

to

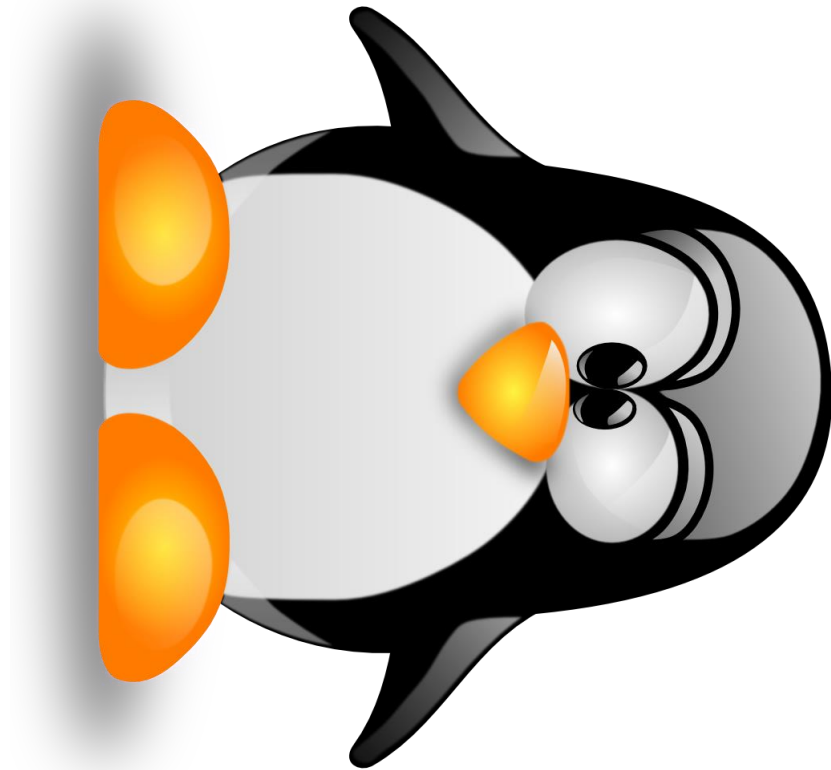
Introduction

Linux



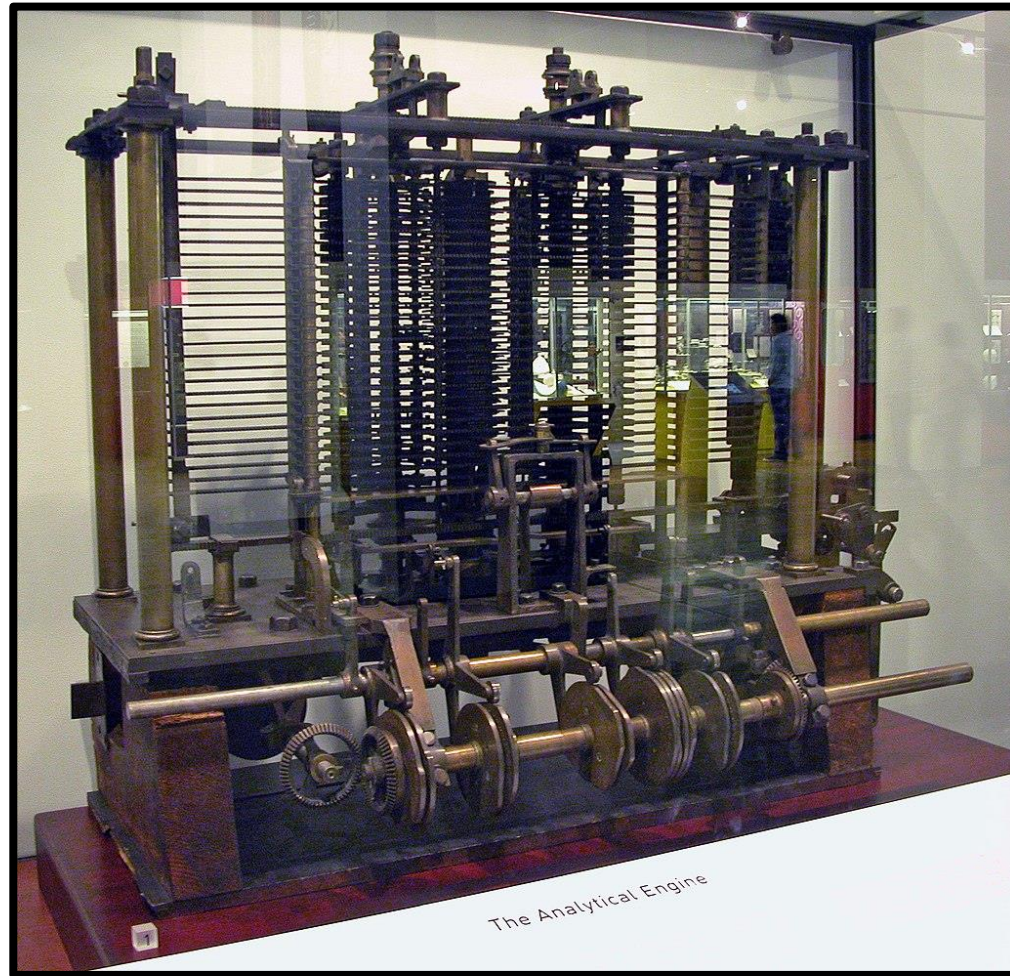
Nikolas Papanikolaou

brief history of computers



Computer History – A Timeline

The Analytical Engine (1837)



Charles Babbage's Analytical Engine & punched cards used to program the machine.

Source: Wikipedia

Computer History – A Timeline

The Analytical Engine (1837)

Ada Lovelace
The “first
programmer”

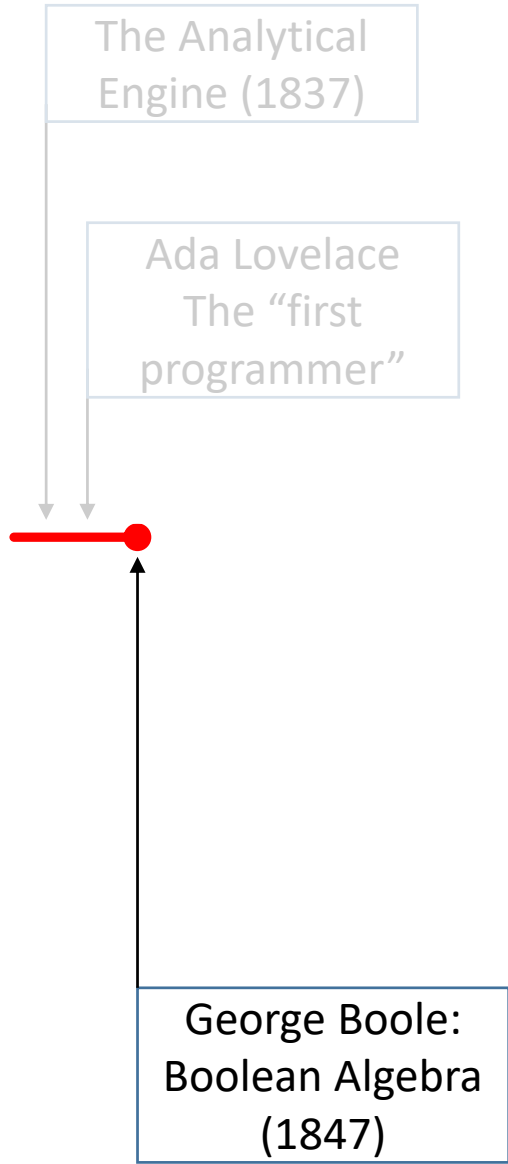


Computer History – A Timeline

The Analytical Engine (1837)

Ada Lovelace
The “first programmer”

George Boole:
Boolean Algebra
(1847)



Computer History – A Timeline

The Analytical Engine (1837)

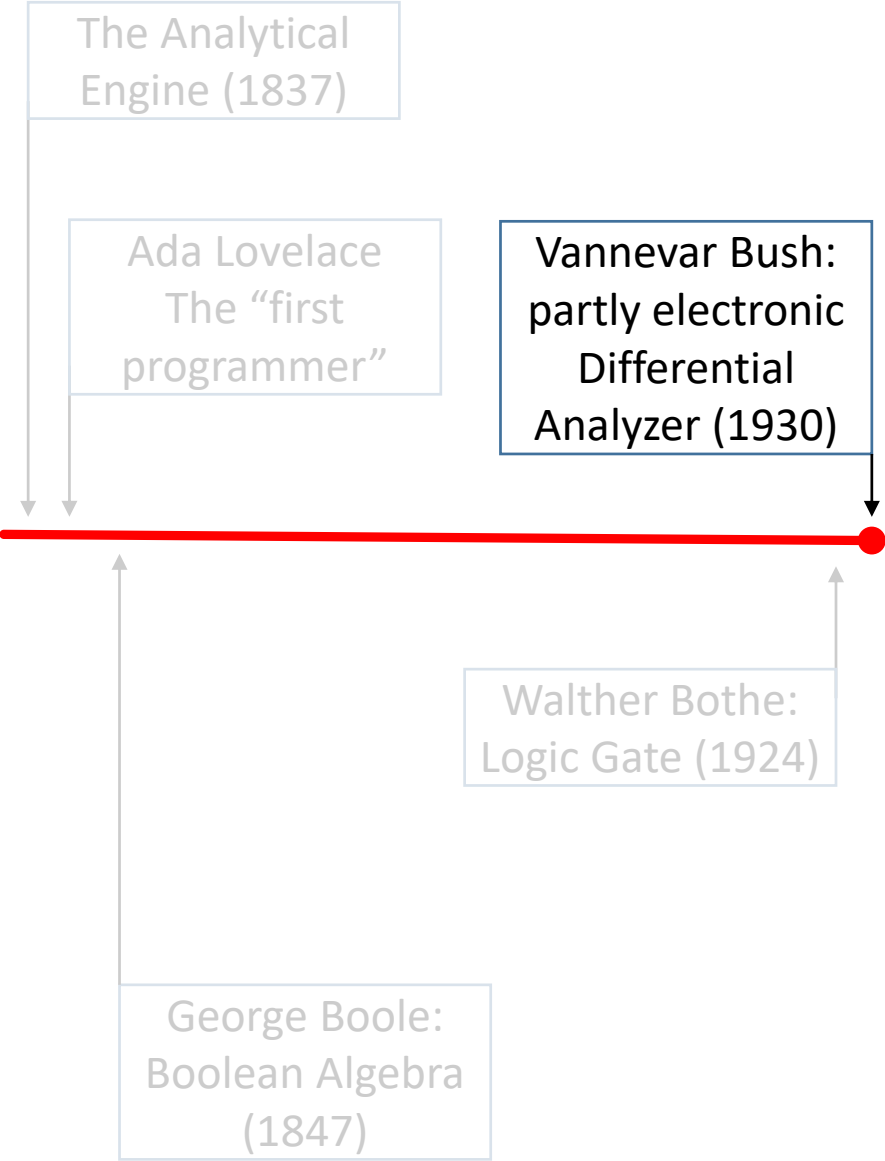
Ada Lovelace
The “first programmer”

Walther Bothe:
Logic Gate (1924)

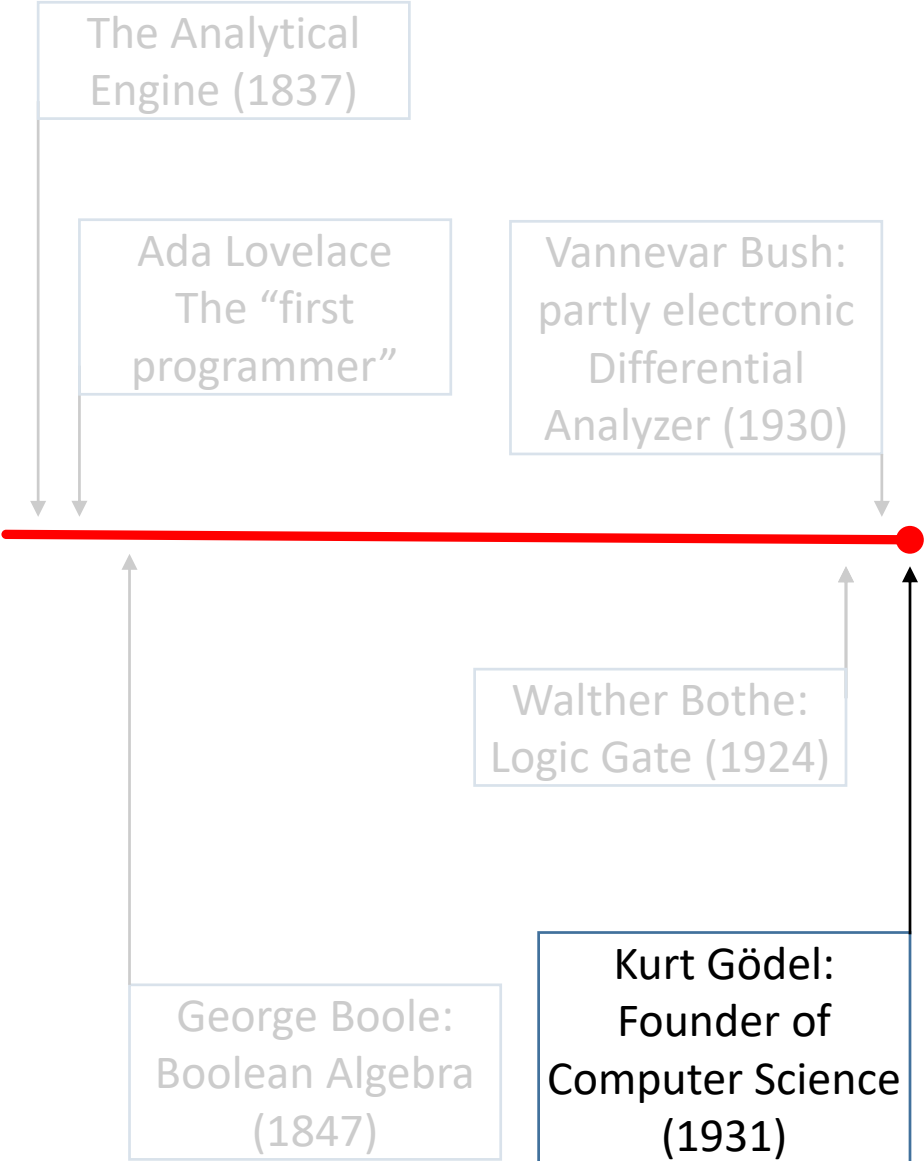
George Boole:
Boolean Algebra
(1847)



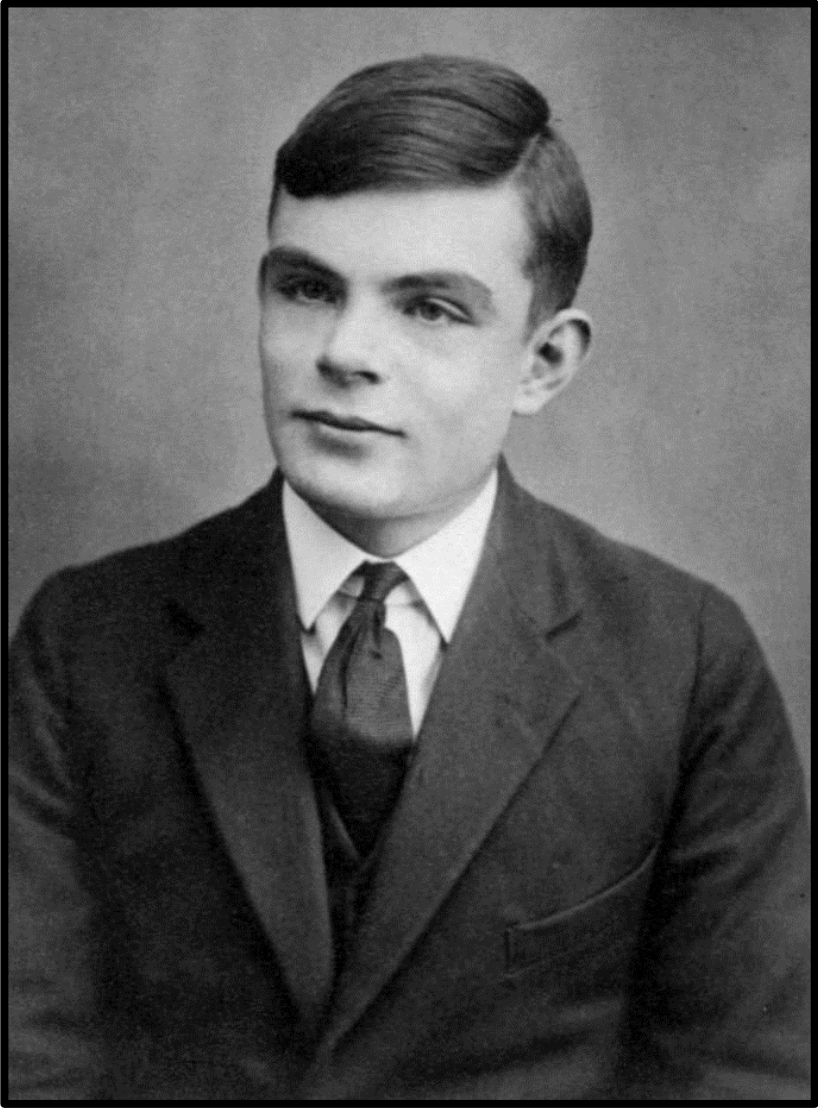
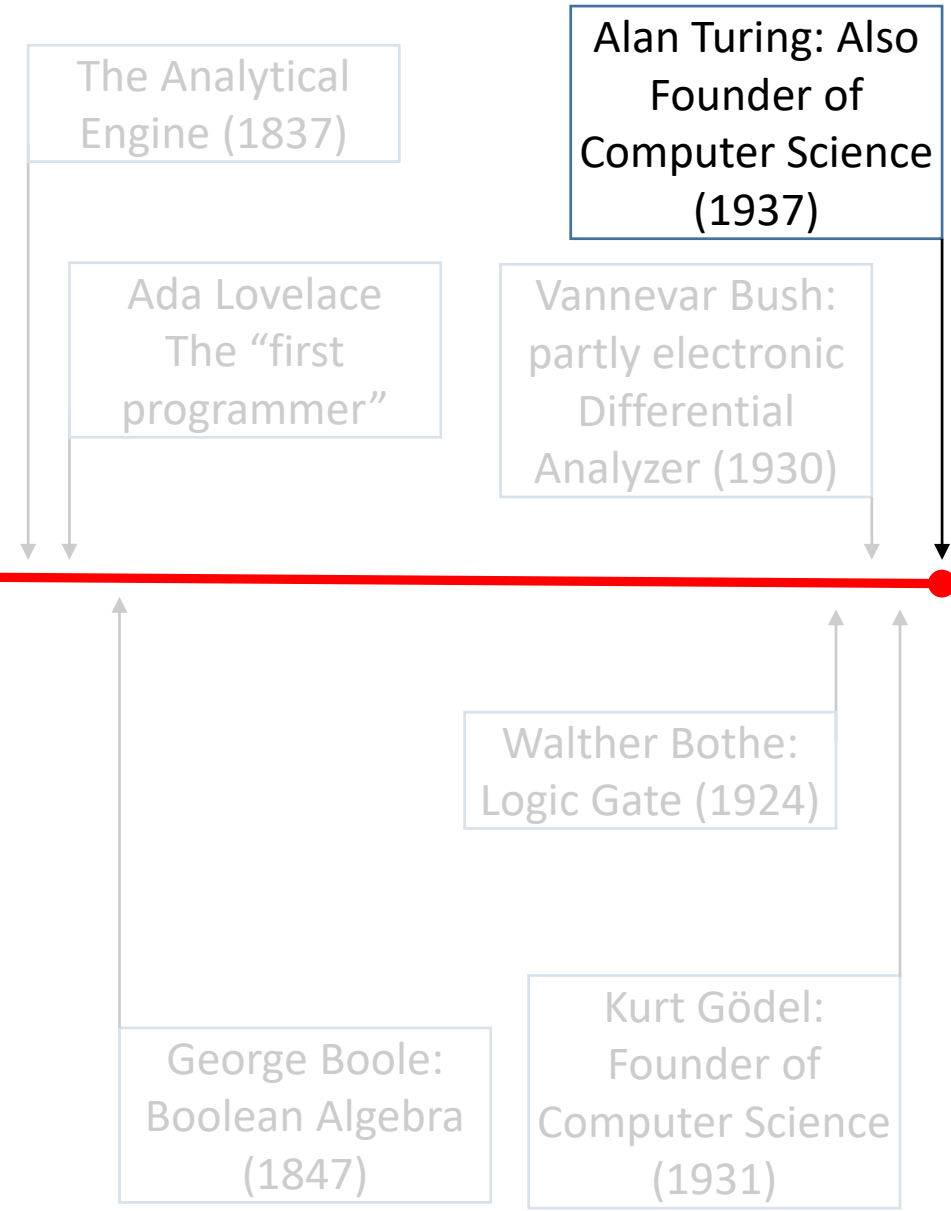
Computer History – A Timeline



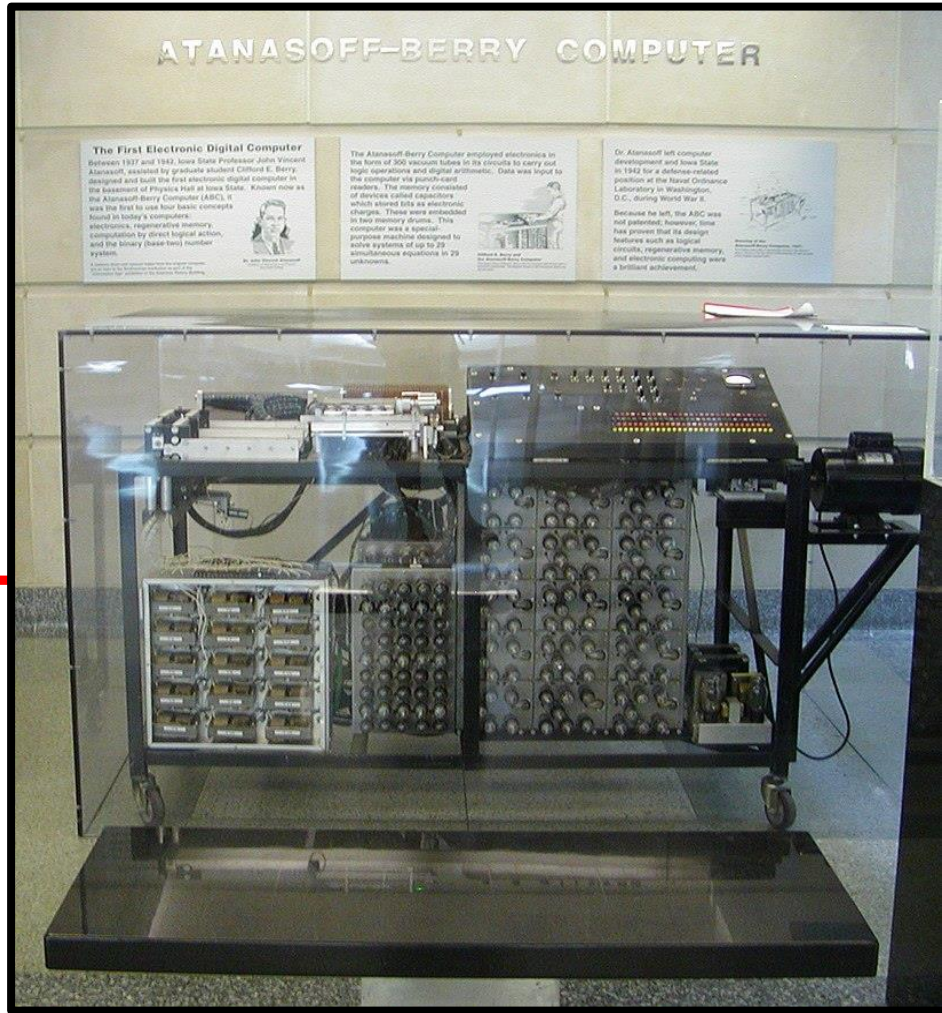
Computer History – A Timeline



Computer History – A Timeline



Computer History – A Timeline



Atanasoff-Berry computer replica at Durham Center, Iowa State University

Source: Wikipedia

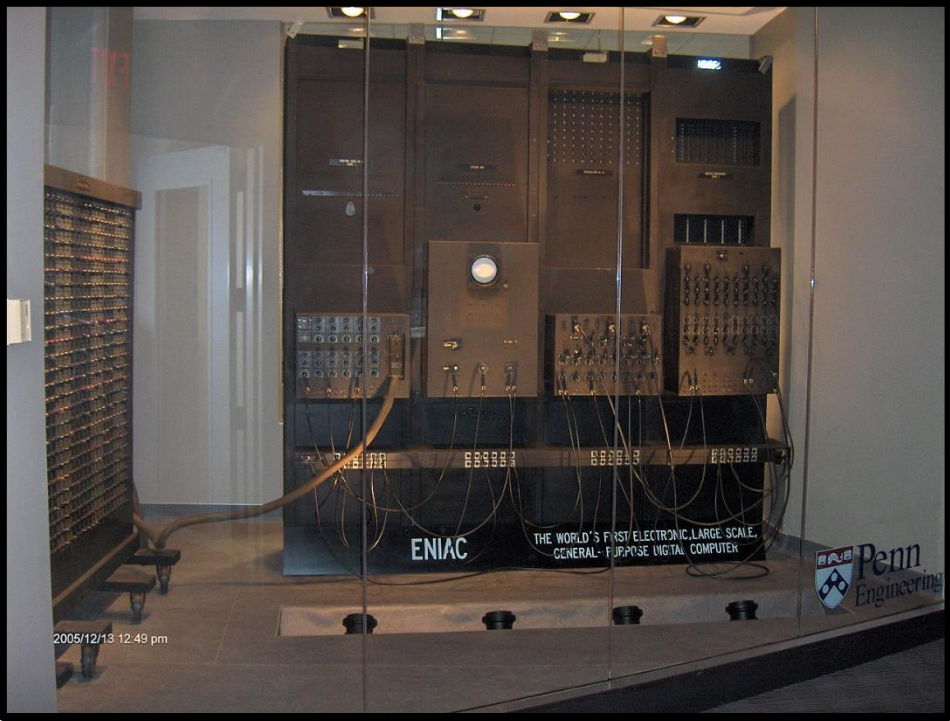
the ABC (Atanasoff-Berry Computer) prototype (1939)

Computer History – A Timeline

The Analytical Engine (1837)

Alan Turing: Also Founder of Computer Science

ENIAC (1945)



ENIAC panels and one of its three function tables, University of Pennsylvania

Source: Wikipedia

the ABC (Atanasoff-Berry Computer) prototype (1939)

(1847)

(1831)

Computer History – A Timeline

ENIAC (1945)

9/9


0800 Antam started
1000 " stopped - antam ✓

1300 (032) MP-MC { 1.2700 9.037 847 025
 (033) PRO 2 2.130476415 } 9.037 846 895 correct
 correct 2.130476415 4.615925059(-2)

Relays 6-2 in 033 failed special speed test
in relay .. 11,000 test.

Relays changed

1100 Started Cosine Tape (Sine check)
1525 Started Multi-Adder Test.

1545  Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1630 Antam started.
1700 closed down.

Relay 3145
Relay 337

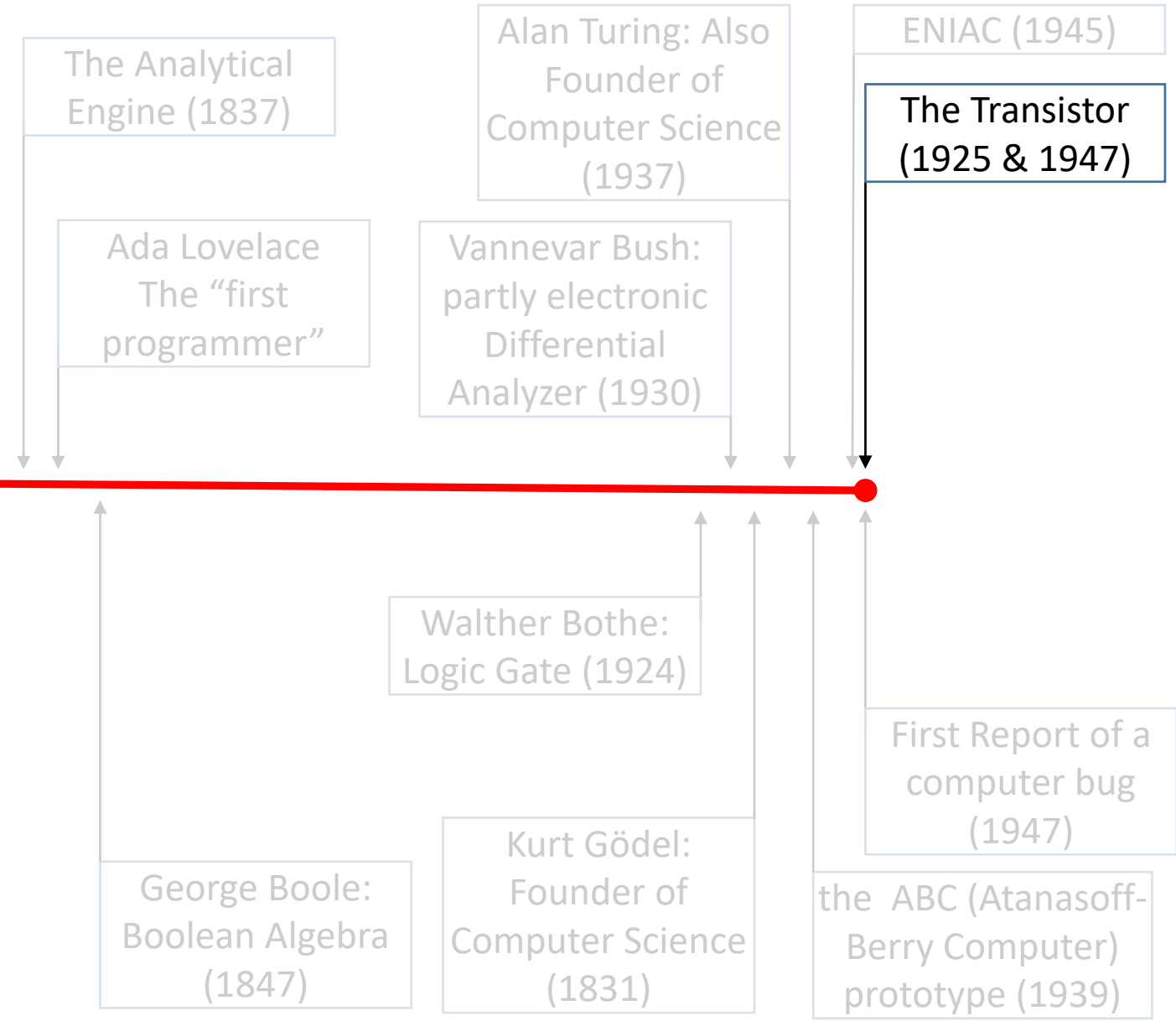
First Report of a
computer bug
(1947)

George Boole:
Boolean Algebra
(1847)

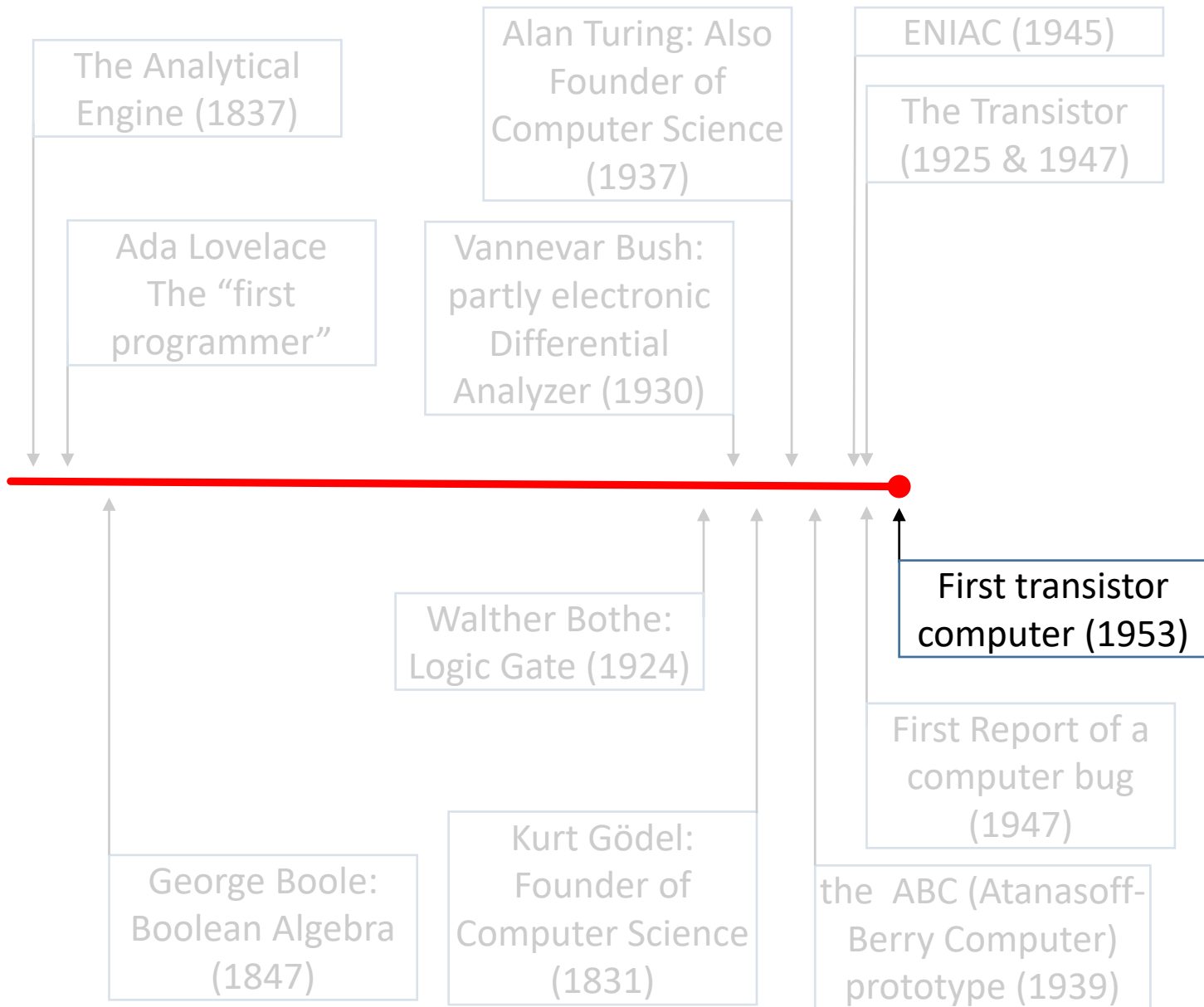
Kurt Gödel:
Founder of
Computer Science
(1931)

the ABC (Atanasoff-
Berry Computer)
prototype (1939)

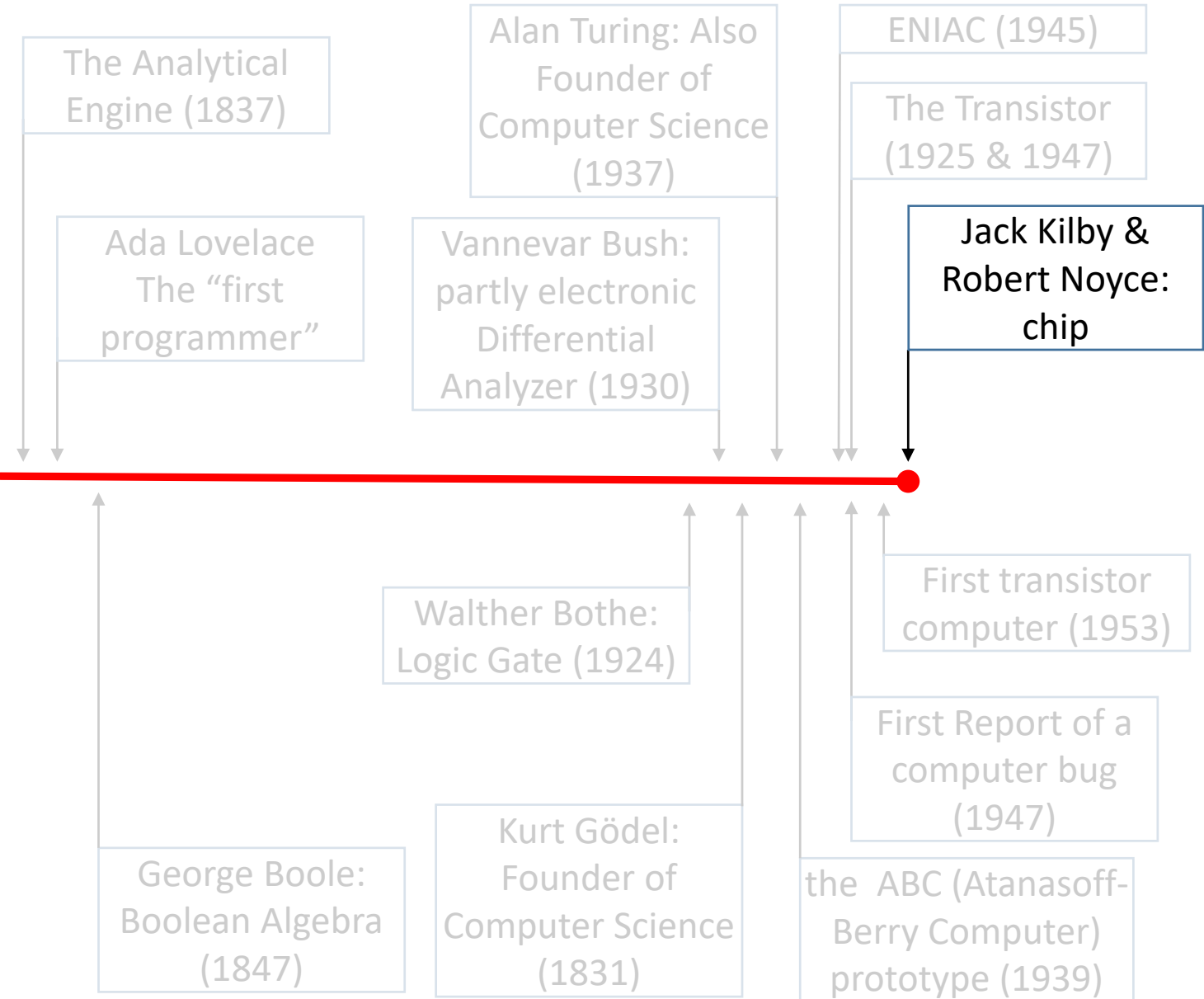
Computer History – A Timeline



Computer History – A Timeline



Computer History – A Timeline



Computer History – A Timeline



The Analytical Engine (1837)

Ada Lovelace
The "first programmer"

Alan Turing: Also

ENIAC (1945)

The Transistor (1925 & 1947)

Jack Kilby & Robert Noyce: Chip (1958)

Xerox Alto (1973)

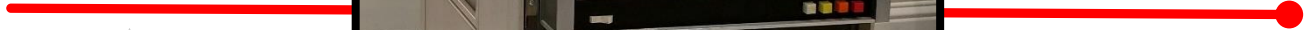
First transistor computer (1953)

First Report of a computer bug (1947)

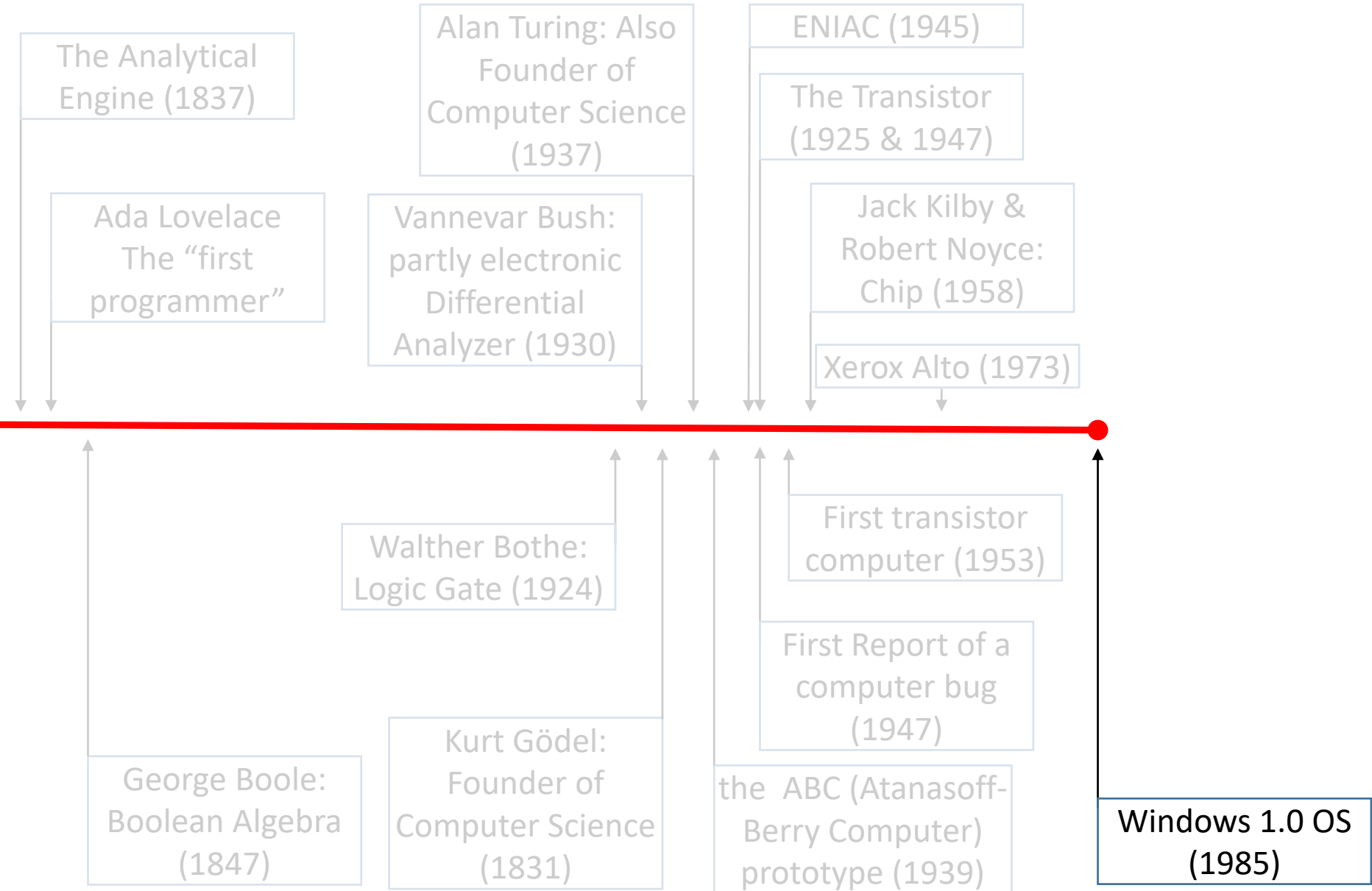
George Boole: Boolean Algebra (1847)

Founder of Computer Science (1831)

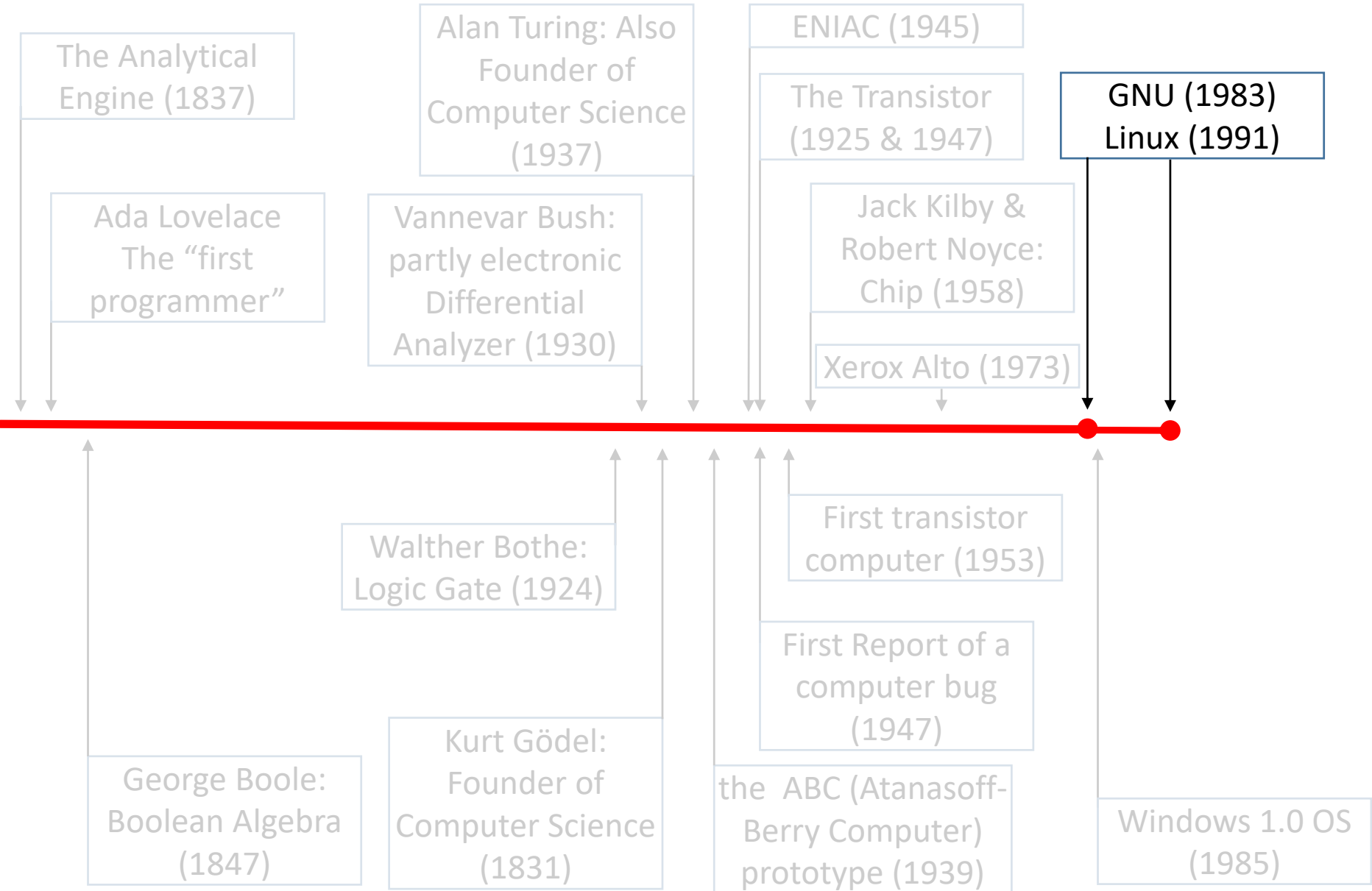
the ABC (Atanasoff-Berry Computer) prototype (1939)



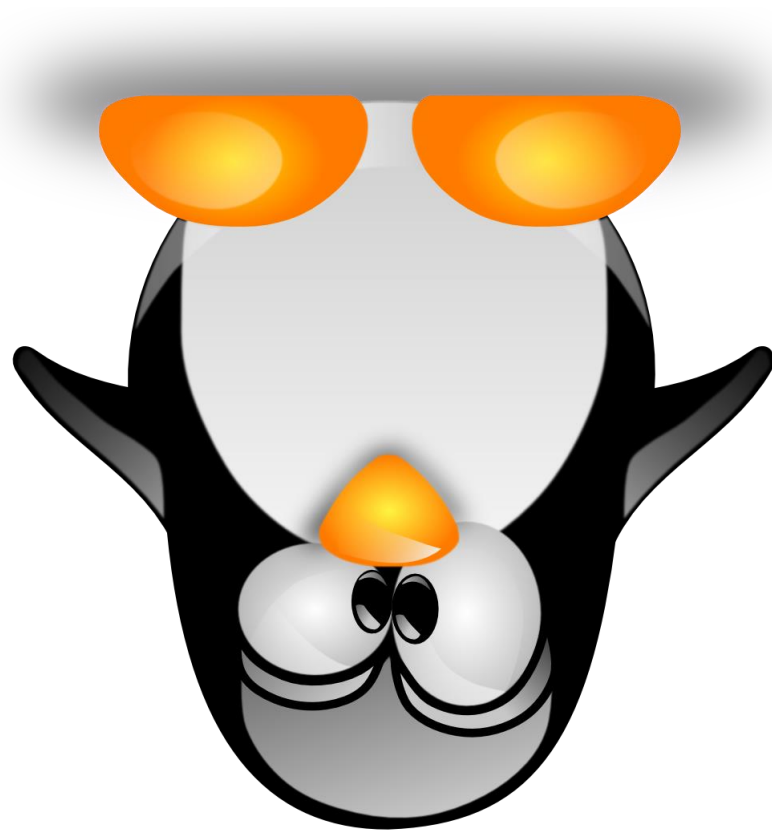
Computer History – A Timeline



Computer History – A Timeline



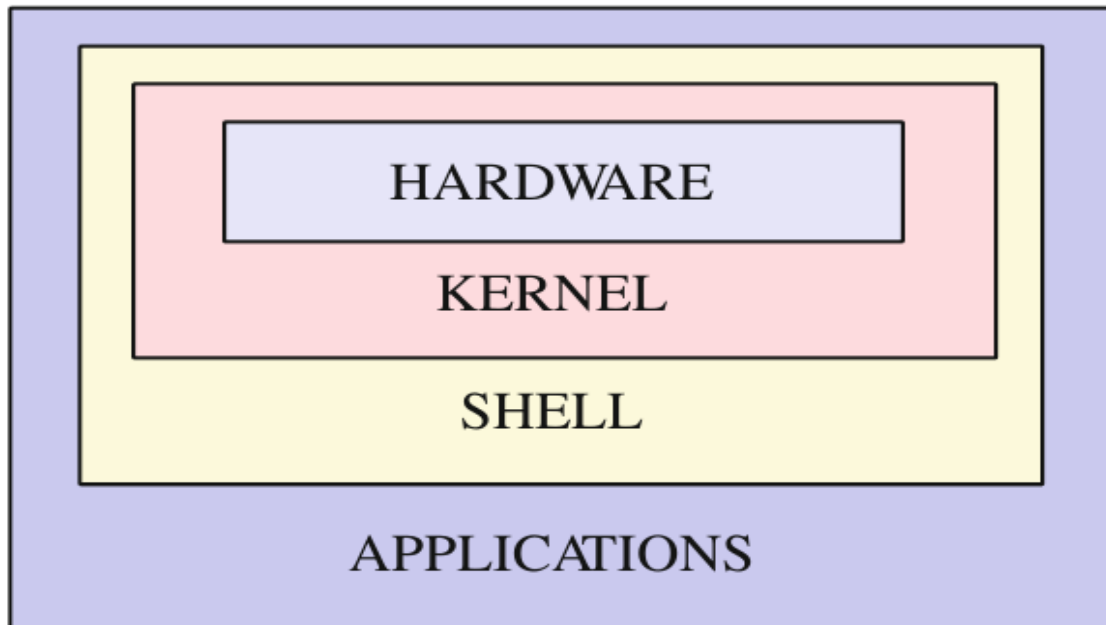
Linux, the basics



Linux OS

- What is an OS?

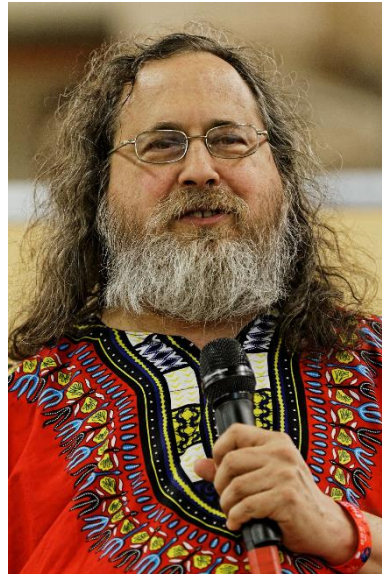
- Connects various pieces of hardware comprising the computer
- Allocates resources
- Allows other programs to run on it
(Almost all software runs on top of the OS)



Linux or GNU/Linux?



Linus Torvalds
Linux Kernel
(a UNIX clone)



Richard Stallman
GNU utilities (e.g. as
cp, mv, ls, date,
bash etc)

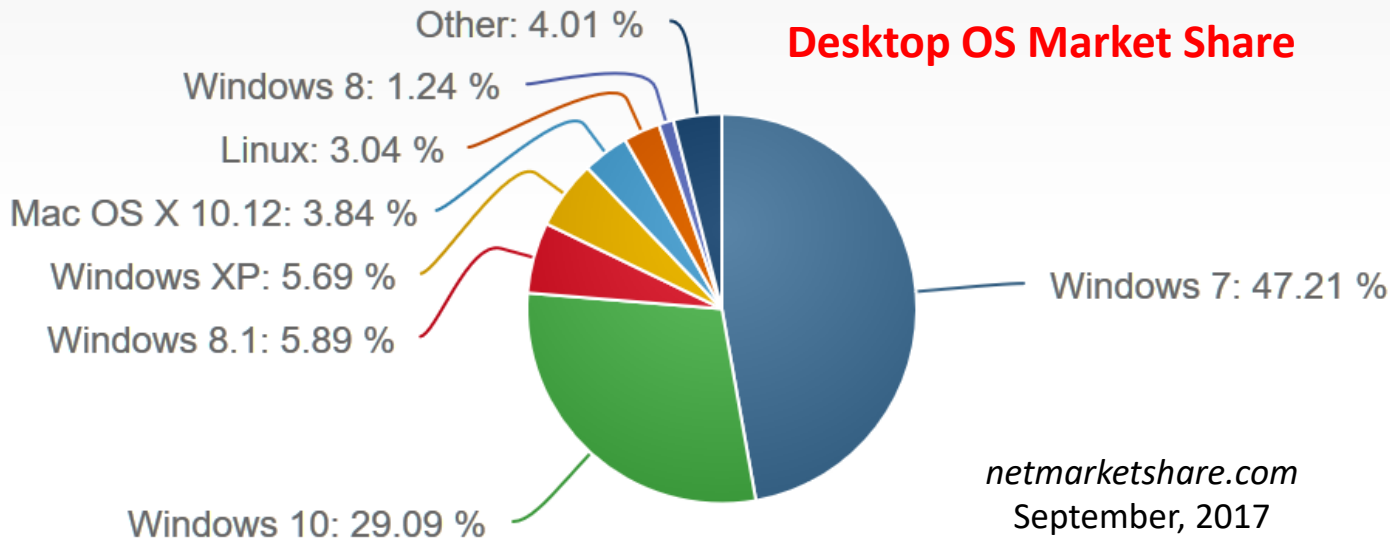


Bill Gates

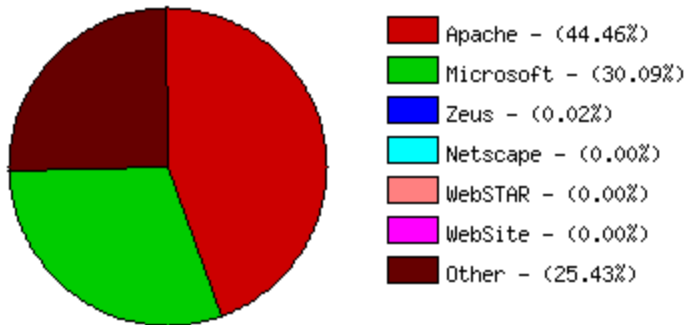
In trying to understand the Linux phenomenon, then, we have to look not at a single innovator but to a sort of bizarre Trinity : Linus Torvalds, Richard Stallman, and Bill Gates. Take away any of these three and Linux would not exist.

Linux Market Share

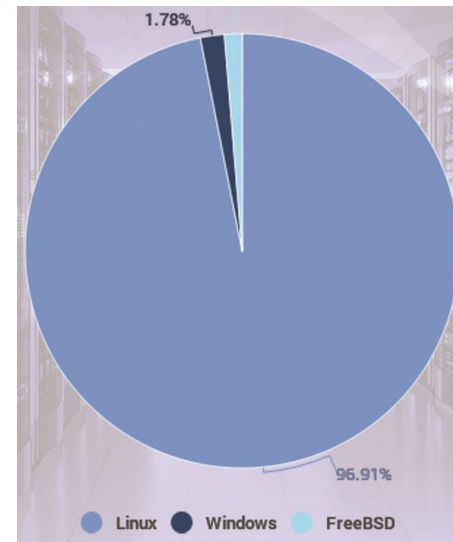
Desktop OS Market Share



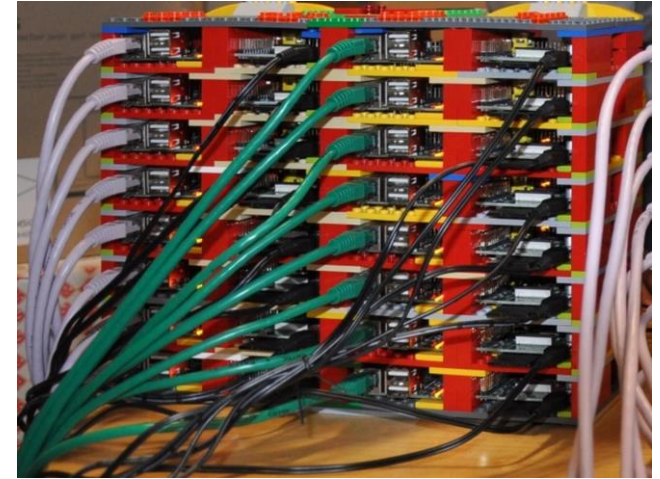
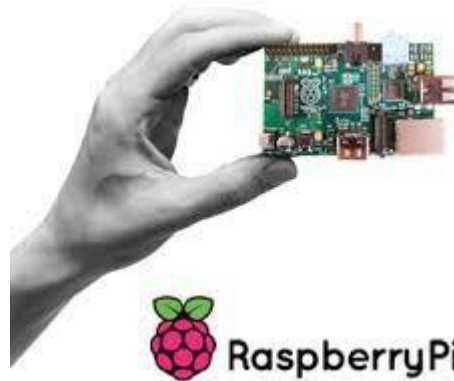
Web Server Market Share



Top 1M Server OS Market Share



Linux – compatible Hardware



Why Linux in Bioinformatics

- **Free**
- Large Community
- Powerful command line (shells)
- Ideal for remote connections
- Multiuser
- Modular & scalable
- Much more user friendly than before

Why Linux in Bioinformatics

Versatility

- Huge number of distributions

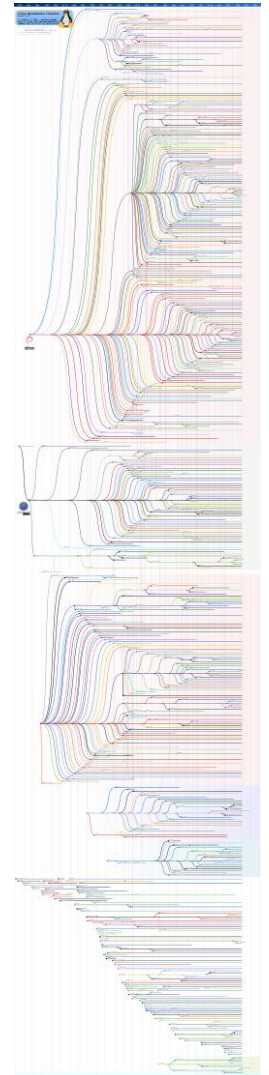


- Large number of Desktop Environments

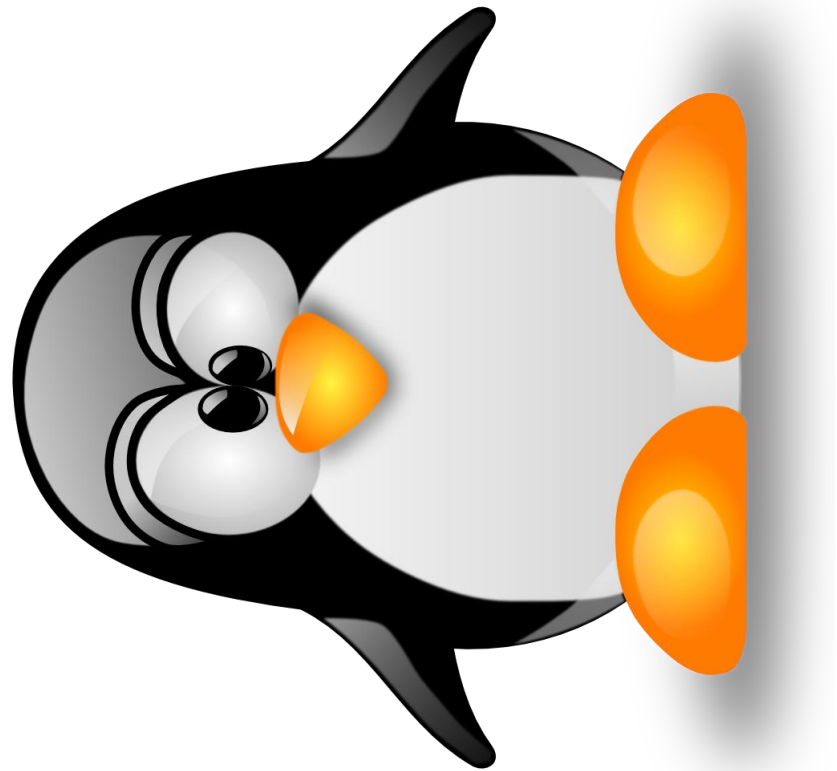


- Lots of software

Linux Distribution Timeline,
wikipedia

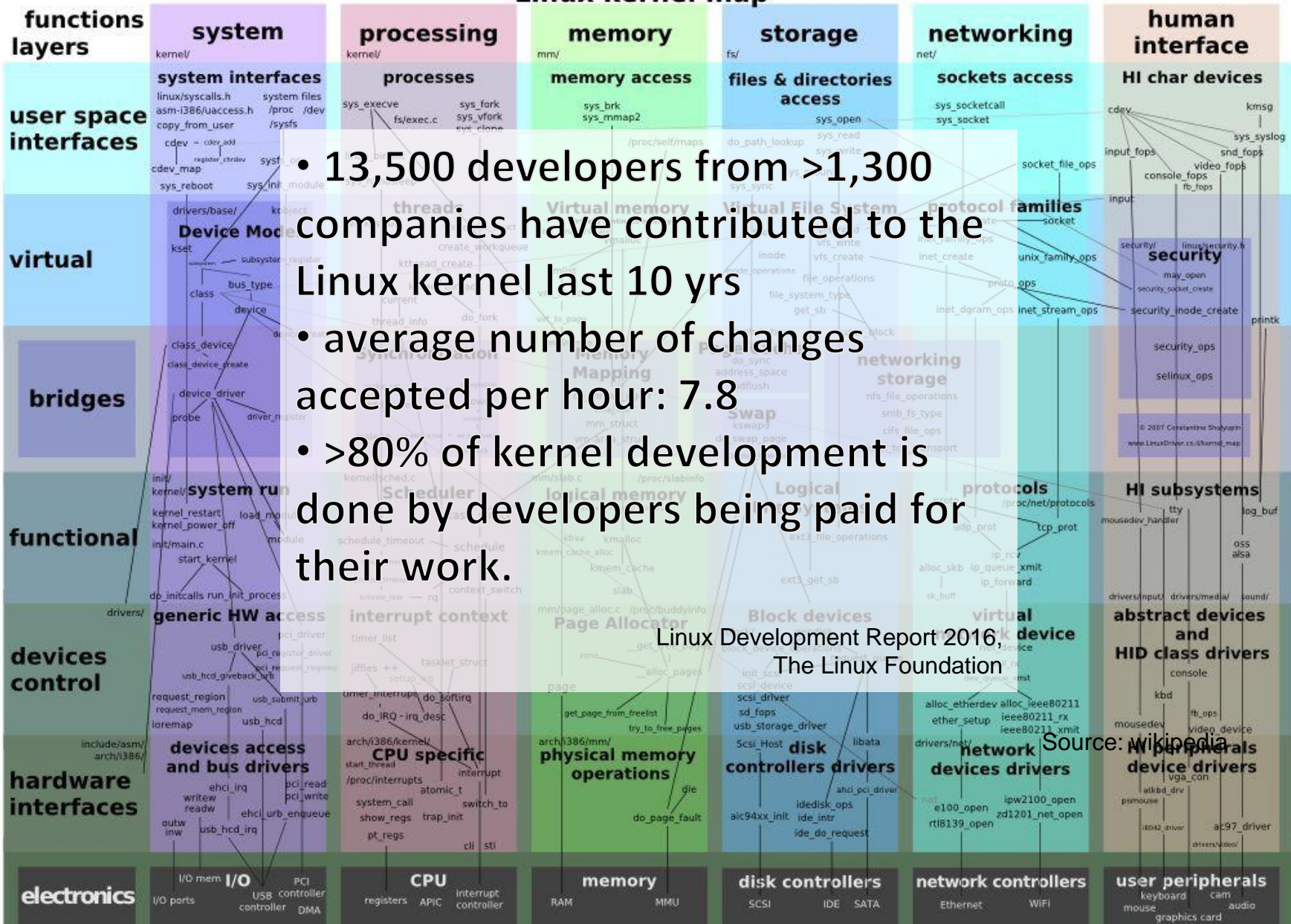


Linux, few
more technical
details



Linux Architecture - Kernel

Linux kernel map



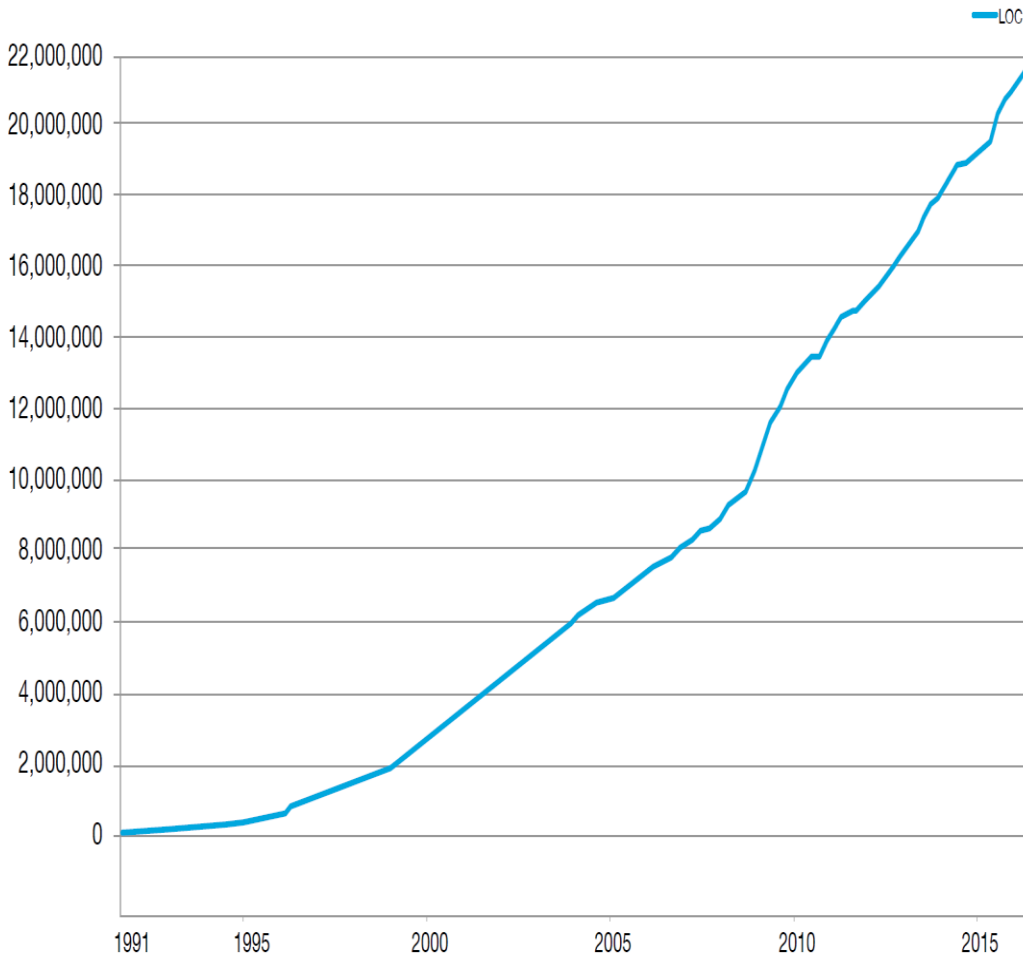
- 13,500 developers from >1,300 companies have contributed to the Linux kernel last 10 yrs
- average number of changes accepted per hour: 7.8
- >80% of kernel development is done by developers being paid for their work.

Linux Development Report 2016,
The Linux Foundation

Source: wikipedia

The Linux Kernel

Total Lines of Code in the Linux Kernel



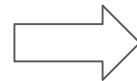
The Linux Foundation, August 2016

Linux Kernel linecount (mid-2015)

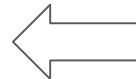
Item	Lines	%
./usr	845	0.0042
./init	5,739	0.0283
./samples	8,758	0.0432
./ipc	8,926	0.0440
./virt	10,701	0.0527
./block	37,845	0.1865
./security	74,844	0.3688
./crypto	90,327	0.4451
./scripts	91,474	0.4507
./lib	109,466	0.5394
./mm	110,035	0.5422
./firmware	129,084	0.6361
./tools	232,123	1.1438
./kernel	246,369	1.2140
./Documentation	569,944	2.8085
./include	715,349	3.5250
./sound	886,892	4.3703
./net	899,167	4.4307
./fs	1,179,220	5.8107
./arch	3,398,176	16.7449
./drivers	11,488,536	56.6110

wikipedia

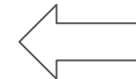
Linux Architecture - Shell



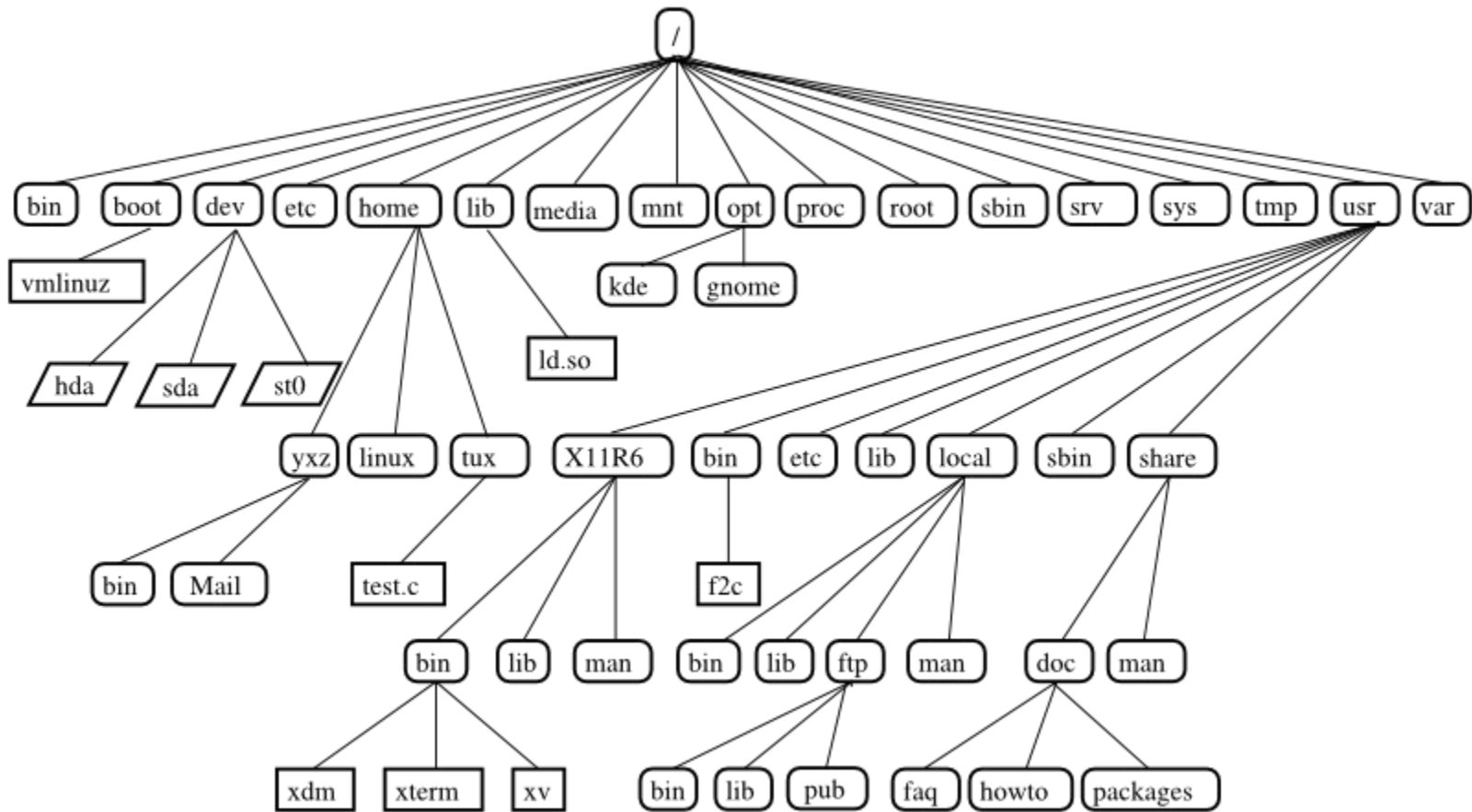
```
pavlos@mypc:~$ ls
```



```
pavlos@mypc:~$ ls  
Documents Music  
Public myfile.txt  
Downloads Pictures  
Videos
```



Linux Directory Structure



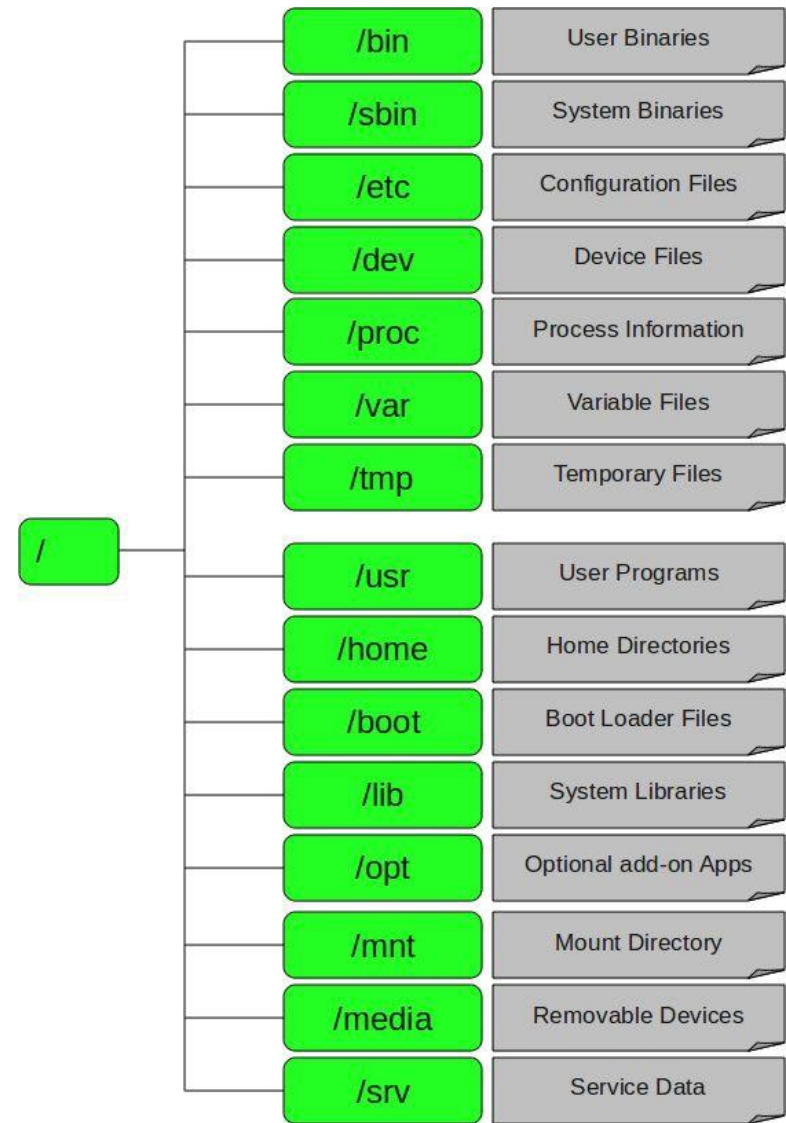
Source:

https://www.suse.com/documentation/sles10/book_sle_reference/data/sec_bash.html

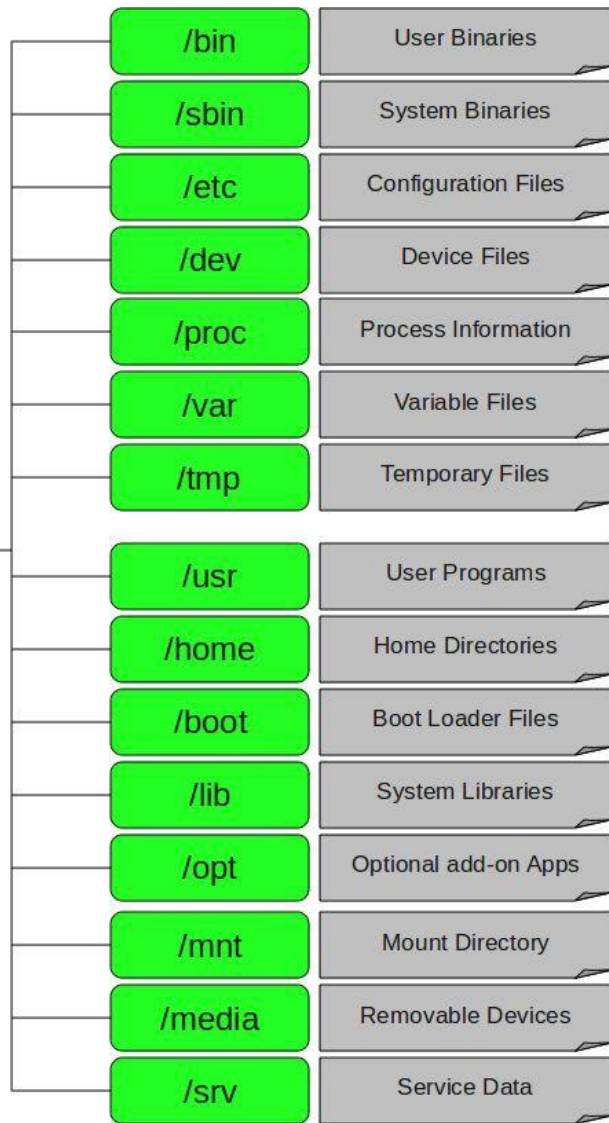
Linux Directory Structure

/ root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.



Linux Directory Structure



/bin User Binaries

- binary executables.
- Common commands for single-user modes
- Commands used by all users of the system
- example: ps, ls, ping, grep, cp.

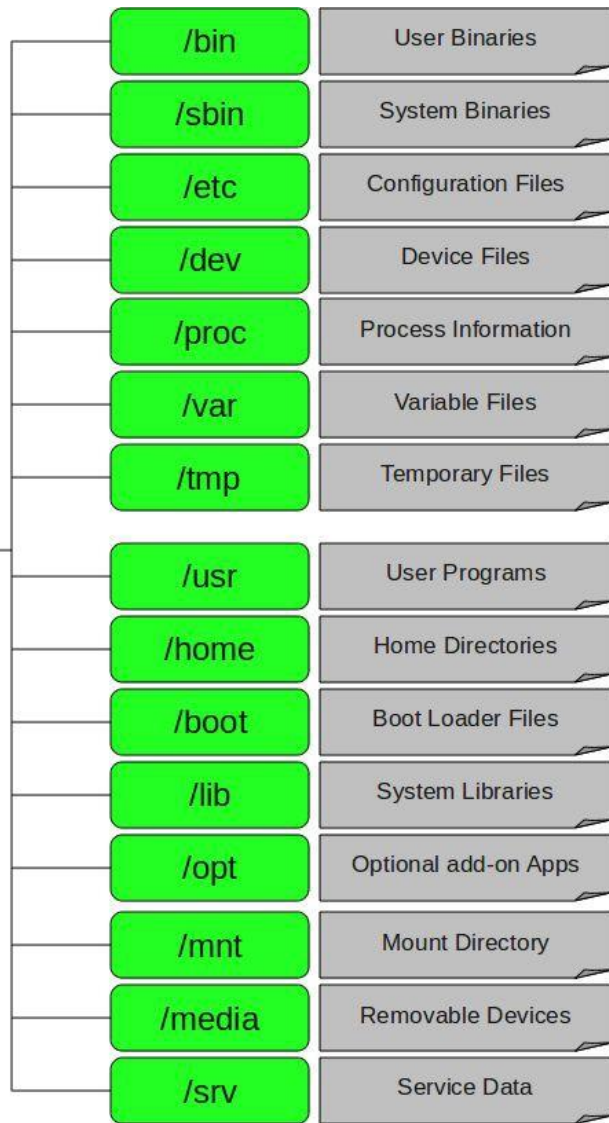
/sbin System Binaries

- also contains binary executables.
- commands typically used by system administrator, for system maintenance
- example: iptables, reboot, fdisk, ifconfig

/etc Configuration Files

- configuration files required by all programs.
- startup and shutdown shell scripts used to start/stop individual programs.

Linux Directory Structure



/dev Device Files

- device files.
- terminal devices, usb, or any device attached to the system.

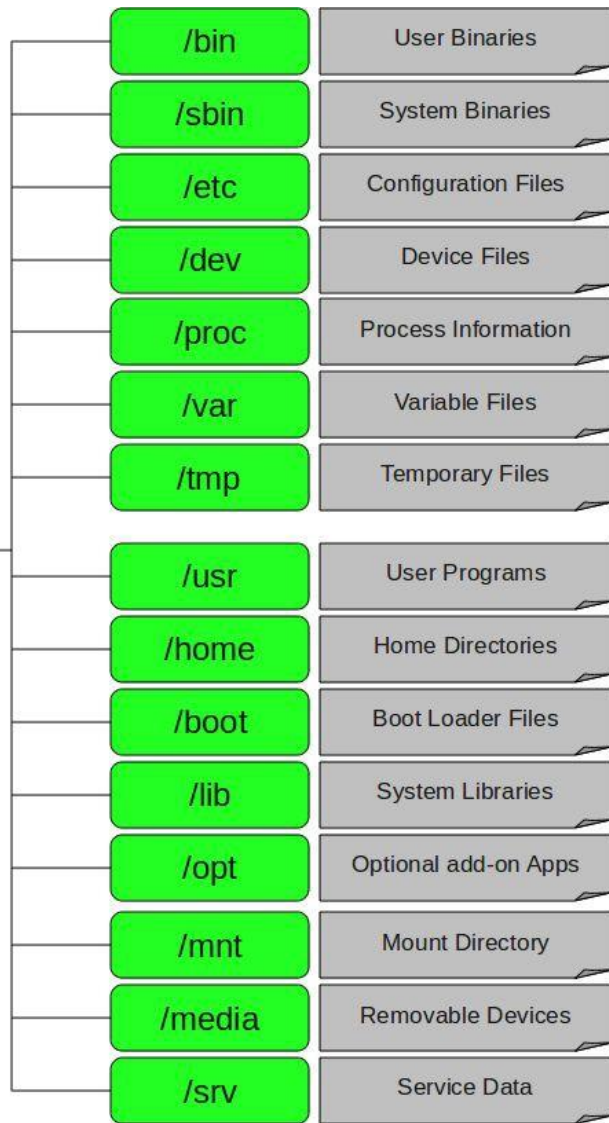
/proc Process Information

- information about running system processes.
- pseudo filesystem.
- example: `/proc/{pid}` contains information about the process with that particular pid.

/var Variable Files

- variable files - files that are expected to grow
- example: system log files (`/var/log`)
packages and database files (`/var/lib`)
emails (`/var/mail`)
(`/var/spool`)
temp files for reboots (`/var/tmp`)

Linux Directory Structure



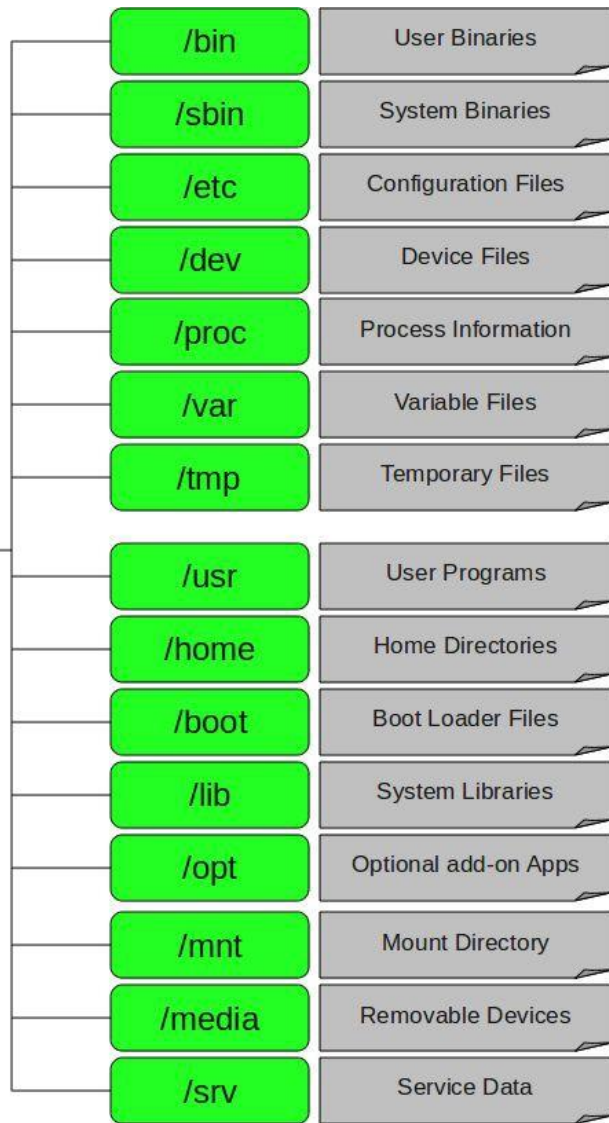
/tmp Temporary Files

- temporary files created by system and users.
- files are deleted when system is rebooted.

/usr User Programs

- usr binaries, libraries, documentation
- /usr/bin contains binary files for user programs.
example: awk, cc, less, scp
- /usr/sbin contains binary files for system administrators.
example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin & /usr/sbin
- /usr/local contains users programs that you install from source
Example: when you install apache from source, it goes under /usr/local/apache2

Linux Directory Structure



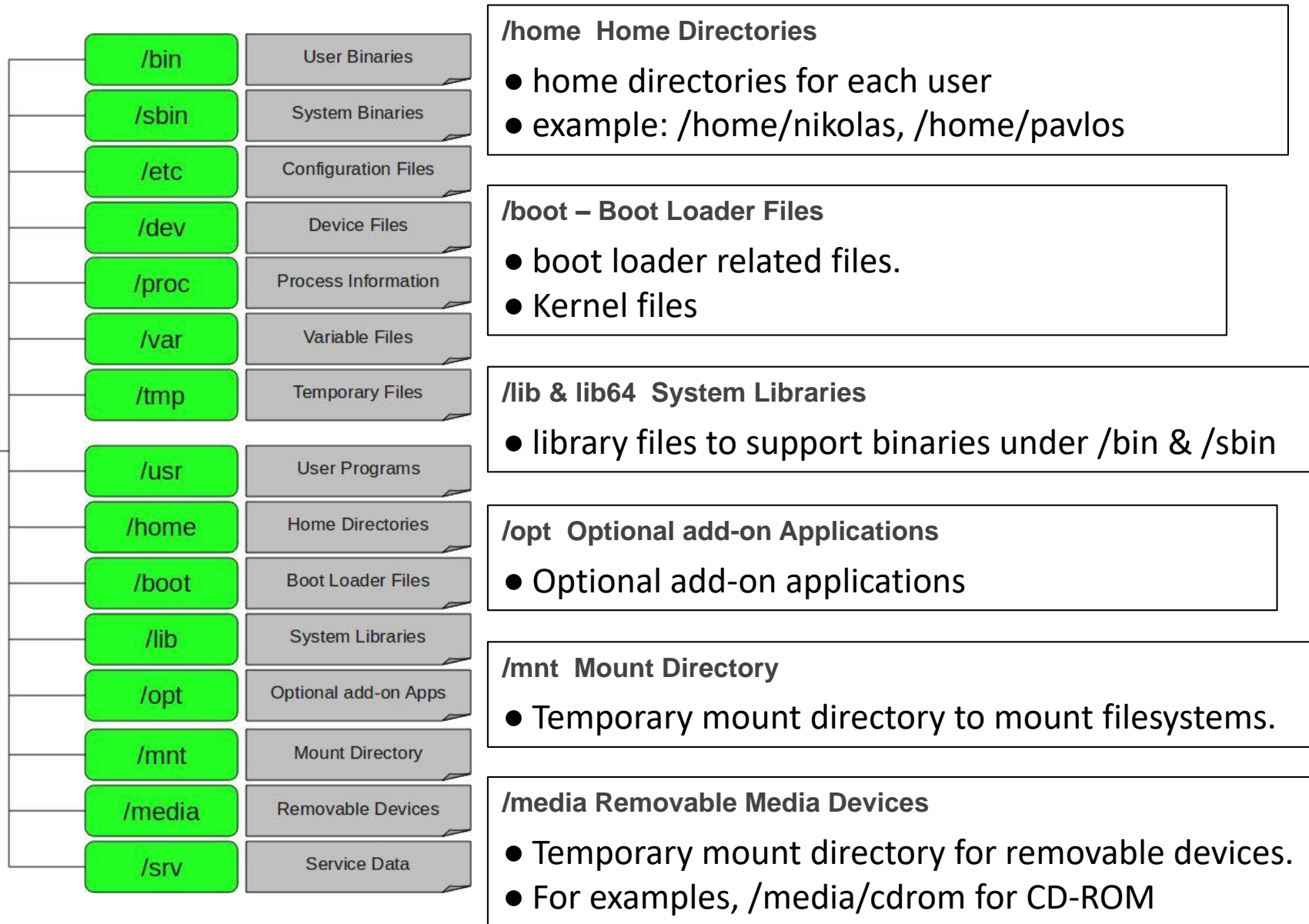
/tmp Temporary Files

- temporary files created by system and users.
- files are deleted when system is rebooted.

/usr User Programs

- usr binaries, libraries, documentation
- /usr/bin contains binary files for user programs.
example: awk, cc, less, scp
- /usr/sbin contains binary files for system administrators.
example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin & /usr/sbin
- /usr/local contains users programs that you install from source
Example: when you install apache from source, it goes under /usr/local/apache2

Linux Directory Structure



Linux, basic
commands



Basic shell commands

- Show current directory:

```
pwd
```

- List files:

- `ls`

- `ls -l` (list file details)

- `ls -lh` (list file with human readable filesize)

- `ls -a` (list all files, including those that start with .)

- Go to directory *directory_name*:

```
cd directory_name
```

- Go one directory up:

```
cd ..
```

- Go to home directory:

```
cd ~ (or just cd)
```

Linux man pages

man pages (e.g. manual pages) are concise documentation pages

e.g.: `man ls`

```
NAME
    ls - list directory
    contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by
    Sort entries alphabetically if none of -cftuvSUX nor -- is speci-
    sort fied.

    Mandatory arguments to long options are mandatory for options
    short too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        with -l, print the author of each
        file
```

Create files & directories

- Create directory:

```
mkdir
```

- Multiple ways for files:

```
>newfile
```

```
touch newfile1 newfile2
```

```
/dev/null > newfile
```


History

- **history**

displays a history of all commands

```
me@mypc:~$ history
1300 ls -l
1301 cd /home/me/data/
```

typing:

!number_of_command

```
me@mypc:~$ !1300
```

repeats the command

Some shortcuts

- `Ctrl + L`

Clears the screen. Alternatively, one can use the `clear` command

```
me@mypc:~$ clear
```

- `Ctrl + R`

Search for a previously typed command. As you type, the first matching command is displayed. Pressing `Ctrl+R` again, displays next match. `Ctrl+S` again, displays previous match

Copying, moving, deleting

- **cp** (copy)

```
cp <file_to_be_copied> <target_directory>
```

- **mv** (move file - also rename)

```
mv <file_to_be_moved> <target_directory>
```

```
mv <file_to_be_renamed> <new_name>
```

- **rm** (delete file), **rmdir** (delete directory)

Creating custom commands

- `alias <custom_name>='<command>'`

eg:

```
alias ll='ls -l'
```

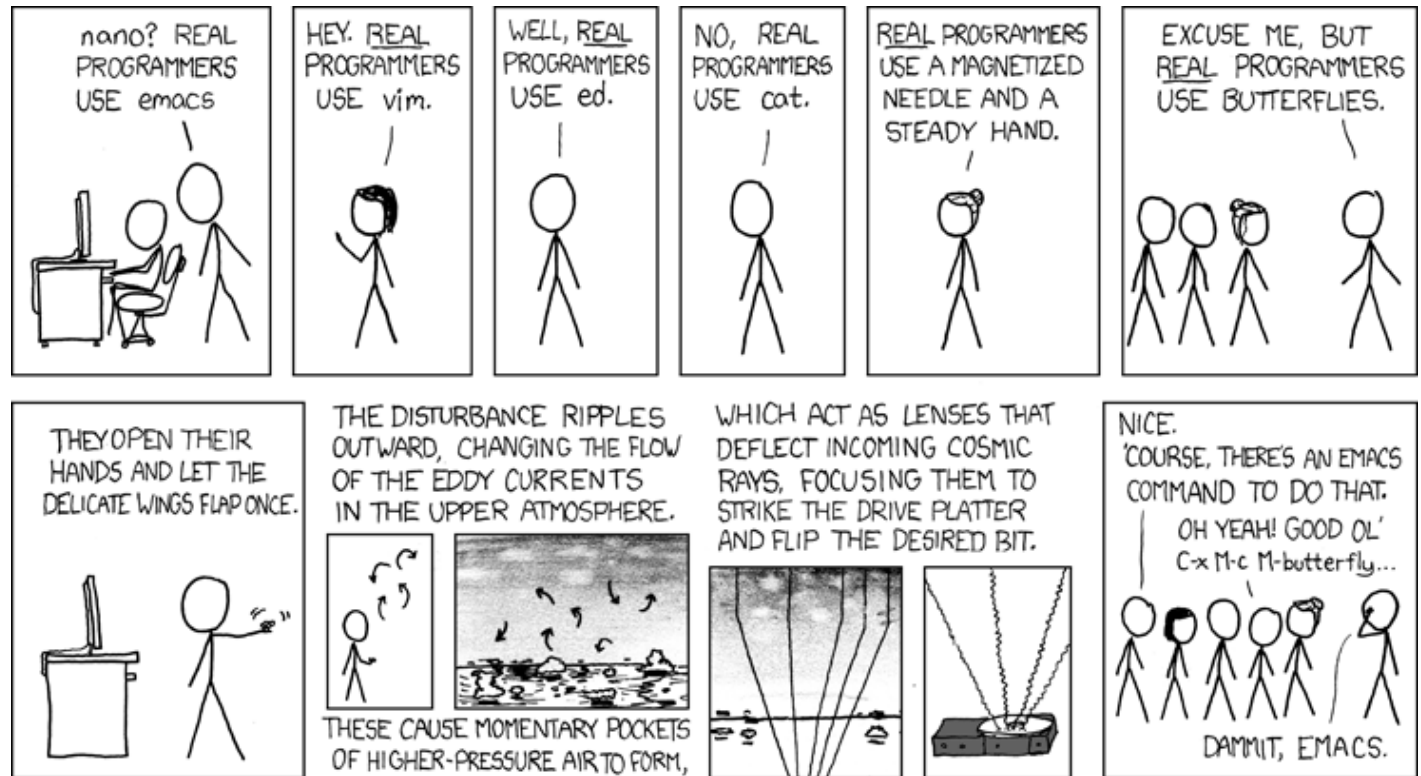
pico, nano, vi, vim, ed, emacs ...

- nano **or** nano <filename>

- help: ^G

- save: ^O

- exit: ^X



Adding commands in .bashrc

1. `nano ~/.bashrc`

2. At the end of file, add line:

```
alias ll='ls -l'
```

3. Save and exit

4. Force changes to take effect now (instead of next time shell opens)

```
source ~/.bashrc
```

Reading files

- **more** OR **cat** OR **less** (show file contents)
- **head** & **tail**: view first lines or last lines of a file

Installing stuff

- `apt-get update`
downloads the package lists from the repositories
- `apt-get install <package_name>`
downloads and installs package *package_name*
e.g.:
`apt-get install sl`
`apt-get install fortune cowsay`
- `apt-get remove <package_name>`
`apt-get purge <package_name>`
removes package and purges configuration files

Redirection: piping

- Connects Standard Input of one command with the Standard Output of another command

```
command1 | command2 | command3
```



Redirection: piping

- `ls -l | more`
- `fortune | cowsay`

Redirection: piping

- Show directory contents omitting the first line (i.e. starting from the second line)

```
ls -l | tail -n +2
```

- Show 2 lines from directory contents after omitting the first line (i.e. show lines 2 & 3)

```
ls -l | tail -n +2 | head -n 2
```

Redirection: >

- `echo hall > myfile.txt`
- `cat > myfile.txt`
line1
line2
line3
CTRL+D
- `ls -l | tail -n +2 | head -n 2 > myfile.txt`
- **Appending (adding at the end of file instead of replacing):**
use `>>` instead of `>`

tail -f

- tail -f in action

1. `ping www.google.com > mytestfile.txt &`

2. `tail -f mytestfile.txt`

3. CTRL+C to stop tail

- How do we stop (kill) the running command?

1. `ps aux | grep ping` or
`ps -u <username>`

2. Note the PID of the command

3. `kill -9 <PID>`