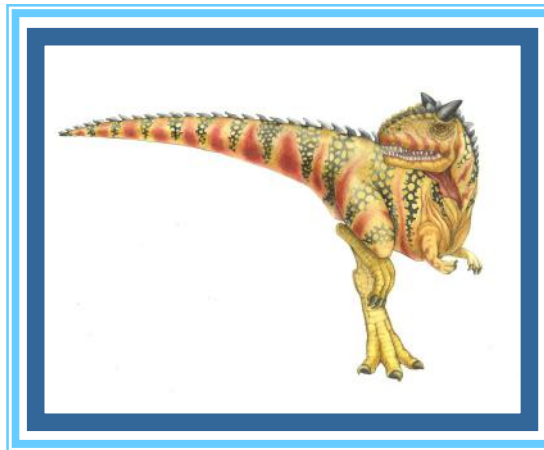


# Introduction





# Course info

---

## □ Instructors:




Dr. Kan Chen  
[kan.chen@singaporetech.edu.sg](mailto:kan.chen@singaporetech.edu.sg)



Dr. David Miguel Sanan Baena  
[david.miguel@singaporetech.edu.sg](mailto:david.miguel@singaporetech.edu.sg)



# Course info

- Website: CSD2180-Operating Systems [2022/23 T1] on xsite.singaporetech.edu.sg
    - Syllabus, lecture notes, assignments, announcements,...
    - Post questions regarding material covered in class or an assignment to Discussions forum on course web page
  - Class Schedule                      Lecture: Mon 11-1pm. LT6C,  
    Thu 11-1pm. LT6C  
    Tutorial: TBA
  - Office hours
    - Mon/Thu 2:00pm-3:00pm, SIT@SP4. Other times appointment by email.
  - TA:
    - Matthias, [ong.s@digipen.edu](mailto:ong.s@digipen.edu)
    - TBA
- 





# Assessment

---

- ❑ Participation/Attendance: 2%
- ❑ Quizzes: 18%
- ❑ Programming Assignments: 40% (7 + 7 + 12 + 7 + 7)
- ❑ Midterm Test: 15%
- ❑ Final Test: 25%
- ❑ Final letter grade algorithm: A: 93 – 100%; A–: 90 – 92.99%; B+: 87 – 89.99%; B: 83 – 86.99%; B–: 80 – 82.99%; C+: 77 – 79.99%; C: 73 – 76.99%; C–: 70 – 72.99%; D: 60 – 69.99; F: < 60%
- ❑ A “D” grade or above is considered as “Pass” and an “F” grade is considered as “Fail”. Students with grades “D”, or “D+” will be given the option to repeat the module (i.e., re-module) but the maximum GPA for the repeated attempt will be capped at 2.0. Marks are final after endorsement by the Board of Examiners. Do note that the criteria for acceptable standing in any given trimester is maintaining a minimum Cumulative Grade Point Average (CGPA) of 2.0. Refer to the SIT Academic Guide for further details.





# Acknowledgment

---

- First part of this course borrows materials from the following sources:
  - Operating System Concepts, 10th edition, by Abraham Silberschatz, Peter Galvin, and Greg Gagne.
  - Modern Operating Systems, 4th Edition, by Andrew Tanenbaum and Herbert Bos.
  - Operating Systems: Three Easy Pieces, by Andrea Arpaci-Dusseau and Remzi Arpaci-Dusseau.
  - Operating Systems, Haibo Chen/Yubin Xia, Institute of parallel and distributed systems (IPADS), Shanghai Jiao Tong University.
  - Operating Systems, Yanyan Jiang, Nanjing University.
  - The CSE451 Web, © 1993-2022, Department of Computer Science and Engineering, University of Washington.





# Chapter 1: Introduction

---

- What is Operating System
  - What does an Operating System Do
  - Operating System History
- Why do you need to study Operating System?
- How to study Operating System





# Objectives

---

- Provide examples of operating systems
  - Describe what can operating system do
- Discuss the challenges of operating system
- Describe the importance of operating system





# What is an Operating System?







# What is an Operating System?

---

- Which is/are operating system(s)?
  - A. Windows 11 and its software
  - B. Ubuntu kernel and the drivers
  - C. The software on a new Samsung phone
  - D. The software on a Dji drone
  - E. Microsoft
  - F. Raspberry Pi
  - G. The C library, libc
  - H. Graphical User Interface





# What is an Operating System?

---

- How do you think an operating system should be defined?
  - I don't know.
  - Nobody knows.
  - The textbook claims to know – see next.
  - They're programs – big programs
    - ▶ The Linux source has over 30M lines of C
    - ▶ Windows has way, way more
  - They're programs that make writing other programs easier –  
The raw hardware is an unforgiving place to execute...





# What is an Operating System?

- A program that acts as **an intermediary between** a user of a computer **and** the computer hardware
- Operating system goals:
  - **Execute user programs** and make solving user problems **easier**
  - Make the computer system **convenient to use**
  - Use the computer hardware **in an efficient manner**





# Computer System Structure

---

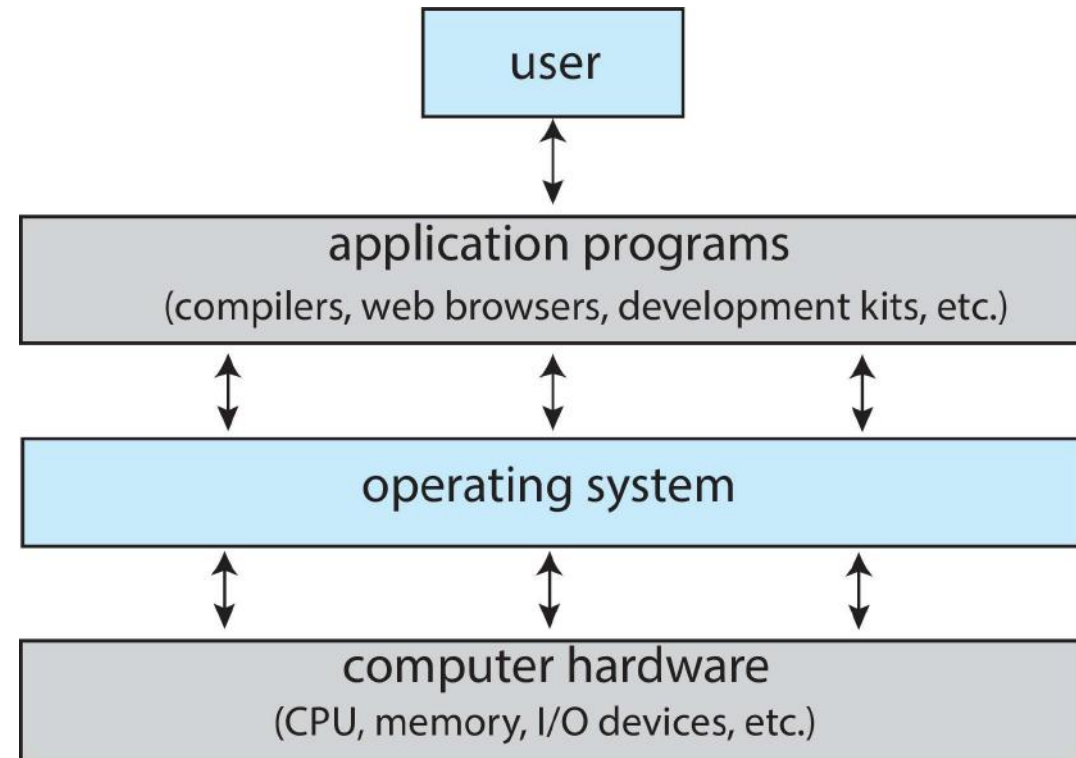
- **Computer system can be divided into **four** components:**
  - **Hardware** – provides basic computing resources
    - ▶ CPU, memory, I/O devices
  - **Operating system**
    - ▶ Controls and coordinates use of hardware among various applications and users
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
    - ▶ Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - ▶ People, machines, other computers





# OS: Between App and HW?

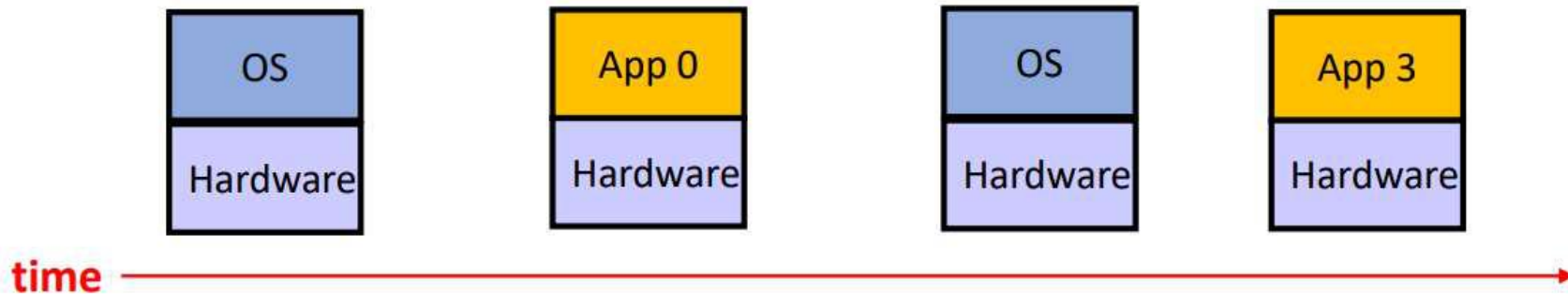
- Your application cannot directly manipulate (a lot of) the hardware, it has to ask the OS to do it on its behalf
  - This is the basis of security
- Various apps/hardware - OS Content and extension are constantly changing



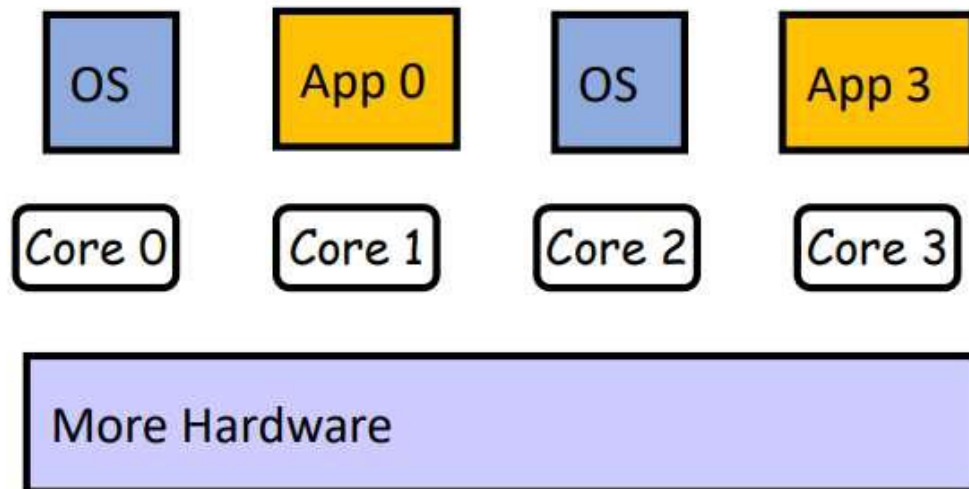


# OS: Between App and HW?

- The OS isn't always between you and (some of) the hardware

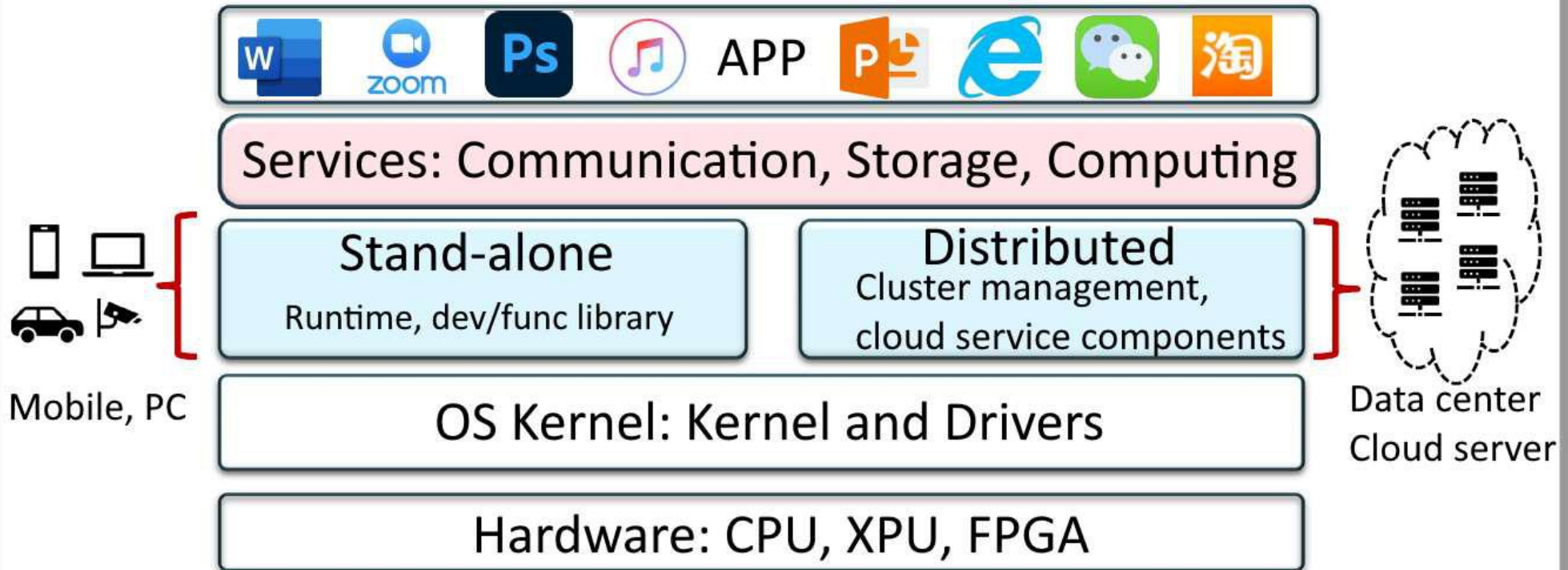


- More updated





# Manage HW and build ecosystem for Apps





# From Hello World!

- What does OS do when run ./hello?

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

```
$ gcc hello.c -o hello
```

```
#run once
```

```
$ ./hello
```

```
Hello World!
```

```
#run twice
```

```
$ ./hello & ./hello
```

```
[1] 3308
```

```
Hello World!
```

```
Hello World!
```

```
[1]+  Done ./hello
```

```
$
```







# What Operating Systems Do for Apps

---

- ❑ Some Issues considered by the Operating System
  - ❑ Where is the hello executable stored? How is it stored?
  - ❑ How is the hello executable loaded into the CPU to run?
  - ❑ How does the hello executable print the line "Hello World!" to the screen?
  - ❑ How can two hello programs run on one CPU while they are running at the same time?
  
- ❑ For applications, operating system needs to:
  - ❑ **Serve** the applications
  - ❑ **Manage** the resources





# Some services provided by the OS

---

- An **abstraction** that provides computing resources for applications
  - CPU: Processes/threads, not limited by the physical CPU numbers
  - Memory: virtual memory, not limited by physical memory
  - I/O devices: abstract various devices into files with a unified interface
- Provides **synchronization** between threads for applications
  - Applications can implement their own synchronization primitives (like spinlock)
  - The operating system provides more efficient synchronization primitives (in conjunction with thread switching, such as `pthread_mutex`)
- Provides **communication** for applications
  - Applications can use the network for inter-process communication (such as loopback devices)
  - The OS provides a more efficient local communication mechanism (with richer semantics like pipe), such as Shell Pipe





# Some management done by the OS

---

- Application life cycle management
  - Operations such as loading, migrating, and destroying applications
- Allocation of computing resources
  - CPU: scheduling mechanism for threads
  - Memory: allocation of physical memory
  - I/O devices: multiplexing and distribution of devices
- Security and isolation
  - Application itself: access privilege control
  - Between applications: isolation mechanisms, including error isolation and performance isolation





# Management and Services

---

- Managing and servicing applications may conflict
  - The goal of the service: maximize the operating efficiency of a single application
  - Goal of management: maximize the overall utilization of system resources
  - Example: Scheduling strategies that simply emphasize fairness tend to have low resource utilization
    - ▶ As small time slot/slice or quantum for round-robin usually leads to a lot of context switches





# Operating System Definition

---

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is a good approximation
  - But varies wildly
- “The one program in memory at all times on the computer” is the **kernel**, part of the operating system
- Everything else is either
  - a **system program** (ships with the operating system, but not part of the kernel) , or
  - an **application program**, all programs not associated with the operating system





# Operating System Definition (Cont.)

---

- Services + management
- From the structural point of view: Operating system kernel + system software (framework)
- Definition from hardware point of view: Efficiently abstract finite/discrete resources into infinite/continuous resources
- Today's OSES for general purpose and mobile computing also include **middleware** – a set of software frameworks that provide additional services to application developers such as **databases, multimedia, graphics**





# Questions

---

- Question 1: If a machine has only one application, when the machine is on, the application will be running automatically and will not exit, does it still need an operating system?
  
- Question 2: If an application wants to fully control the PC hardware directly, instead of using the abstractions provided by the operating system, does it still need an operating system?





# App-OS interface: System Calls

- What are system calls?
  - The mechanism for the application to call the operating system to service
  - E.g. `printf()` -> `write()` -> `sys_write()`
    - ▶ `write(1, "Hello World!\n", 13)`
- Make Calling OS services like calling normal API functions
- `strace hello.c`

```
/* run the hello programm*/
execve("./hello", ["./hello"], 0x7ffed5a79e80 /* 64 vars */) = 0
...
/* write `Hello World!\n' to std output, 1: stdout, 13: 13 chars*/
write(1, "Hello World!\n", 13Hello World!) = 13

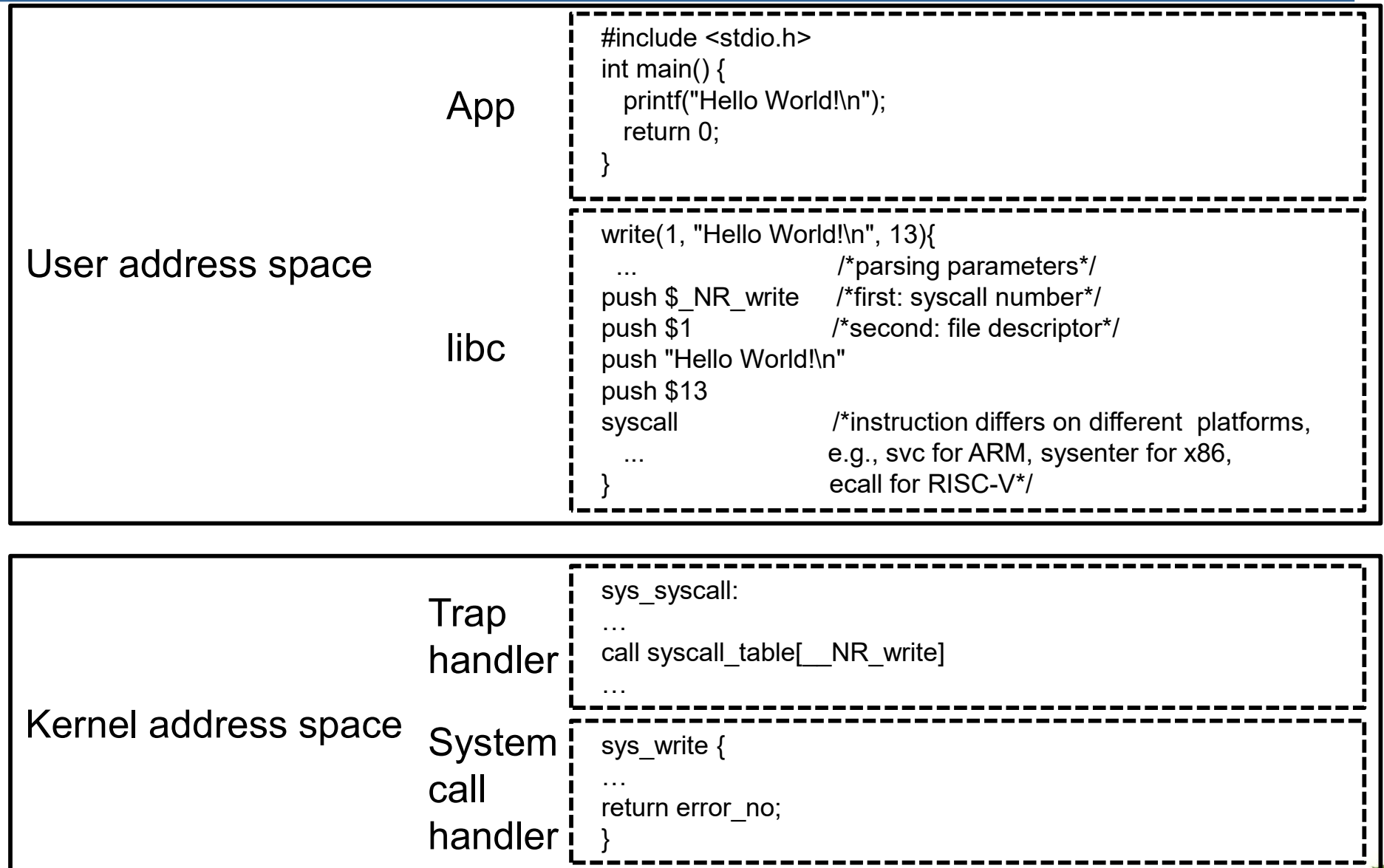
/* exit hello*/
exit_group(0)
```







# System Call Steps





# OS: Management

---

- ❑ Avoid one malicious (rogue) application occupying all resources
- ❑ Method-1: A clock interrupt occurs every 10ms (time slice)
  - ❑ The scheduler decides the next task to run
- ❑ Method-2: The current task execution can be interrupted by signals, etc.
  - ❑ E.g., kill -9 1951

rogue.c

```
int main () {  
    while (1);  
}
```





# OS: Management

---

- ❑ How to freeze the OS?
  - ❑ e.g., rogue2 can fork a lot processes
- ❑ How to solve this problem?
  - ❑ Limit resource usage: cgroup/Linux
  - ❑ Virtualization: virtual machine – more resources
  - ❑ The universal method: reboot
  - ❑ Platform check: AppStore's app submission review

rogue2.c

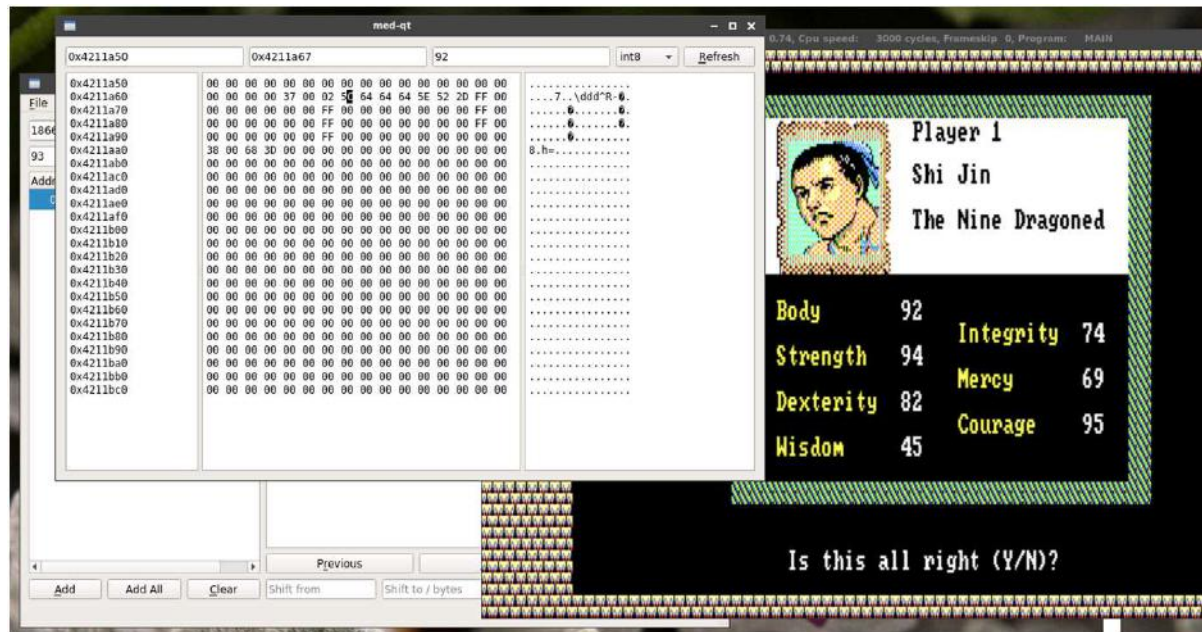
```
int main () {  
    while (1)  
        fork();  
}
```





# OS : Management

- ❑ Avoid one malicious application from accessing/modifying restricted (others') resources
- ❑ OS should provide us protection, otherwise we can easily hack a game:





# Question



**Jack Ma**

Working as a programmer since 2022

**Why I got fired after committing this code?  
Is it because I did not write comments?**

```
public static Date getNextDay() {  
    try {  
        Thread.sleep(24*60*60*1000);  
    } catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
    return new Date();  
}
```

↑ 144





# Operating System History

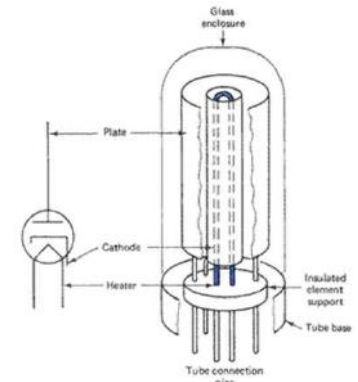




# OS History

## □ 1940s

- ENIAC (1946.2.14), extraordinary genius design but simple
- Logic gate: vacuum tube, memory: delay lines, io: card/lights
- NO OS
  - ▶ It's amazing to be able to put the program on it, programs operate hardware directly with instructions



## □ Real Bug

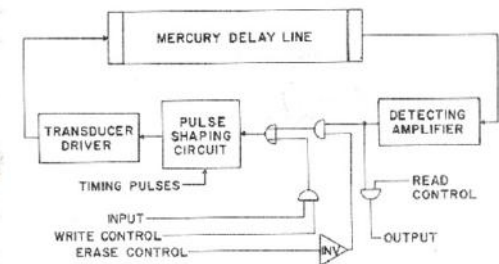
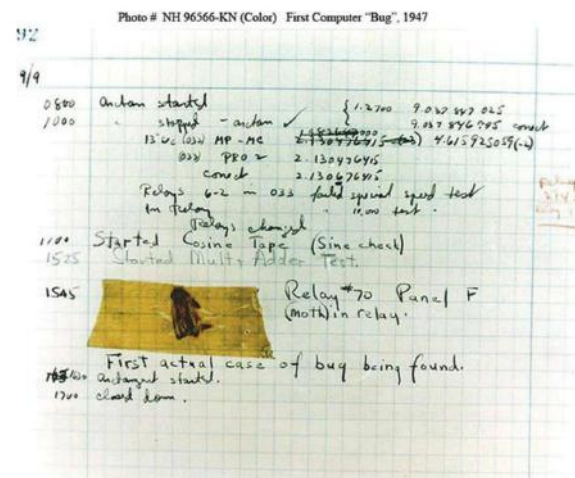


Figure 15-2. An acoustical delay line memory system







# OS History

- 1950s
  - Batch OS: GM-NAA I/O
  - Robert L. Patrick and Owen Mock (1956, IBM 704)

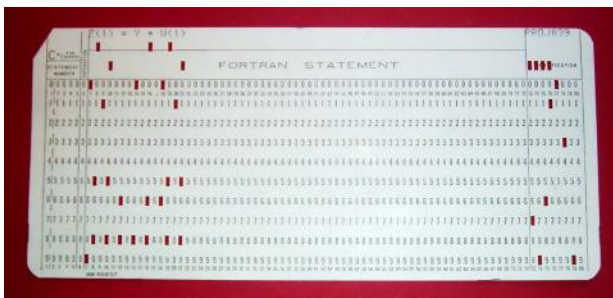






# OS History

- 1950s
  - The concept of OS emerges: the system that operates the tasks (jobs)
  - A library function and scheduler that manages multiple programs that are queued to run in sequence.
  - Writing/running programs is very labor-intensive (e.g., if you write an infinite loop...)
  - Very expensive (\$50,000-\$1,000,000), only one per university
  - For central computers: multi-user queues to share computers
  - "Batch system" = automatic switching of programs (card change) + library function API
  - Disk Operating Systems (DOS) Objects and APIs such as "devices", "files", "tasks", etc. begin to appear in the operating system



```
C---- THIS PROGRAM READS INPUT FROM THE CARD READER,  
C---- 3 INTEGERS IN EACH CARD, CALCULATE AND OUTPUT  
C---- THE SUM OF THEM.  
100 READ(5,10) I1, I2, I3  
10 FORMAT(3I5)  
   IF (I1.EQ.0 .AND. I2.EQ.0 .AND. I3.EQ.0) GOTO 200  
   ISUM = I1 + I2 + I3  
   WRITE(6,20) I1, I2, I3, ISUM  
20 FORMAT(7HSUM OF , I5, 2H , , I5, 5H AND , I5,  
   * 4H IS , I6)  
   GOTO 100  
200 STOP  
END
```





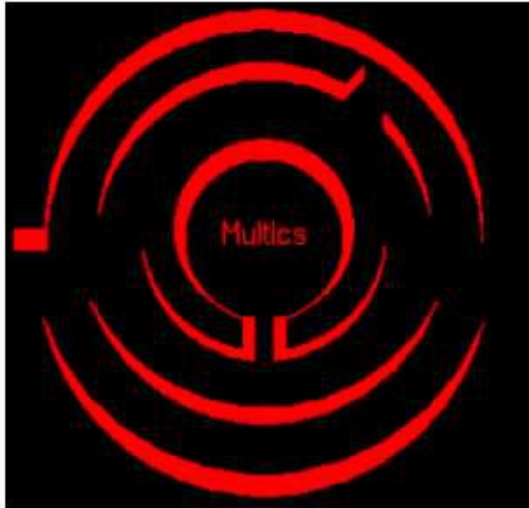
# OS History

- 1960s
  - IBM System/360 OS, 1964
  - The first general-purpose modern operating system
  - Separation of operating system and computer
    - ▶ Architect: Gene Amdahl (Amdahl's Law)
    - ▶ Project Manager: Fred Brooks ("The Mythical Man-Month", 1999 Turing Award)





# Time-sharing and multitasking

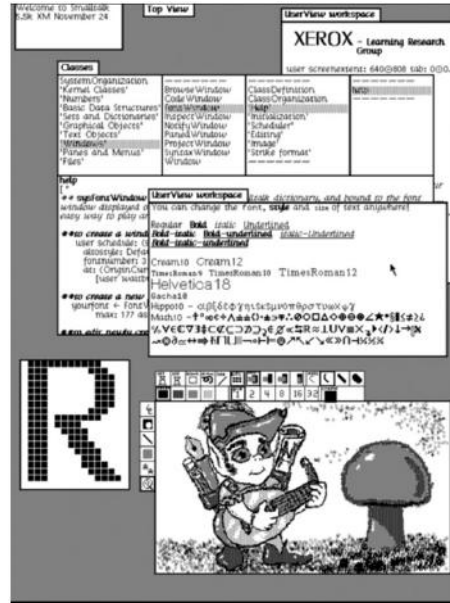


- ❑ Multics:
  - ❑ Fernando Corbató (1990 Turing Award) MIT/GE, 1964
  - ❑ Time-sharing, file system, dynamic linking, etc.
- ❑ Unix:
  - ❑ Ken Thompson, Dennis Ritchie (1983 Turing Award), 1969
  - ❑ Shell, a hierarchical file system
- ❑ Linux:
  - ❑ Linus Torvalds, 1991
  - ❑ Most Popular Open-Source Operating Systems





# GUI: Xerox Alto/MacOS/Windows



- Xerox Alto (1973)
  - The first graphical operating system, mouse was introduced (Chuck Thacker, Turing Award 2009)
- Mac OS (Apple LISA, 1983)
  - 1979 Jobs visited Xerox PARC
- Windows 1.0 (1985)





# Why study OS?

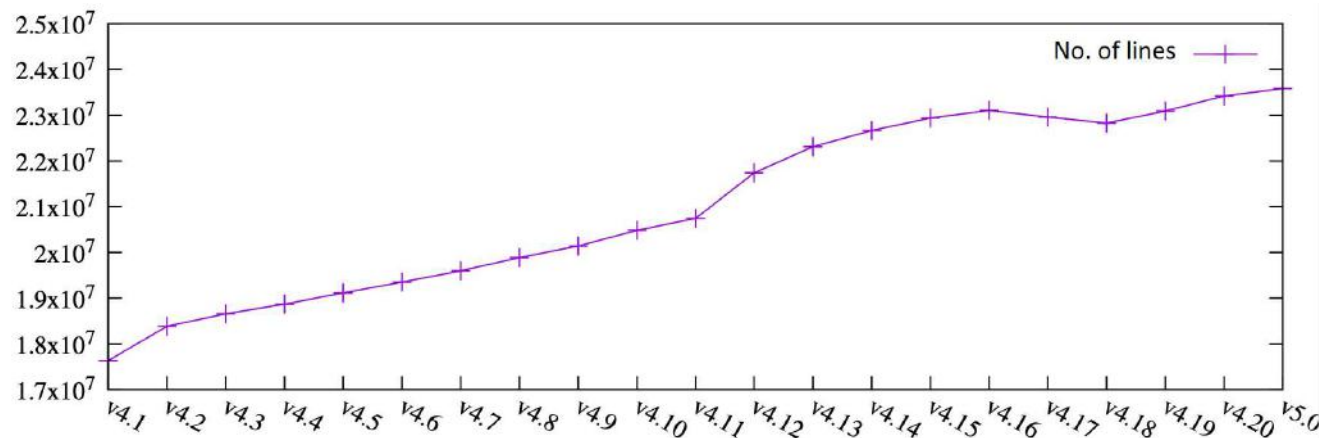




# Challenges: Opportunities

- Game, Metaverse, AI, IoT, 5G, Cloud, Fintech, Autonomous driving...
  - 100X throughput, 100x faster, 100X computing power, 100X devices
- Trend-1: From closed to open to closed
  - IOS vs Linux vs Android, RedHat ...
- Trend-2: From dedicated to general to dedicated
  - From general-purpose computing to domain computing, various xPUs/devices emerge – GPU, TPU, NPU, IPU, FPGA SSD
- Trend-3: From simple to complex to more complex
  - Linux has exceeded 30 million lines of code, Linux's legacy...
  - Still growing/updated at 2 million lines per year

CAN'T WAIT TO HAVE BING INSTALLED ON MY PHONE —  
Google to charge Android OEMs as much as \$40 per phone in EU  
After the EU ruling, OEMs can unbundle Google's Android apps, but it will cost them.  
IT'S OFFICIAL: IBM is acquiring software company Red Hat for \$34 billion

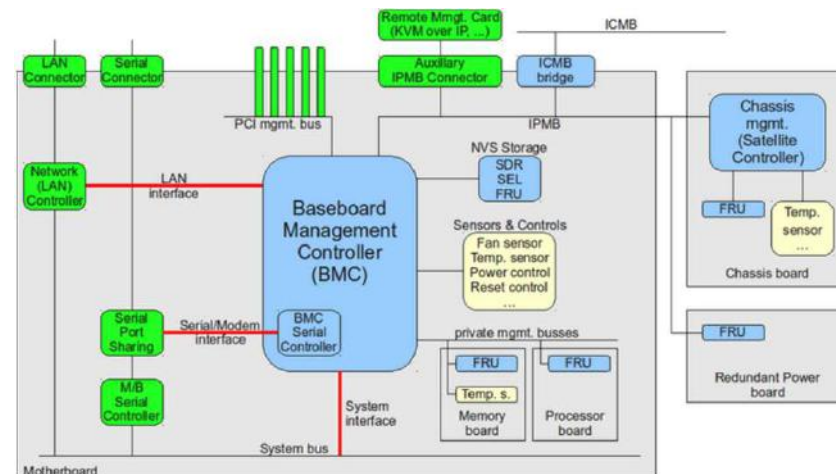
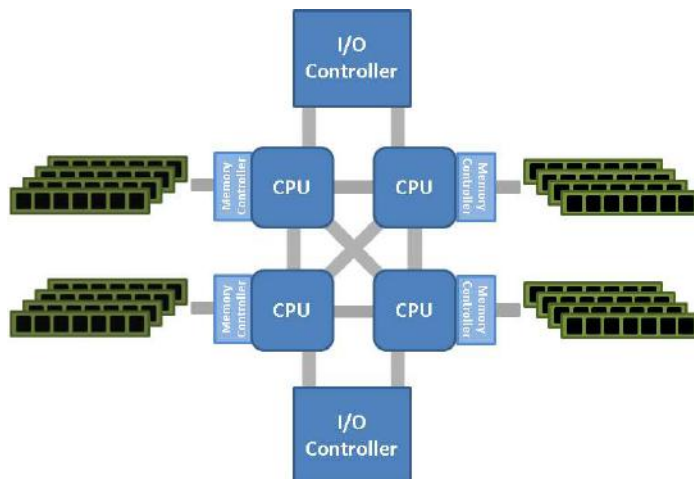






# Challenges: Opportunities

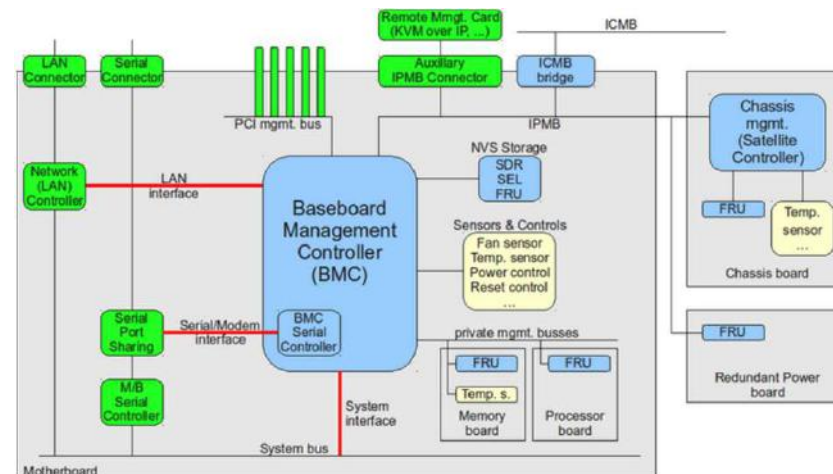
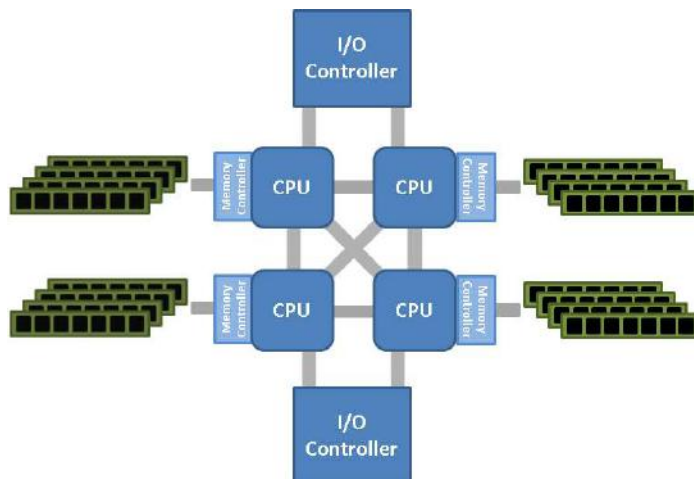
- Trend-4: Hardware advances
  - ARM/Intel, big and small cores (big.medium.little, Intel P/E-cores) ...
  - Non-uniform Memory Access (NUMA) ...
  - Intel-VT/AMD-V ...
  - New programmable devices (smart network card, intelligent storage (SSD), AI accelerator)
- Trend-5: Distributed, Programmable heterogeneous network
  - A set of OSes running on a computer/on-chip
  - MINIX? The most popular OS in the world, thanks to Intel





# Challenges: Opportunities

- Trend-4: Hardware advances
  - ARM/Intel, big and small cores (big.medium.little, Intel P/E-cores) ...
  - Non-uniform Memory Access (NUMA) ...
  - Intel-VT/AMD-V ...
  - New programmable devices (smart network card, intelligent storage (SSD), AI accelerator)
- Trend-5: Distributed, Programmable heterogeneous network
  - A set of OSes running on a computer/on-chip
  - MINIX? The most popular OS in the world, thanks to Intel

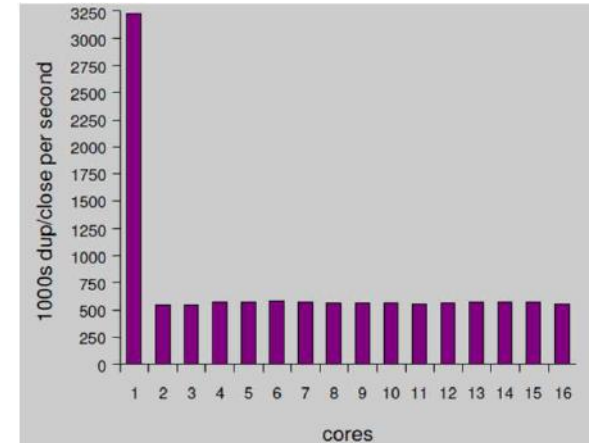
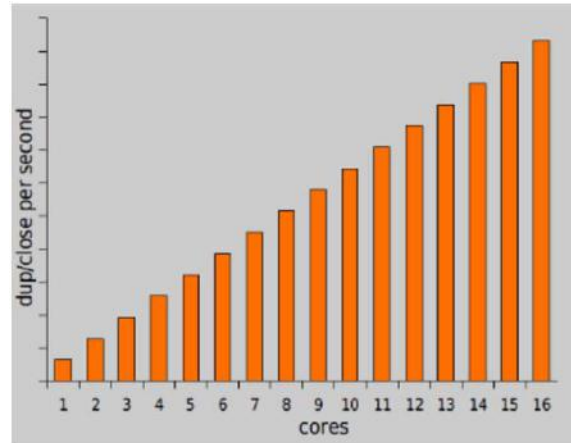






# Challenges: Opportunities

## □ Scalability



## □ Tradeoff between Isolation and Performance

- Slow inter-process communication: 16 times slower than function calls
- 30+ years optimization – IPC can cost 30% of the program runtime
- “But there is no separate thread or process with SQLite. SQLite runs in the same address space as the application, using the same program counter and heap storage. SQLite does no inter-process communication (IPC). ...SQLite interprets the SQL in the same thread as the caller.”

### Function call



1. Prepare stack
  2. Save registers
  3. JMP
- ~30 cycles

### Synchronized IPC



\* domain  
means  
a process

1. Domain\* switch ~ 500 cycles (RISC-V)
2. Message copying ~ 4000 cycles (4KB)





# Challenges: Security

---

CVE-2015-8370

New Technology

## The Simplest Hack: Hitting The Backspace 28 Times Will Break You Into a Linux Computer

A hack so ingenious it seems fake [but is all too real.]



// BY JOHN WENZ DEC 16, 2015

CVE-2016-4484

News, Security

## You Can Own A Linux System By Holding Down Enter Key For 70 Seconds, Here's The Fix

Adarsh Verma November 16, 2016





# Challenges: Security

**do\_brk()**

**Bug**

```
Asmlinkage unsigned long sys_brk(unsigned long brk) {  
    ...  
    /* OK, looks good – let it rip. */  
    if (do_mmap(NULL, oldbrk, newbrk-oldbrk,  
                PROT_READ|PROT_WRITE|PROT_EXEC,  
                MAP_FIXED|MAP_PRIVATE, 0) != oldbrk)  
+     if (do_brk(oldbrk, newbrk-oldbrk) != oldbrk)  
        goto out;
```

- Purpose of do\_brk() was to speed up anonymous mmaps from sys\_brk()
- do\_mmap() checked things properly, do\_brk() removed almost all checking

The brk and sbrk functions are historical curiosities left over from earlier days before the advent of virtual memory management. --- **man brk**

**do\_brk()**

**fix**

```
if (!len)  
    return addr;  
  
+ if ((addr + len) > TASK_SIZE || (addr + len) < addr)  
+ return -EINVAL;  
+
```

- The TASK\_SIZE is typically set to 0xc0000000 in 32-bit OS, the start of kernel memory





# New OSes

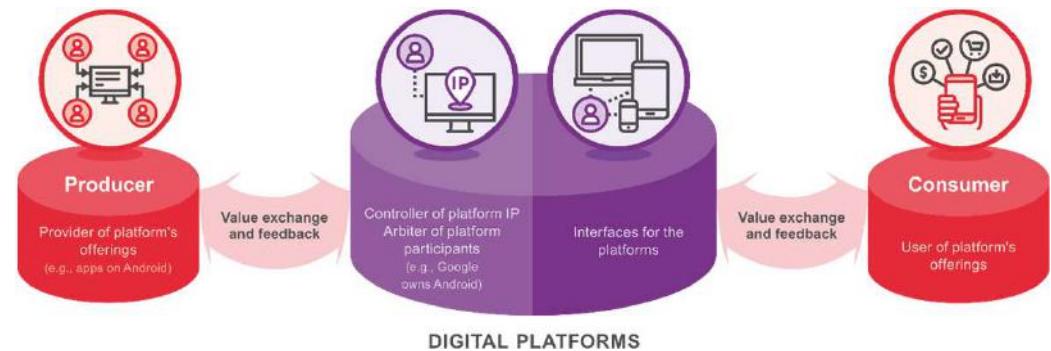
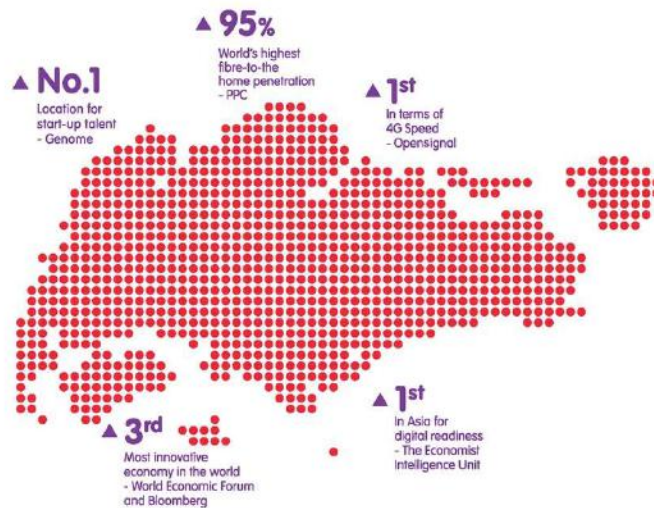
- ❑ OS is a mature field. Can a new OS to solve tackle the challenges?
  - ❑ Windows once dominated
  - ❑ One of the most complex systems ever, difficult for users to customize
  - ❑ "New" operating systems keep emerging–Redox, Harmony, Fuchsia...
  - ❑ What OS does DJI drone/Google's data center/NASA rover use?
    - ▶ The First Bug on Mars: OS Scheduling, Priority Inversion, and the Mars Pathfinder
    - ▶ After landed, begin system resets
      - Pathfinder's VxWorks RTOS causing resets every time Pathfinder started collected weather data.





# OSes as Digital platforms

- In Singapore: IMDA, Smart Nation, Ministry of Trade & Industry, ...



- In United Nation Digital Economy Report 2019
  - ... create environments for code and content producers to develop applications and software in the form of, for example, **operating systems** (e.g. Android or Linux) or technology standards (e.g. MPEG video) ...







# Why study OS

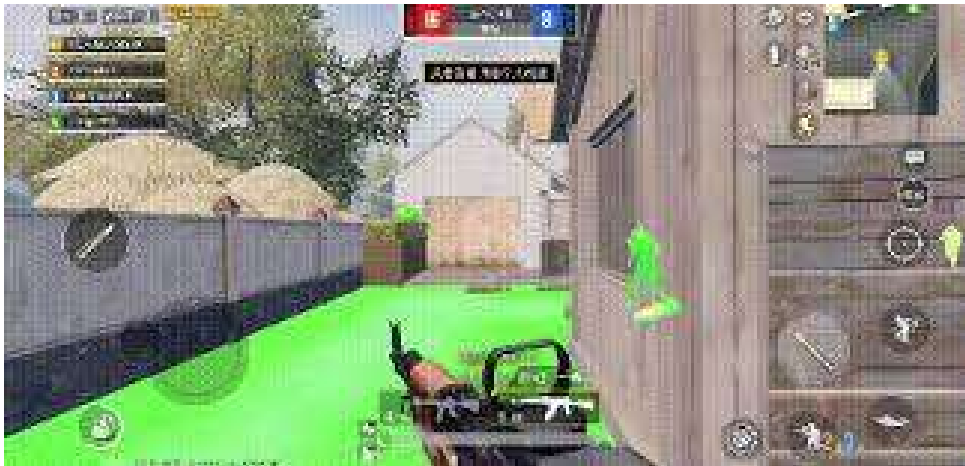
- ❑ OS is the cornerstone of the academic “System” field (OSDI/SOSP)
- ❑ Many “OS” companies: Microsoft, Google, IBM, Vmware, Huawei ...
  - ❑ Cluster, GFS, MapReduce, BigTable ...
- ❑ Interview questions:
  - ❑ What happens before main()? Explain “volatile” in C? ...
- ❑ Learn more
  - ❑ You use software everyday, but you don’t understand
    - ▶ Fully utilize hardware you get from the sellers in Sim Lim Square
    - ▶ How to write browsers, compilers, IDEs, task managers, antivirus software, viruses?
  - ❑ OS/Game Engine/Browser: software engineering masterpieces
    - ▶ Caching, Syncing, Pool, ...





# Why study OS

- Game development
  - Plugin, Hotfix/patch, Memory, Anti-Cheat
    - ▶ `render(objects) → render_hacked(objects)`
  - GPU, Vulkan semaphore/fence, Multithreaded ...





# How to study OS?







# How to study OS?

- Understand
  - What is a Program ?
    - ▶ State machine, states: PC, registers, stack, ...
  - What services does the operating system provide to programs?
    - ▶ OS = Object + API
  - How does operating system manage the programs?
    - ▶ OS = C program, after init, it becomes interrupt/trap/fault handler

How to run a program?



After study OS



Double click

...





# How to study OS?

- ❑ Don't be afraid
  - ❑ Programming: C, Assembly , ...
  - ❑ (correct) Tools: command line, gcc, strace, gdb, IDE (vscode is ok), ...
  - ❑ Readings: Code, RTFM, Textbooks, ...

