

# Chapter 4: Process Management

## Operating system

Dr. Do Tien Dung  
[dungdt@ptit.edu.vn](mailto:dungdt@ptit.edu.vn)



Posts and Telecommunications  
Institute of Technology  
Faculty of Information Technology 1



Faculty of Information Technology 1  
Posts and Telecommunication Institute of Technology

August 15, 2022

## Definitions

What is the process?

Status of process

Information describing process

Table and process list

Process operations

## Thread

Definitions

Resources of process and thread

Advantage of multithread model

Thread at kernel and user level

## Schedule

Definitions

Scheduling types

1. Definitions
2. Thread
3. Scheduling
4. Synchronization
5. Deadlock

**Process** *is the running program.*

Program	Process
Static entity	Dynamic entity
Do not own any specific resources	Allocated some resources to store process and run commands

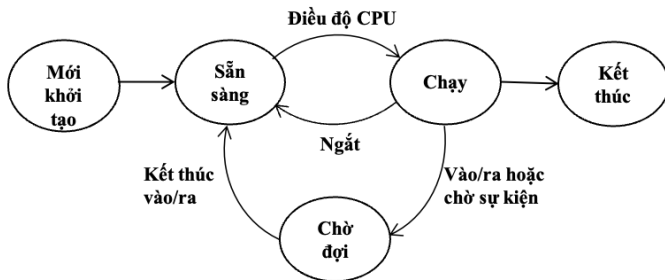
► Process is started when a program is loaded into memory:

- User process
- System process

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time	PID	User
WindowServer	21,2	37:43:17,44	25	140	Apple	19,9	4:08:25,06	362	_windowserver
kernel_task	11,7	23:43:41,58	563	1340	Apple	0,0	0,00	0	root
Safari Graphics and Media	6,0	2:17:58,10	18	164	Apple	0,0	1:18,67	45904	dinhxuantrunq
launchd	4,5	3:06:10,31	3	1	Apple	0,0	0,00	1	root
coreaudiod	2,5	6:22:48,77	12	103	Apple	0,0	0,00	399	_coreaudiod
screencapture	2,0	0,51	3	0	Apple	0,0	0,00	98518	dinhxuantrunq
https://www.youtube.com	1,8	4:48:39	9	24	Apple	0,0	0,00	87145	dinhxuantrunq
Activity Monitor	1,7	5,48	5	1	Apple	0,0	0,00	98516	dinhxuantrunq
Microsoft Word	1,0	7:37,22	38	27	Apple	0,0	1,48	92547	dinhxuantrunq
https://www.icicib22.scrs.in	0,9	55,46	4	24	Apple	0,0	0,00	97975	dinhxuantrunq
loginwindow	0,9	2:13:53,12	14	726	Apple	0,0	1,27	365	dinhxuantrunq
OneDrive	0,8	3:05:29,57	17	30	Intel	0,0	0,00	10545	dinhxuantrunq
sysmond	0,8	1:04:01,33	3	0	Apple	0,0	0,00	823	root

- ▶ During proceeding, the process change status
- ▶ Status of process is a part of current actions of process, including 5 status:
  - **New:** the process is starting
  - **Ready:** the process wait for CPU allocation to be proceed
  - **Running:** commands is proceeded by CPU
  - **Waiting:** the process is waiting for a event happen (blocked)
  - **Terminated:** the process finish but is not deleted yet

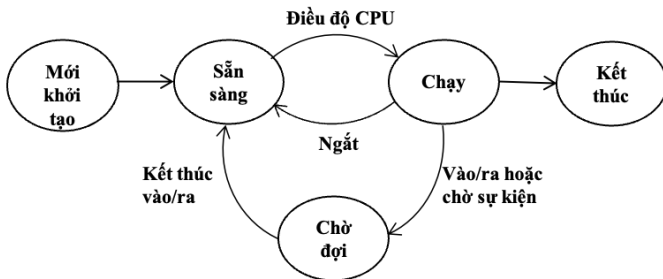
## Transition between status:



- ▶ **Start -> Ready**: the process is started and loaded in memory, waiting for CPU allocation.
- ▶ **Ready -> Run**: depending on CPU schedule of OS, process is allocated CPU and change to running status

# Definitions (cont.)

Status of process



- ▶ **Run -> Ready:** OS allocate CPU for another process. Depending on scheduling/deadlock, process change to "Ready" status and wait for CPU allocation to continue running.
- ▶ **Run -> Wait:** Process runs when an event happens, change to "Wait" status to wait for CPU allocation.
- ▶ **Run -> Finish:** a process finish proceeding

The process information is stored in a data structure called Process Control Block (PCB)

► Main information in PCB:

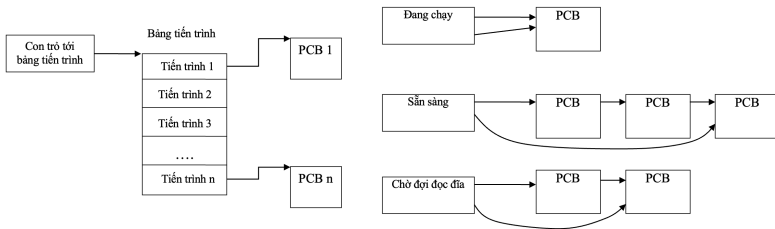
- Process Identifier(PID)
- Process status: one of five status
- CPU registers:
  - Command pointer register: pointing to the next command.
  - Stack pointer register: save parameters/function status at calling time/thủ tục của chương trình
  - Condition and status registers
  - Multifunction registers
- Memory management Information
- Usable resources Information

process state
process number
program counter
registers
memory limits
list of open files
...

## Roles of PCB:

- ▶ *Store information* for OS to restore the content of registers, allow restoring the status before stopping.
- ▶ *Schedule Information*: the priority of process, position in stack, the amount of using resourcess.
- ▶ *Memory information of process*: the position in the memory
- ▶ *List of other resources*: opening file, I/O that process is using
- ▶ *Statistic information for management*: CPU using time, time limit

- ▶ OS stores and identifies the position of PCB by using process table having pointer to PCB of whole processes system
- ▶ PCB of process are linked to a list, each list are processes with status.



The operations of process management include:

- ▶ Create new process
- ▶ End process
- ▶ Change between processes
- ▶ Scheduling
- ▶ Synchronization
- ▶ Ensure information exchange between processes

## Create new process:

- ▶ To create a new process, OS does following steps:
  - *Give ID* to a new process and create a cell in process table
  - *Create memory space* for a process and PCB
  - *Initiate PCB*: OS give values for components of PCB
  - *Link process PCBL* in the management list, for instance in the in proceeding process

## End process:

- ▶ Normal end: request OS to close itself by system request calling `exit()`.
- ▶ A process is forced exiting in following cases:
  - Forced quit by father process
  - Due to errors
  - Request more memories than available memories of system
  - Proceed longer than time limit
  - Due to management system or OS ends it

### Exchange between processes - Exchange context:

- ▶ In processing, CPU can change from this process to another one.
- ▶ Information about current process in PCB is context.
- ▶ Process exchange happens at:
  - Having interrupt: due to clock or I/O interrupt
  - Process calls call system
- ▶ When changing another process, process saves in PCB
- ▶ When restoring CPU allocation, context is restored from PCB into registers and table.

## Exchange between processes - Exchange context:

Context include which information?

- ▶ Simple case: OS proceed I/O process, then continue proceeding process:
  - Context includes information that could be changed by interrupt function
  - Registers content, CPU status
- ▶ Sophiscated cases: After interrupting, OS proceed another process
  - Change process status
  - Update information in PCB
  - Change PCB of process into list of new status
  - Update PCB of new chosen process
  - Update register conten, and CPU status
- ▶ Process exchange needs time

- ▶ Process is considered from 2 aspects:
  - Process is resource owner
  - Process is an unit proceeding calculations
- ▶ Previous OS: each process is one processing unit (each process can not proceed more than one work at the same time)
- ▶ Modern OS: allow separating role of instruction proceeding of process.

*Each unit does commands of process, which means that a series of command is allocated CPU to proceed independently a **thread***

- ▶ New OSs support multithreading:
  - A process may have many threads
  - Allow many series of commands, doing many tasks at the same time.



tiến trình gồm một luồng



tiến trình gồm nhiều luồng

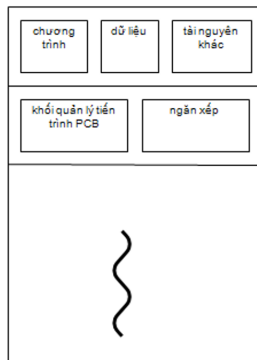
} = chuỗi lệnh

- ▶ In multithread system, a process is an unit of resource allocation of OS
- ▶ Each process has share resources:
  - Process memory(logic): contain program (commands), data of process.
  - Other resources: opening file, I/O device or ports

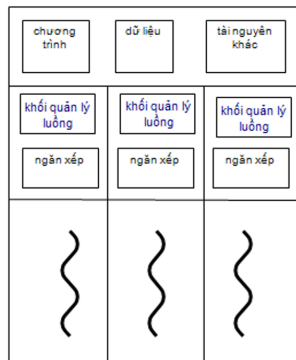
### Single thread and multithread process.

Single thread process	Multithread process
Process has PCB management contain information of status process, register	Each thread need to manage command pointer, content registers
Stack has parameters, routines, function, sub-program	Thread has its own state storing inside thread management Thread has its own stack
When proceeding, thread will control content of registers and command pointers	All threads of process share memory space and resources

## Resources of process and thread.



Tiến trình đơn luồng



Tiến trình nhiều luồng

### **Advantage of multithread model:**

- ▶ Improve efficiency and save time
- ▶ Easy to share resources and information
- ▶ Improve response
- ▶ Make use of many CPU architecture
- ▶ Easy to organize program

- ▶ Can create and manage thread at two levels:
  - User level
  - Kernel level
- ▶ Thread in user level: is created and managed without the support of OS
- ▶ Thread in kernel level: is created and managed by OS

## Thread in user level:

- ▶ Application create and manage
- ▶ CPU allocation is done for whole process
- ▶ OS consider process as an unit with only one status
- ▶ Advantages:
  - Save time because exchanging threads does not require to change to kernel mode
  - Application can have its own schedule, does not depend on schedule of OS
  - May use when OS does not support multithreads
- ▶ Disadvantages
  - Do not use the advantage of responses of multithread model
  - Do not use the many CPU architecture

### Thread in kernel level:

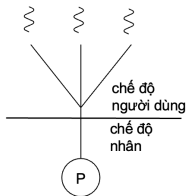
- ▶ Thread in kernel mode is created and managed by OS
- ▶ OS provide programming interface
- ▶ Provide system calls that application request to create, delete thread and change parameters relating
- ▶ Advantages:
  - Improve the response and the ability of proceeding multiple threads simultaneously
- ▶ Disadvantages
  - Low speed due to creating and changing threads are in kernel mode
- ▶ Windows and Linux support thread at kernel mode

### Combine thread at kernel and user mode:

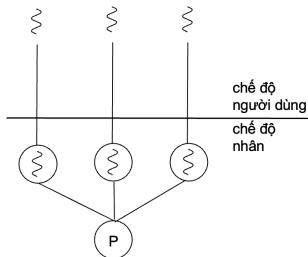
- ▶ May use thread at kernel and user mode
- ▶ Theo các tổ chức này:
  - Thread at user mode is created in user mode based on library of application
  - Thread at user mode is mapped to the number equivalent or fewer than threads at kernel mode
  - Number of thread at kernel depends on systems, for instance, system with many CPU will have more thread at kernel.

# Thread (cont.)

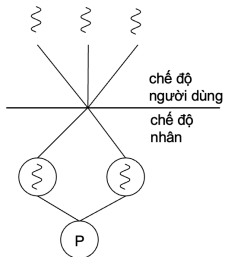
Thread at kernel and user mode



a) Mô hình mức người dùng



b) Mô hình mức nhân



c) Mô hình kết hợp

~ LUÔNG mức người dùng

~ LUÔNG mức nhân

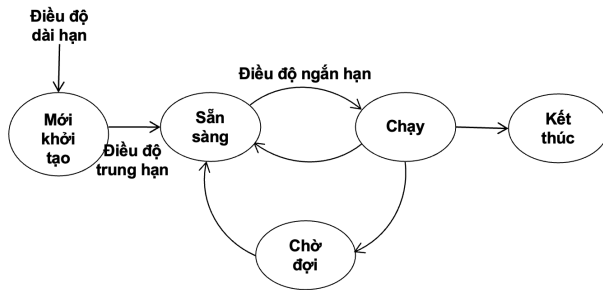
P tiến trình

- ▶ *In multi program-supported system, many process can exist and run at the same time.*
- ▶ *Multiprogramming techniques have many advantages because they allow efficient use of CPU, meeting the user's computing needs.*
- ▶ *However, it raises many sophisticated issues to OS*

*Scheduling* decide which resources and the time that a process uses.

- ▶ Scheduling of CPU is to decide the number and time using CPU
- ▶ Scheduling process and thread:
  - Previous system: process: process is main unit, scheduling with process
  - Thread support system: thread at kernel mode is a unit allocated by OS, not process.
  - Process schedule is used equivalently to thread schedule

## Long term shedcule and short term schedule



### ► Long term schedule:

- At the beginning of process creating
- OS decide whether process is added in working list?
- If accepted, system will have a new process. Otherwise, it waits for new time to create and proceed.

- Affect to multiprogramming mode
- ▶ Midterm schedule:
  - Decide whether a process is allocated memory to use?
- ▶ Short-term schedule:
  - Decide which process is allocated CPU to use
  - Proceed with process at ready state

### Preemptive and non preemptive schedule:

- ▶ Preemptive:
  - OS use a mechanism to redeem CPU of running process
- ▶ Nonpreemptive:
  - A process at running state will use CPU until below circumstances:
    - ▶ The process ends
    - ▶ The process change to waiting state due to I/O interrupt
  - Cooperative Schedule: is proceed when a process cooperate and give away CPU
  - If the process is not cooperative, CPU is used unlimited and other process is not allocated CPU(Windows 96, NT)

### **Advantages of preemptive schedule:**

- ▶ OS is more active, does not depend on the process operation
- ▶ Ensure real time sharing
- ▶ Require hardware having timer and some other supports
- ▶ Process management is more complicated

### Some used criteria:

- ▶ Number of used processes:
  - The number of used processes in amount of time
  - Measure the efficiency of system
- ▶ CPU efficiency
  - Try to let CPU resting time as less as possible
- ▶ Life time of process:
  - From start to end of process
- ▶ Waiting time:
  - Total time between waiting state and CPU allocation
  - Affect directly to process schedule

### Schedule criteria:

- ▶ Response time:
  - This is user criterion and often used in direct interact system.
- ▶ Predicting ability:
  - Life time, waiting time, response time should b stable, and do not depend on system load
- ▶ Fairness:
  - Processes with the same priority have to behave equally

### Schedule algorithms:

1. FCFS: First Come, First Served
2. RR: round robin
3. SPF: Shortest Job First
4. SRTF: Shortest Remaining Time First
5. Priority Scheduling

### FCFS: First Come, First Served

#### Principle:

- ▶ The process use CPU according to the order of appearing
- ▶ The process owns CPU until ending or waiting at I/O interrupt

Tiến trình		Thời gian	
P1		24	
P2		3	
P3		3	

0	24	27
24	3	3
P1	P2	P3

Waiting time of P1, P2, P3 are: 0, 24, 27.

Average waiting time  $(0 + 24 + 27)/3 = 17$

## FCFS: First Come, First Served

### Principle:

- ▶ The process use CPU according to the order of appearing
- ▶ The process owns CPU until ending or waiting at I/O interrupt
- ▶ Characteristics
  - Simple, easy to use
  - Short waiting time process has to wait same as long waiting time process

### RR: round robin

#### Principle:

- ▶ Each process is allocated the amount of  $t$  time to run
- ▶ When time ends, the process is redeemed CPU and put to the end of stact
- ▶ If there are  $n$  processes, waiting time is  $(n-1)t$

Ví dụ:

Tiến trình		Độ dài chu kỳ CPU	
P1		10	
P2		4	
P3		2	

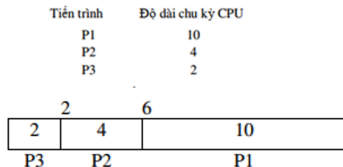
2	4	6	8	10	12	14
2	2	2	2	2	2	2
P1	P2	P3	P1	P2	P1	P1

### SPF: Shortest Job First

#### Principle:

- ▶ Choose in the start the process have shortest CPU time to allocate CPU
- ▶ If there are many processes having same CPU using time, choose the front process
- ▶ Average use time is much shorter than FCFS

Ví dụ:



### **SRTF: Shortest Remaining Time First**

#### **Principle:**

- ▶ When having a new process in stack, OS compare remaining CPU time of running process with remaining time of new process
- ▶ If time is shorter, OS redeem CPU of running process and allocate to new one

Ví dụ:

Tiến trình	Thời điểm xuất hiện	Độ dài chu kỳ CPU
P1	0	8
P2	0	7
P3	2	2

### Priority Scheduling Schedule:

- ▶ Each process has one priority
- ▶ Higher priority will be allocated first
- ▶ Same priority processes will apply FCFS
- ▶ Priority is determined by many criteria

Ví dụ:

Tiến trình	Mức ưu tiên
P1	4
P2	1
P3	3

P2	P3	P1

## Chapter 4 Process management

- ▶ Definitions
- ▶ Thread
- ▶ Scheduling

## Chapter 4 Process management

- ▶ Synchronize processes
- ▶ Tình trạng bế tắc và đói

1. **Câu 1:** Cho các tiến trình với thời điểm xuất hiện, thời gian (chu kỳ) CPU tiếp theo và số ưu tiên như trong bảng sau (số ưu tiên nhỏ ứng với độ ưu tiên cao). Vẽ biểu đồ thể hiện thứ tự và thời gian cấp phát CPU cho các tiến trình sử dụng thuật toán :

Tiến trình	Thời điểm xuất hiện (c)	Thời gian (độ dài)	Số ưu tiên
P1	0	10	1
P2	2	8	3
P3	4	5	2
P4	5	3	2

- Điều độ theo mức ưu tiên không có phân phối lại
- Điều độ ưu tiên tiến trình ngắn nhất
- Điều độ ưu tiên thời gian còn lại ngắn nhất

Tính thời gian chờ đợi trung bình cho từng trường hợp