IT5002 Tutorial 6

AY 2025/26 Semester 1

Prepared by Michael Yang

Slides adapted from Theodore Leebrant, Prof. Colin and Prof. Aaron

1. Using Google or otherwise, research and show how bootstrapping works on Windows, LINUX or MacOS (pick ONE OS to talk about).

General Flow:

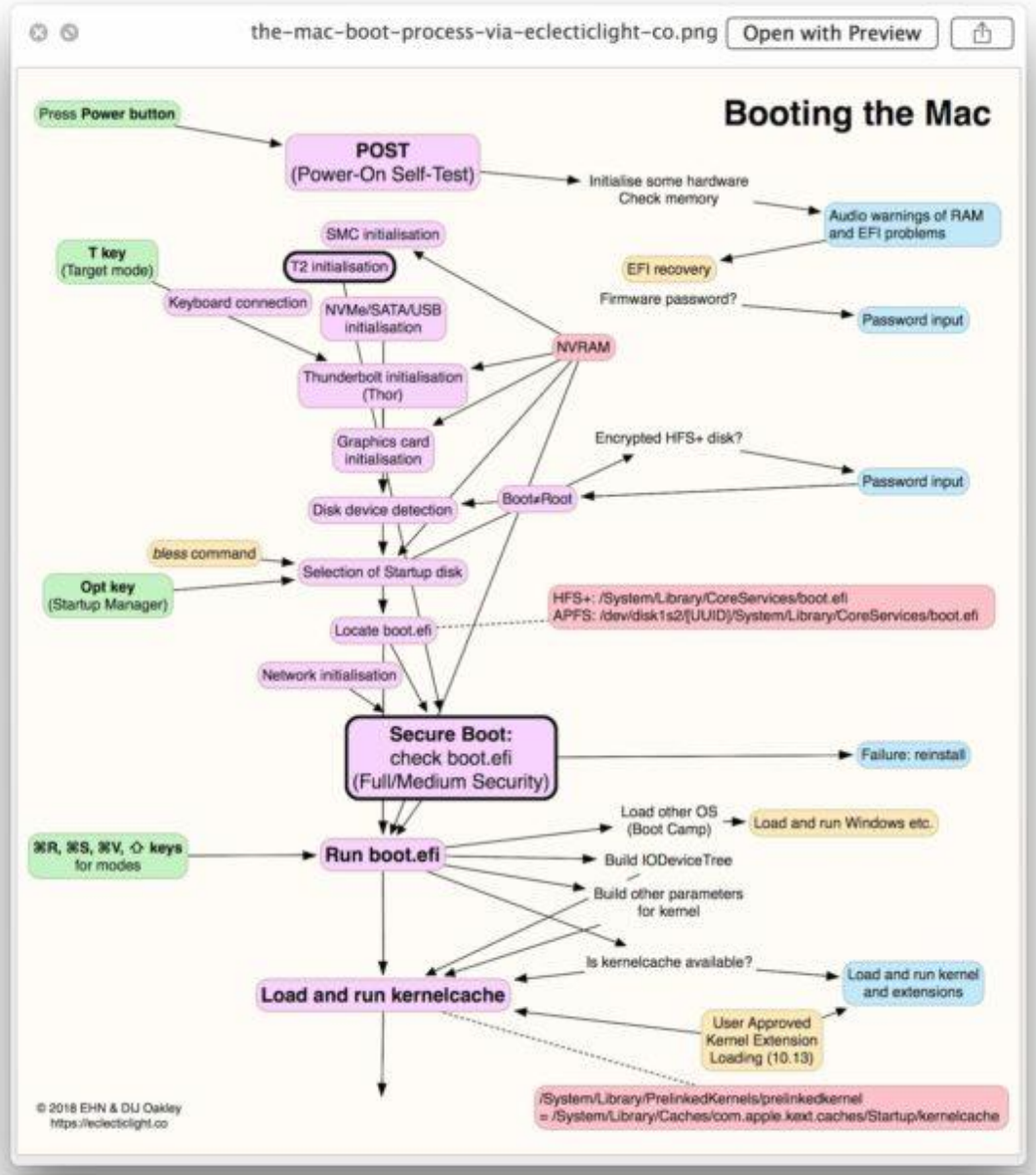
- Power on
- Execute BIOS/UEFI from Read Only Memory (ROM)
- Perform Power On Self Test (POST)
- Load bootloader
- Bootloader locates and loads kernel into RAM
- Kernel starts master process (init/systemd)
- Initialize device drivers, kernel modules, background processes
- Launch user GUI

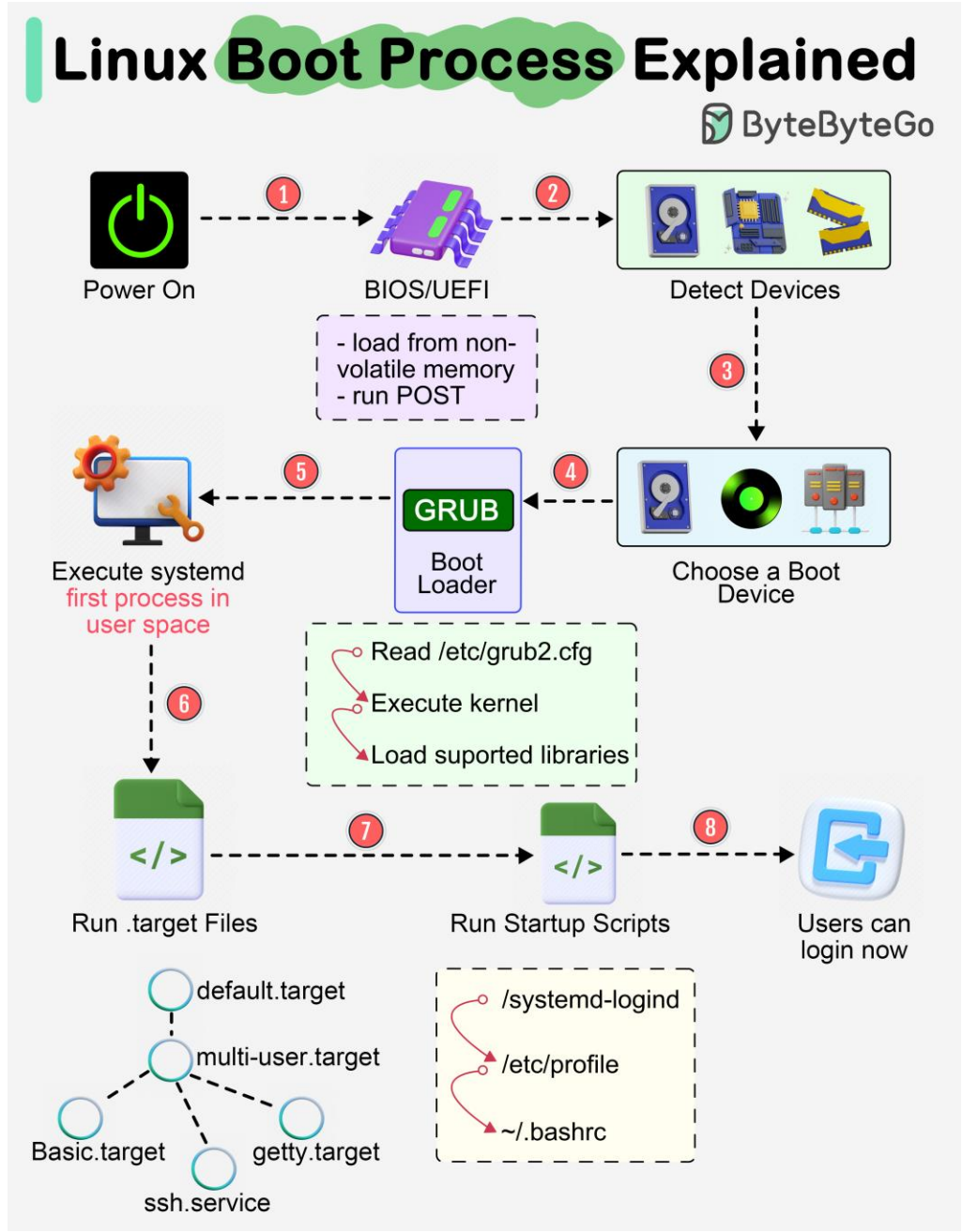
Mac Specific:

- https://support.apple.com/en-sg/guide/security/secac71d5623/web
- https://support.apple.com/en-sg/guide/security/sec5d0fab7c6/web
- https://osxdaily.com/2007/01/22/what-happens-in-the-mac-os-x-boot-process/

Linux:

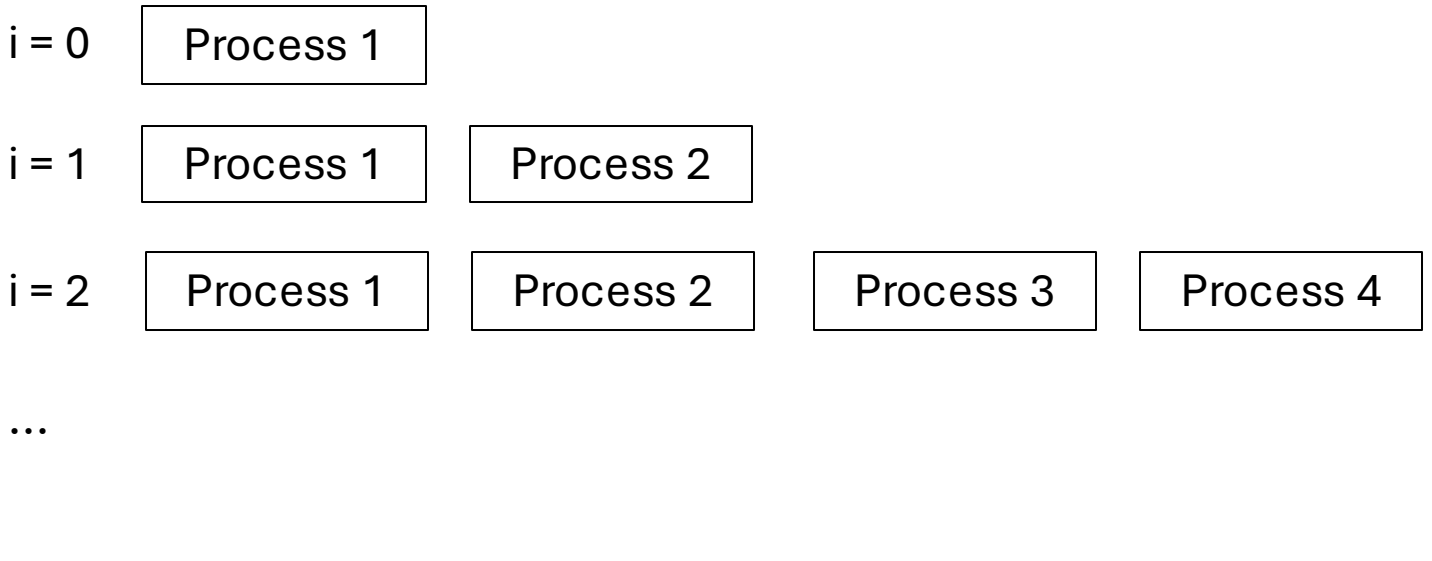
- https://www.youtube.com/watch?v=XpFsMB6FoOs


the-mac-boot-process-via-eclecticlight-co.png
Open with Preview
Booting the Mac
Press Power button
POST
(Power-On Self-Test)
Initialise some hardware
Check memory
Audio warnings of RAM and EFI problems
EFI recovery
SMC initialisation
T2 initialisation
T key
(Target mode)
Keyboard connection
NVMe/SATA/USB initialisation
Firmware password?
Password input
NVRAM
Thunderbolt initialisation (Thor)
Graphics card initialisation
Encrypted HFS+ disk?
Password input
Boot≠Root
Disk device detection
bless command
Selection of Startup disk
Opt key
(Startup Manager)
HFS+: /System/Library/CoreServices/boot.efi
APFS: /dev/disk1s2/[UUID]/System/Library/CoreServices/boot.efi
Locate boot.efi
Network initialisation
Secure Boot:
check boot.efi
(Full/Medium Security)
Failure: reinstall
⌘R, ⌘S, ⌘V, ⇧ keys
for modes
Run boot.efi
Load other OS
(Boot Camp)
Load and run Windows etc.
Build IODeviceTree
Build other parameters for kernel
Is kernelcache available?
Load and run kernel and extensions
Load and run kernelcache
User Approved Kernel Extension Loading (10.13)
/System/Library/PrelinkedKernels/prelinkedkernel
= /System/Library/Caches/com.apple.kext.caches/Startup/kernelcache
© 2018 EHN & DU Oakley
https://eclecticlight.co


Linux Boot Process Explained
ByteByteGo
Power On
1
BIOS/UEFI
2
Detect Devices
- load from non-volatile memory
- run POST
3
Choose a Boot Device
4
GRUB
Boot Loader
5
Execute systemd
first process in user space
6
Read /etc/grub2.cfg
Execute kernel
Load suported libraries
Run .target Files
7
Run Startup Scripts
8
Users can login now
default.target
multi-user.target
Basic.target
getty.target
ssh.service
/systemd-logind
/etc/profile
~/.bashrc

2. How many processes are created in the following program? Include the original process created when the program is first run.

```
for (int i=0; i<10; i++) fork();
```



3. Explain, with an example, why the highest priority task can still be interrupted by the lowest priority interrupt, and how this affects ISR design. (Hint: Interrupts are implemented in hardware in the CPU itself).

- Even if a task has a high priority, it can still be interrupted by a low-priority interrupt
- **Hardware** interrupts and **software** task priorities operate on **different levels of control**
- Interrupts are handled by the CPU's interrupt controller, not the OS scheduler. When an interrupt occurs, it **preempts** whatever the CPU was doing, even if that's the highest priority task.
- The CPU has no concept of 'task priority'; it just fetches and executes instructions. The task priority is only relevant to the task scheduler
- Interrupt lines are checked at the end of each instruction execution cycle, so they are always serviced regardless of their priority level

4. We know that to support multitasking we need to save the registers, program counter, stack pointers and machine status word (SREG in the lecture notes). What other pieces of information does the OS need to save about a process? Explain each piece.

File handles / Open File Table

- Which files are opened by the process, current location inside a file, access permissions, file open mode (read/write/append) etc.

Pending signals

- E.g. SIGTERM, SIGKILL, SIGINT

Process Running State

- Ready/Running/Suspended/Terminated etc.

Accounting Information

- How much CPU time the process has used, how much disk space, network activity

Process ID

- Unique number identifying the process

5. Given four batch jobs T1, T2, T3 and T4 with running times of 190 cycles, 300 cycles, 30 cycles and 130 cycles, find:

i) The average waiting time to run if the jobs are executed in order.

ii) The average time to run if the jobs are executed SJF.

From your answer and otherwise, explain the advantage and disadvantage of SJF. In particular, what if the number of jobs running is not fixed and new jobs can be added at any time?

FCFS

190	300	30	130

T1 Wait: 0
T2 Wait: 190
T3 Wait: 190+300=490
T4 Wait: 190+300+30=520
Avg Wait Time: (0+190+490+520)/4 = 300 cycles

SJF

30	130	190	300

T1 Wait: 30+130=160
T2 Wait: 30+130+190=350
T3 Wait: 0
T4 Wait: 30
Avg Wait Time: (160+350+0+30)/4 = 135 cycles

SJF:

- Less average wait time
- Prone to starvation if you keep submitting short tasks

Slides uploaded to https://github.com/michaelyql/IT5002

Email: e1121035@u.nus.edu

Anonymous feedback: https://bit.ly/feedback-michael

(or scan the QR below)