



COMPUTER ENGINEERING



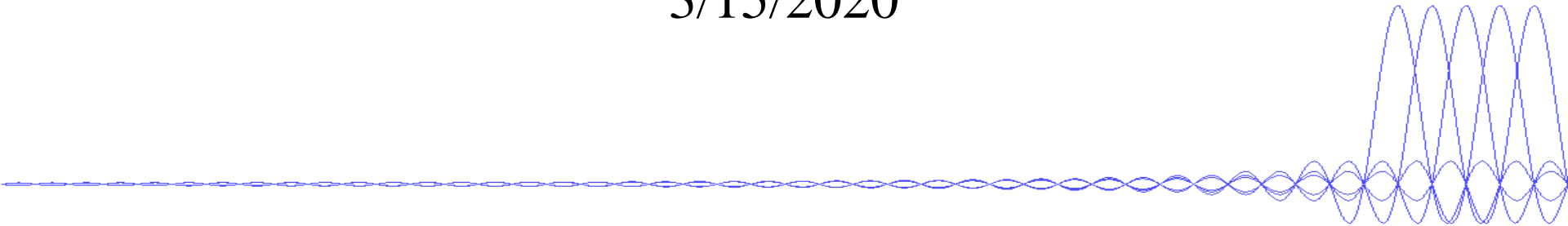
**UIT**  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# HỆ ĐIỀU HÀNH

## Chương 4 (3)

### Định thời CPU

3/15/2020





# Câu hỏi ôn tập chương 4 (2)

## ■ Các giải thuật định thời

- Round-Robin (RR)
- Highest Response Ratio Next (HRRN)
- Multilevel Queue
- Multilevel Feedback Queue



## Nội dung chương 4 (3)

- Định thời tiêu trình (Thread scheduling)
- Định thời đa bộ xử lý (Multiple-processor scheduling)
- Định thời theo thời gian thực (Real-time CPU scheduling)
- Định thời trên một số hệ điều hành
  - Linux
  - Windows
  - Solaris



# Định thời tiến trình

- Trên các hệ điều hành hiện đại có hỗ trợ tiến trình, tiến trình được định thời, không phải tiến trình.
- Có sự phân biệt giữa tiến trình người dùng và tiến trình hạt nhân khi định thời.
- Tiến trình người dùng được định thời thông qua các thư viện quản lý tiến trình:
  - Phạm vi định thời là bên trong tiến trình (process-contention scope - PCS)
  - Thường được thực hiện bằng cách thiết lập độ ưu tiên (bởi người lập trình).
- Tiến trình hạt nhân được định thời trên tất cả các CPU khả dụng. Phạm vi định thời là toàn hệ thống (system-contention scope - SCS).



# Định thời đa bộ xử lý

- Định thời CPU trở nên phức tạp hơn khi hệ thống có nhiều bộ xử lý.
- Khái niệm đa bộ xử lý có thể là một trong các dạng sau:
  - CPU có nhiều lõi vật lý (Multicore CPUs)
  - CPU có nhiều luồng xử lý trên một lõi (Multithreaded cores)
  - Hệ thống NUMA (non-uniform memory access)
  - Đa xử lý không đồng nhất (Heterogeneous multiprocessing)
- Có hai cách tiếp cận phổ biến: đa xử lý bất đối xứng (asymmetric multiprocessing) và đa xử lý đối xứng (symmetric multiprocessing - SMP).



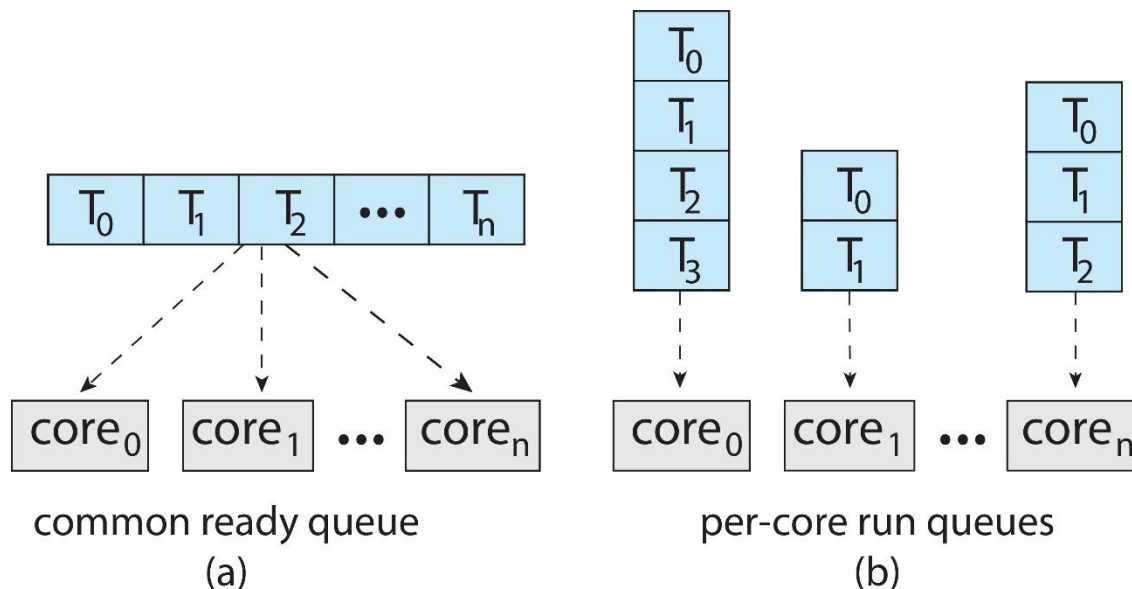
# Đa xử lý bất đối xứng

- Tất cả các thao tác lập lịch, xử lý I/O được thực hiện bởi một bộ xử lý – master server.
- Các bộ xử lý còn lại chỉ thực thi user code.
- Ưu điểm:
  - Đơn giản: chỉ một bộ xử lý truy xuất dữ liệu hệ thống, không cần chia sẻ dữ liệu.
- Nhược điểm:
  - Master server có thể bị nghẽn cổ chai (bottleneck), làm giảm hiệu năng của hệ thống



# Đa xử lý đối xứng

- Mỗi bộ xử lý tự định thời cho chính nó.
- Hai hướng tiếp cận để tổ chức các tiểu trình cần định thời:
  - Tất cả tiểu trình nằm trong cùng một hàng đợi ready (a)
  - Mỗi bộ xử lý tự tổ chức hàng đợi của riêng nó (b)





# Đa xử lý đối xứng

- Tất cả tiểu trình nằm trong cùng một hàng đợi ready:
  - Tiểu trình có thể không được bộ xử lý nào chọn ?
  - Xuất hiện vùng tranh chấp: Nhiều bộ xử lý có thể chọn định thời cùng một tiểu trình => Cần có cơ chế kiểm tra và khóa (lock) việc truy xuất tiểu trình => Hiệu năng hệ thống có thể giảm do nghẽn cổ chai.
- Mỗi bộ xử lý tự tổ chức hàng đợi của riêng nó:
  - Hiệu năng không bị ảnh hưởng do các vấn đề khi dùng chung một hàng đợi => Hướng tiếp cận phổ biến trên các hệ thống SMP.
  - Vấn đề: Khối lượng công việc của các bộ xử lý khác nhau?





# Cân bằng tải (Load balancing)

- Một bộ xử lý có quá nhiều tải, trong khi các bộ xử lý khác rỗi => Cần đảm bảo các bộ xử lý đều được sử dụng hiệu quả.
- Mục tiêu của cân bằng tải là phân phối khối lượng công việc (workload) đều nhau cho các CPU.
- Có hai cách cân bằng tải:
  - Push migration: Một tác vụ đặc biệt sẽ kiểm tra định kỳ tải của từng CPU. Nếu tình trạng quá tải xuất hiện, hệ thống sẽ di chuyển (đẩy) tác vụ từ CPU bị quá tải sang các CPU khác.
  - Pull migration: CPU rỗi kéo (pull) tác vụ đang chờ từ CPU bận.



# Processor affinity

- Khi một tác vụ chạy trên một bộ xử lý, bộ nhớ đệm (cache) của bộ xử lý đó lưu trữ dữ liệu được truy xuất bởi tác vụ => tác vụ có affinity với bộ xử lý - “processor affinity”.
- Cân bằng tải sẽ ảnh hưởng đến processor affinity, cụ thể là khi một tác vụ được dời sang bộ xử lý khác:

- Cache của bộ xử lý mới phải nạp lại (repopulate)
- Cache của bộ xử lý cũ phải được giải phóng (invalidate)

=> Phí tổn

- Có 2 dạng processor affinity:
  - Soft affinity: Hệ thống sẽ cố giữ tác vụ chỉ chạy trên bộ xử lý đó (nhưng không đảm bảo).
  - Hard affinity: Cho phép tiến trình chọn một tập các bộ xử lý mà nó có thể chạy trên đó.



# Processor affinity

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtualizati...
Microsoft.Photos.exe	4872	Suspended	ntthien	00	0 K	Disabled
MicrosoftEdge.exe	5096	Suspended	n	00	0 K	Disabled
MicrosoftEdgeCP.exe	10172	Suspended	n	00	0 K	Disabled
MicrosoftEdgeSH.exe	8664	Suspended	n	00	0 K	Disabled
msdtc.exe	6760	Running	ORK ...	00	20 K	Not allowed
MsMpEng.exe	4816	Running	EM	00	91,228 K	Not allowed
Music.UI.exe	10584	Suspended	n	00	0 K	Disabled
NisSrv.exe	216	Running	L SER...	00	988 K	Not allowed
notepad++.exe	3328	Running	n	00	15,224 K	Disabled
o2flash.exe	9352	Running	EM	00	24 K	Not allowed
OfficeClickToRun.exe	3996	Running	EM	00	4,404 K	Not allowed
openvpn-gui.exe	9744	Running	n	00	88 K	Disabled
openvpnserv.exe	4284	Running	EM	00	16 K	Not allowed
POWERPNT.EXE	5924	Running	n	00	87,332 K	Disabled
POWERPNT.EXE	9792	Running	n	00	86,100 K	Disabled
PresentationFontCac...	7356	Running	L SER...	00	20 K	Not allowed
PrivacyIconClient.exe	1124	Running	n	00	5,820 K	Disabled
RAVBg64.exe	3560	Running	SYSTEM	00	116 K	Not allowed
RAVBg64.exe	8788	Running	ntthien	00	156 K	Disabled

Processor affinity

Which processors are allowed to run "POWERPNT.EXE"?

- ☒ <All Processors>
- ☒ CPU 0
- ☒ CPU 1
- ☒ CPU 2
- ☒ CPU 3

OK Cancel

End task



# Định thời theo thời gian thực

- Có nhiều thách thức do yêu cầu về tính chất thời gian thực.
- Có 2 dạng hệ thống thời gian thực:
  - Soft real-time systems: Các tác vụ quan trọng sẽ được cấp độ ưu tiên lớn nhất, nhưng không đảm bảo bất cứ điều gì khác.
  - Hard real-time systems: Tác vụ phải hoàn thành trong deadline của nó.



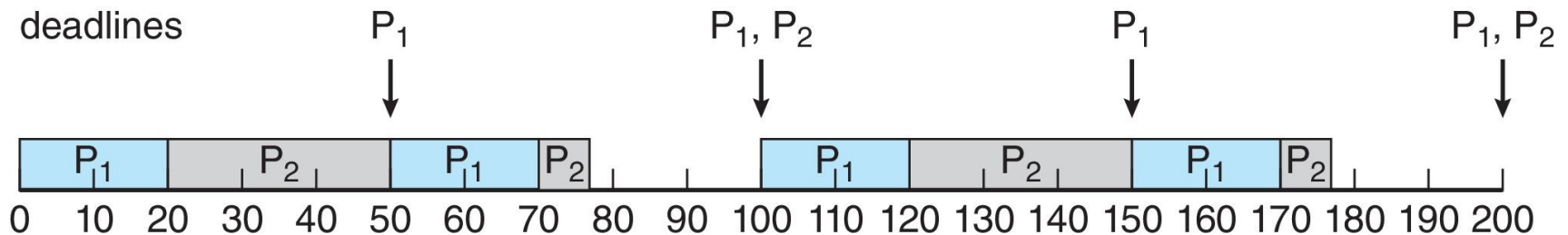
# Định thời theo độ ưu tiên

- Hệ thống thời gian thực phải phản hồi ngay lập tức yêu cầu CPU của một tiến trình => Bộ định thời phải hỗ trợ định thời theo độ ưu tiên với chế độ trung dụng.
- Tiến trình có thêm một đặc trưng mới: tính chu kỳ - yêu cầu CPU trong một khoảng thời gian cố định.
- Khi một tiến trình có chu kỳ yêu cầu CPU, nó có thời gian xử lý  $t$ , thời gian deadline  $d$  (thời gian nó sẽ được phục vụ bởi CPU) và thời gian chu kỳ  $p$ .
  - $0 \leq t \leq d \leq p$
  - Tần suất của tác vụ là  $1/p$ .



# Định thời Rate Monotonic

- Độ ưu tiên được gán dựa trên nghịch đảo của chu kỳ  $\Rightarrow$  Chu kỳ ngắn thì độ ưu tiên cao và ngược lại.
- P1 được gán độ ưu tiên cao hơn P2.





# Định thời trên Linux

- Nhân Linux 2.5 trở về trước sử dụng các phiên bản định thời UNIX tiêu chuẩn.
  - Không hỗ trợ tốt các hệ thống nhiều bộ xử lý.
  - Hiệu năng kém nếu có số lượng lớn các tiến trình trong hệ thống
- Nhân Linux 2.5 sử dụng bộ định thời  $O(1)$ :
  - Chạy với thời gian hằng số.
  - Định thời theo độ ưu tiên với chế độ trung dụng.
  - Có hai khoảng ưu tiên: time-sharing và real-time.
  - Giá trị số nhỏ hơn biểu diễn độ ưu tiên lớn hơn.
  - Hoạt động tốt với các hệ thống SMP nhưng đáp ứng kém với các tiến trình interactive.



# Định thời trên Linux: CFS

- Nhân Linux từ 2.6.23 sử dụng bộ định thời CFS (Completely Fair Scheduler)
  - Định thời theo lớp:
    - Mỗi lớp được gán một độ ưu tiên cụ thể.
    - Bộ định thời chọn tác vụ có độ ưu tiên cao nhất trong lớp có độ ưu tiên cao nhất.
    - Thời gian sử dụng CPU của mỗi tác vụ không dựa trên quantum time cố định mà dựa trên tỷ lệ giờ CPU.
    - Nhân Linux cài đặt sẵn 2 lớp: default và real-time. Các lớp khác có thể được thêm vào.





# Định thời trên Linux: CFS

## □ Thời gian sử dụng CPU:

- Được tính dựa trên giá trị nice được gán cho mỗi tác vụ, có giá trị từ -20 đến 19.
- Giá trị thấp hơn có độ ưu tiên cao hơn.
- Target latency – khoảng thời gian mà một tiến trình cần được chạy ít nhất một lần.
- Target latency có thể tăng lên nếu số lượng tiến trình tăng lên.

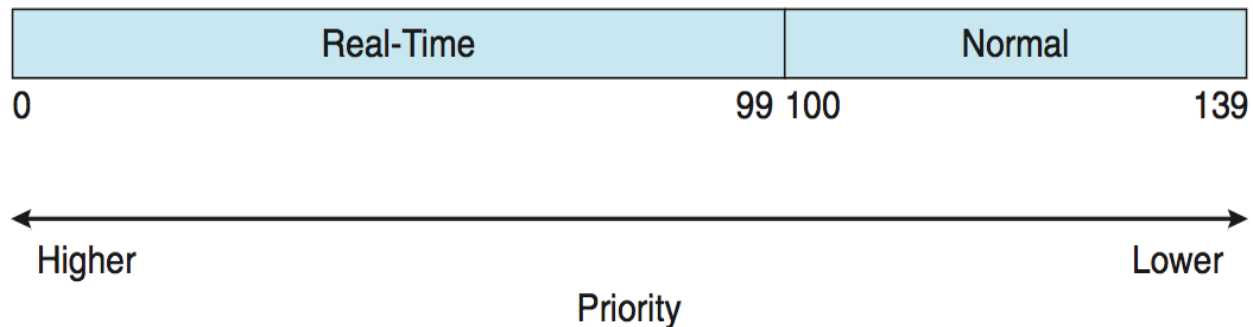
## □ CFS xác định tác vụ được thực thi kế tiếp qua virtual run time:

- Mỗi tác vụ có giá trị virtual run time riêng, được kết hợp với một hệ số đặc biệt dựa trên độ ưu tiên.
- Các tiến trình có độ ưu tiên bình thường có virtual run time tương đương với thời gian chạy thực tế.
- Chọn tiến trình có virtual run time nhỏ nhất để thực thi tiếp.



# Định thời trên Linux: Real-time

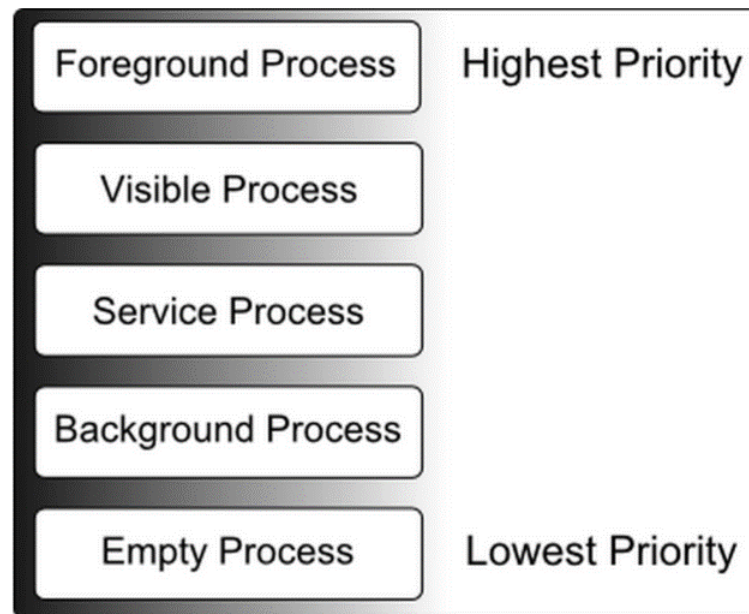
- Định thời real-time dựa trên tiêu chuẩn POSIX.
  - Các tác vụ real-time có độ ưu tiên tĩnh.
- Độ ưu tiên được chia thành 2 phần: real-time (từ 0 đến 99) và normal (từ 100 đến 139).
- Giá trị nice -20 tương ứng với độ ưu tiên 100, +19 tương ứng với độ ưu tiên 139.





# Định thời trên Android

- Sử dụng bộ định thời của Linux.
- Độ ưu tiên được phân chia theo nhóm của các tiến trình:



- Để thu hồi tài nguyên, Android có thể hủy (kill) các tiến trình dựa trên độ ưu tiên của chúng.



# Định thời trên Windows

- Định thời theo độ ưu tiên với chế độ trung dụng.
- Tác vụ có độ ưu tiên cao nhất luôn được chạy tiếp.
- Tiến trình sẽ được thực thi cho đến khi (1) block bởi system call, (2) hết quantum time, (3) bị thay thế bởi một tiến trình khác có độ ưu tiên cao hơn.
- Sử dụng 32 độ ưu tiên, được chia thành 2 lớp: variable (1-15) và real-time (16-31). Độ ưu tiên 0 dành cho quản lý bộ nhớ.
- Mỗi độ ưu tiên có hàng đợi riêng.
- Idle thread được chạy nếu không có bất cứ tác vụ nào trong hàng đợi.



# Định thời trên Windows

- Các hàm thư viện Windows API cung cấp cho tiến trình các lớp ưu tiên sau:
  - ▣ REALTIME\_PRIORITY\_CLASS, HIGH\_PRIORITY\_CLASS, ABOVE\_NORMAL\_PRIORITY\_CLASS, NORMAL\_PRIORITY\_CLASS, BELOW\_NORMAL\_PRIORITY\_CLASS, IDLE\_PRIORITY\_CLASS.
- Tiến trình có thể có các độ ưu tiên tương đối sau:
  - ▣ TIME\_CRITICAL, HIGHEST, ABOVE\_NORMAL, **NORMAL**, BELOW\_NORMAL, LOWEST, IDLE
- Lớp ưu tiên và độ ưu tiên tương đối có thể kết hợp để xác định giá trị ưu tiên.
- Độ ưu tiên cơ sở (lúc khởi tạo) là **NORMAL** bên trong lớp.
- Khi hết quantum, độ ưu tiên có thể giảm nhưng không nhỏ hơn độ ưu tiên cơ sở.



# Định thời trên Windows

## ■ Các độ ưu tiên trên Windows

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1



# Định thời trên Windows

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtualizati...
AdobeUpdateService...	3852	Running	SYSTEM	00	240 K	Not allowed
AGMSERVICE.exe	3876	Running	SYSTEM	00	488 K	Not allowed
AGSSERVICE.exe	3912	Running	SYSTEM	00	96 K	Not allowed
ApMsgFwd.exe	12712	Running	ntthien	01	216 K	Disabled
ApntEx.exe	9236	Running	ntthien	00	188 K	Disabled
Apoint.exe	4880	Running	ntthien	00	440 K	Disabled
ApplicationFrameHos...	3384	Running	ntthien	00	5,144 K	Disabled
backgroundTaskHost...	860	Suspended	ntthien	00	0 K	Disabled
browser_broker.exe	14340	Running	ntthien	00	344 K	Disabled
chrome.exe	5744	Running	ntthien	00	120,876 K	Disabled
chrome.exe			ntthien	00	588 K	Disabled
chrome.exe			ntthien	00	104 K	Disabled
chrome.exe			ntthien	00	241,348 K	Disabled
chrome.exe			ntthien	00	65,972 K	Disabled
chrome.exe			ntthien	00	2,828 K	Disabled
chrome.exe			ntthien	00	1,764 K	Disabled
chrome.exe			ntthien	00	23,884 K	Disabled
chrome.exe			ntthien	00	44,152 K	Disabled
chrome.exe			ntthien	00	30,712 K	Disabled
chrome.exe			ntthien	00	6,964 K	Disabled
chrome.exe			ntthien	00	38,880 K	Disabled
chrome.exe			ntthien	00	49,080 K	Disabled

End task

Provide feedback

End process tree

Set priority

Set affinity

Analyze wait chain

Debug

UAC virtualization

Create dump file

Open file location

Search online

Properties

Go to service(s)

Realtime

High

Above normal

Normal

Below normal

Low

End task



# Định thời trên Windows

- Windows 7 có thêm user-mode scheduling (UMS):
  - Ứng dụng tạo và quản lý tiến trình độc lập với nhân.
  - Hiệu quả hơn trong trường hợp có nhiều tiến trình.
  - Định thời UMS được thực hiện với sự hỗ trợ của các thư viện như C++ Concurrency Runtime (ConcRT).





# Định thời trên Solaris

- Định thời theo độ ưu tiên
- Có 6 lớp, mỗi lớp có độ ưu tiên khác nhau và giải thuật định thời khác nhau:
  - Time sharing (TS) – mặc định
  - Interactive (IA)
  - Real time (RT)
  - System (SYS)
  - Fair Share (FSS)
  - Fixed priority (FP)
- Lớp TS sử dụng giải thuật định thời MFQ.
- Độ ưu tiên càng lớn thì time slice càng nhỏ.



# Định thời trên Solaris

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

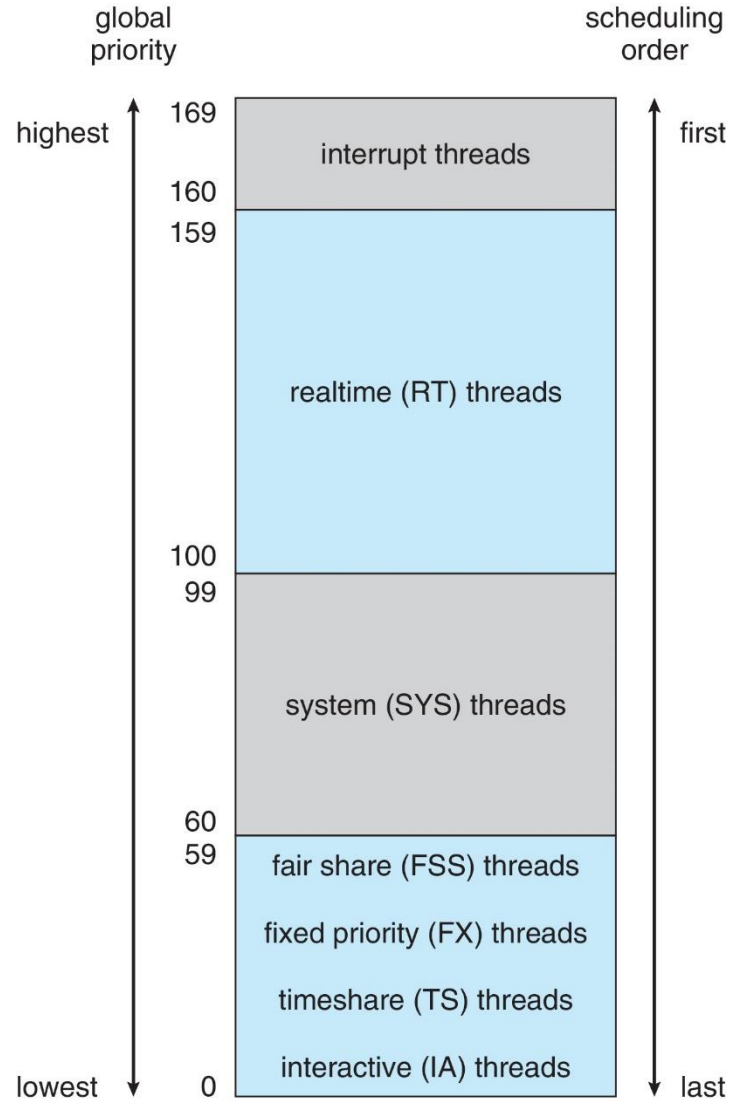


# Định thời trên Solaris

- Bộ định thời chuyển đổi độ ưu tiên theo lớp thành độ ưu tiên toàn cục:
  - Tác vụ có độ ưu tiên cao nhất được chọn chạy tiếp.
  - Tiến trình sẽ được thực thi cho đến khi (1) block, (2) hết quantum time, (3) bị thay thế bởi một tiến trình khác có độ ưu tiên cao hơn.
  - Nếu có nhiều tiến trình có cùng độ ưu tiên, bộ định thời sẽ sử dụng hàng đợi round-robin.



# Định thời trên Solaris





- Policy và Mechanism
- Đánh giá giải thuật định thời CPU
- (Đọc trong tài liệu tham khảo sách gốc tiếng Anh)



# Tóm tắt nội dung buổi học

- Định thời tiêu trình (Thread scheduling)
- Định thời đa bộ xử lý (Multiple-processor scheduling)
- Định thời theo thời gian thực (Real-time CPU scheduling)
- Định thời trên một số hệ điều hành
  - Linux
  - Windows
  - Solaris



COMPUTER ENGINEERING



**UIT**  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# THẢO LUẬN

