Ch.1

- user of the computer ↔ OS ↔ hardware | OS is a resource allocator and a control program
- OS goals: ↳ is interrupt driven
  1- Excute user programs and make solving user problems easier.
  2- Make the computer system convenient to use.
  3- Use the computer hardware in an efficient manner.
- Computer system structure:
  1- hardware 2- OS 3- application programs 4- users
- Kernel is the one program running at all times on the computer.
- Bootstrap program is the program that initializes the os when a computer is powered up or rebooted.
  ↳ typically stored in ROM or EPROM, generally known as firmware.
- main memory ↔ CPU ↔ local buffer ↔ I/O ↔ device
- Interrupt vector contains the addresses of all the service routines.
- A trap or exception is a software-generated interrupt caused either by an error or a user request
- Interrupts: (1) polling (2) vectored
- System call: request to the OS to allow user to wait for I/O completion
- Device-status table: contains entery for each I/O device indicating its type, address and state
- Storage-Device Hierarchy:

registers, cache, main memory	volatile	- expensive / faster / smaller - CPU can access directly / RAM
ssd, magnetic disk, optical disk, magnetic tape	non-volatile (secondary storage)	- divided into tracks and sectors - cheap / slower / larger Tertiary storage ↳ optical storage, magnetic tape

- caching: copying information into faster storage system
- Device → Device controller → Device driver → Kernel
- Direct memory access: device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Multiprocessor (parallel or multicore systems):
  advantages ↳ ① increased throughput. types ↳ ① asymetric
  ② economy of scale. ② symetric
  ③ increased reliability.

Scanned with CamScanner

- Operating system structure:
  - 1. multiprogramming: organizes jobs, so CPU always has one to execute (via jobs scheduling)
  - 2. Timesharing: CPU switches jobs so frequently that users can interact with each job.
- Virtual memory allows execution of processes not completely in memory.
- Dual-mode: usermode and kernel mode / modebit provided by hardware
- multi-mode: virtual machine manager
- Timer: prevents infinite loop and process hogging resources. (to generate an interrupt)
- process is a program in execution / active entity
- program is a passive entity stored on disk (executable file)
- Single threaded process has one program counter specifying location of next instruction to execute.
- multi threaded process has one program counter per thread.
- memory management allows keeping several programs in memory
- cache coherency: must provide the most recent value in their cache
- buffering: storing data temporarily while it is being transferred
- spooling: overlapping of output of one jobs with input of other jobs
- protection: any mechanism for controlling access of processes or users to resources defined by the c
- Security: defense of the system against internal and external attacks.
- privilege escalation: allows user to change to effective ID with more rights
- Emulation: enables one computer system to behave like another computer system
- virtualization: refers to the act of creating a virtual version of something
- public cloud: available via internet to anyone willing to pay.
- private cloud: run by a company for the company's own use.
- hybrid cloud: includes both public and private cloud components.
- Software as a service: word processor
- platform as a service: database server
- infrastructure as a service: storage available for backup

Scanned with CamScanner

Ch.2

- OS functions that are helpful to the user:

(1) user interface (2) program execution (3) I/O operations (4) file system manipulation
(5) communication (6) error detection (7) resource allocation (8) accounting (9) protection & security

- Library function (level 2) ⟶ system call API (level 1) ⟶ system call (level 0)
- API lets the OS manages the requests so the software is less likely to affect other software when it crashe
  ↳ (1) portability (2) ease of use (3) security & protection (4) efficiency (5) access
- System call interface maintains a table indexed according to these numbers.
- System call parameter passing:
  (1) pass parameters in registers
  (2) block address passed as a parameter in a register
  (3) parameters pushed onto the stack by the program and popped off the stack by the OS
- Registry: used to store and retrieve configuration information
- mechanisms determine how to do something, policies decide what will be done
- high-level language easier to port to other hardware but slower.
- OS structure:
  (1) Simple structure (most functionality in the least space)
  (2) Layered approach (0 → hardware, n → user interface) (each layer call the layer under it only)
  (3) Microkernel system structure (communication using message passing) (reliable, secure and extendable)
  (4) Modules (flexible) (call any module anytime, not like layered)
  (5) Hybrid systems
- iOS:
  (1) cocoa touch (2) media services (3) core services (4) core OS
- android based on linix kernel but modified
- Debugging is finding & fixing error & bugs
- failure of an application can generate core dump file capturing memory of the process.
- OS failure can generate crash dump file containing kernel memory.
- System performance:
  (1) trace listings (2) profiling

Sysgen program obtains information concerning the specific configuration of the hardware system
boot lock loads bootstrap loader from disk
GRUB allows selection of kernel from multiple disks, versions and kernal options.

Scanned with CamScanner

ch.3

- program becomes process when executable file loaded into memory
- one program can be several processes.
- process includes multiple parts:

(1) text section (2) program counter (3) Data section (4) Stack & heap

- process state:

new → running → waiting → ready → terminated

- process control block (PCB):

(1) process state (3) CPU registers (5) Memory management information
(2) process counter (4) CPU scheduling information (6) Accounting information (7) I/O info.

- Job queue: set of all processes in the system
- Ready queue: set of all processes residing in main memory, ready and waiting to execute
- Device queues: set of processes waiting for an I/O device
- process scheduler select among available processes for next execution on CPU
- Long term scheduler (or job scheduler) selects which processes should be brought into the ready queue → slow
- Short term scheduler (or CPU scheduler) selects which processes should be executed next and allocates CPU → fast
- Long term scheduler controls the degree of multiprogramming
- I/O bound process: spends more time doing I/O than computations, many short CPU bursts
- CPU bound process: spends more time doing computations, few very long CPU bursts → swapping
- Medium term scheduler: can be added if degree of multiple programming needs to decrease
- context switch: when CPU switches to another process, the system must save the state of the old process and load the saved state for the new process
- context-switch time is overhead, the system does not useful work while switching
  - the more complex the OS & the PCB, the longer the context switch
- parent process create children processes forming a tree of processes
- fork() system call creates new process
- exec() system call used after a fork() to replace the process' memory space with a new program
- process executes last statement and asks the OS to delete it (exit())
- parent may terminate execution of its children processes (abort())
  - all children must also terminated - cascading termination
- processes within a system may be independent or cooperating
  - (1) information sharing (2) computer speedup (3) modularity (4) convenience
- Two models of IPC: (1) Message passing (2) Shared memory

Scanned with CamScanner

- unbounded-buffer: places no practical limit on the size of the buffer
- bounded-buffer: assumes that there is a fixed buffer size
- Message passing:

(1) Direct communication: automatically / one pair / one link / bi-directional

(2) indirect communication: shared mailbox / many / several links / unidirectional or bi-directional

- Sychonization

(1) blocking (synchronous)

(2) Non-blocking (asynchronous)

- if both send and receive are blocking, we have rendezvous
- communications in client-server systems:

(1) Sockets: endpoint for communication (IP address + port number)

↳ to allow a client and a server on the same host to communicate, loopback is used to refer to it

↳ a. connection oriented (TCP) b. connectionless (UDP) c. Multicast socket

(2) Remote procedure calls

(3) pipes (ordinary pipes / named pipes)

(4) Remote method invocation (Java)

Scanned with CamScanner

- concurrent access to shared data may result in data inconsistency

Each process must ask permission to enter critical section. in entry section, may follow section with exit section, then remainder section

Critical section:

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (true);
```

Solution:

(1) Mutual execution: if a process is executing in its critical section, then no process can execute

(2) progress: الى هيدخل الـ critical section لازم يبقى من regular أو critical section ده حاسس قبل كده waiting الـ

(3) Bounded waiting: كل الـ processes لازم تتساوى في عدد مرات دخول الـ critical section

---

(1) Peterson's solution (software solution):

- Two process solution
- int turn; Boolean flag[2];
- Mutual execution ✓ progress ✓ bounded waiting ✓

(2) Synchronization hardware:

a. test-and-set instructions
   - bounded waiting x

b. compare-and swap instructions
   - bounded waiting x

c. Mutex locks (hardware solution بتاعة) (require busy waiting - spinlock)
   - acquire() & release() must be atomic (non-interruptible)

(3) Semaphore: (does not require busy waiting)

- wait() = P() Signal() = V()
- counting semaphore — unrestricted domain
- Binary semaphore — 0 & 1
- block — place the process invoking the operation on the appropriate waiting queue
- wakeup — remove one of processes in the waiting queue and place it in the ready queue

---

- Deadlock: two or more processes are waiting indefinitly for an event that can be caused by only one of the waiting processes سيب وأنا اسيب
- Starvation: a process may never be removed from the semaphore queue in which it is suspended
- priority inversion: scheduling problem when lower-priority process holds a lock needed by higher-priority proce

Scanned with CamScanner

- classical problems of synchronization:

(1) Bounded-buffer problem

2) Reader and writer problem

(3) Dining-philosophers problem

Philosopher:

```
do{
    wait(chopstick[i]);
    wait(chopstick[(i+1)%5]);
    Signal(chopstick[i]);
    Signal(chopstick[(i+1)%5]; }while(true);
```

producer:

```
do{
    wait(empty);
    wait(mutex);
    Signal(mutex);
    wait(full);
  }while(true);
```

consumer:

```
do{
    wait(full);
    wait(mutex);
    Signal(mutex);
    Signal(empty);
  }while(true);
```

reader:

```
do{
    wait(mutex);
    read_count++;
    if(read_count==1)
       wait(rw-mutex);
    Signal(mutex);
    wait(mutex);
    read_count--;
    if(read_count==0)
       Signal(rw-mutex);
    Signal(mutex)
do{
    [ ] 1
    // critical section
    [ ] 2
  }while true
```

writer:

```
do{
    wait(rw-mutex);
    Signal(rw-mutex);
  }while(true);
```

a. Software

1 while(test_and_set(&lock));

2 lock=false;

b. Hardware

1 acquire(mutex);

2 release(mutex);

c. Semaphore

1 wait(semaphore);

2 release(semaphore);

Scanned with CamScanner