ***/etc/grub.d/****
- to confirm changes: ***sudo update-grub***

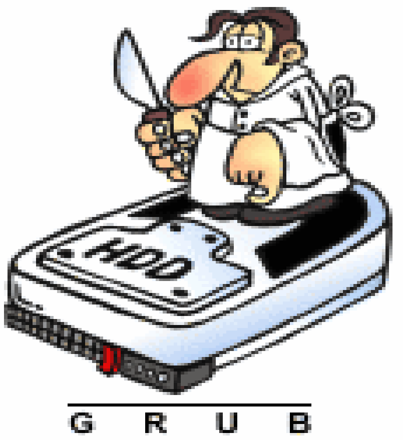
GRUB2: /etc/default/grub

- `GRUB_DEFAULT=0`
- `GRUB_HIDDEN_TIMEOUT=0`
- `GRUB_HIDDEN_TIMEOUT_QUIET=true`
- `GRUB_TIMEOUT=10`
- ``GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian` ``
- `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"`
- `GRUB_CMDLINE_LINUX=""`
- more information:
  - *info -f grub -n 'Simple configuration*'

the file in full length

GRUB Legacy

- examples of configuration
- ***/boot/grub/menu.lst***



default 0 – default will be loaded the first one

timeout 10 – menu display time

(in our class rooms relatively long)

title Debian GNU/Linux, kernel 2.6.28-11-generic

root (hd0,2)

kernel /boot/vmlinuz-... root=.. ro single

initrd /boot/initrd.img-...

Shutting down

- *init* will be invited to close the *user space* functionality in controlled way
- *init* will be closed
- *kernel* will run the closing process of itself

Multiple operating systems?

- *dual boot, triple boot, etc*
  - MS Windows + GNU/Linux
  - many same operating systems
  - MS Windows + GNU/Linux + macOS
- hardware virtualization – using multiple operating systems simultaneously
  - VirtualBox
  - VMware
  - etc (please see the comparison)

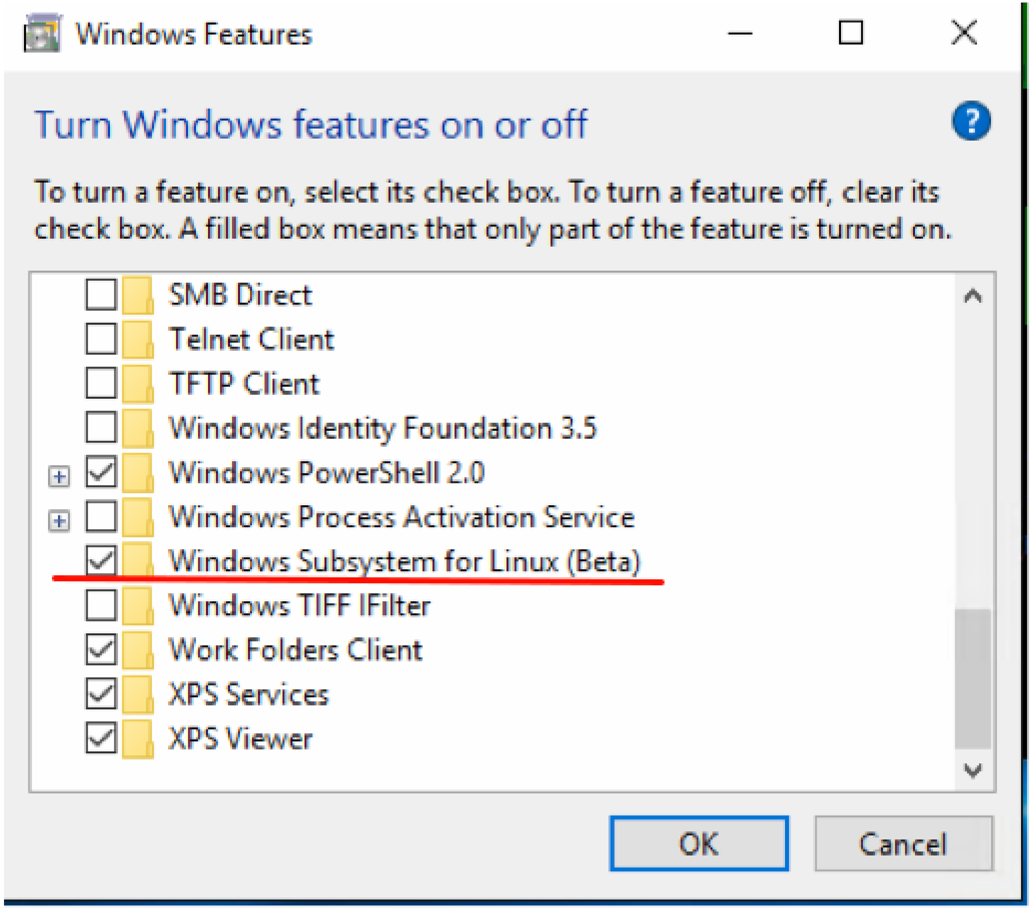
MS Windows 10 + WSL (Windows Subsystem for Linux)

https://msdn.microsoft.com/en-us/commandline/wsl/install_guide

Linux is working on top of Windows kernel

Powershell command to enable WSL:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- □ SMB Direct
- □ Telnet Client
- □ TFTP Client
- □ Windows Identity Foundation 3.5
- ☑ Windows PowerShell 2.0
- □ Windows Process Activation Service
- ☑ Windows Subsystem for Linux (Beta)
- □ Windows TIFF IFilter
- ☑ Work Folders Client
- ☑ XPS Services
- ☑ XPS Viewer

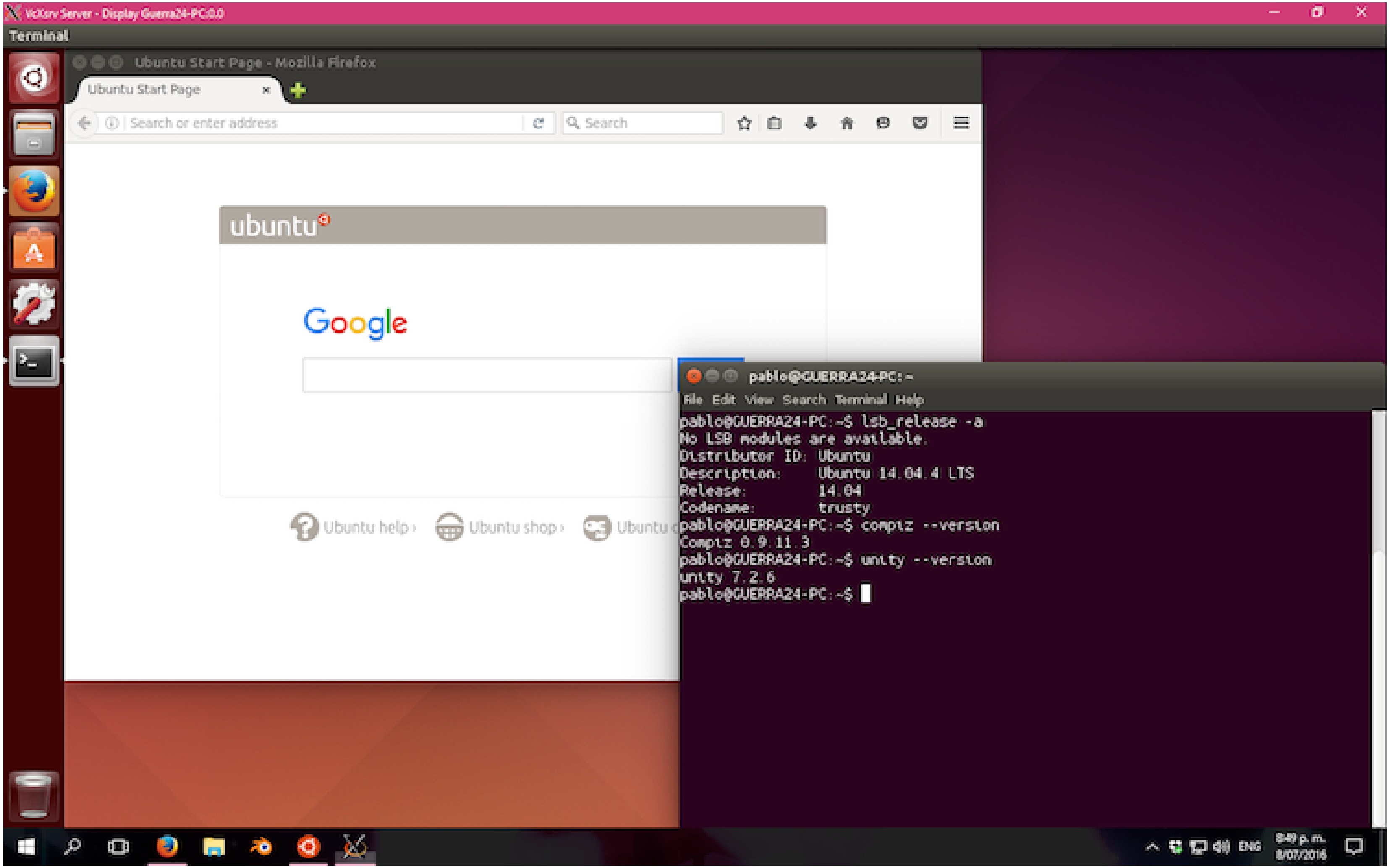
http://blog.dustinkirkland.com/2016/08/howdy-windows-six-part-series.html

http://www.omgubuntu.co.uk/2016/07/someone-just-installed-unity-windows

https://github.com/RoliSoft/WSL-Distribution-Switcher

https://www.suse.com/communities/blog/make-windows-green-part-1/

Ubuntu Linux working using WSL in Windows 10



```
pablo@GUERRA24-PC:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
pablo@GUERRA24-PC:~$ compiz --version
Compiz 0.9.11.3
pablo@GUERRA24-PC:~$ unity --version
unity 7.2.6
pablo@GUERRA24-PC:~$ █
```

Links

- Debian booting in Estonian http://kuutorvaja.eenet.ee/wiki/Debiani_alglaadimine
- https://en.wikipedia.org/wiki/Linux_startup_process
- https://en.wikipedia.org/wiki/Windows_startup_process
- https://en.wikipedia.org/wiki/Booting
- http://www.computerhope.com/unix/dmesg.htm
- Wikipedia – BIOS http://en.wikipedia.org/wiki/BIOS
- Coreboot https://en.wikipedia.org/wiki/Coreboot
- Libreboot https://en.wikipedia.org/wiki/Libreboot
- IBM - Inside the Linux boot process http://www.ibm.com/developerworks/library/l-linuxboot/
- GRUB2 https://help.ubuntu.com/community/Grub2
- https://wiki.ubuntu.com/Booting
- https://help.ubuntu.com/community/BootOptions
- http://askubuntu.com/questions/592740/how-does-the-ubuntu-boot-process-work

Links (2)

- multiple MS Windows to same computer - http://www.howtogeek.com/197647/how-to-dual-boot-windows-10-with-windows-7-or-8/
- legal free versions of Microsoft software - https://www.microsoft.com/en-us/evalcenter/
- ready to use virtual machines from Microsoft (also for Linux, macOS) - https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
- MS Windows + Ubuntu Linux - https://help.ubuntu.com/community/WindowsDualBoot
- Ubuntu + macOS https://help.ubuntu.com/community/DualBoot/MacOSX
- Ubuntu Linux + another OS (MS Windows, macOS etc) -
  https://help.ubuntu.com/community/DualBoot
  https://help.ubuntu.com/community/MultiOSBoot
  http://ubuntuguide.org/wiki/Multiple_OS_Installation
- virtualization based on Ubuntu - https://help.ubuntu.com/community/CategoryVirtualization
- Ubuntu + Windows 10 https://www.youtube.com/watch?v=JvBZBfY5Pfc

Questions?

Thank you for your attention!