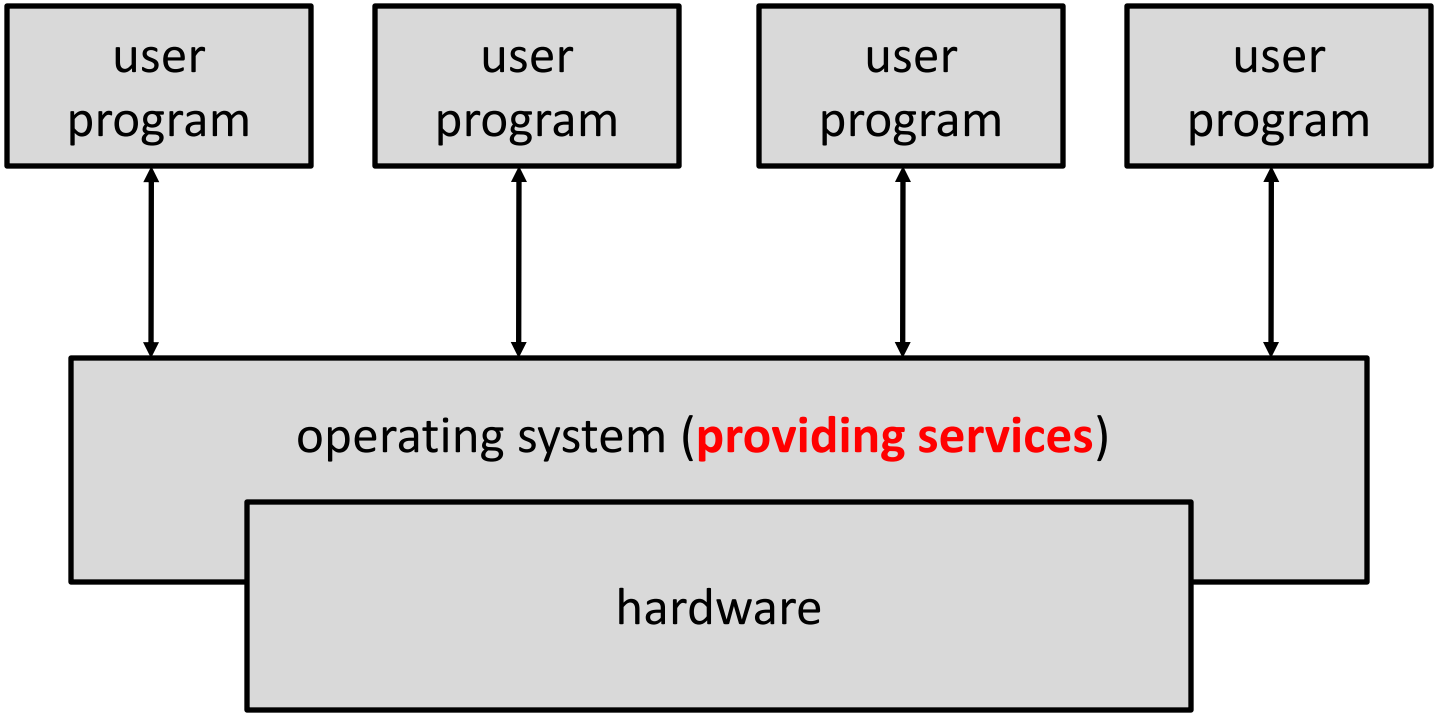
แก้ไขครั้งล่าสุดเมื่อวันที่ 21 กันยายน 2567

Operating systems (OS)



This can be done by trap routine.

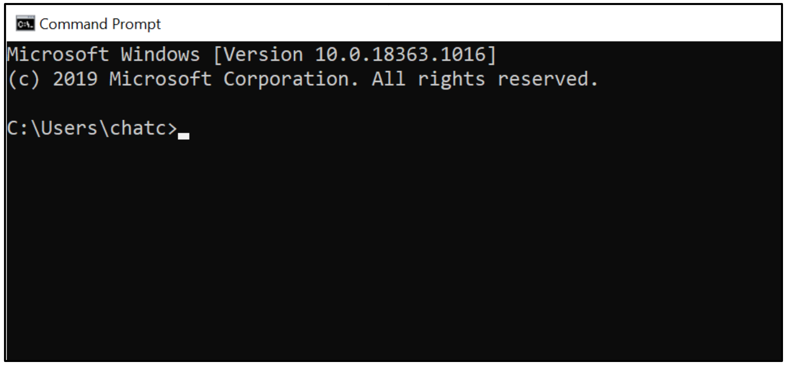
Operating-system services

1. User interface
2. Program execution
3. I/O operations
4. File-system manipulation
5. Communication (เช่น copy & paste, drag & drop หรือสื่อสารข้ามเครื่อง)
6. Error detection
7. Resource allocation
8. Accounting (ระบบ user / group)
9. Protection and security (privilege / hacker)

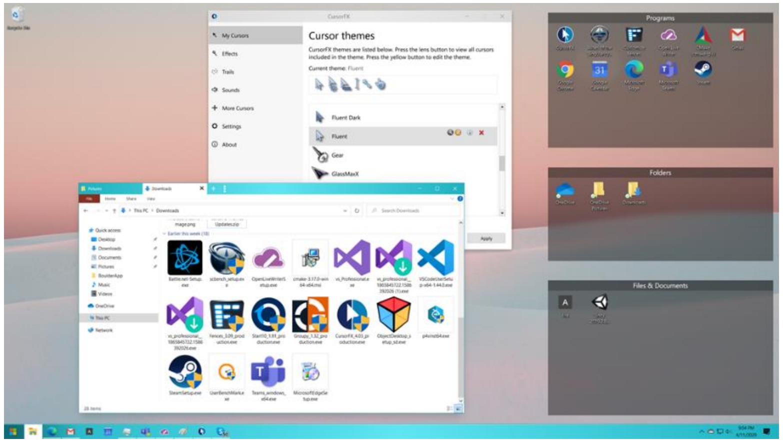
1. User interface

- **Command-line interface (CLI)**
  - DOS
  - Unix / Linux

- **Graphical user interface (GUI)**
  - Windows
  - Mac OS
  - Unix / Linux
    - X-Windows systems
    - K Desktop Environment (KDE)
    - GNOME desktop



```
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\chatc>_
```



2. Program execution

The program must be able to end its execution, either normally or abnormally (indicating error).

Examples

- **hardware resource running low (CPU, memory, and battery).**
- **I/O error (fail to read or write I/O devices, parity error).**
- **a connection failure on a network.**
- **a lack of papers in the printer.**
- **arithmetic overflow.**
- **dangling pointer (read/write a memory block with no ownership).**

Genuine OS is supposed to be responsive and handle common errors.

3. I/O operations

For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.

Efficiency

- OS services are written by professionals (and optimized for speed).
- User programs are small (เรียกใช้ device driver).
- User programs are portable (ย้ายไปใช้ hardware ยี่ห้อ/รุ่นอื่น ๆ ได้).
- OS can manage concurrency access and cache.

Device drivers

Protection

- Users must have permission to access I/O devices.
- User programs can deteriorate some I/O devices.

4. File-system manipulation

File-system services are create/delete /rename/search files and folders. The rationale for file-system services is similar to that of I/O operations.

5. Communications

There are many circumstances in which one process needs to exchange information with another process เช่น copy & paste, drag & drop

Example

A running database program has to exchange information with several programs that may be on the same computer or may be not.

6. Error detection

See example in Program execution.

7. Resource allocation

When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

Resources

- CPU
- Memory
- File storage
- Printer & modem
- Other peripheral devices

8. Accounting

We want to keep track of which users how much and what kinds of computer resources. This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics. Usage statistics may be a valuable tool for researchers who wish to reconfigure the system to improve computing service.

9. Protection and security

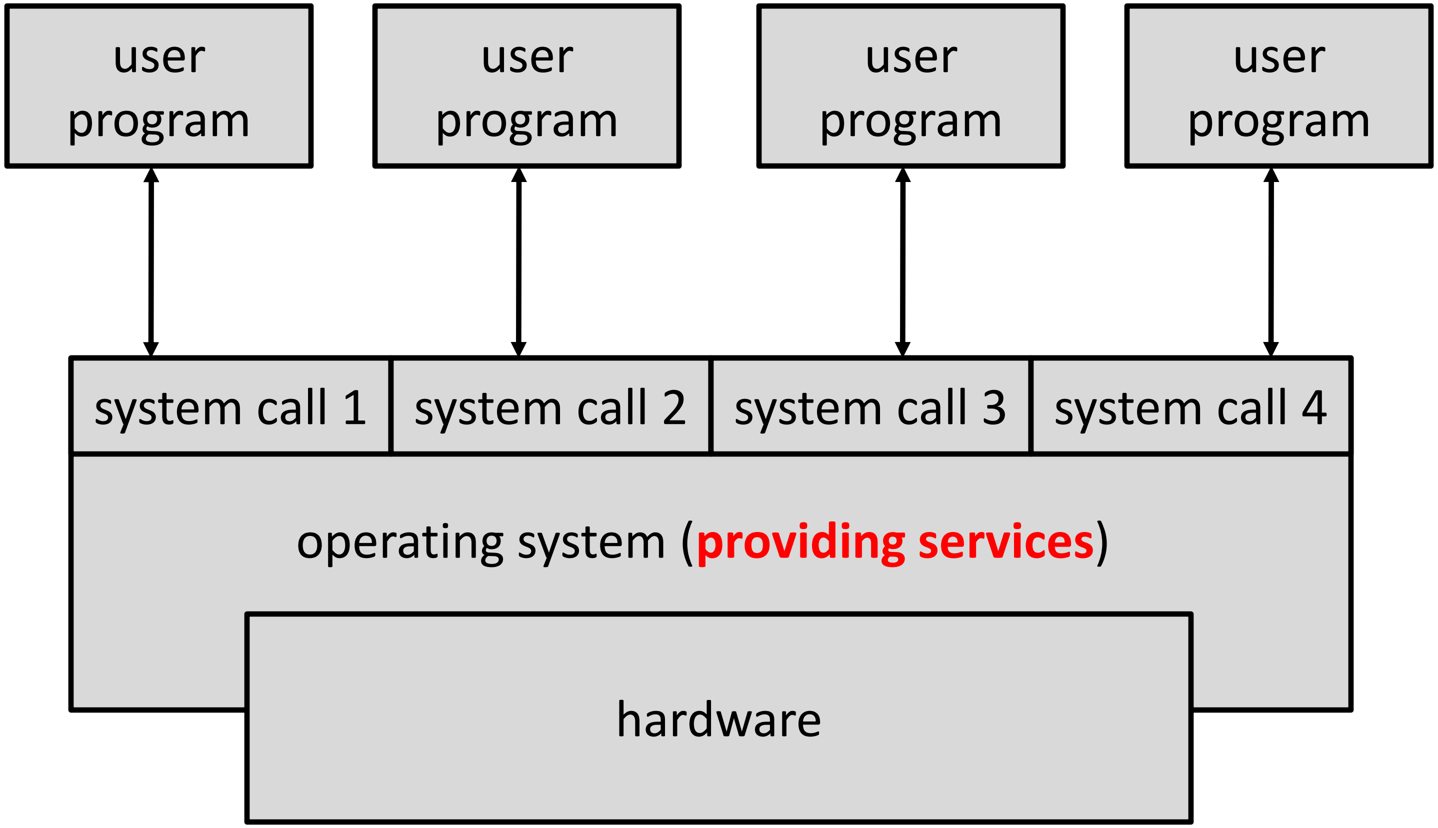
Protection involves ensuring that all access to system resources is controlled. It should not be possible for one process to interfere with the others or with the operating system itself. เช่น memory protection

Security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access the system resources. It extended to defending external I/O devices, including modems and network adapters, from invalid access attempts and to recording all such connections for detection of break-ins.

Authentication	การพิสูจน์ตัวจริง
Authorization	การอนุญาต, การให้อำนาจ

System calls (or trap routine)

User programs cannot control resources directly.



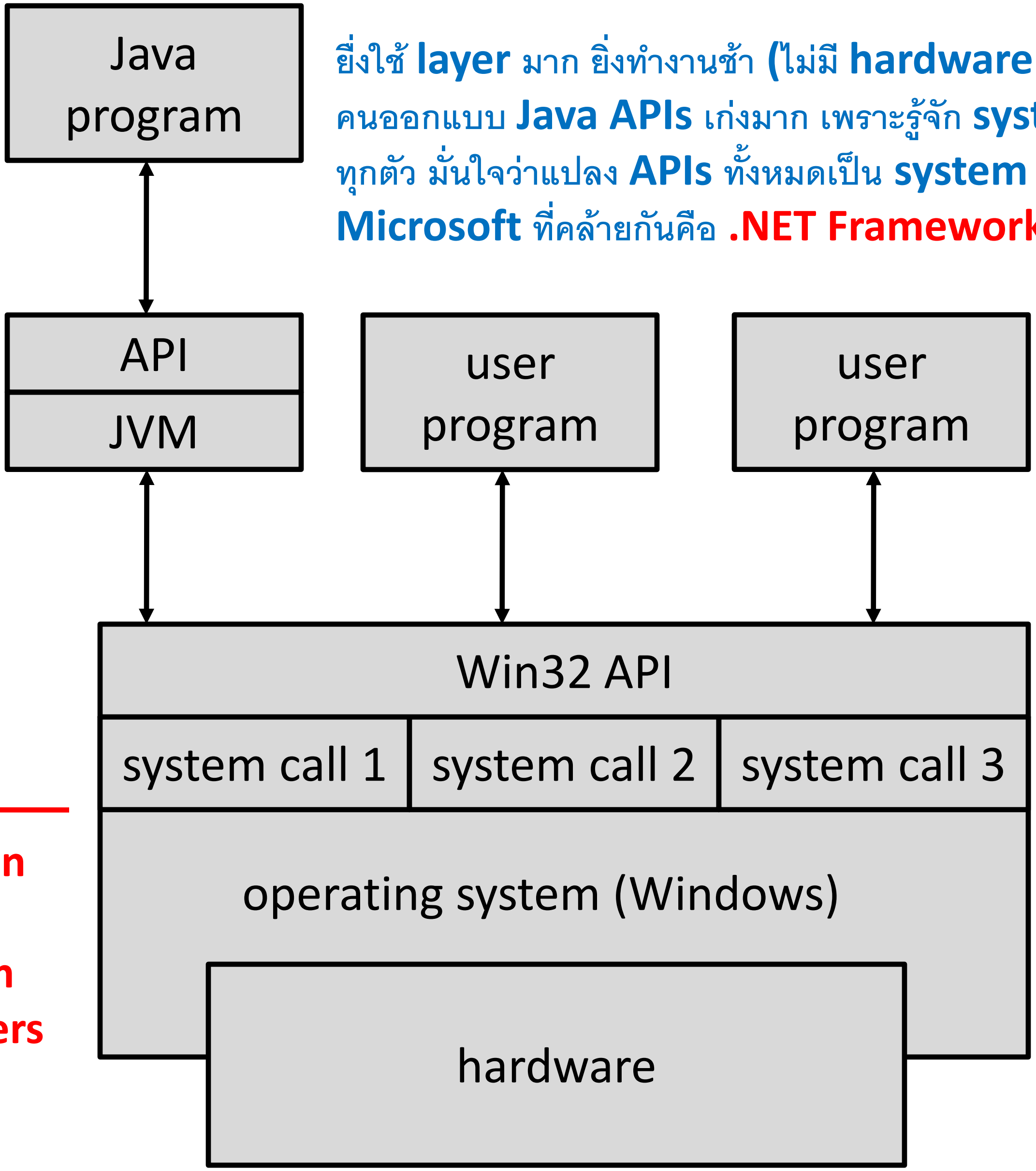
```
> myprog.exe
> Enter source file: myfile1.dat
> Enter destination file: myfile2.dat
```

An example: copying a file

1. Read the first argument	
Print to screen	system call
Read keyboard	system call
2. Read the second argument	
Print to screen	system call
Read keyboard	system call
3. Open the input file	system call
Error: file is not found	
4. Create output file	system call
Error: file exists	
5. Loop	
Read	system call
Error: hardware failure (parity error)	
Write	system call
Error: no more disk space	
1. Close input file	system call
2. Close output file	system call

System call ≠ Application programming interface (API)

ยิ่งใช้ **layer** มาก ยิ่งทำงานช้า **(ไม่มี hardware ที่ execute byte code)** คนออกแบบ **Java APIs** เก่งมาก เพราะรู้จัก **system calls** ของ **OS** แทบจะทุกตัว มั่นใจว่าแปลง **APIs** ทั้งหมดเป็น **system calls** ได้ เทคโนโลยีของ **Microsoft** ที่คล้ายกันคือ **.NET Framework** และ **.NET Core**



Abstraction layer for Java programmers

Abstraction layer for system programmers

Abstraction layer for Windows programmers

สำหรับงานที่ต้องการ **performance** และ **compatibility** เช่น เกม

An example of Win32 API

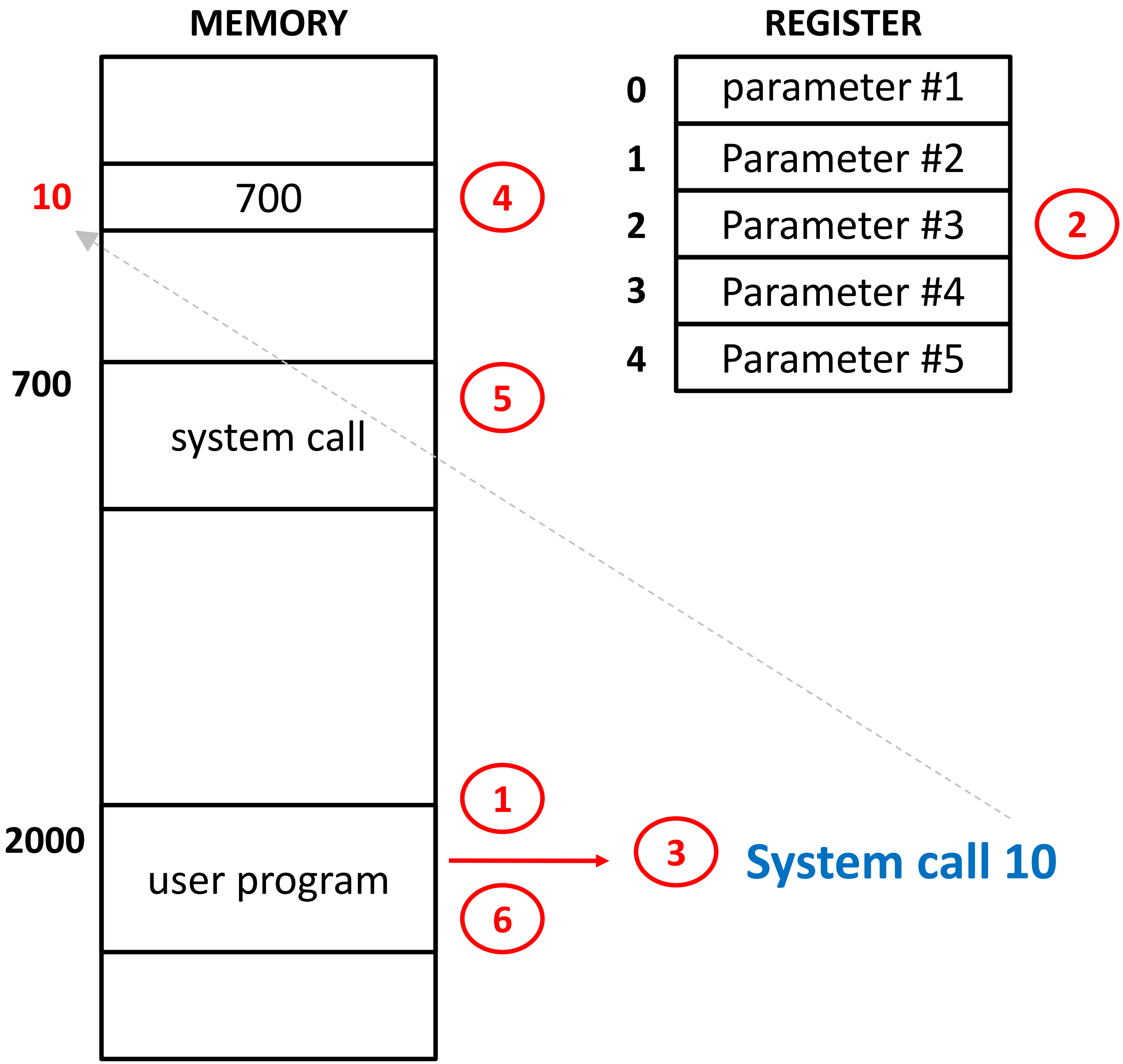
```
BOOL WINAPI ReadFile(
        __in HANDLE hFile,
        __out LPVOID lpBuffer,
        __in DWORD nNumberOfBytesToRead,
        __out_opt LPDWORD lpNumberOfBytesRead,
        __inout_opt LPOVERLAPPED lpOverlapped
);
```

Ref: http://msdn.microsoft.com/en-us/library/aa365467(VS.85).aspx

Inside ReadFile() involves several system calls which is specific to OS.

The executable code compiled using Win32 API are compatible with Windows OS (Windows 98, Windows 2000, Windows XP, etc).

System call: passing of parameters as registers (LC3 processor)



MEMORY

10

700

4

700

system call

5

2000

user program

1

3

6

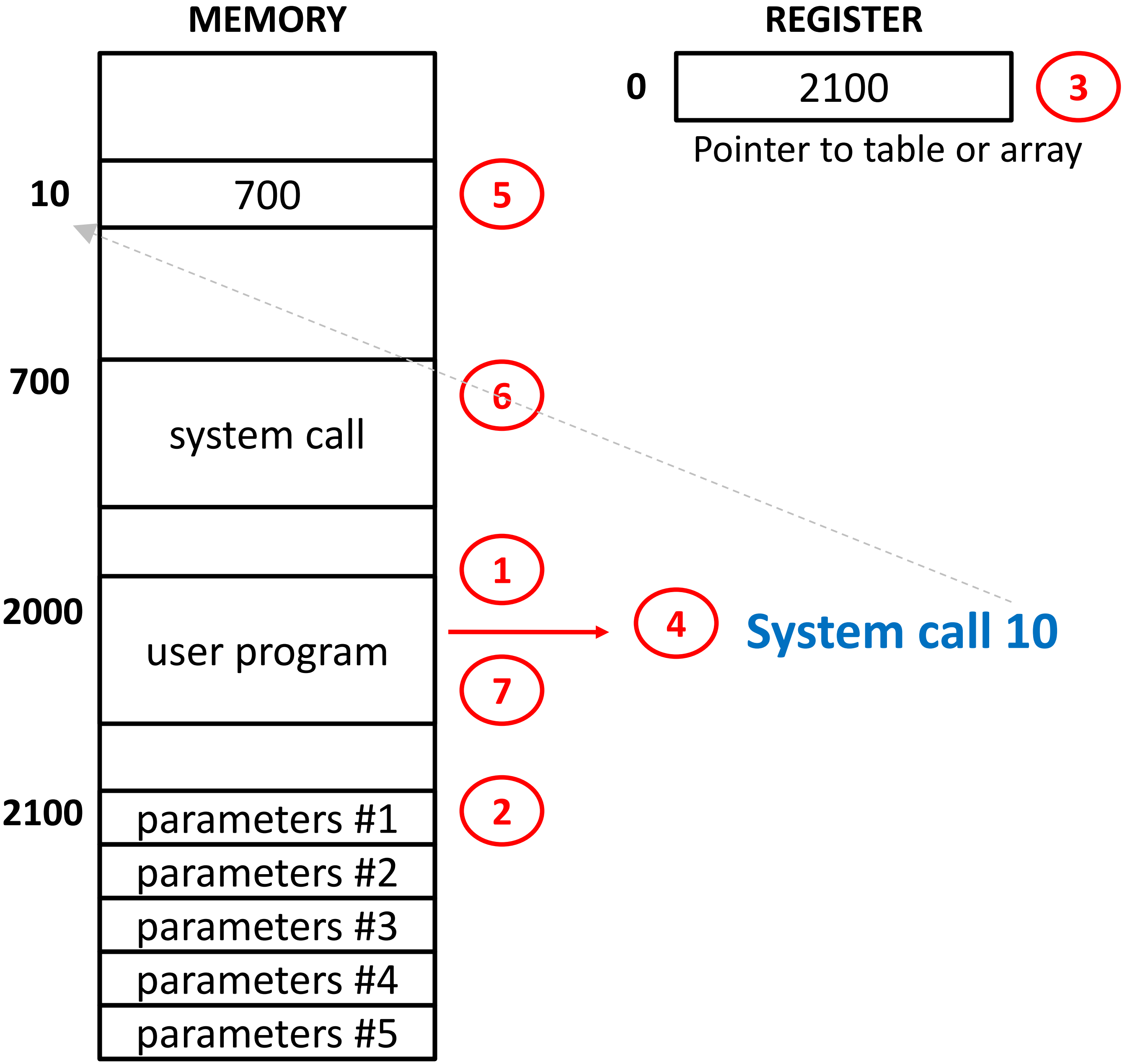
System call 10

REGISTER

0	parameter #1
1	Parameter #2
2	Parameter #3
3	Parameter #4
4	Parameter #5

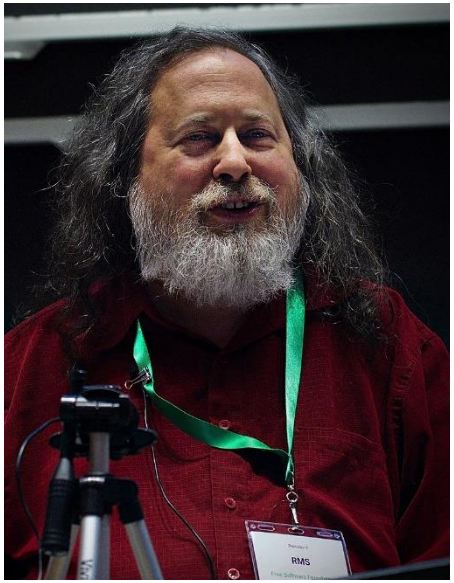
2

System call: passing of parameters as a table (LC3 processor)



Types of system calls

1. **Process control**
   end, abort, load, execute, create/terminate process,
   wait for time, wait event, signal event, allocate and free memory,
   get/set process attributes, lock & release
2. **File management**
   create file, delete file, open, close, read, write, reposition,
   get/set file attributes
3. **Device management**
   request device, release device, read, write reposition,
   get/set device attributes, logically attach/detach devices
4. **Information maintenance**
   get/set time/date, get/set system data, dump, single step
   get/set process, file, or device attributes
5. **Communications**
   create, delete communication connection,
   send/receive messages, transfer status information,
   attach/detach remote devices, message-passing, shared-memory model
6. **Protection**
   set/get permission, multi-user to networking environment



Richard Stallman
ให้กำเนิด Free Software
Foundation (FSF)

GNU Project
กนู

Linus Torvalds
ลีนึส ตูร์วัลดส์

Compiler / Debugger / Editor

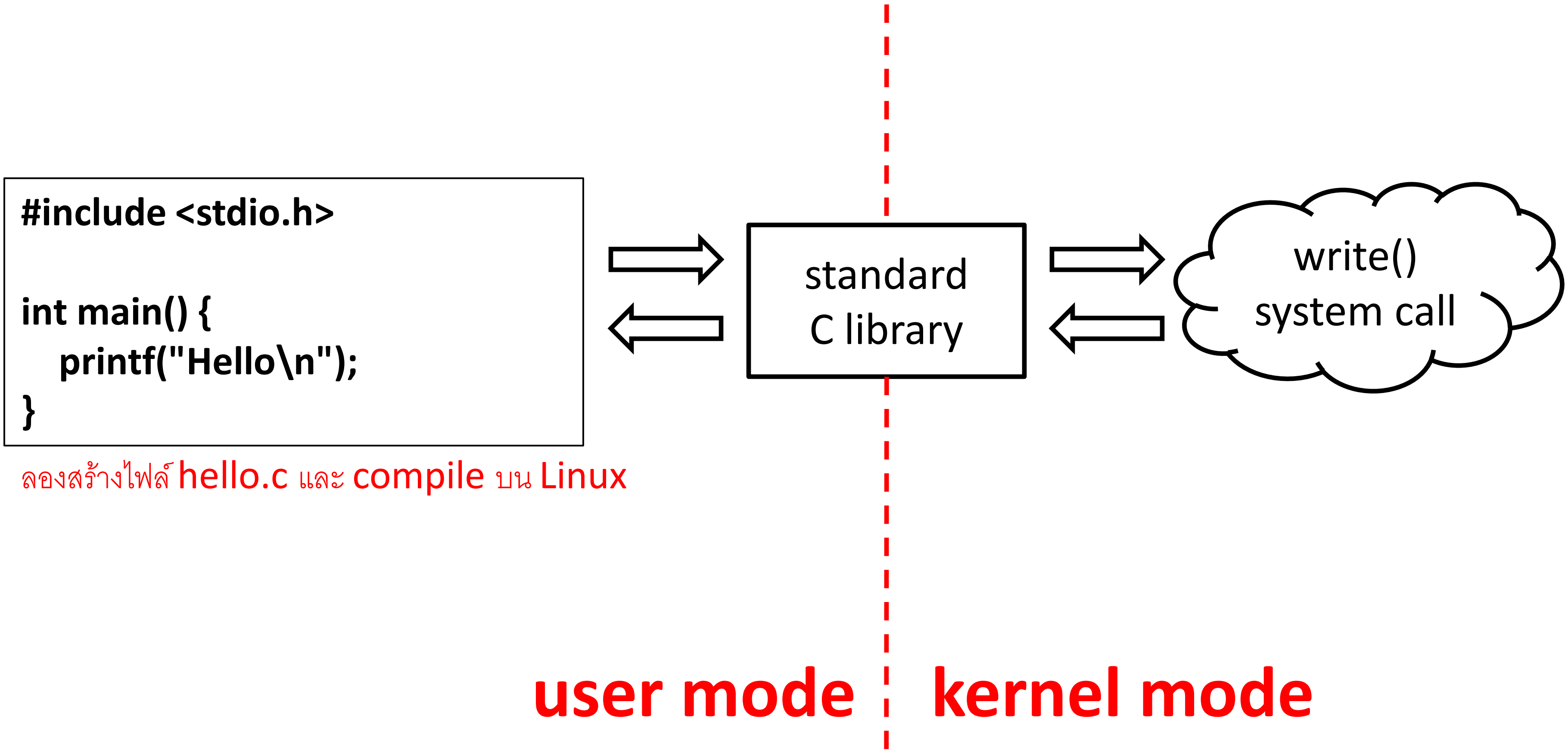
รันบน Unix ที่มีอยู่ในขณะนั้น บริษัทต่าง ๆ มี Unix ของตัวเอง

Linux kernel

เกิด Linux distribution ต่าง ๆ ขึ้นมากมาย

GNU = GNU's Not Unix

Standard C library



```
#include <stdio.h>

int main() {
   printf("Hello\n");
}
```

ลองสร้างไฟล์ hello.c และ compile บน Linux

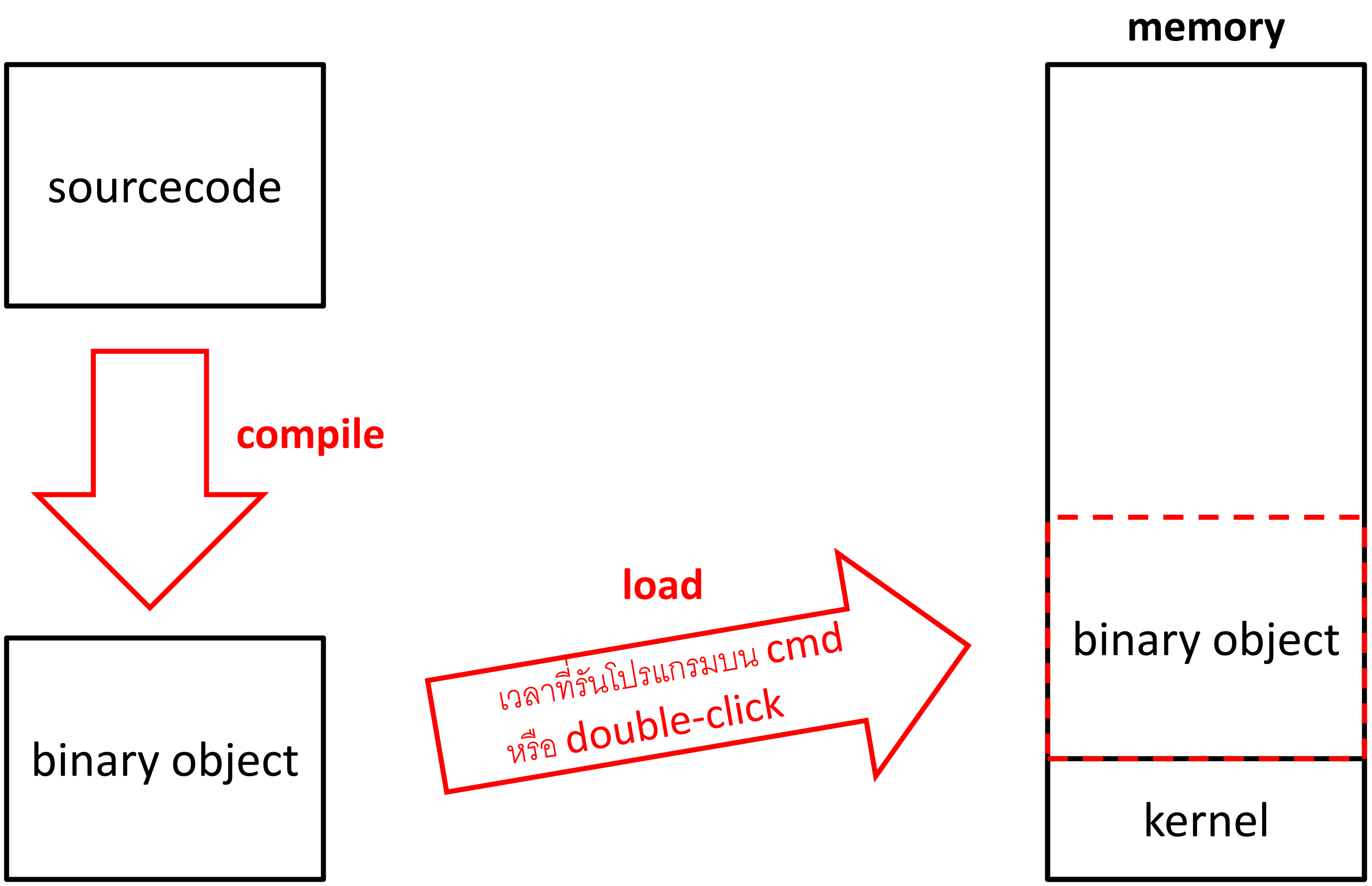
standard
C library

write()
system call

user mode

kernel mode

Loaders



memory

sourcecode

compile

binary object

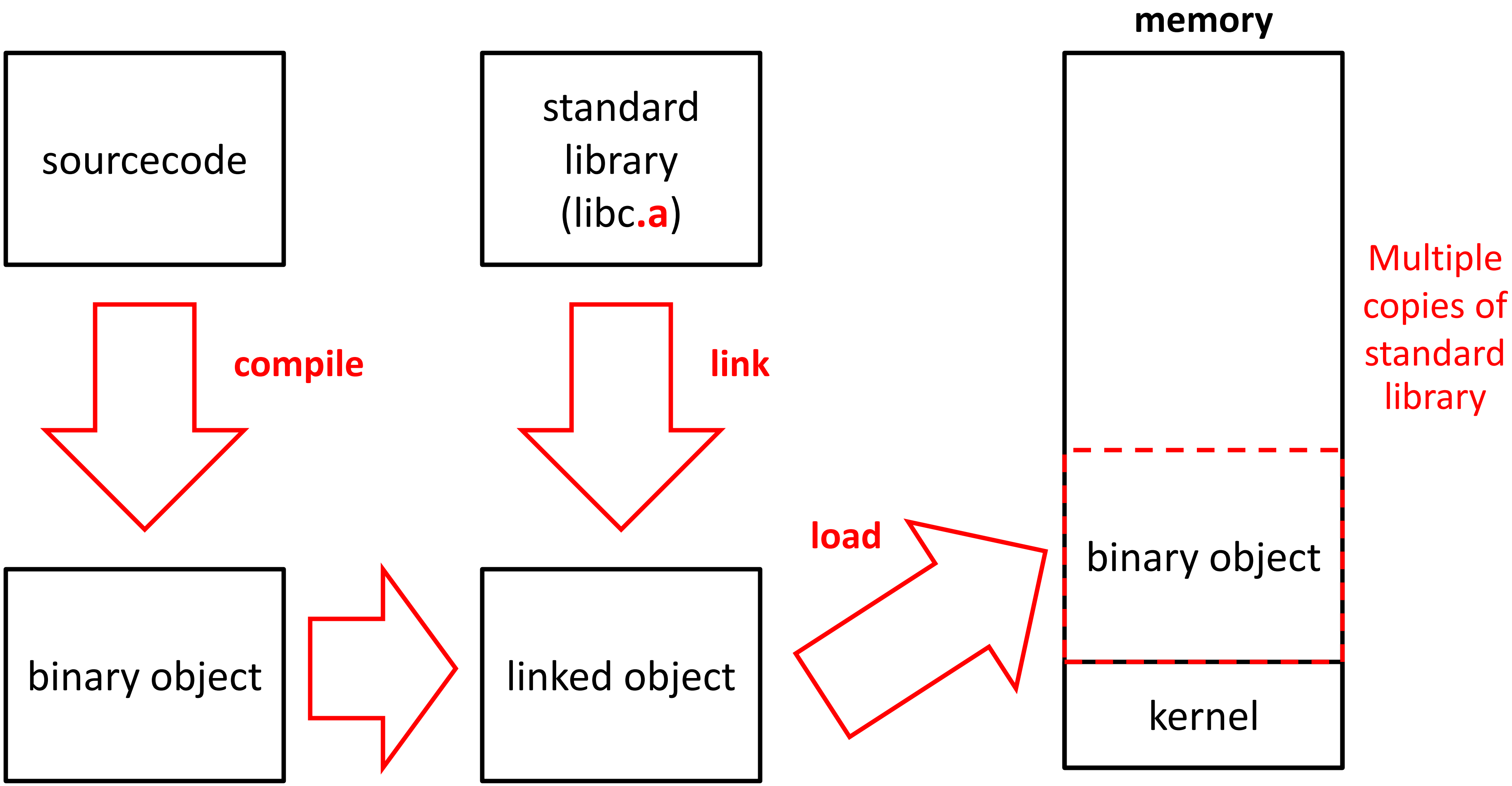
load

เวลาที่รันโปรแกรมบน cmd
หรือ double-click

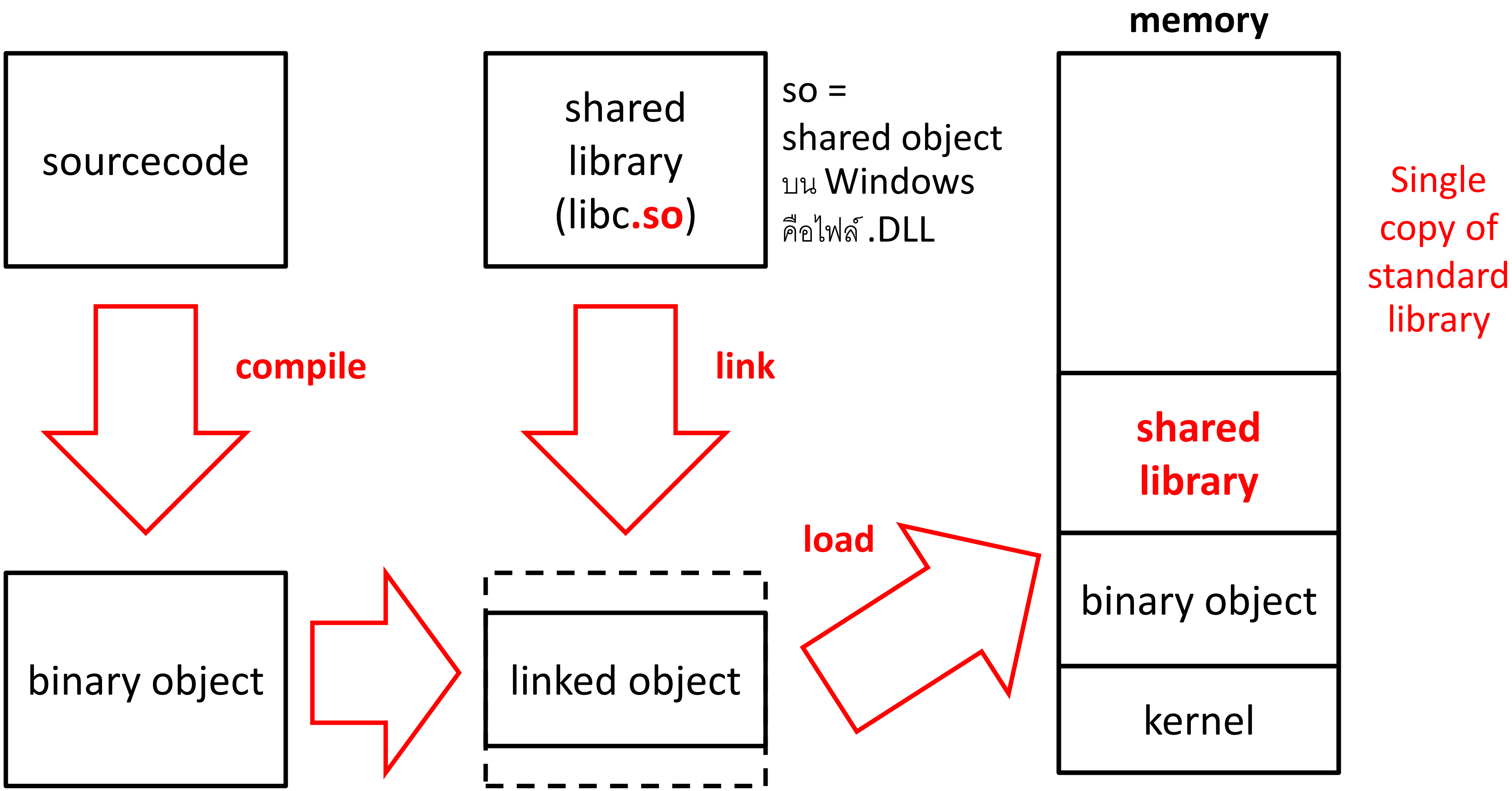
binary object

kernel

Linkers (static)

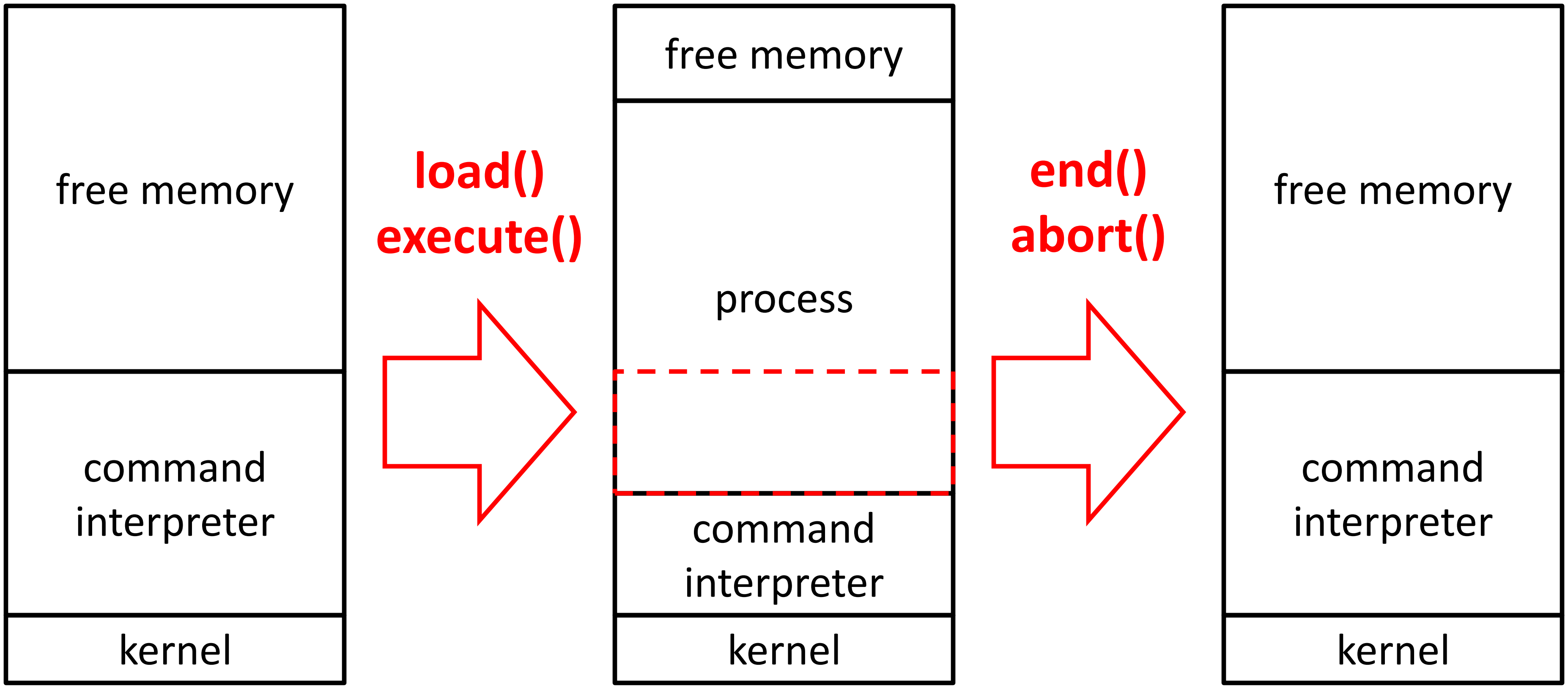


Linkers (dynamic)



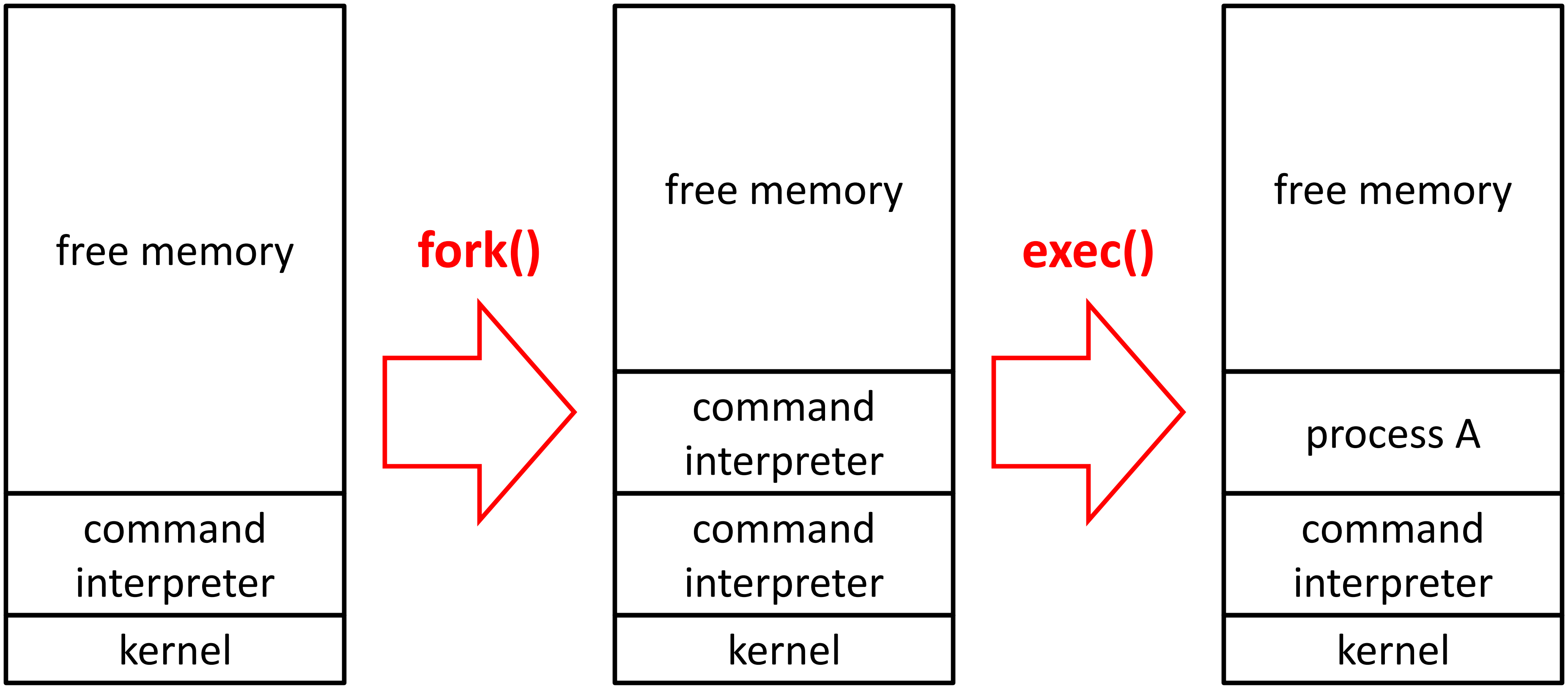
so = shared object บน Windows คือไฟล์ .DLL

Single copy of standard library



MS-DOS

A portion of command interpreter is unloaded from memory.



Free BSD

fork() and exec() are system calls for creating a process.

System programs

1. **File management**
   create, delete, copy, rename, etc.

2. **Status information**
   date/time, cpu/memory/disk usage

3. **File modification**
   text editors (nano, notepad)

4. **Programming-language support**
   compilers, assemblers, and debuggers (gdb)

5. **Program loading and execution**
   Linkers and loaders, and debuggers

6. **Communications**
   connection among processes, users, computer systems
   browse web pages, send e-mail, remote login, transfer file

Operating-system design & implementation

1. **Design Goals**

   User

   convenient to use, easy to learn and to use, reliable, safe, fast

   System

   easy to design, implement, and maintain, flexible, reliable, error free

   (server, desktop, real-time, embedded systems)

2. **Mechanisms vs. Policies**

   Mechanism determines <u>how to do something</u>

   Policy determines <u>what to be done</u>

   Mechanism - Policy อื่น ๆ เช่น

   Password หรือ Fingerprint - Authentication

   X-window - K Desktop Environment (KDE) หรือ GNOME

   Cooperative/Preemptive (Timer Interrupt) - CPU scheduler แบบ Time Sharing

3. **Implementation**

   Assembly (MS-DOS), C/C++ (Unix/Linux)

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

https://stackoverflow.com/questions/4784500/policy-and-mechanism

Operating-system structure

1. Simple structure
   MS-DOS, Unix (monolithic structure), see Figure 2.13

2. Layer approach
   pros and cons, see Figure 2.14

3. Microkernels
   Mach (mid-1980), see Figure 2.15
   Remove all non-essential components from the kernel
   Microkernels provides minimal process, memory management, and communication facility (message passing).
   Performance decrease (first-release Windows NT)

4. Modules
   Core kernel + loadable kernels
   Any module can call any other module (no message passing).
   Apple Mac OS X = Mach + BSD (see Figure 2.16)

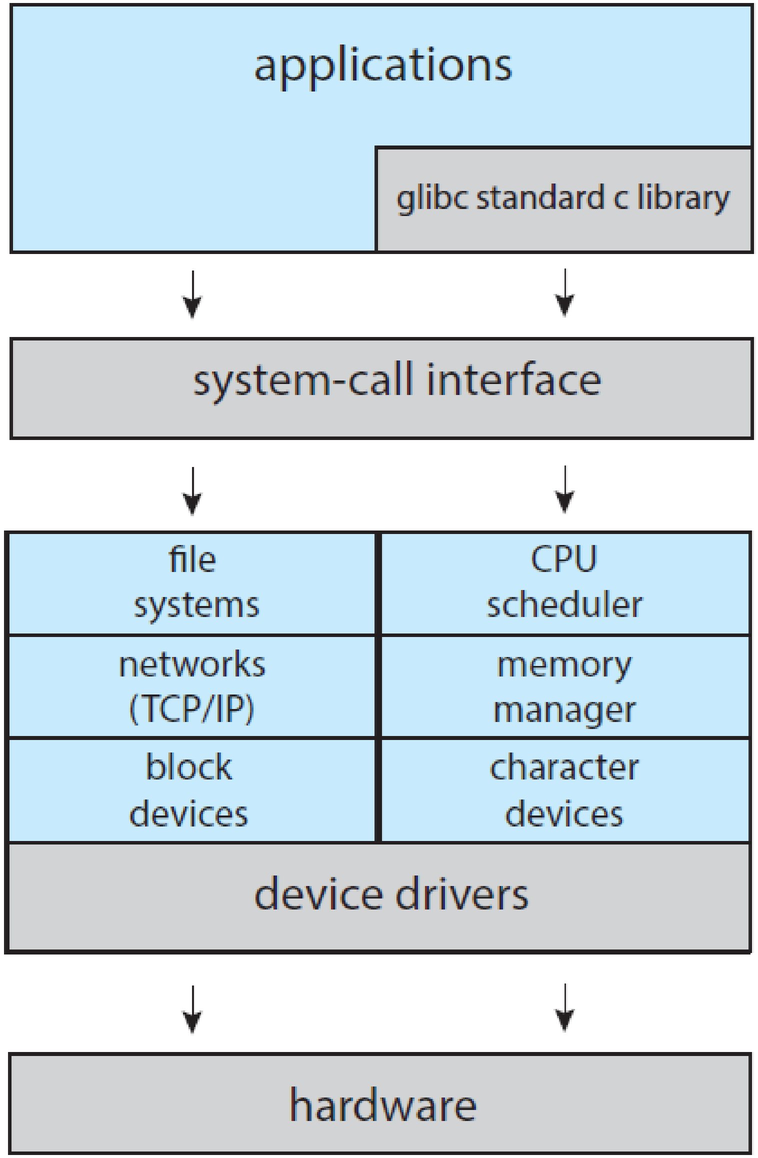


Figure 2.13 Linux system structure.

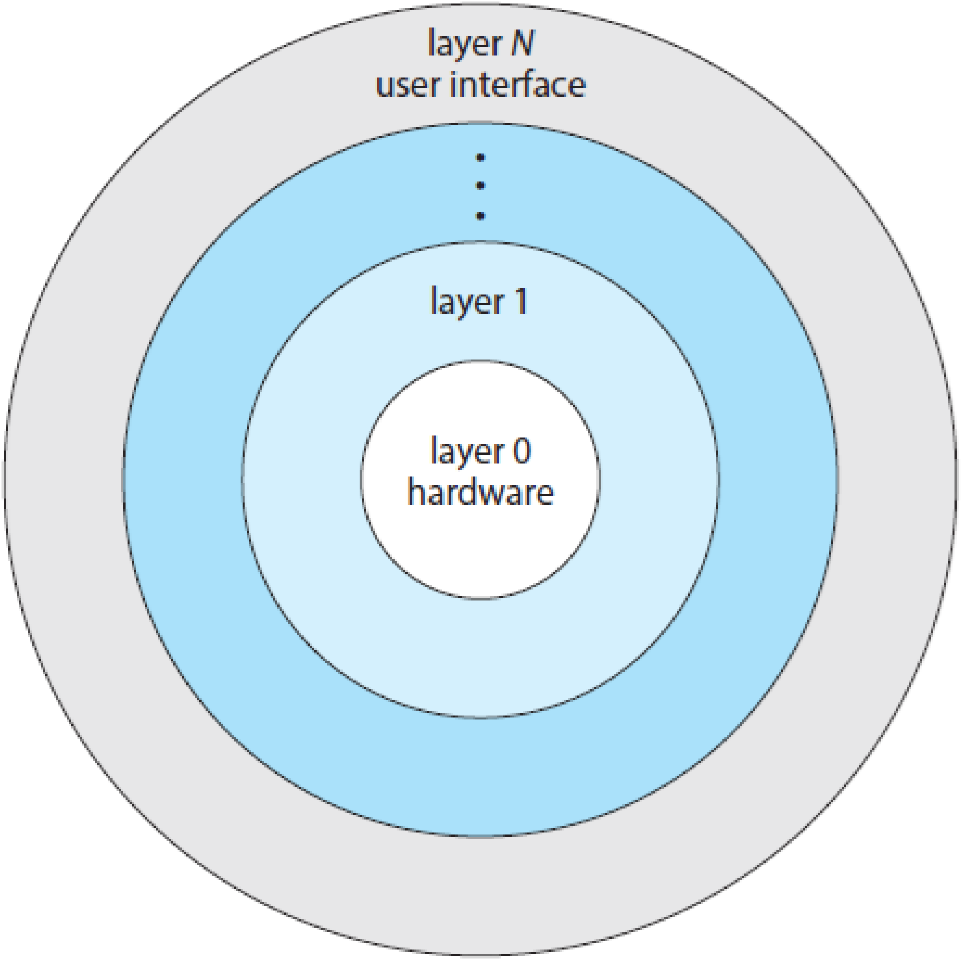
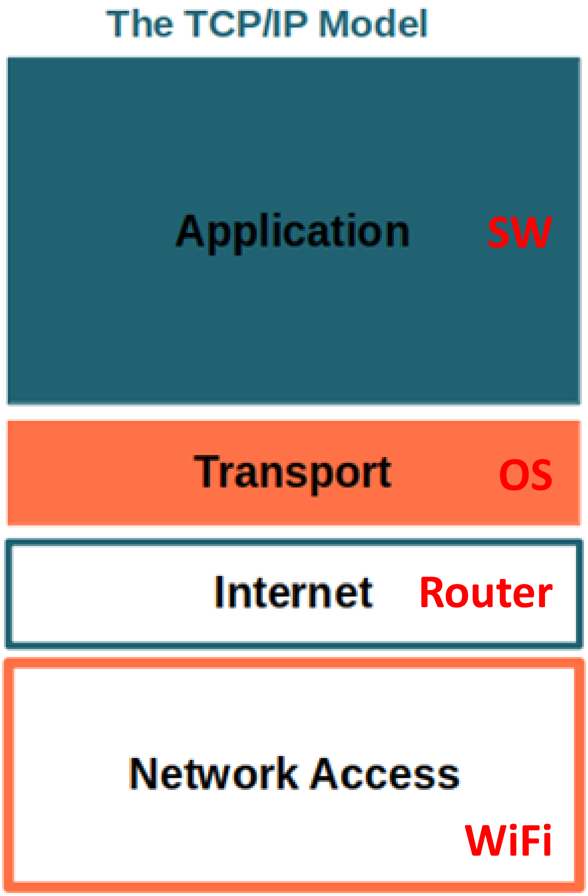


Figure 2.14 A layered operating system.



The TCP/IP Model

Application SW

Transport OS

Internet Router

Network Access WiFi

ตัวอย่างเช่น Windows NT
ใน TCP/IP ก็ใช้หลักการนี้

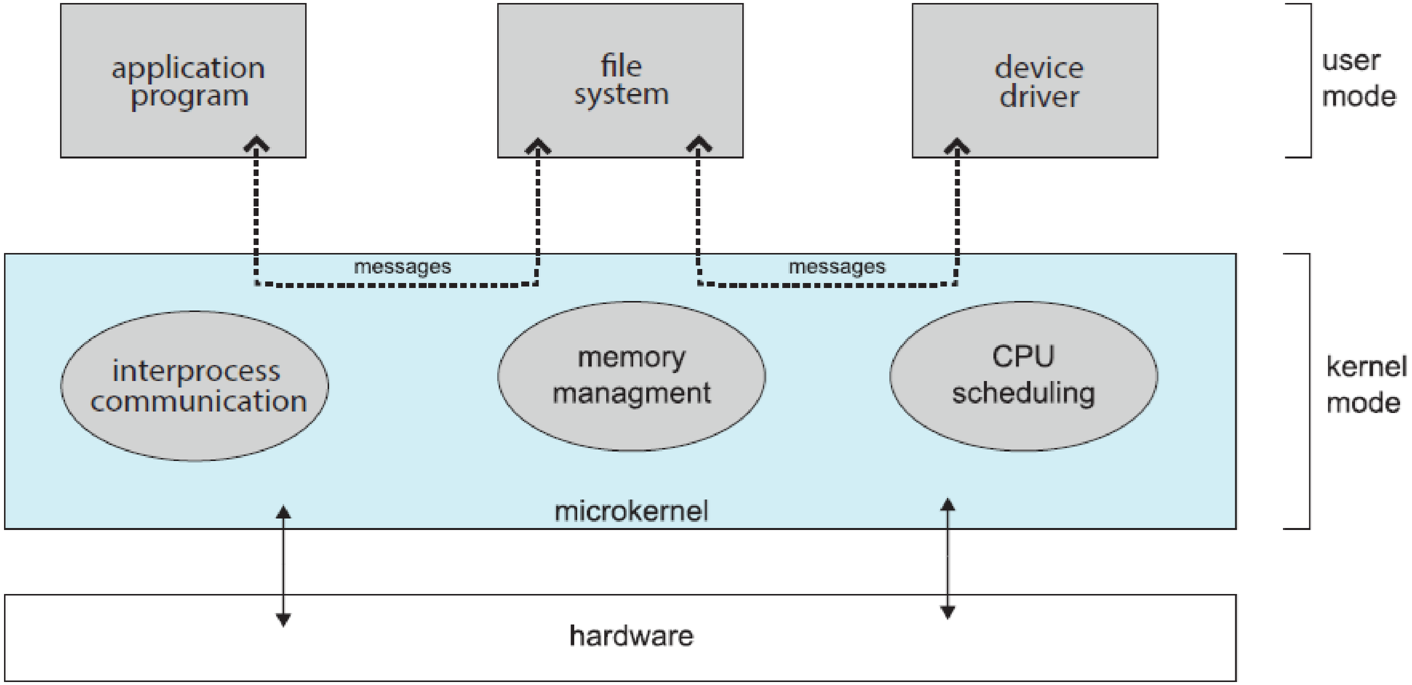
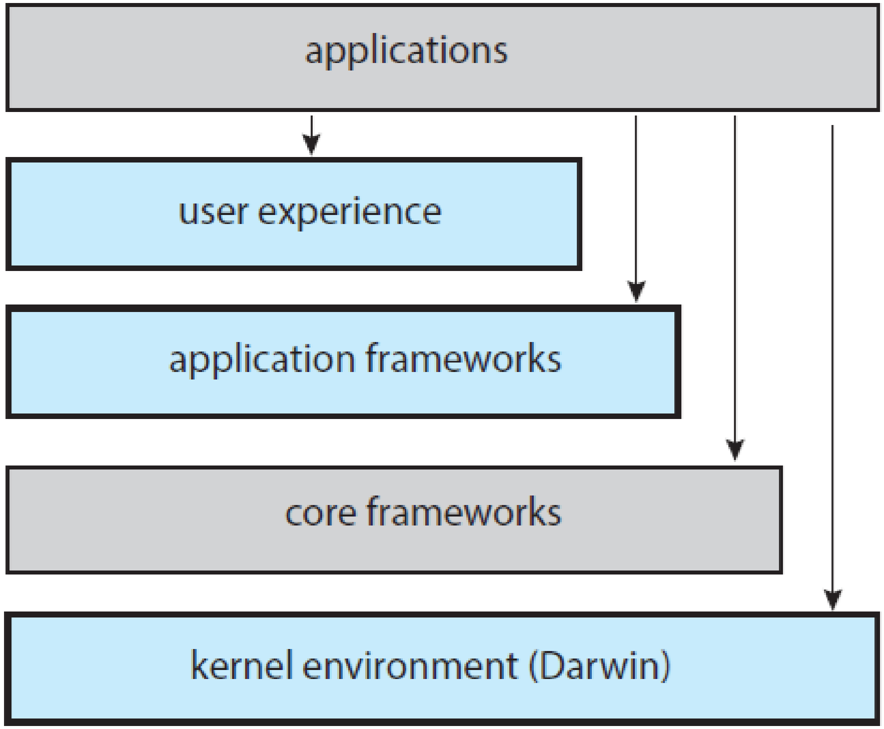


Figure 2.15 Architecture of a typical microkernel.



User Interaction
ใช้ mouse หรือ keyboard

API for Objective-C
macOS / iOS มี feature ต่างกัน

API ที่ใช้ร่วมกันได้
เช่น decode วีดีโอ / render 3D

Kernel ของ iOS จะต่างจาก macOS
power / memory / security

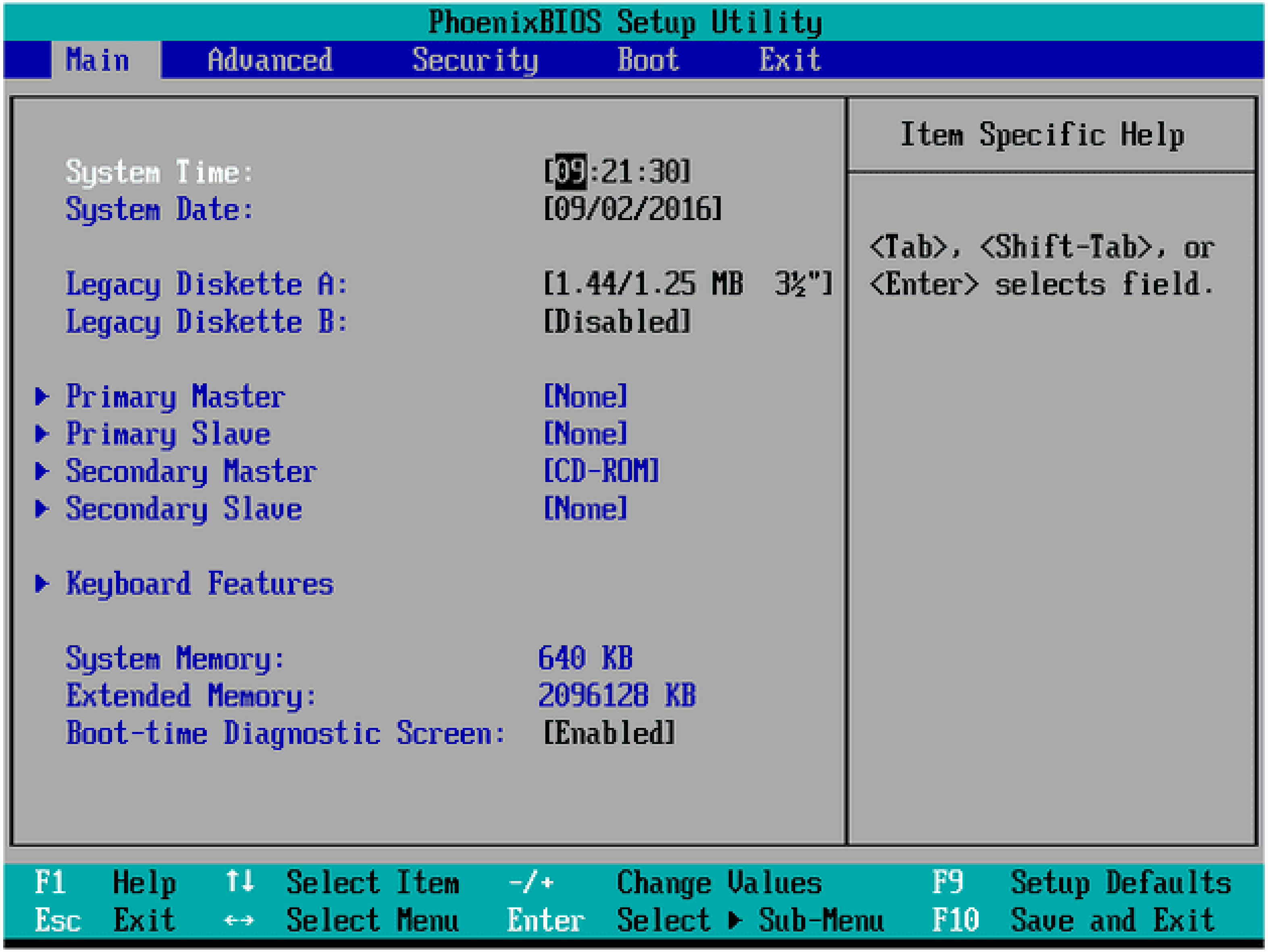
Figure 2.16 Architecture of Apple’s macOS and iOS operating systems.

System boot

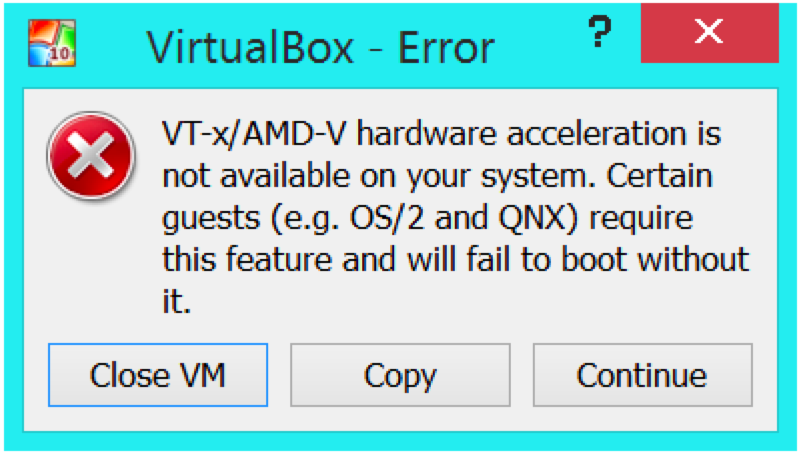
1. โปรแกรมแรก เรียกว่า **Bootstrap program**
2. **Basic input/output system (BIOS)** เป็น **firmware** ตัวหนึ่ง
3. บรรจุอยู่ใน **Read-only memory (ROM)** หรือ **EPROM** หรือ **EEPROM**
4. **BIOS** จะตรวจเช็คอุปกรณ์ และหา **OS** ใน **boot block** ที่อยู่ใน **disk**
5. **GRUB** เป็น **boot loader** ในกรณีที่ติดตั้ง **OS** ไว้มากกว่า **1** ตัว **BIOS** จะโหลด **GRUB** ก่อน จากนั้น **GRUB** จะแสดงเมนูให้ผู้ใช้เลือกว่าจะ **boot OS** ตัวใด แต่ต้องเลือกใช้ทีละหนึ่งตัว
   ปัจจุบันนิยมใช้ **vm** รัน **guest OS** ได้หลายตัวพร้อมกัน สลับไปมาระหว่าง **host/guest OS** ได้ทันที

P = Programmable
E = Erasable
E = Electrical

เฟิร์มแวร์ (firmware) ในระบบคอมพิวเตอร์ คือซอฟต์แวร์ที่ฝังอยู่ในฮาร์ดแวร์ โดยที่ผู้ใช้จะสามารถอ่าน และเรียกใช้งานเฟิร์มแวร์ได้ แต่ไม่สามารถแก้ไข เขียน หรือ ลบเฟิร์มแวร์ได้



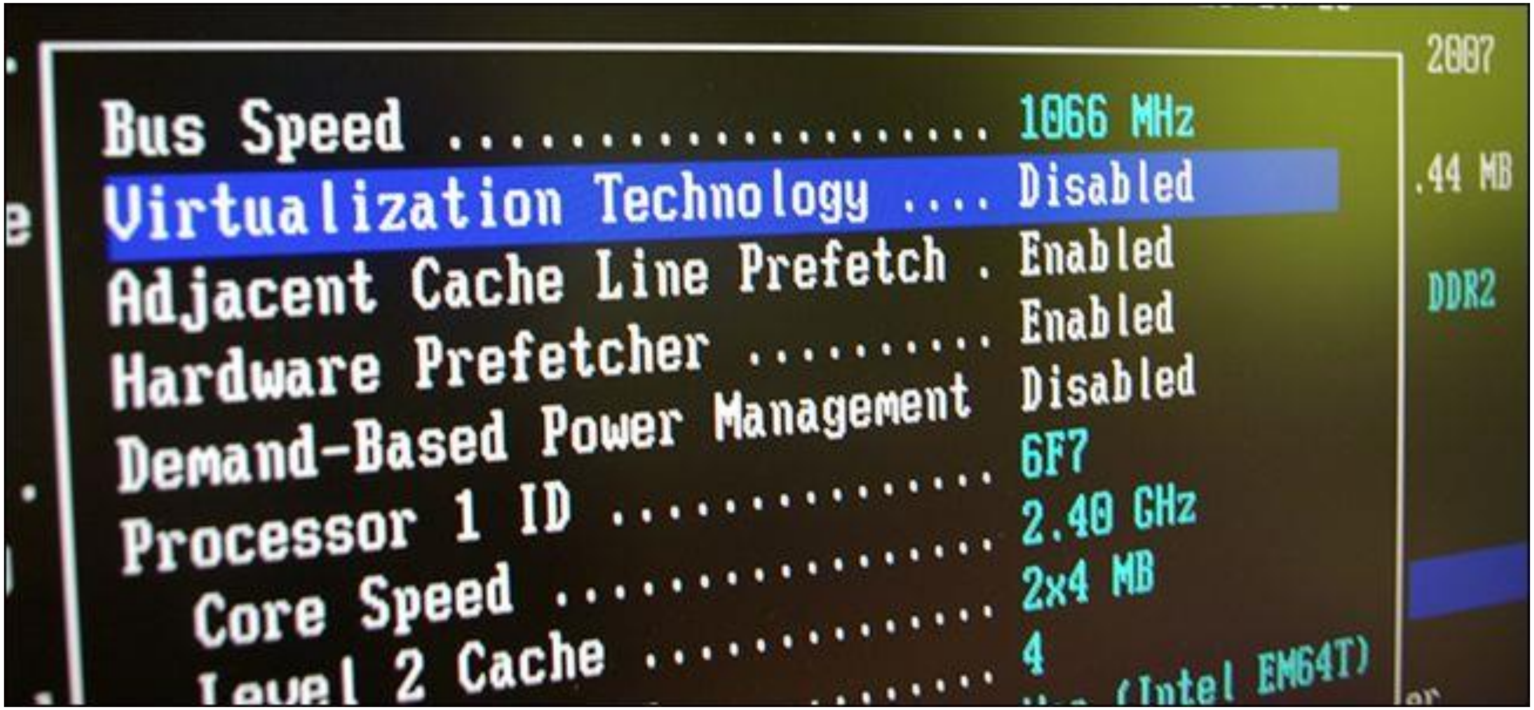
```
                         PhoenixBIOS Setup Utility
   Main      Advanced     Security     Boot     Exit

                                                    Item Specific Help

   System Time:                  [09:21:30]
   System Date:                  [09/02/2016]
                                                    <Tab>, <Shift-Tab>, or
   Legacy Diskette A:            [1.44/1.25 MB  3½"]  <Enter> selects field.
   Legacy Diskette B:            [Disabled]

 ▶ Primary Master                [None]
 ▶ Primary Slave                 [None]
 ▶ Secondary Master              [CD-ROM]
 ▶ Secondary Slave               [None]

 ▶ Keyboard Features

   System Memory:                640 KB
   Extended Memory:              2096128 KB
   Boot-time Diagnostic Screen:  [Enabled]

 F1   Help   ↑↓  Select Item   -/+    Change Values       F9   Setup Defaults
 Esc  Exit   ↔   Select Menu   Enter  Select ▶ Sub-Menu   F10  Save and Exit
```



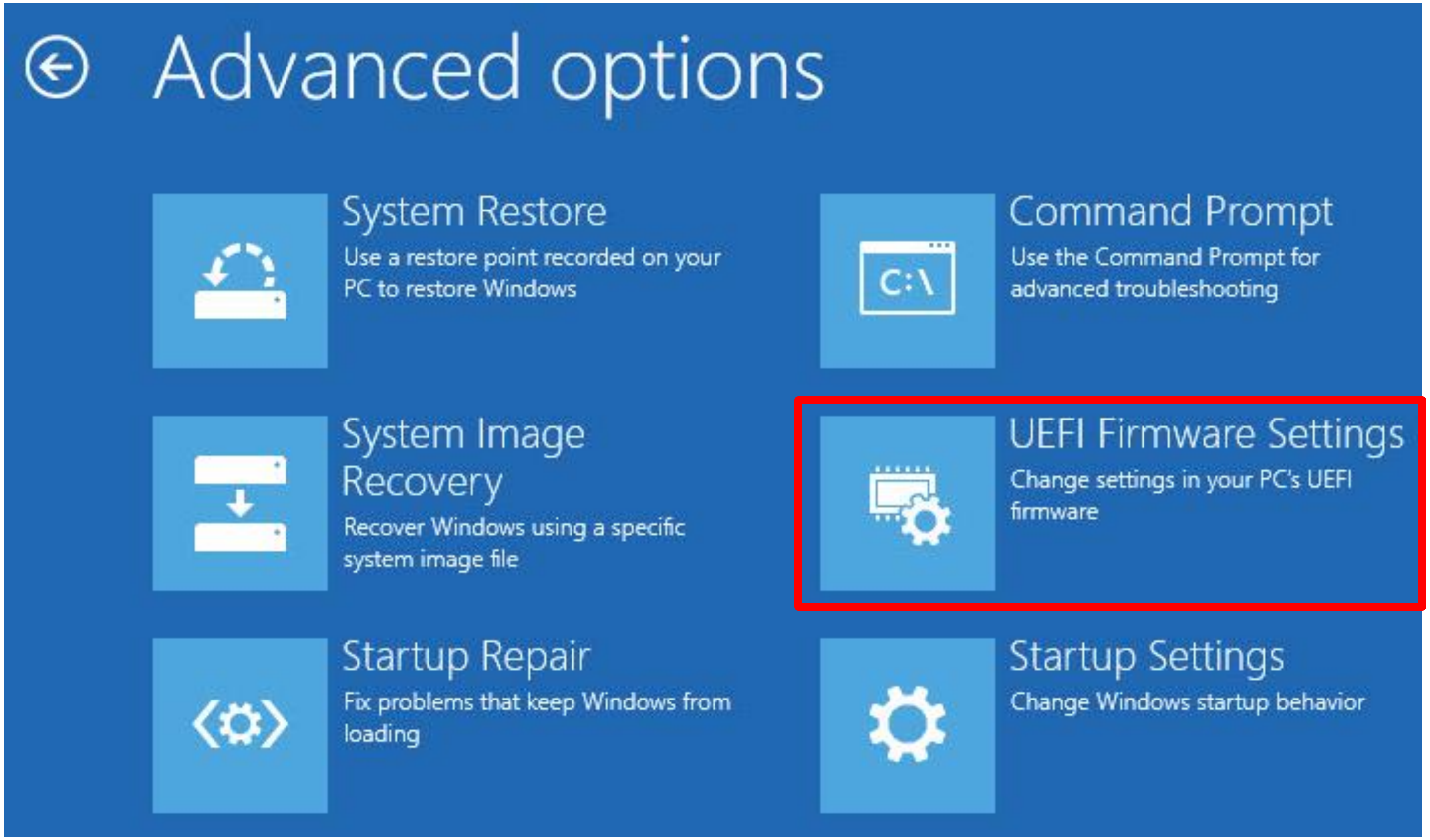
VirtualBox - Error

VT-x/AMD-V hardware acceleration is not available on your system. Certain guests (e.g. OS/2 and QNX) require this feature and will fail to boot without it.

เฉพาะ Windows
เท่านั้น Mac ไม่มี



```
Bus Speed ......................... 1066 MHz
Virtualization Technology .... Disabled
Adjacent Cache Line Prefetch . Enabled
Hardware Prefetcher .......... Enabled
Demand-Based Power Management  Disabled
Processor 1 ID ............... 6F7
   Core Speed ................ 2.40 GHz
   Level 2 Cache ............. 2x4 MB
                               4
                                  (Intel EM64T)
```



Advanced options

System Restore
Use a restore point recorded on your PC to restore Windows

Command Prompt
Use the Command Prompt for advanced troubleshooting

System Image Recovery
Recover Windows using a specific system image file

UEFI Firmware Settings
Change settings in your PC's UEFI firmware

Startup Repair
Fix problems that keep Windows from loading

Startup Settings
Change Windows startup behavior

UEFI is the abbreviation of Unified Extensible Firmware Interface, which is a firmware interface for computers and it works as a "middleman" to connect a computer's firmware to its operating system. It is used to initialize the hardware components and start the operating system stored on the hard disk drive when the computer starts up.

UEFI possesses many new features and advantages that cannot be achieved through the traditional BIOS and it is aimed to completely replace the BIOS in the future.

UEFI stores all the information about initialization and startup in a .efi file, a file stored on a special partition called EFI System Partition (ESP). The ESP partition will also contain the boot loader programs for the operating system installed on the computer.

It is because of this partition, UEFI can directly boot the operating system and save the BIOS self-test process, which is an important reason for UEFI faster booting.

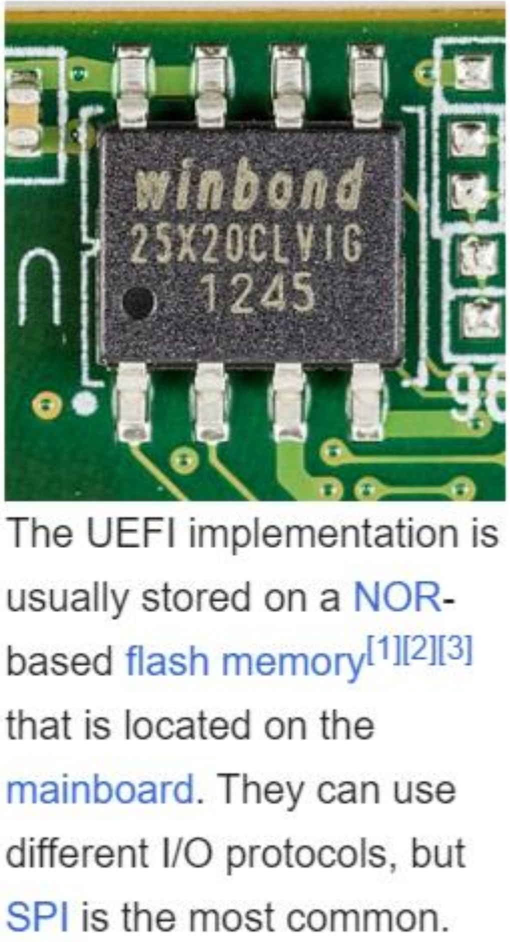
สรุป

OS ทำ Power-On Self Test (POST) เร็วกว่า BIOS เพราะโปรแกรม BIOS เล็กและเขียนซับซ้อนมากไม่ได้ เช่น รันโหมด 16-bit ใช้ memory ได้แค่ 1 MB มันจะค่อย ๆ เช็คอุปกรณ์ฮาร์ดแวร์ไปทีละตัว ในขณะที่ os สามารถเช็ค hw พร้อมกันทีละหลาย ๆ ตัวได้ เป็นต้น

BIOS ไม่สามารถ boot OS จากไดรฟ์ที่ใหญ่กว่า 2.1 TB ได้ ใกล้จะชนขีดจำกัดแล้ว

Windows 11 ใช้ UEFI เท่านั้น

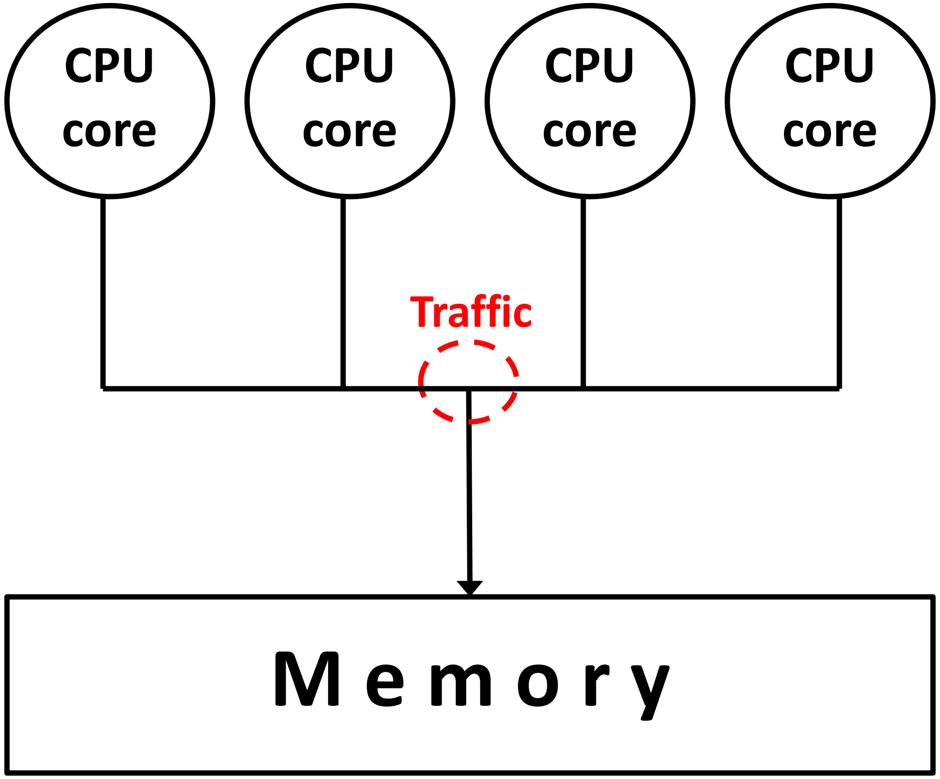
มี device driver เป็น byte code ที่รันบน processor ใดก็ได้
เพื่อให้ใช้ network, graphics ได้ ก่อนที่จะโหลด OS



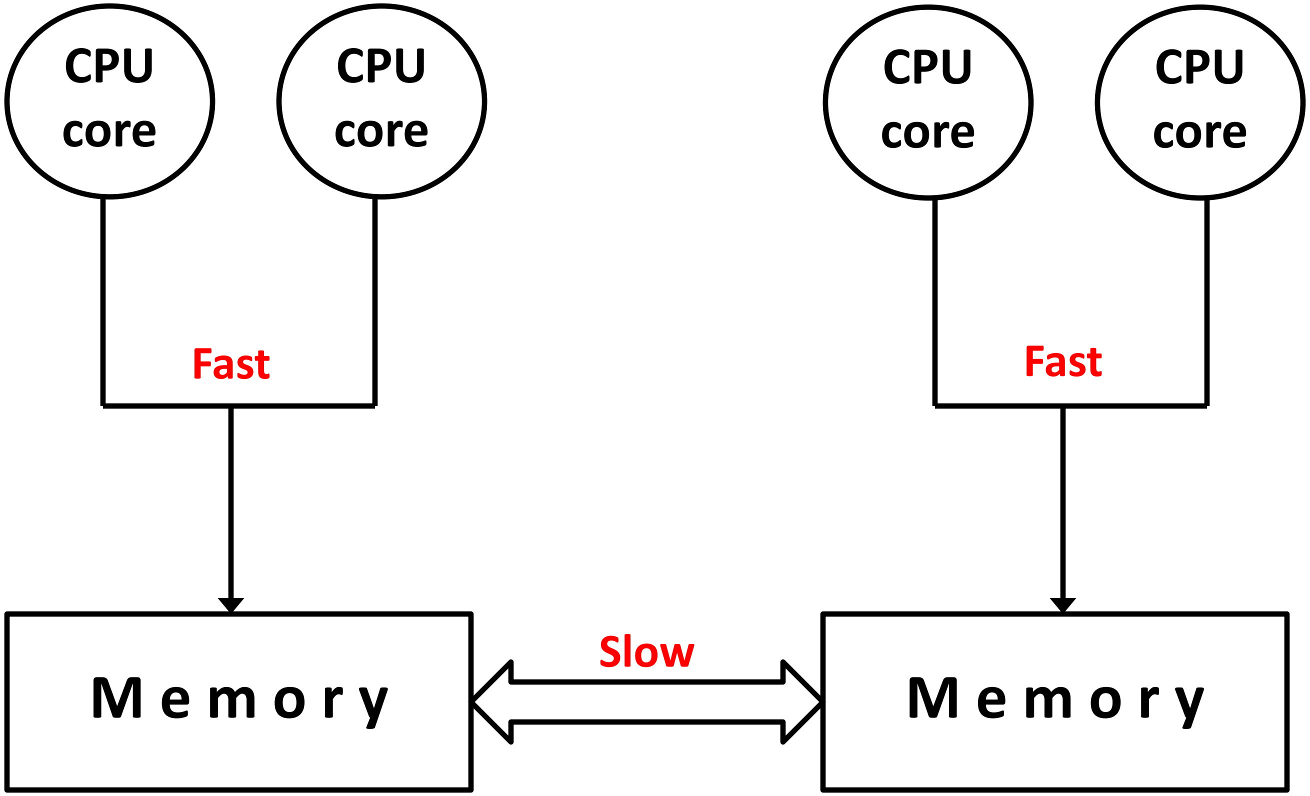
The UEFI implementation is usually stored on a NOR-based flash memory[1][2][3] that is located on the mainboard. They can use different I/O protocols, but SPI is the most common.

	BIOS	UEFI
Bootstrap program	ROM	Flash memory
Boot loader	Master Boot Record (MBR)	EFI system partition (ESP)
Mode	16 บิต (ใช้ mem ได้ 1 MB)	32/64 บิต
Pre-OS environment	-	Network, GUI, multilanguage
Programming Lang.	Assembly	C, Python
Windows product key	OS disk ผู้ใช้ป้อน key เพื่อ activate	UEFI firmware ผู้ผลิต PC ใส่ key มาให้เลย

Memory Bottleneck



Non-Uniform Memory Access (NUMA)



OS สมัยใหม่รู้จักกระจายงานไปรันในแต่ละ core และพยายามรันโปรแกรมด้วย core & memory ที่อยู่บน node เดียวกัน