OS_110_CH3

Process Concept

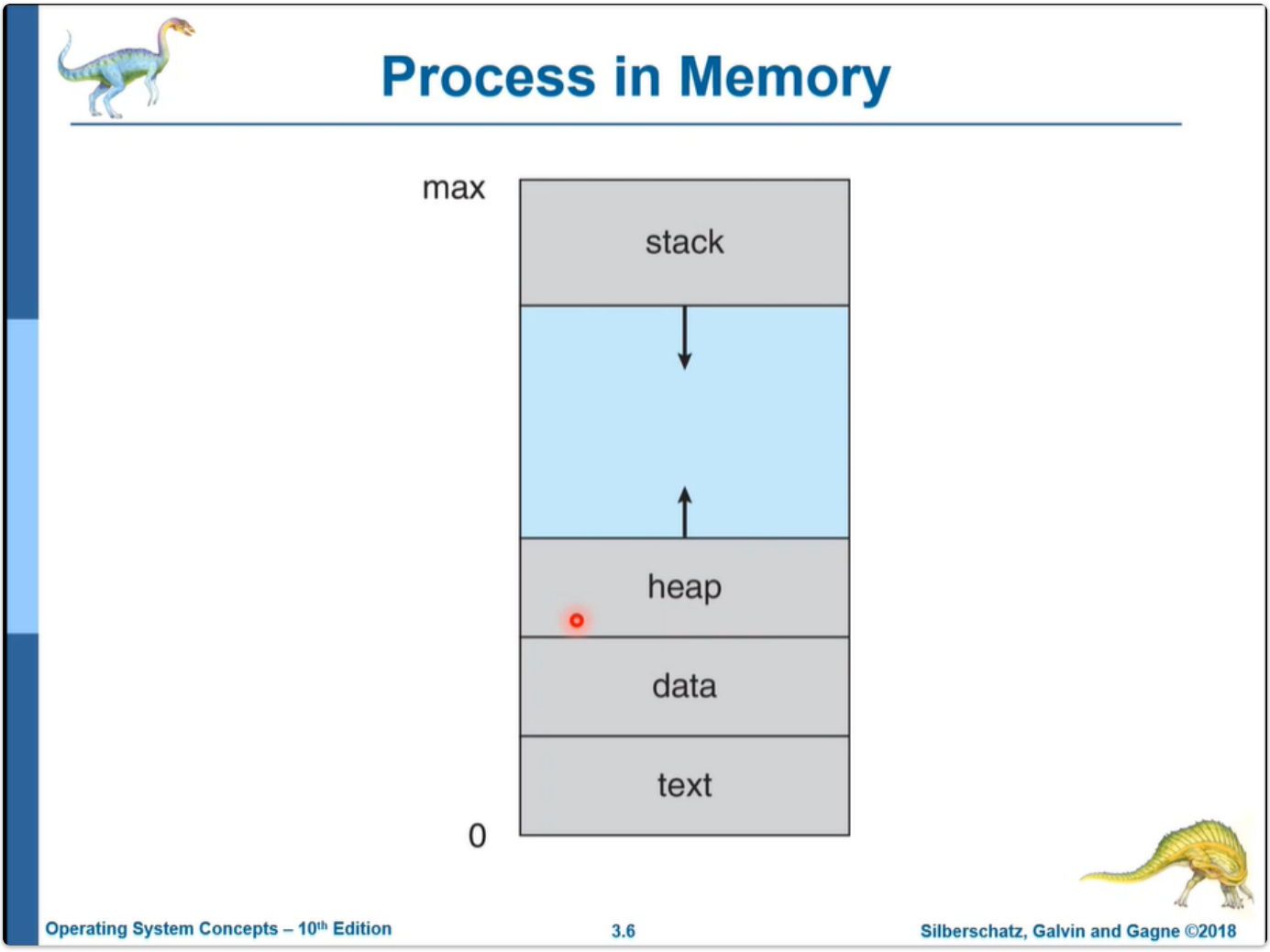
Prcess

- Process : active entity，在運作的行程，在記憶體裡面被執行

- Program : passive entity，單純程式碼，二進位的方式被儲存在硬碟裡

- includes
  + Code - text section
  + Data section - global variables
  + Stack - local variables and functions
  + Heap - dynamic variables
  + Program counter - 程式執行中，Program counter 會指向下一行要執行的程式碼

+ Resources



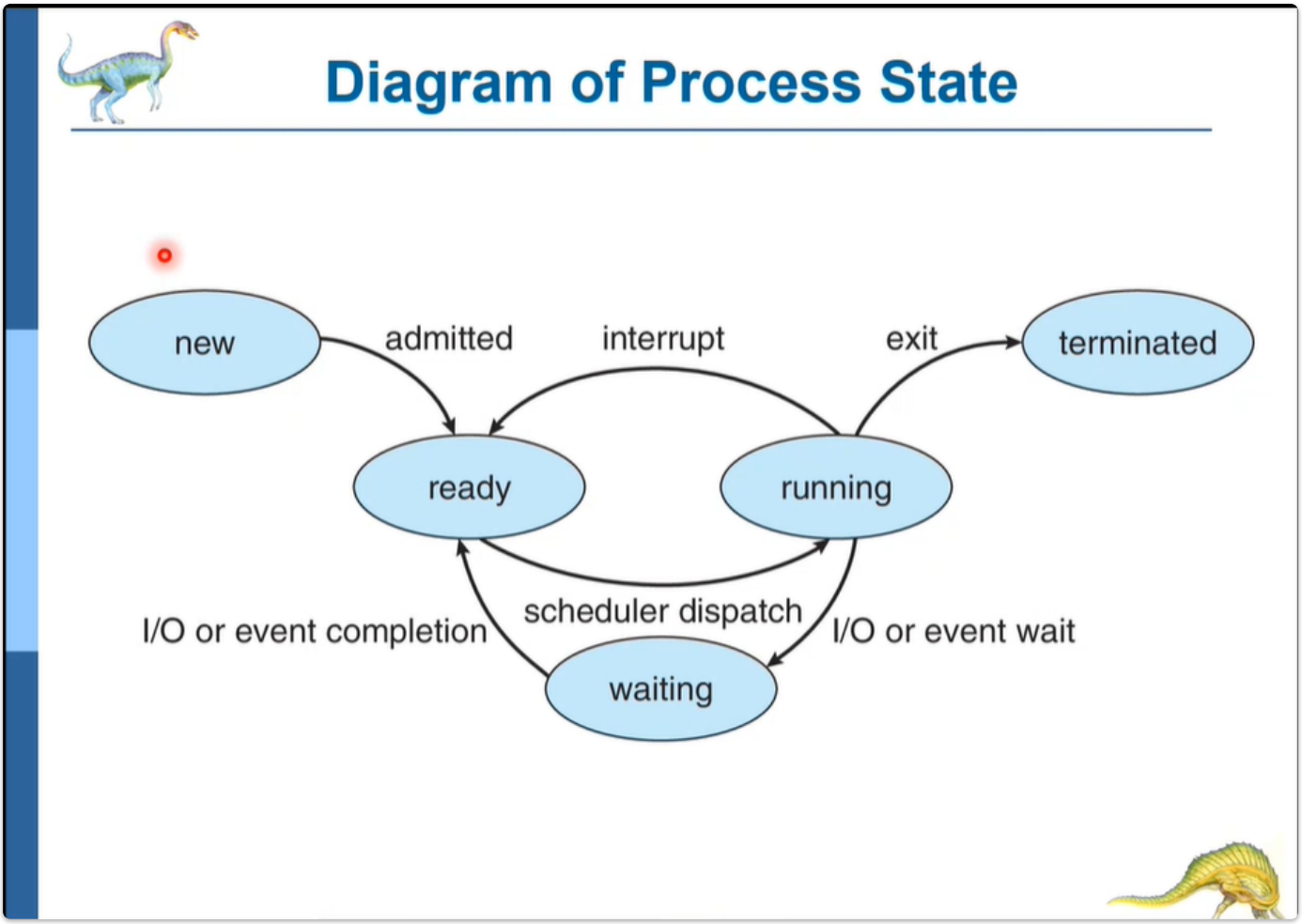
Threads

- 所有的線程屬於同一個 process 時，會共用 code section, data section, OS resources
- Register set, Stack, Program counter 不會共用

Process State

- New 創建一個新的 Process，將程式碼配置到 memory 裡面
- Ready 等著 OS 排程到 CPU 裡面去執行
- Running CPU 執行
- Waiting 當中斷發生

- Terminated 將所有資源釋放



Process Control Block (PCB)

- Process state
- Program counter
- CPU register
- CPU scheduling information
- Memory-management information
- I/O status information
- Accounting information

Context switch

令多個行程(process)可以分享單一CPU資源的計算過程。要交換CPU上的行程時，必須先行儲存目前行程的狀態，再將欲執行的行程之狀態讀回CPU中。

下列兩項都是 Context switch

- 在運作的程式中被 interrupt 的時候，會將狀態儲存到 PCB 裡面
- 而要再執行時，會重新載入 PCB

Process Scheduling

Process Scheduling Queues

- Job queue : all processes
- Ready queue : ready and waiting to exec
- Device queue : waiting I/O device

Schedulers

- Short-term (CPU scheduler) (Ready → Run)
- Long-term (Job scheduler) (New → Ready)
- Medium-term (Ready → Wait) swapped in/out memory

Long-Term Scheduler

- Control degree of multiprogramming 控制電腦裡面的 process 數量
- 選擇一個好組合，使得 CPU 跟 I/O overlap 提高

Short-Term Scheduler

- 執行頻率非常高
- 效率必須非常 well
- 演算法不能太複雜

Medium-Term

- swap out
- swap in
- 目的 : improve process mix, free up memory
- virtual memory

Operations on Process

Tree of Process

每個 process 都有一個 unique 的 pid

Process Creation

- Resource sharing
  - share all 親子共享
  - subset 靜態記憶體共享
  - no share 非共享
- 執行順序
  - 親子同時執行，使 OS 自己決定誰先誰後都可以
  - 子必須先執行完
- 空間
  - 完全複製 (不同記憶體位置)
  - 直接 load 程式碼進去 (同一個)

Linux/UNIX Process Creation

System call

- fork
  - 會完全複製程式碼
  - Child 執行完會回傳 1
  - Parent 會知道 Child 的 pid
- execlp
  - load a new binary file into memory，可以執行做不一樣的程式碼
  - 重置該 process 執行的東西
- wait
- exit
  - dellocated by the OS

Interprocess Communication

- IPC：溝通的機制
- Purposes
  - information sharing
  - computation speedup
  - convenience
  - modularity

Communication Methods

如果在用 pointer → Shared memory
function → Message passing

- Shared memory
  - 優：快
  - 缺：user synchronization (同步問題)
  - 人少
- Message passing
  - 缺：慢
  - 優：同步
  - 人多
- Sockets
  - connect by IP & port
  - exchange **unstructured stream of bytes**
- Remote Procedure Calls
  - 可以呼叫其他 process 的 procedure call
  - 參數跟回傳值都透過 message passing

Shared Memory

- 需要有一個記憶體空間，所以要跟 OS 配置
- 使用者自己決定如何使用

- 當資料被同時讀取寫入時，會有同步衝突問題
  - Consumer & Producer Problem

Message-Passing

- 溝通和同步的機制
- IPC
  - Send
  - Receive
- 需要有一個 link 的存在
  - 硬體
  - 邏輯、程式

Direct communication

- 必須明確定義雙方
  - Send
  - Receive
- Links 會自動建立
- one to one
- 單向的
- 對方不能交換，除非要重新來過

Indirect communication

- 非指定 ID
- 像是送到一個 mail box 裡面，讓別人去拿
- 可以 many to many
- OS 管理 mail box
- Mailbox sharing

    - 可能會有多個人拿到
    - 解決辦法
        - 一個 link 只能有兩個人用
        - 限制同一時間只有一個人接收
        - 當兩個同時要接收時，會將其中一個延遲，再讓它接收

Synchronization

- blocking 同步化溝通，直到收到回傳
- non-blocking 不同步溝通，放著就好

Buffer implementation

- Zero
- Bounded
- Unbounded

Sockets

- 透過 IP address 跟 port number 溝通

Remote PRocedure Calls (RPC)

- 是一個 remote 的 function call
- Stubs
    - client-side proxy for the actual procedure on the server

Client and Server Stubs

- Packs parameters into a message (i.e. parameter marshaling)