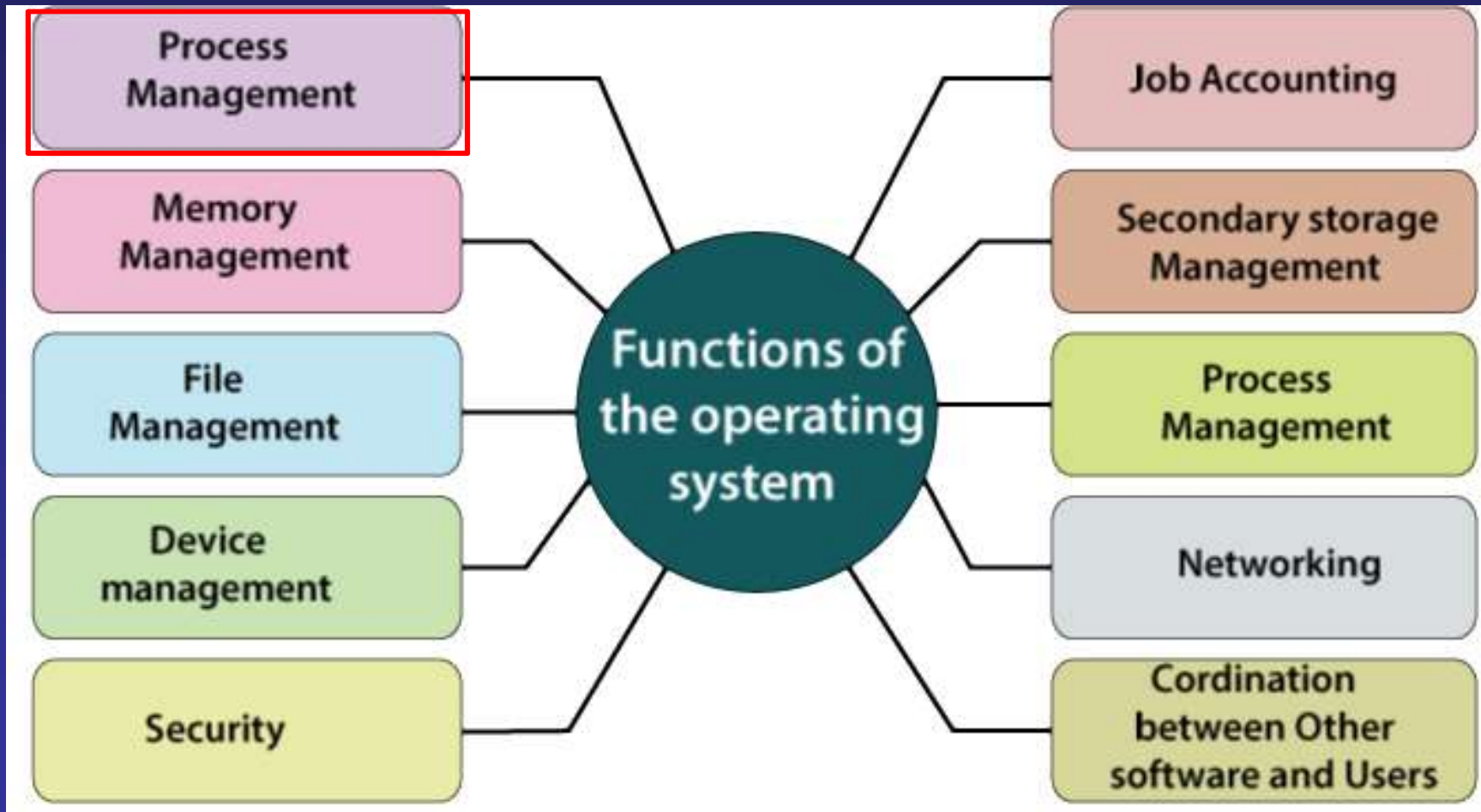


# Operating System

**Nguyen Tri Thanh**  
**ntthanh@vnu.edu.vn**



# Review

- Which of the following is **not** a part of an OS kernel ?
  - A. Process management
  - B. Network management
  - C. Memory management
  - D. Database management systems

# Review

- Which of the following should **NOT** be fixed in an OS?
  - A. SATA driver (a disk driver)
  - B. Process management module
  - C. Network management
  - D. Memory management

# Review

- Which of the following is **incorrect** about a time sharing OS?
  - A. Allow multiple processes to run on a single CPU machine
  - B. Utilize resources more effectively
  - C. Only utilize CPU more effectively
  - D. Even suitable for multi-CPU machines

# Review

- Which of the following is **incorrect** about a batch OS?
  - A. A simple type of OSes
  - B. It works in First-comes-first-served order
  - C. Allow multiple users to use the system concurrently
  - D. Not the same as multiprogramming systems

# Review

- Which of the following is incorrect about a multi-user OS?
  - A. Allow multiple processes to run on a single CPU machine
  - B. Allow each user run multiple processes
  - C. Allow multiple users to use the system concurrently
  - D. Be the same as multiprogramming systems

# Review

- Which of the following devices DOESN'T have an embedded system?
  - A. mp3 player
  - B. TV
  - C. calculator
  - D. laptop



# Process and Process Scheduling

# Objectives

- Present what a process is
- Present process scheduling approaches
- Scheduling in multi-queue systems
- Implement the scheduling algorithms

# Reference

- Chapter 3, 6 of **Operating System Concepts**

# Question

- What is a **process**?
  - A. A file on disk
  - B. An application
  - C. A program **running** on the system
  - D. A library
- **Job, task** and **process** may be used interchangeably

# Process statistic

```
ntthanh@turing:~$ top - 18:46:39 up 6 days, 2:40, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 96 total, 1 running, 95 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 0.0%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1542448k total, 1224996k used, 317452k free, 195084k buffers
Swap: 3112952k total, 0k used, 3112952k free, 736040k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
18038	ntthanh	15	0	12736	1052	800	R	0.3	0.1	0:00.05	top
1	root	15	0	10348	676	568	S	0.0	0.0	0:01.99	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:01.15	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.68	migration/1
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.91	events/0
9	root	10	-5	0	0	0	S	0.0	0.0	0:00.04	events/1
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	khelper
23	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
28	root	10	-5	0	0	0	S	0.0	0.0	0:00.19	khlockd/0
29	root	10	-5	0	0	0	S	0.0	0.0	0:00.04	khlockd/1
30	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
99	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
100	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/1

# Process statistic

```
ntthanh@turing:~$ top - 18:46:39 up 6 days, 2:40, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 96 total, 1 running, 95 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 0.0%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1542448k total, 1224996k used, 317452k free, 195084k buffers
Swap: 3112952k total, 0k used, 3112952k free, 736040k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
18038	ntthanh	15	0	12736	1052	800	R	0.3	0.1	0:00.05	top
1	root	15	0	10348	676	568	S	0.0	0.0	0:01.99	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:01.15	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.68	migration/1
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.91	events/0
9	root	10	-5	0	0	0	S	0.0	0.0	0:00.04	events/1
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	khelper
23	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
28	root	10	-5	0	0	0	S	0.0	0.0	0:00.19	kblockd/0
29	root	10	-5	0	0	0	S	0.0	0.0	0:00.04	kblockd/1
30	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
99	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
100	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/1

# Process statistic

```

ntthanh@turing:~$ top - 18:46:39 up 6 days, 2:40, 1 user, load
Tasks: 96 total, 1 running, 95 sleeping, 0
Cpu(s): 0.2%us, 0.0%sy, 0.0%ni, 99.8%id, 0.0%wa
Mem: 1542448k total, 1224996k used, 317452k free
Swap: 3112952k total, 0k used, 3112952k free

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM
18038 ntthanh  15   0 12736 1052  800  R   0.3   0.3
  1 root      15   0 10348  676  568  S   0.0   0.0
  2 root      RT   -5     0     0     0  S   0.0   0.0
  3 root      34  19     0     0     0  S   0.0   0.0
  4 root      RT   -5     0     0     0  S   0.0   0.0
  5 root      RT   -5     0     0     0  S   0.0   0.0
  6 root      34  19     0     0     0  S   0.0   0.0
  7 root      RT   -5     0     0     0  S   0.0   0.0
  8 root      10  -5     0     0     0  S   0.0   0.0
  9 root      10  -5     0     0     0  S   0.0   0.0
 10 root      10  -5     0     0     0  S   0.0   0.0
 23 root      11  -5     0     0     0  S   0.0   0.0
 28 root      10  -5     0     0     0  S   0.0   0.0
 29 root      10  -5     0     0     0  S   0.0   0.0
 30 root      15  -5     0     0     0  S   0.0   0.0
 99 root      15  -5     0     0     0  S   0.0   0.0
100 root      15  -5     0     0     0  S   0.0   0.0

```

Image Name	User Name	CPU	Mem Usage
chrome.exe	thanhnt	00	39,400 K
POWERPNT.EXE	thanhnt	00	107,232 K
chrome.exe	thanhnt	13	49,424 K
chrome.exe	thanhnt	00	44,808 K
TSVNCache.exe	thanhnt	00	5,072 K
taskmgr.exe	thanhnt	01	5,160 K
chrome.exe	thanhnt	00	21,416 K
chrome.exe	thanhnt	00	16,568 K
chrome.exe	thanhnt	00	10,928 K
chrome.exe	thanhnt	00	56,740 K
chrome.exe	thanhnt	00	48,684 K
UniKeyNT.exe	thanhnt	00	3,532 K
chrome.exe	thanhnt	00	68,888 K
chrome.exe	thanhnt	00	19,548 K
chrome.exe	thanhnt	00	7,396 K
OSPPSVC.EXE	NETWORK SERVICE	00	10,188 K
iPodService.exe	SYSTEM	00	4,324 K
alg.exe	LOCAL SERVICE	00	3,784 K
svchost.exe	LOCAL SERVICE	00	4,068 K
chrome.exe	thanhnt	00	23,004 K
chrome.exe	thanhnt	00	28,252 K
svchost.exe	NETWORK SERVICE	00	4,392 K
svchost.exe	SYSTEM	00	30,068 K
svchost.exe	NETWORK SERVICE	00	4,792 K
TOTALCMD.EXE	thanhnt	00	12,608 K
svchost.exe	SYSTEM	00	5,052 K
iTunesHelper.exe	thanhnt	00	8,904 K
lsass.exe	SYSTEM	00	1,560 K
services.exe	SYSTEM	00	3,644 K
winlogon.exe	SYSTEM	00	3,124 K

☐ Show processes from all users

End Process

# Process classification

Processes are classified into 2 categories

## A. System processes

- ✓ Created by system account
- ✓ Run essential services

## B. User processes

- ✓ Created by user accounts
- ✓ Usually are application processes (Word, Excel, YM,...)



# Question

What is the correct relation among application, process and program concepts

- A. An application may have multiple processes, a process may have multiple programs
- B. An application only has one program, a program only has one process
- C. An application may have multiple programs, a program may have multiple processes
- D. An application may have many programs, a program only has one process

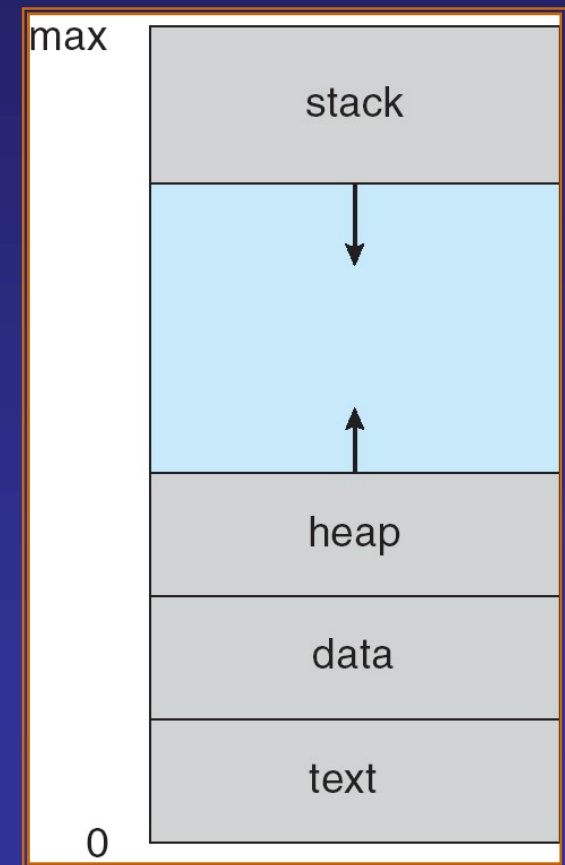
# Question

Select the best description of resources a **pure** computer may have

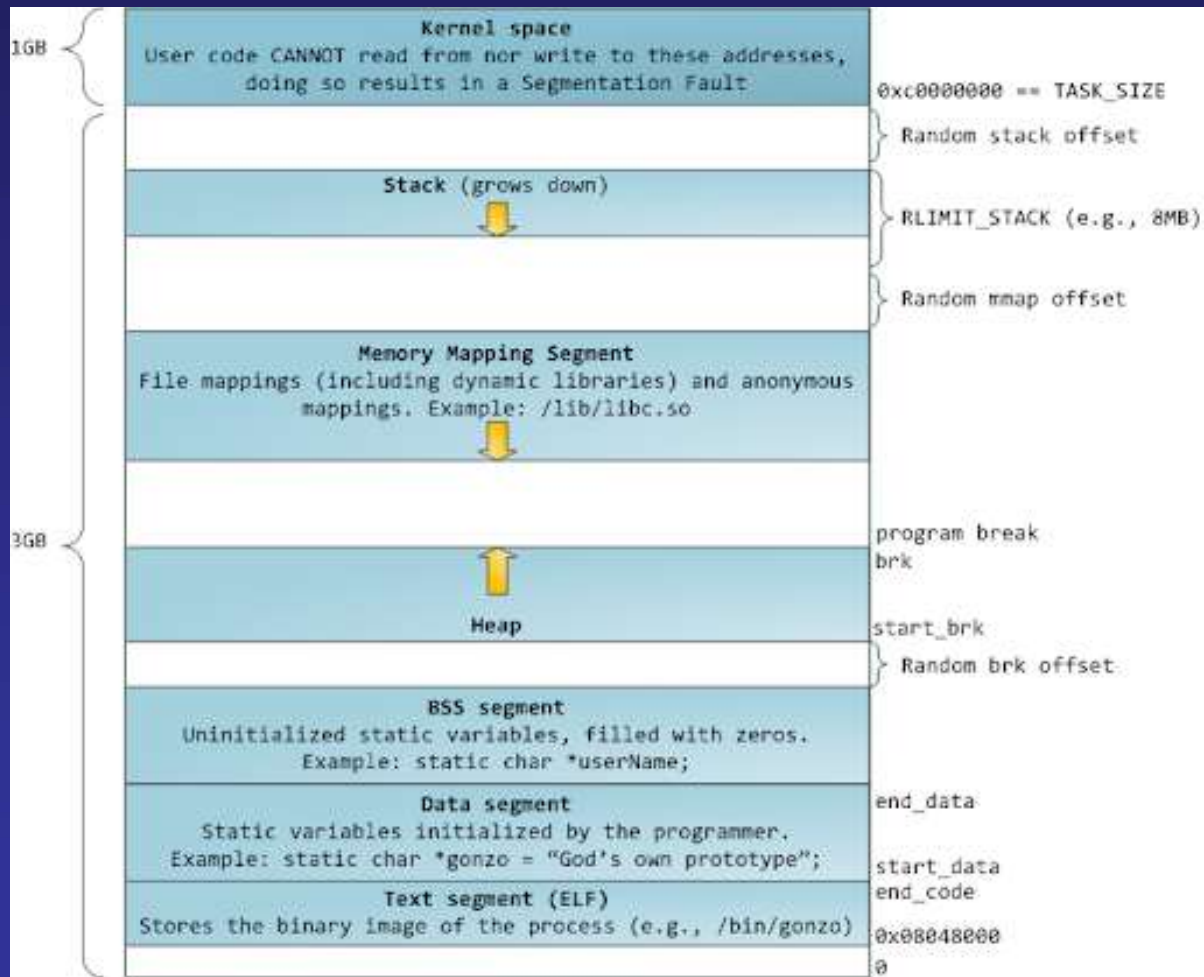
- A. CPU, RAM and anything that can connect to the computer, such as CD, network card, ...
- B. CPU, RAM, Disks
- C. CPU, RAM, Disk, printer
- D. CPU, RAM, Disk, printer, monitor

# Process structure

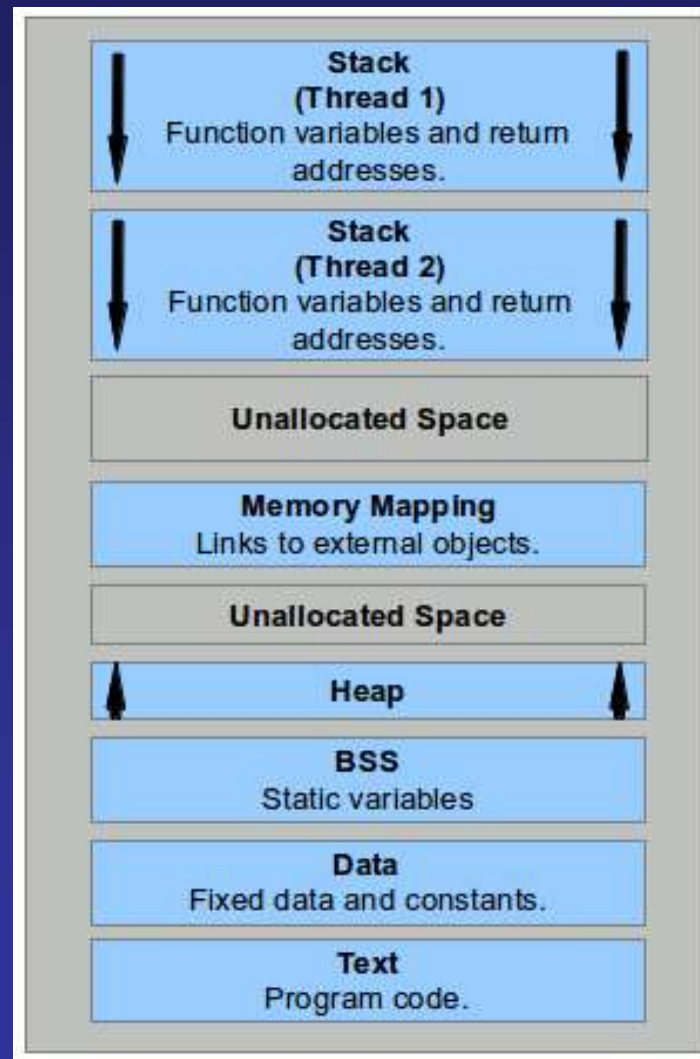
- A process at least consists of
  1. A **program counter** (Instruction Pointer)
  2. Text (code)
  3. Stack + Heap
  4. Data section
- Other information is included
- The process structure is different among OSes



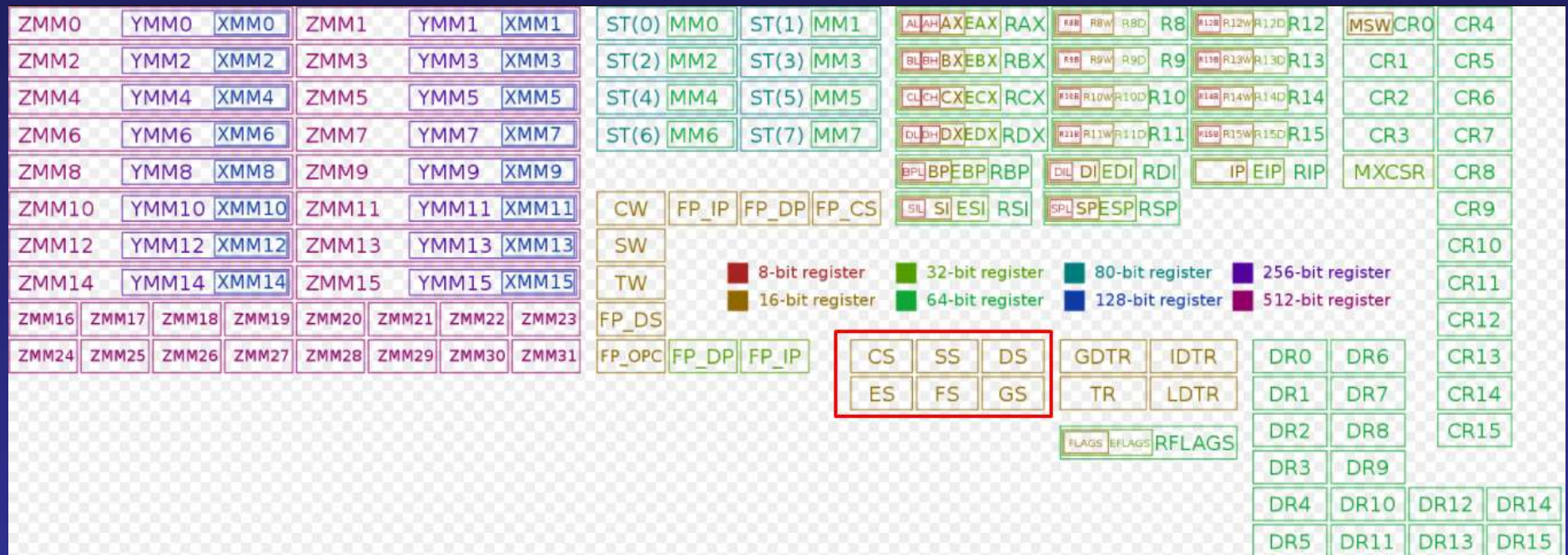
# Process structure



# Process structure



# X86-64 Intel registers



<https://en.wikipedia.org/wiki/X86>

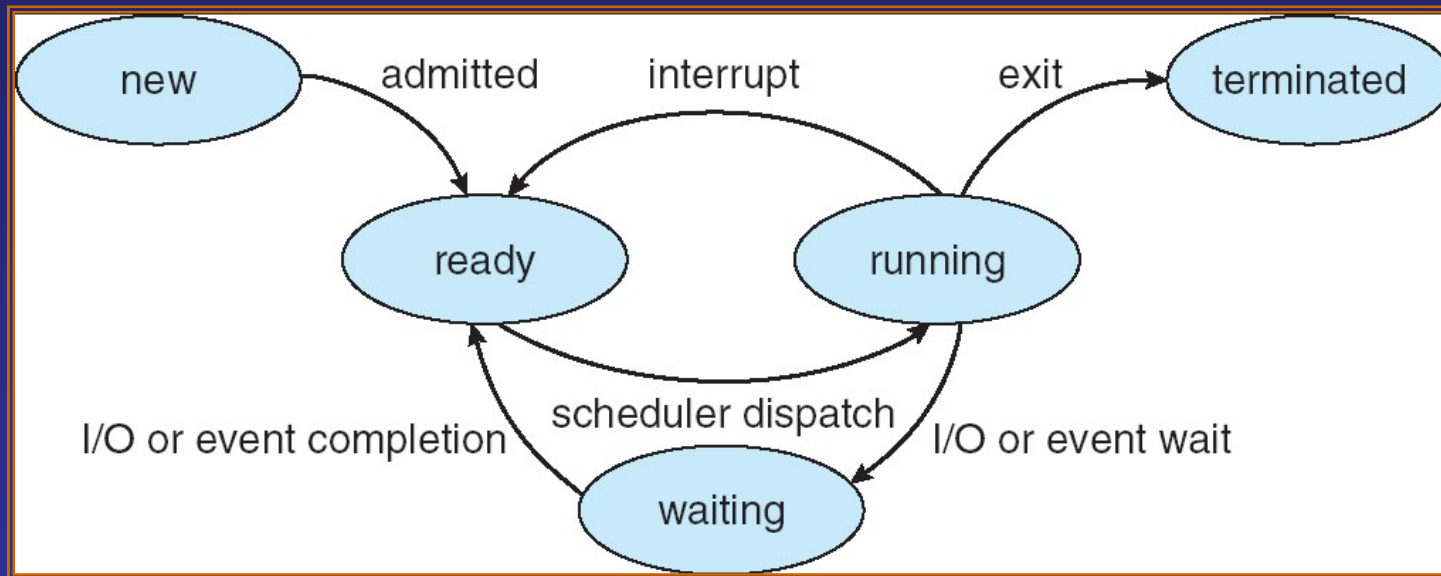
# Process control block (PCB)

Information associated with each process

- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information
- PCB is different among OSes

# Process states

- A process has many states during its life



- The number of states is different among OSes

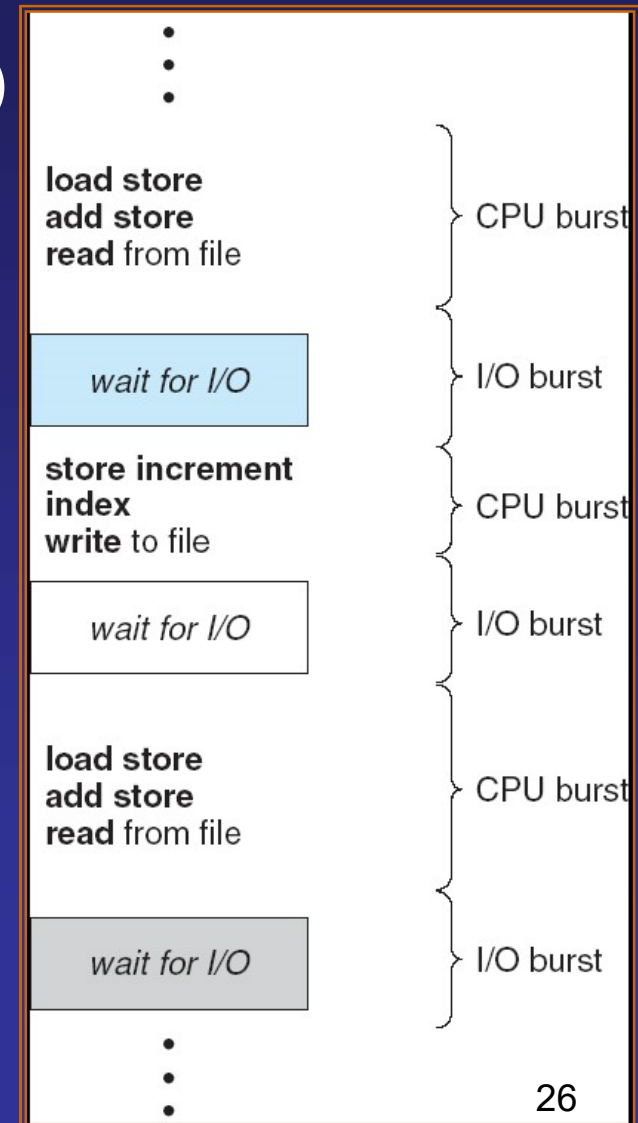


# Process states

- New
  - a new process is initiated
- Running
  - Process instructions are being run
- Waiting
  - Process is waiting for a certain resource or event
- Ready
  - Process just waits for its turn to run
- Terminated
  - The process completes

# CPU And I/O Bursts

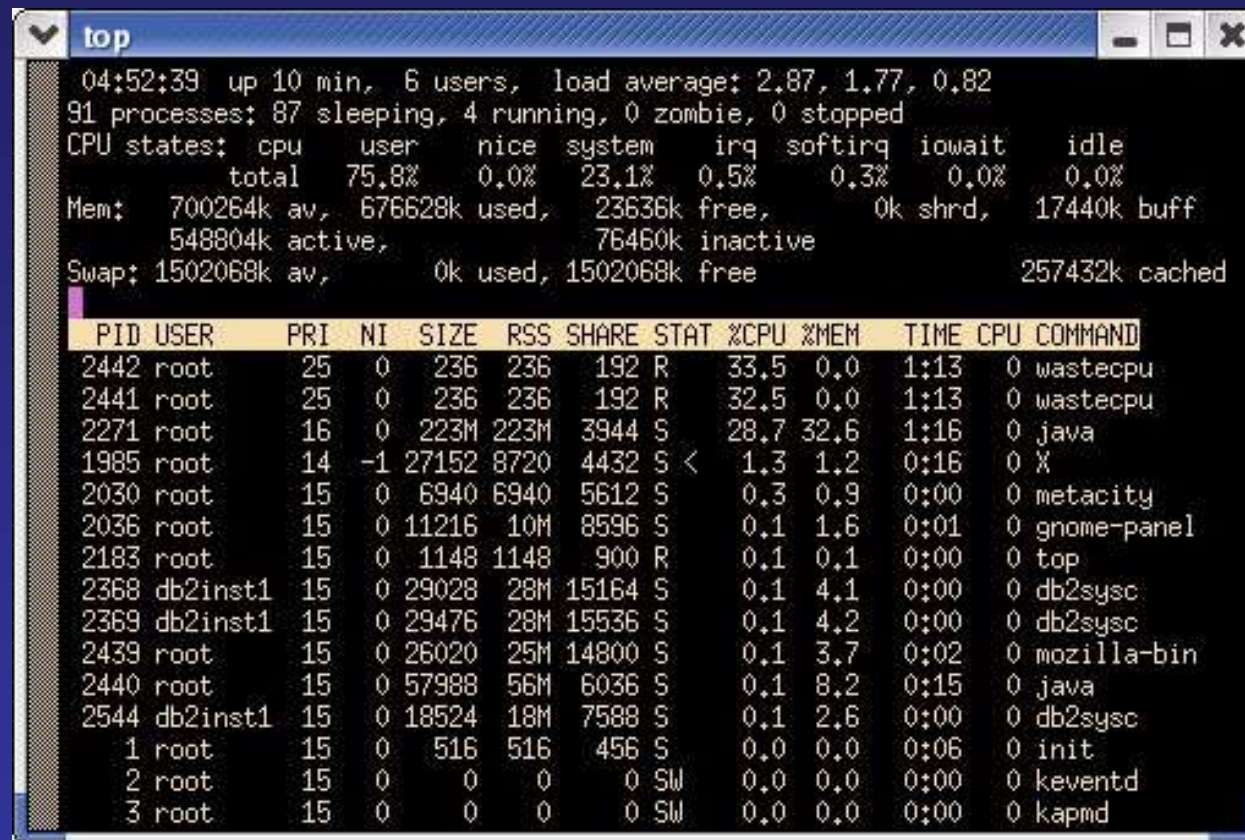
- Burst – a time span (duration)
- Two burst types
  - IO burst
  - CPU burst



# Process classification

- CPU-bound process
  - uses CPU a lot (for computation)
- IO-bound process
  - does IO a lot (data manipulation)
- These types of processes affect schedulers

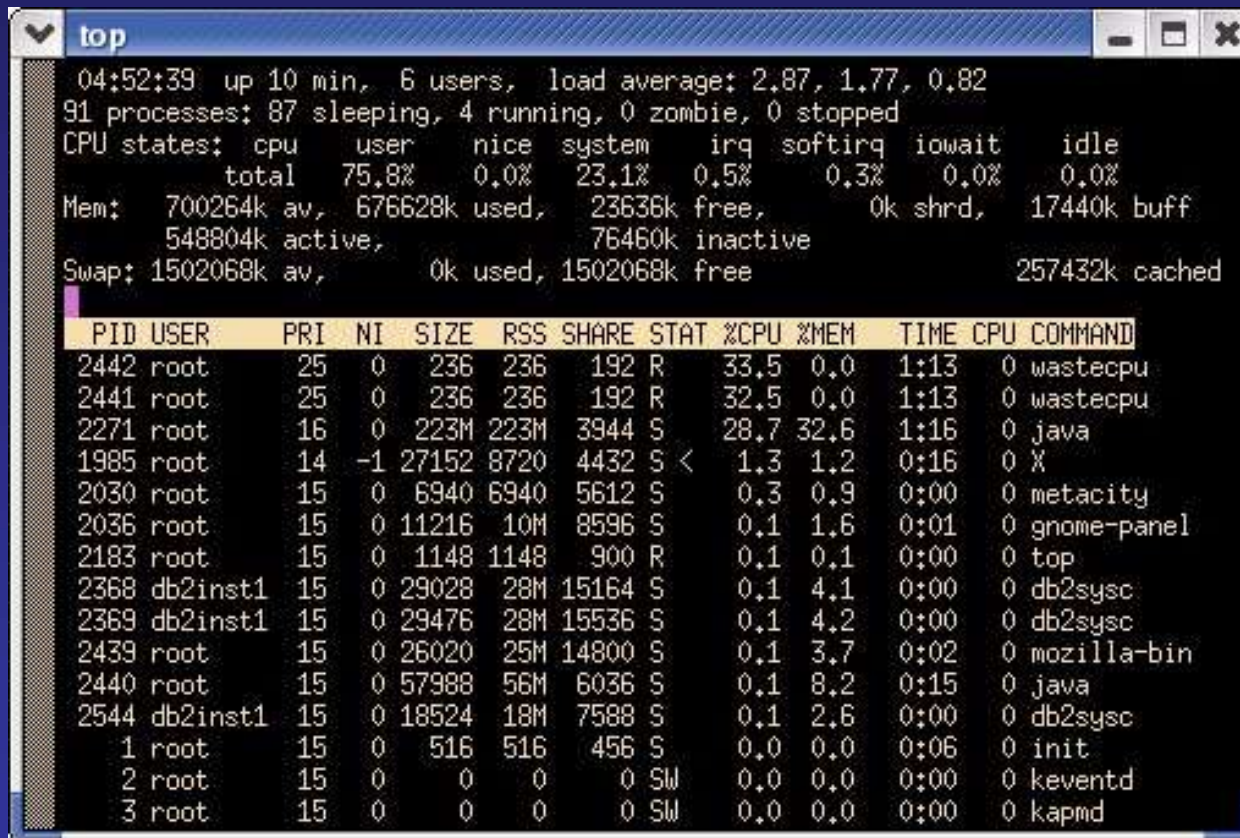
# Process classification



```
04:52:39 up 10 min, 6 users, load average: 2.87, 1.77, 0.82
91 processes: 87 sleeping, 4 running, 0 zombie, 0 stopped
CPU states:  cpu    user    nice    system    irq    softirq    iowait    idle
              total    75.8%    0.0%    23.1%    0.5%    0.3%    0.0%    0.0%
Mem:  700264k av, 676628k used, 23636k free, 0k shrd, 17440k buff
      548804k active, 76460k inactive
Swap: 1502068k av, 0k used, 1502068k free 257432k cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	CPU	COMMAND
2442	root	25	0	236	236	192	R	33.5	0.0	1:13	0	wastecpu
2441	root	25	0	236	236	192	R	32.5	0.0	1:13	0	wastecpu
2271	root	16	0	223M	223M	3944	S	28.7	32.6	1:16	0	java
1985	root	14	-1	27152	8720	4432	S<	1.3	1.2	0:16	0	X
2030	root	15	0	6940	6940	5612	S	0.3	0.9	0:00	0	metacity
2036	root	15	0	11216	10M	8596	S	0.1	1.6	0:01	0	gnome-panel
2183	root	15	0	1148	1148	900	R	0.1	0.1	0:00	0	top
2368	db2inst1	15	0	29028	28M	15164	S	0.1	4.1	0:00	0	db2sysc
2369	db2inst1	15	0	29476	28M	15536	S	0.1	4.2	0:00	0	db2sysc
2439	root	15	0	26020	25M	14800	S	0.1	3.7	0:02	0	mozilla-bin
2440	root	15	0	57988	56M	6036	S	0.1	8.2	0:15	0	java
2544	db2inst1	15	0	18524	18M	7588	S	0.1	2.6	0:00	0	db2sysc
1	root	15	0	516	516	456	S	0.0	0.0	0:06	0	init
2	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	keventd
3	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kapmd

# Process classification



04:52:39 up 10 min, 6 users, load average: 2.87, 1.77, 0.82  
91 processes: 87 sleeping, 4 running, 0 zombie, 0 stopped  
CPU states: cpu user nice system irq softirq iowait idle  
 total 75.8% 0.0% 23.1% 0.5% 0.3% 0.0% 0.0%  
Mem: 700264k av, 676628k used, 23636k free, 0k shrd, 17440k buff  
 548804k active, 76460k inactive  
Swap: 1502068k av, 0k used, 1502068k free 257432k cached

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	CPU	COMMAND
2442	root	25	0	236	236	192	R	33.5	0.0	1:13	0	wastecpu
2441	root	25	0	236	236	192	R	32.5	0.0	1:13	0	wastecpu
2271	root	16	0	223M	223M	3944	S	28.7	32.6	1:16	0	java
1985	root	14	-1	27152	8720	4432	S <	1.3	1.2	0:16	0	%
2030	root	15	0	6940	6940	5612	S	0.3	0.9	0:00	0	metacity
2036	root	15	0	11216	10M	8596	S	0.1	1.6	0:01	0	gnome-panel
2183	root	15	0	1148	1148	900	R	0.1	0.1	0:00	0	top
2368	db2inst1	15	0	29028	28M	15164	S	0.1	4.1	0:00	0	db2sysc
2369	db2inst1	15	0	29476	28M	15536	S	0.1	4.2	0:00	0	db2sysc
2439	root	15	0	26020	25M	14800	S	0.1	3.7	0:02	0	mozilla-bin
2440	root	15	0	57988	56M	6036	S	0.1	8.2	0:15	0	java
2544	db2inst1	15	0	18524	18M	7588	S	0.1	2.6	0:00	0	db2sysc
1	root	15	0	516	516	456	S	0.0	0.0	0:06	0	init
2	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	keventd
3	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kapmd

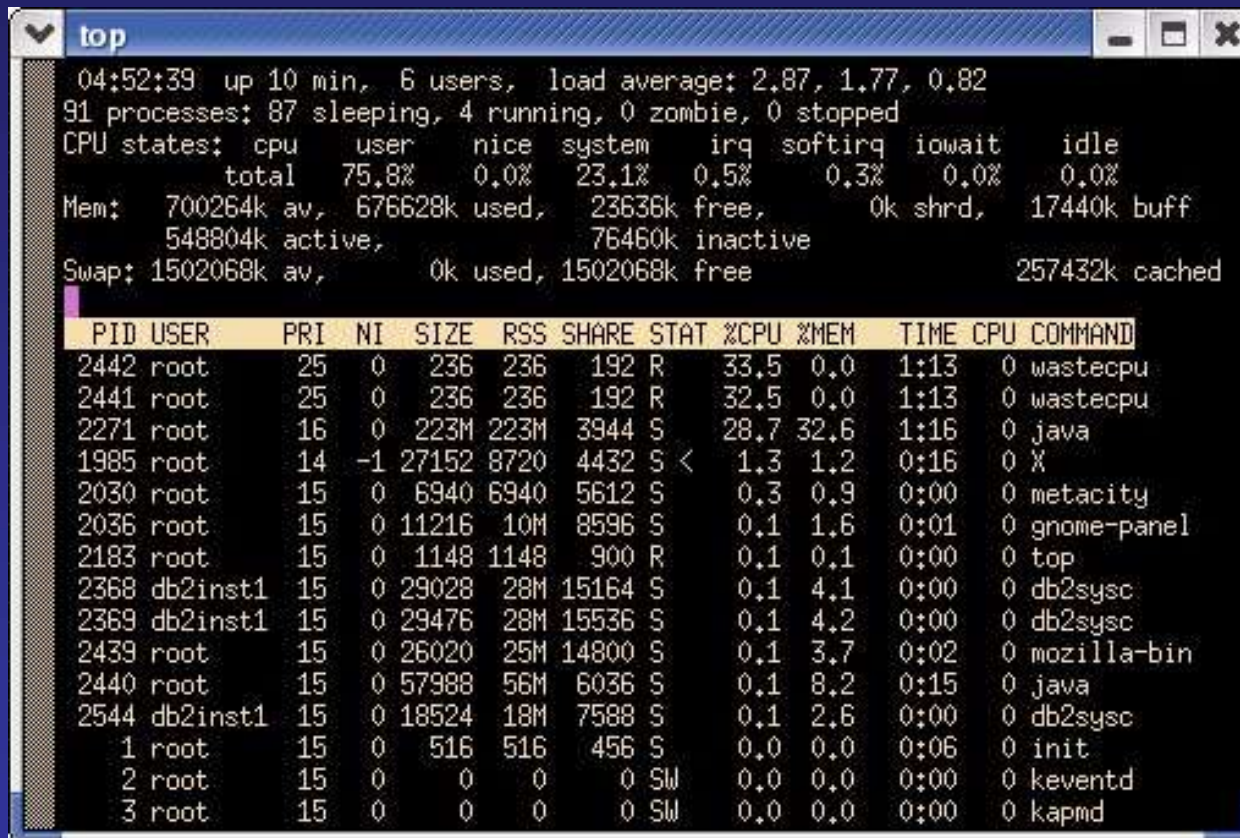


# Process classification

```
root@Clip-Streaming1:~  
top - 15:44:16 up 8 days, 1:08, 2 users, load average: 6.82, 8.39, 9.32  
Tasks: 326 total, 1 running, 325 sleeping, 0 stopped, 0 zombie  
Cpu(s): 0.7%us, 2.3%sy, 0.0%ni, 25.3%id, 70.5%wa, 0.0%hi, 1.3%si, 0.0%st  
Mem: 8165812k total, 5106372k used, 3059440k free, 3820k buffers  
Swap: 2096472k total, 0k used, 2096472k free, 3964488k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
12378	root	15	0	545m	316m	2384	S	6.0	4.0	184:13.86	glusterfs
2442	root	15	0	83664	4412	1800	S	1.3	0.1	12:31.15	lighttpd
2441	root	15	0	82852	3564	1224	S	1.0	0.0	6:35.30	lighttpd
2271	root	15	0	83500	4236	1800	S	1.0	0.1	10:00.00	lighttpd
1985	root	15	0	83852	4592	1992	S	1.0	0.1	12:55.71	lighttpd
2030	root	15	0	82356	3084	964	S	0.7	0.0	0:43.26	lighttpd
2036	root	15	0	82656	3320	1032	S	0.7	0.0	5:58.00	lighttpd
2183	root	15	0	82796	3472	1160	S	0.7	0.0	6:18.00	lighttpd
2368	db2ins	15	0	83036	3696	1288	S	0.7	0.0	7:59.84	lighttpd
2369	db2ins	15	0	83124	3876	1480	S	0.7	0.0	8:04.68	lighttpd
2439	root	15	0	83108	3768	1352	S	0.7	0.0	8:27.63	lighttpd
2440	root	15	0	84120	4792	2252	S	0.7	0.1	11:10.48	lighttpd
2544	db2ins	15	0	83296	4008	1416	S	0.7	0.0	11:44.15	lighttpd
1	root	15	0	83744	4412	1864	S	0.7	0.1	11:43.98	lighttpd
2	root	16	0	83744	4412	1864	S	0.7	0.1	11:43.98	lighttpd
3	root	15	0	83600	4304	1608	S	0.7	0.1	14:21.61	lighttpd
4371	root	15	0	12876	1264	808	R	0.3	0.0	0:00.10	top
16149	root	15	0	144m	3060	904	S	0.3	0.0	1:35.41	lighttpd

# Process classification



04:52:39 up 10 min, 6 users, load average: 2.87, 1.77, 0.82  
91 processes: 87 sleeping, 4 running, 0 zombie, 0 stopped  
CPU states: cpu user nice system irq softirq iowait idle  
 total 75.8% 0.0% 23.1% 0.5% 0.3% 0.0% 0.0%  
Mem: 700264k av, 676628k used, 23636k free, 0k shrd, 17440k buff  
 548804k active, 76460k inactive  
Swap: 1502068k av, 0k used, 1502068k free 257432k cached

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	CPU	COMMAND
2442	root	25	0	236	236	192	R	33.5	0.0	1:13	0	wastecpu
2441	root	25	0	236	236	192	R	32.5	0.0	1:13	0	wastecpu
2271	root	16	0	223M	223M	3944	S	28.7	32.6	1:16	0	java
1985	root	14	-1	27152	8720	4432	S<	1.3	1.2	0:16	0	%
2030	root	15	0	6940	6940	5612	S	0.3	0.9	0:00	0	metacity
2036	root	15	0	11216	10M	8596	S	0.1	1.6	0:01	0	gnome-panel
2183	root	15	0	1148	1148	900	R	0.1	0.1	0:00	0	top
2368	db2inst1	15	0	29028	28M	15164	S	0.1	4.1	0:00	0	db2sysc
2369	db2inst1	15	0	29476	28M	15536	S	0.1	4.2	0:00	0	db2sysc
2439	root	15	0	26020	25M	14800	S	0.1	3.7	0:02	0	mozilla-bin
2440	root	15	0	57988	56M	6036	S	0.1	8.2	0:15	0	java
2544	db2inst1	15	0	18524	18M	7588	S	0.1	2.6	0:00	0	db2sysc
1	root	15	0	516	516	456	S	0.0	0.0	0:06	0	init
2	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	keventd
3	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kapmd



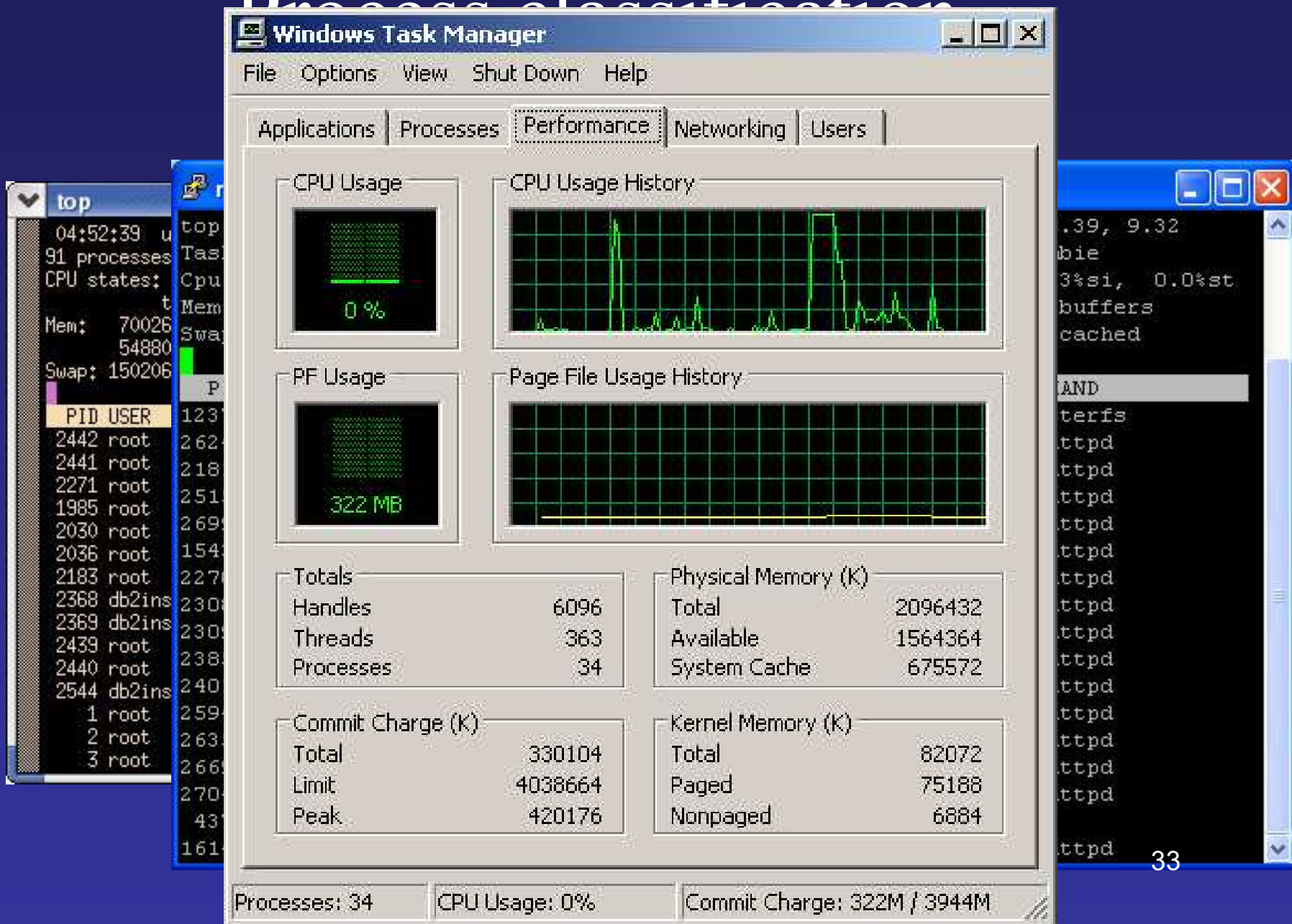
# Process classification

```
root@Clip-Streaming1:~  
top - 15:44:16 up 8 days, 1:08, 2 users, load average: 6.82, 8.39, 9.32  
Tasks: 326 total, 1 running, 325 sleeping, 0 stopped, 0 zombie  
Cpu(s): 0.7%us, 2.3%sy, 0.0%ni, 25.3%id, 70.5%wa, 0.0%hi, 1.3%si, 0.0%st  
Mem: 8165812k total, 5106372k used, 3059440k free, 3820k buffers  
Swap: 2096472k total, 0k used, 2096472k free, 3964488k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
12378	root	15	0	545m	316m	2384	S	6.0	4.0	184:13.86	glusterfs
2442	root	15	0	83664	4412	1800	S	1.3	0.1	12:31.15	lighttpd
2441	root	15	0	82852	3564	1224	S	1.0	0.0	6:35.30	lighttpd
2271	root	15	0	83500	4236	1800	S	1.0	0.1	10:00.00	lighttpd
1985	root	15	0	83852	4592	1992	S	1.0	0.1	12:55.71	lighttpd
2030	root	15	0	82356	3084	964	S	0.7	0.0	0:43.26	lighttpd
2036	root	15	0	82656	3320	1032	S	0.7	0.0	5:58.00	lighttpd
2183	root	15	0	82796	3472	1160	S	0.7	0.0	6:18.00	lighttpd
2368	db2ins	15	0	83036	3696	1288	S	0.7	0.0	7:59.84	lighttpd
2369	db2ins	15	0	83124	3876	1480	S	0.7	0.0	8:04.68	lighttpd
2439	root	15	0	83108	3768	1352	S	0.7	0.0	8:27.63	lighttpd
2440	root	15	0	84120	4792	2252	S	0.7	0.1	11:10.48	lighttpd
2544	db2ins	15	0	83296	4008	1416	S	0.7	0.0	11:44.15	lighttpd
1	root	15	0	83744	4412	1864	S	0.7	0.1	11:43.98	lighttpd
2	root	16	0	83744	4412	1864	S	0.7	0.1	11:43.98	lighttpd
3	root	15	0	83600	4304	1608	S	0.7	0.1	14:21.61	lighttpd
27045	root	15	0	12876	1264	808	R	0.3	0.0	0:00.10	top
4371	root	15	0	144m	3060	904	S	0.3	0.0	1:35.41	lighttpd
16149	root	15	0								



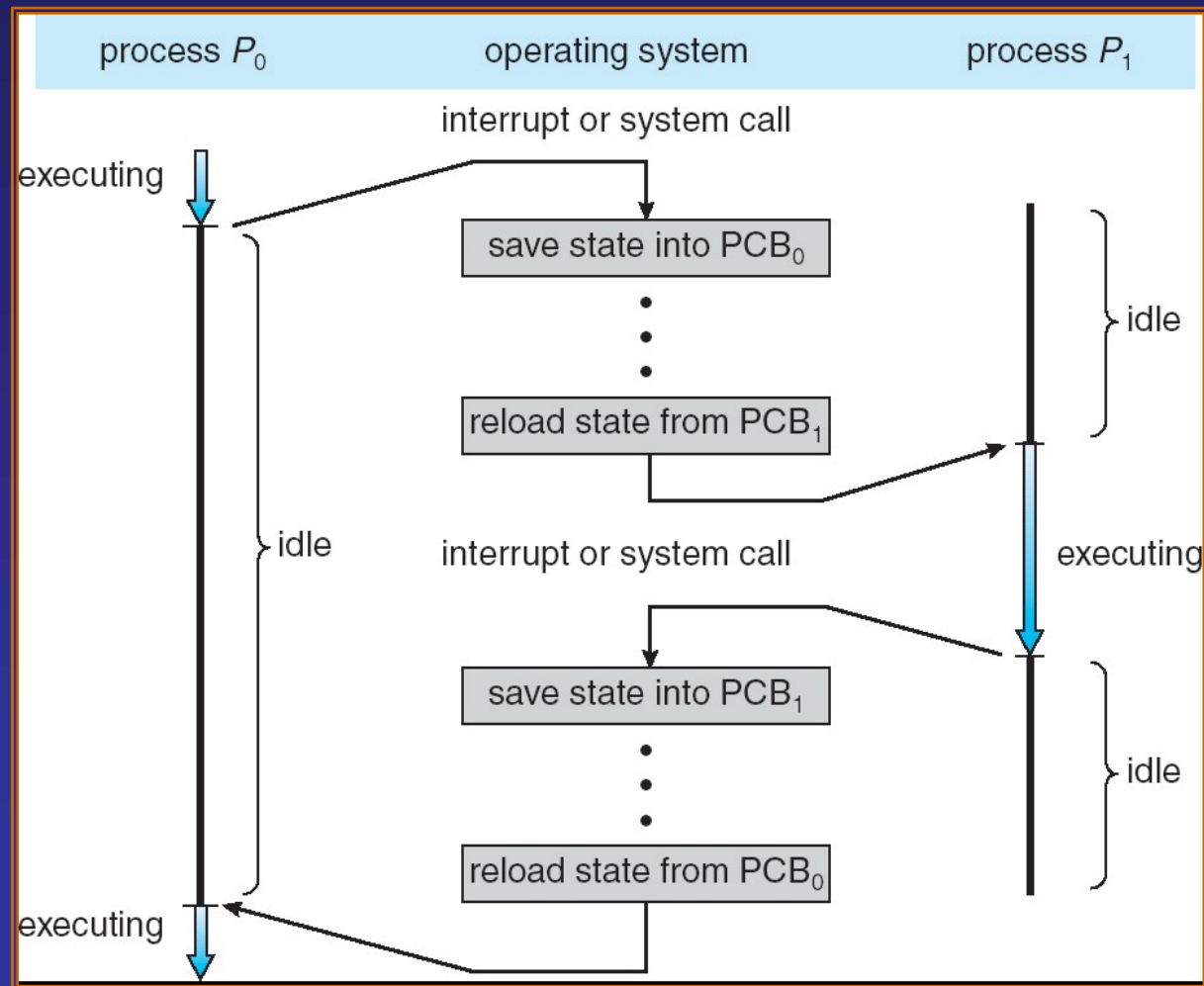
# Process classification



# Process context switch

- Context switch
  - CPU stops current process and runs another one
- Progress steps
  - save the state of the current process
  - put it into the READY/WAITING queue
  - pick the target process
  - restore the state of the target process
  - run the target process

# Process context switch



# Question

- Which of the following is **incorrect** about context switch?
  - A. the steps of changing from current process to the target one
  - B. the current process will be put into the waiting queue
  - C. the target process will be run
  - D. the state of the current process will be saved

# Process scheduling introduction

# Problem

- You have 5 exams within a week
  - How do you manage to study?
- You have several courses to select
- A shop salesperson has many customers waiting
  - How does he/she do?
- At a buffet where several dishes are available
  - How do you eat?
- A CPU has several processes
  - How does it run them

# Problem

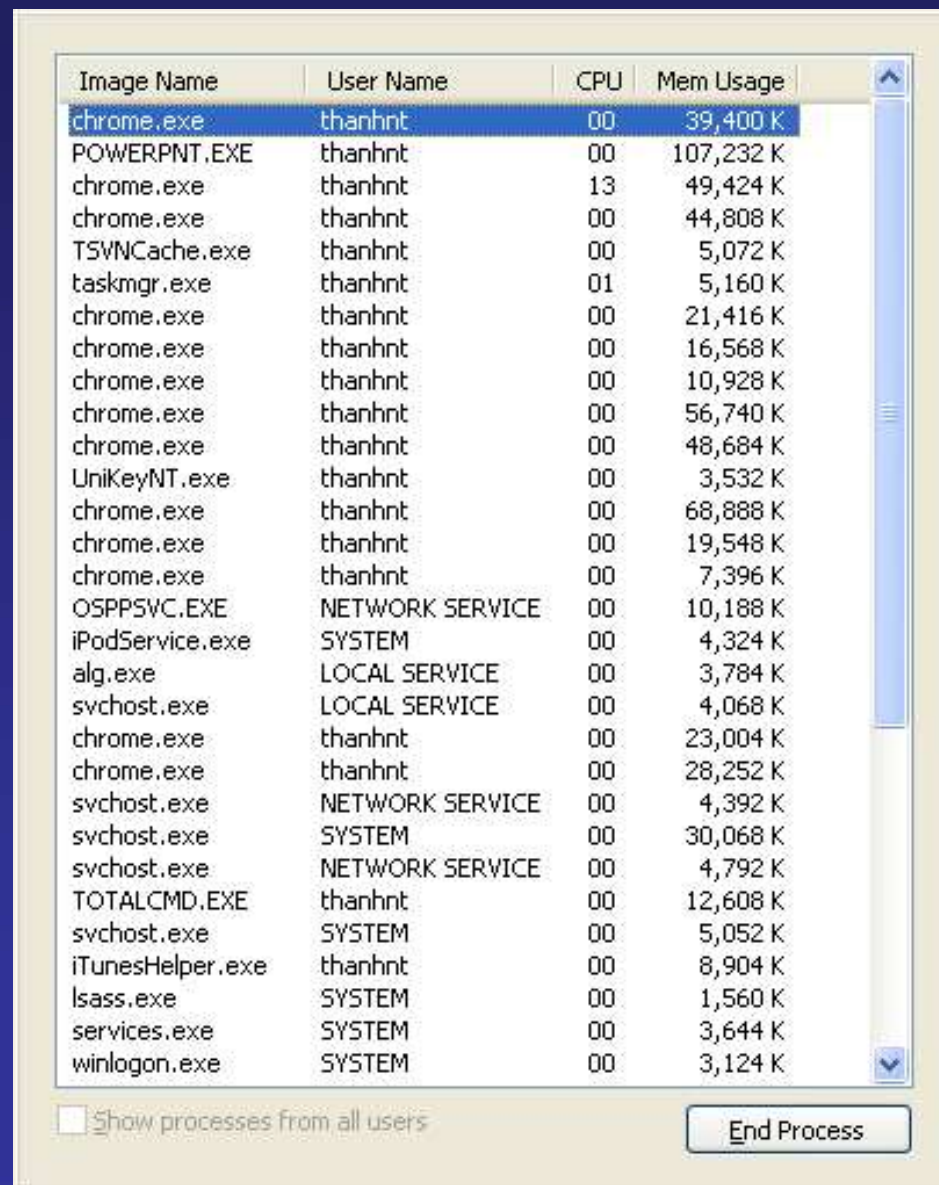


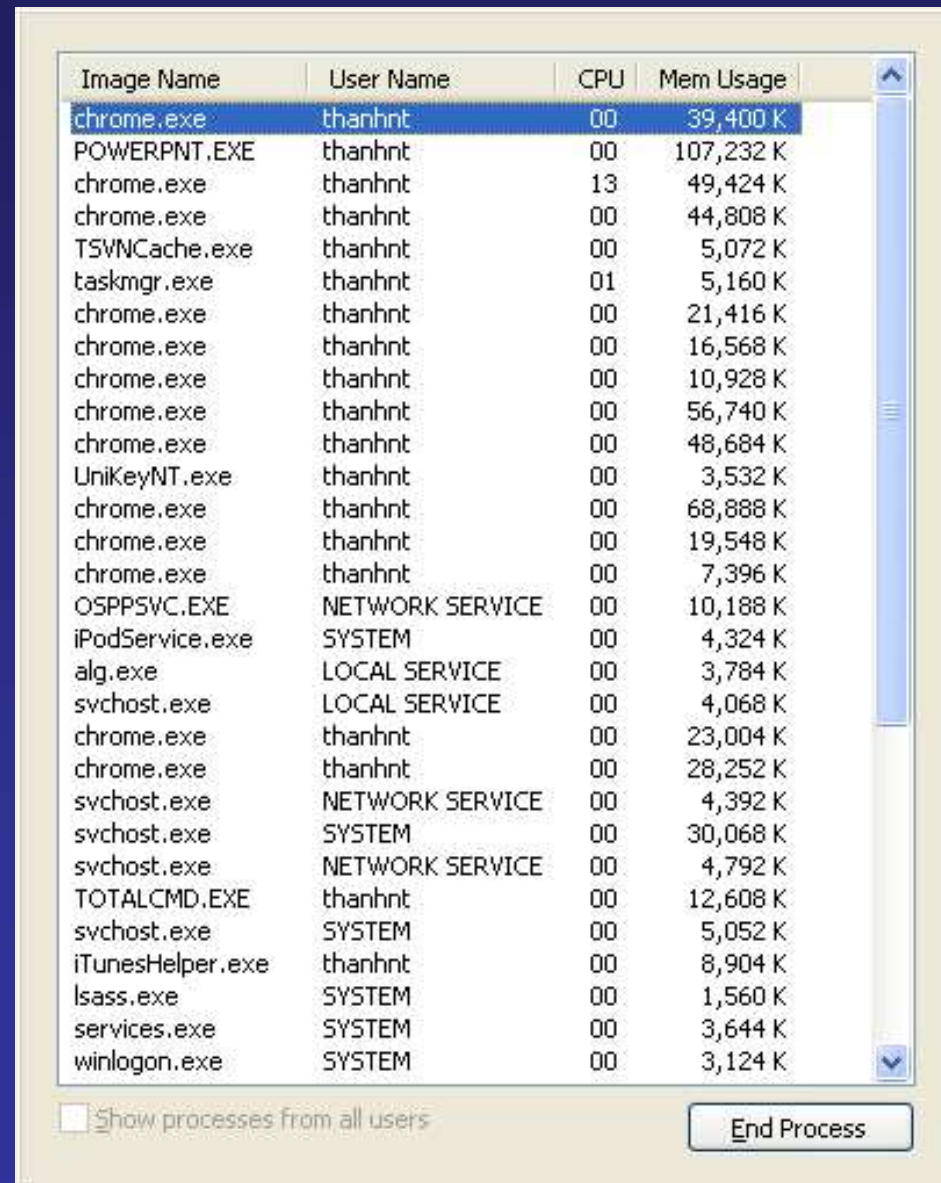
Image Name	User Name	CPU	Mem Usage
chrome.exe	thanhnt	00	39,400 K
POWERPNT.EXE	thanhnt	00	107,232 K
chrome.exe	thanhnt	13	49,424 K
chrome.exe	thanhnt	00	44,808 K
TSVNCache.exe	thanhnt	00	5,072 K
taskmgr.exe	thanhnt	01	5,160 K
chrome.exe	thanhnt	00	21,416 K
chrome.exe	thanhnt	00	16,568 K
chrome.exe	thanhnt	00	10,928 K
chrome.exe	thanhnt	00	56,740 K
chrome.exe	thanhnt	00	48,684 K
UniKeyNT.exe	thanhnt	00	3,532 K
chrome.exe	thanhnt	00	68,888 K
chrome.exe	thanhnt	00	19,548 K
chrome.exe	thanhnt	00	7,396 K
OSPPSVC.EXE	NETWORK SERVICE	00	10,188 K
iPodService.exe	SYSTEM	00	4,324 K
alg.exe	LOCAL SERVICE	00	3,784 K
svchost.exe	LOCAL SERVICE	00	4,068 K
chrome.exe	thanhnt	00	23,004 K
chrome.exe	thanhnt	00	28,252 K
svchost.exe	NETWORK SERVICE	00	4,392 K
svchost.exe	SYSTEM	00	30,068 K
svchost.exe	NETWORK SERVICE	00	4,792 K
TOTALCMD.EXE	thanhnt	00	12,608 K
svchost.exe	SYSTEM	00	5,052 K
iTunesHelper.exe	thanhnt	00	8,904 K
lsass.exe	SYSTEM	00	1,560 K
services.exe	SYSTEM	00	3,644 K
winlogon.exe	SYSTEM	00	3,124 K

☐ Show processes from all users

End Process

# Problem

How to run these processes?



A screenshot of the Windows Task Manager window, specifically the 'Processes' tab. The window displays a list of running processes with columns for Image Name, User Name, CPU usage, and Memory Usage. The first process, 'chrome.exe', is highlighted in blue. The list includes various system and user processes, such as 'POWERPNT.EXE', 'TSVNCache.exe', 'taskmgr.exe', and several instances of 'chrome.exe'. At the bottom, there is a checkbox labeled 'Show processes from all users' and a button labeled 'End Process'.

Image Name	User Name	CPU	Mem Usage
chrome.exe	thanhnt	00	39,400 K
POWERPNT.EXE	thanhnt	00	107,232 K
chrome.exe	thanhnt	13	49,424 K
chrome.exe	thanhnt	00	44,808 K
TSVNCache.exe	thanhnt	00	5,072 K
taskmgr.exe	thanhnt	01	5,160 K
chrome.exe	thanhnt	00	21,416 K
chrome.exe	thanhnt	00	16,568 K
chrome.exe	thanhnt	00	10,928 K
chrome.exe	thanhnt	00	56,740 K
chrome.exe	thanhnt	00	48,684 K
UniKeyNT.exe	thanhnt	00	3,532 K
chrome.exe	thanhnt	00	68,888 K
chrome.exe	thanhnt	00	19,548 K
chrome.exe	thanhnt	00	7,396 K
OSPPSVC.EXE	NETWORK SERVICE	00	10,188 K
iPodService.exe	SYSTEM	00	4,324 K
alg.exe	LOCAL SERVICE	00	3,784 K
svchost.exe	LOCAL SERVICE	00	4,068 K
chrome.exe	thanhnt	00	23,004 K
chrome.exe	thanhnt	00	28,252 K
svchost.exe	NETWORK SERVICE	00	4,392 K
svchost.exe	SYSTEM	00	30,068 K
svchost.exe	NETWORK SERVICE	00	4,792 K
TOTALCMD.EXE	thanhnt	00	12,608 K
svchost.exe	SYSTEM	00	5,052 K
iTunesHelper.exe	thanhnt	00	8,904 K
lsass.exe	SYSTEM	00	1,560 K
services.exe	SYSTEM	00	3,644 K
winlogon.exe	SYSTEM	00	3,124 K

☐ Show processes from all users

End Process



# Question

Which **best** describes the reason why we need process scheduling?

- A. Because we have many processes
- B. Because we have many processes and want them to be treated fairly
- C. Many reasons
  - Many processes
  - Utilize resources effectively
  - Don't let users wait
  - ...
- D. Because we want to utilize RAM effectively <sub>41</sub>

# Queue

- When there are several people waiting at the counter (in a supermarket)
  - What do they do?

# Queue



# Queue



www.shutterstock.com · 1415951



# Queue





# Queue



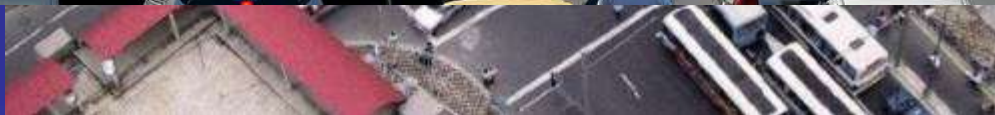


# Queue

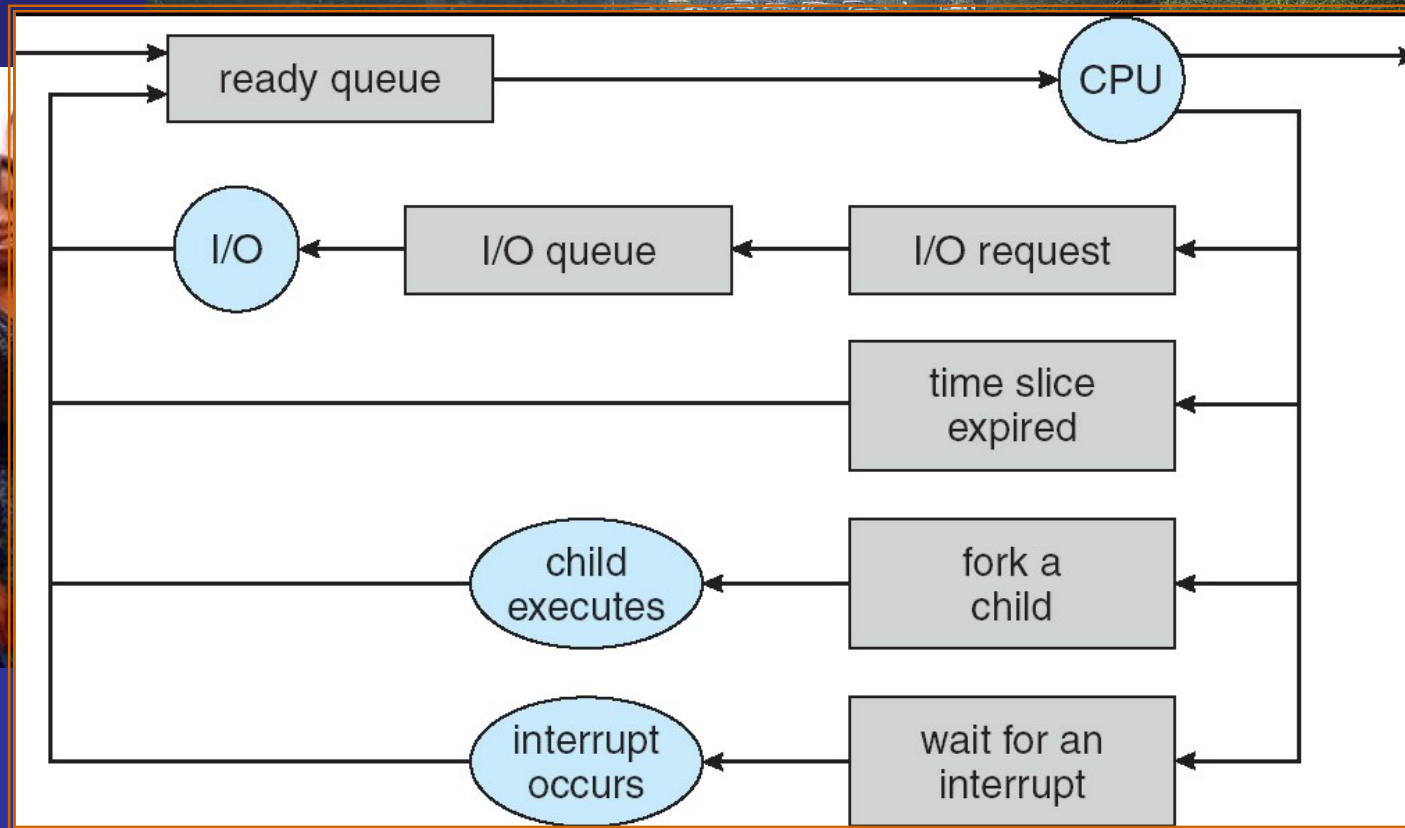




# Queue

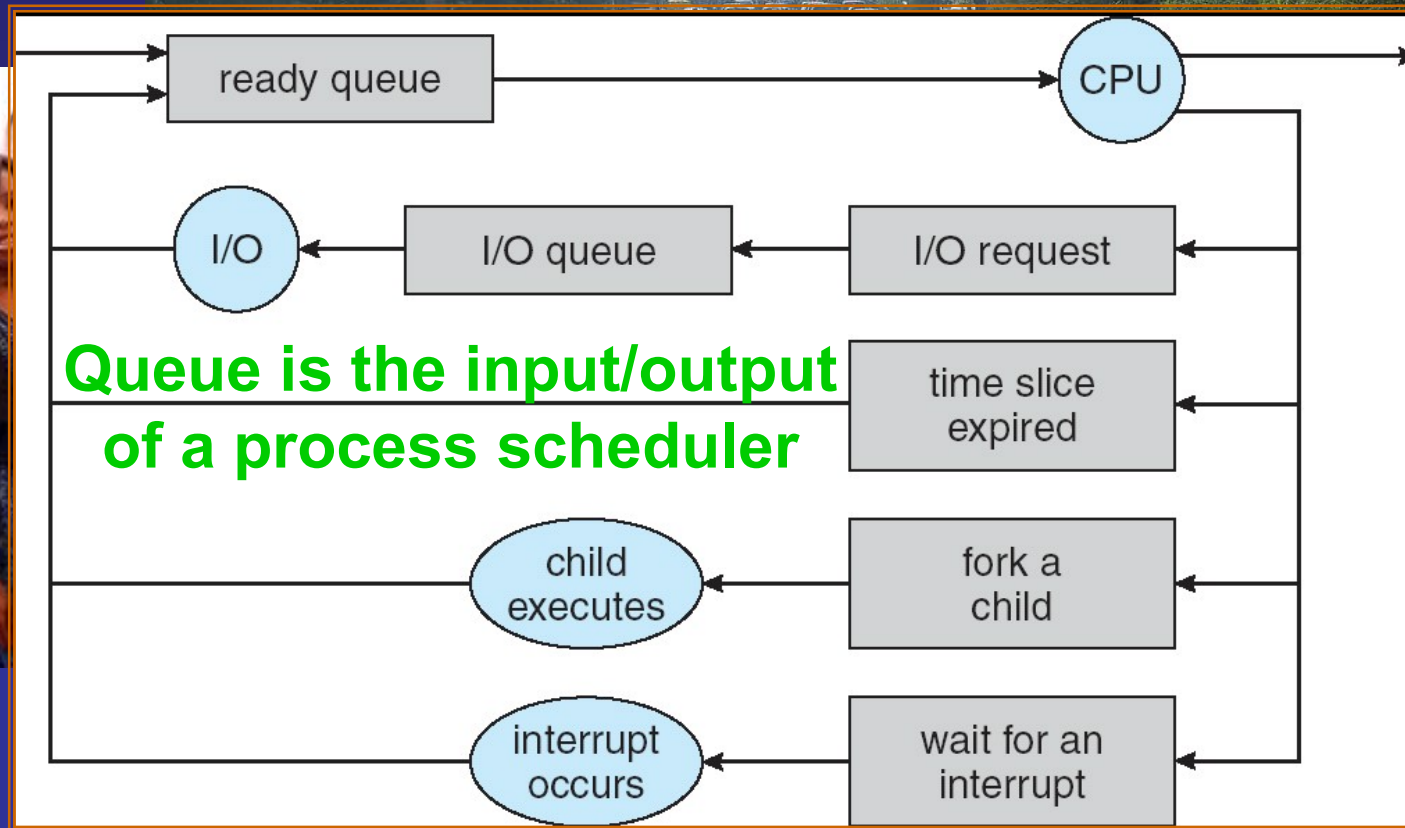


# Queue





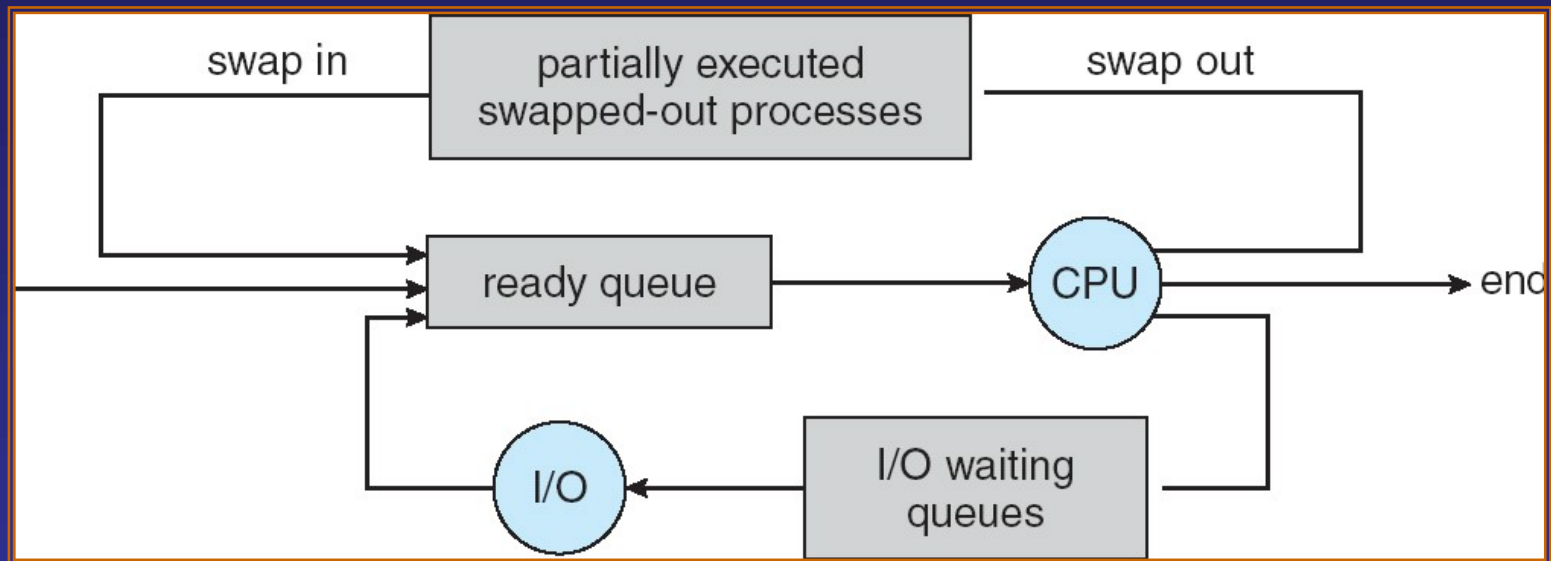
# Queue



# Different schedulers

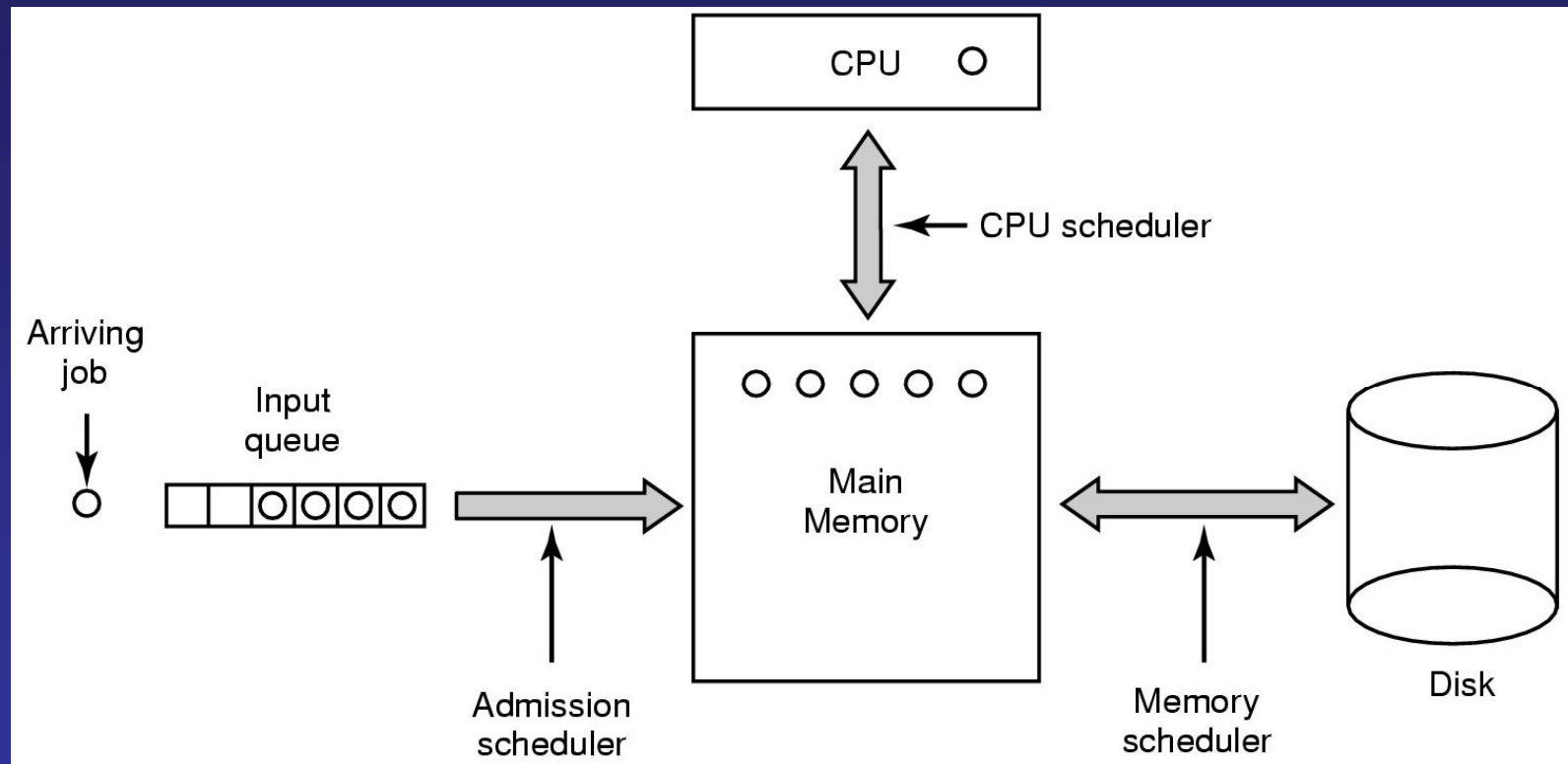
- **Long-term scheduler** (or job scheduler)
  - selects which processes should be brought into the ready queue
- **Short-term scheduler** (or CPU scheduler)
  - selects which process should be executed next
- **Medium-term scheduler**
  - selects which process to temporarily swap out (of the MEM)

# Different schedulers



Where is the position of the 3 schedulers?

# Scheduling in Batch Systems



Three level scheduling

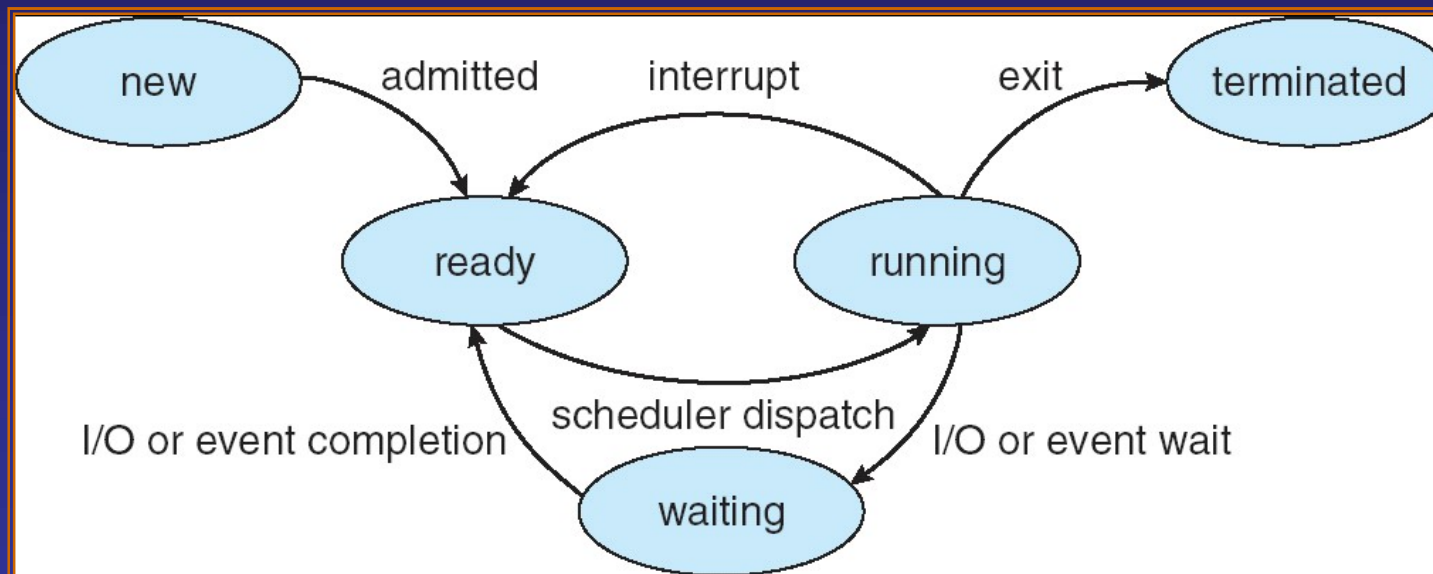
# Question

What is wrong when the CPU scheduler is called?

- A. A process changes from RUNNING to READY
- B. A process is stopped
- C. A process is admitted
- D. A different process will be run



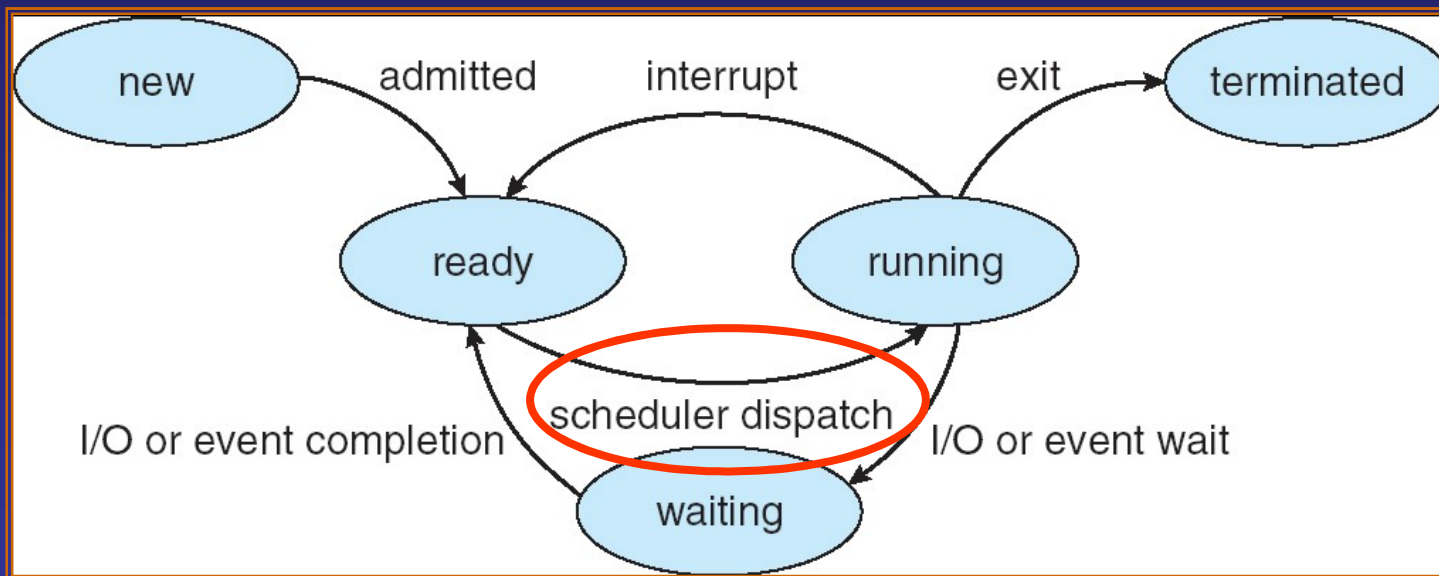
# Position of CPU scheduler



# Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - run the process
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running

# Dispatcher



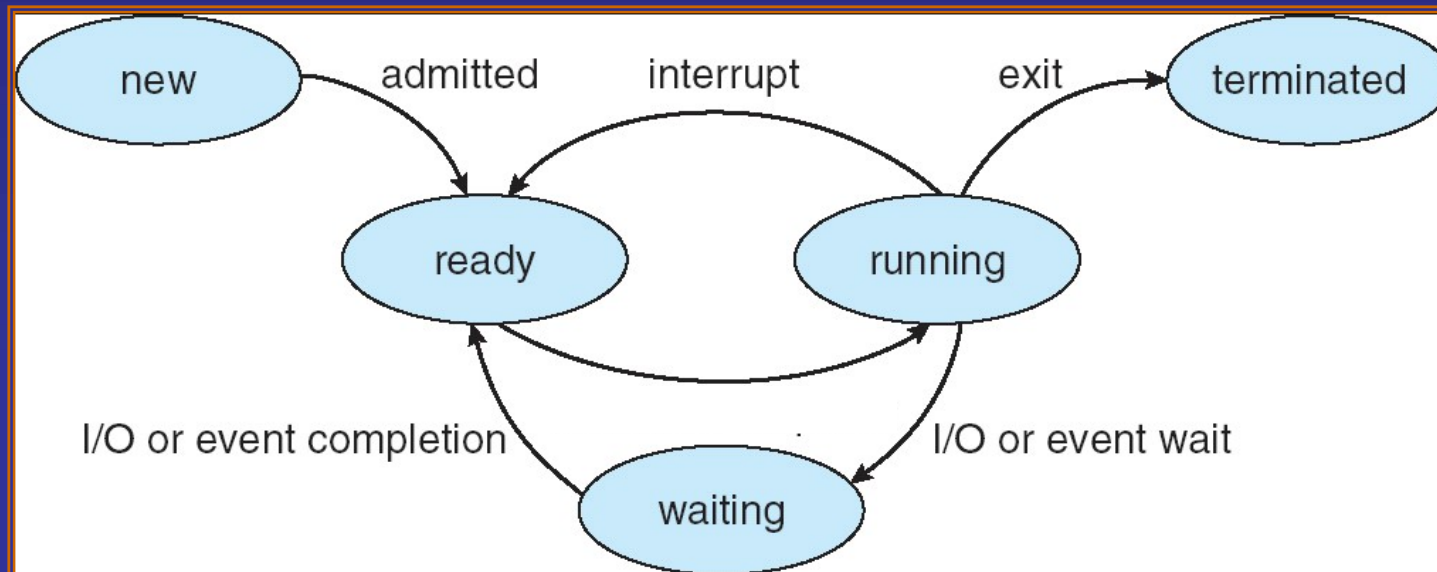
# CPU scheduling

# CPU scheduling

What is CPU scheduling?

- A. Select program to be initialized
- B. Select process to swap out
- C. Select process to change into the idle state
- D. Select process to run

# CPU scheduling



# CPU scheduling

Where is the position of CPU scheduler?

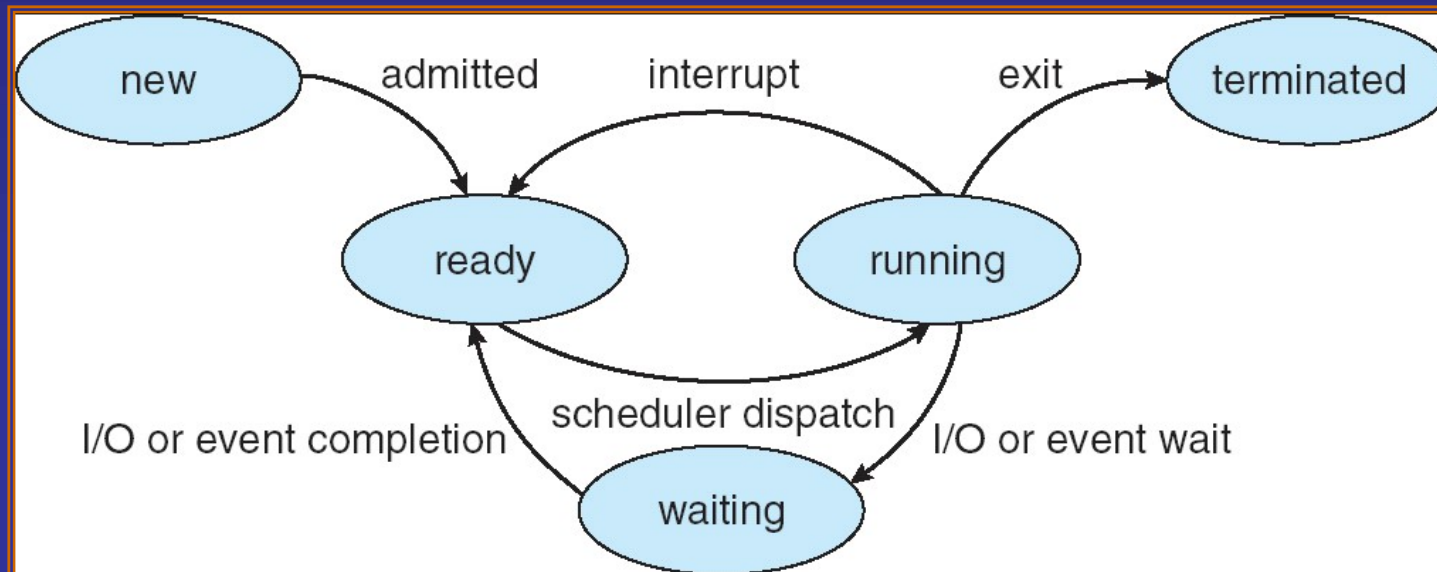
- A. Between NEW and READY states
- B. Between RUNNING and READY states
- C. Between RUNNING and TERMINATED states
- D. Between RUNNING and WAITING states



# CPU scheduler type

- Non pre-emptive
  - running process has privilege to use CPU until it terminates or changes into WAITING state
  - Ex: Apple Macintosh, Windows 3.1
- Pre-emptive
  - running process may be forced to release CPU
  - Ex: Current Windows versions, Linux, Unix
- Which type is more effective?

# Question

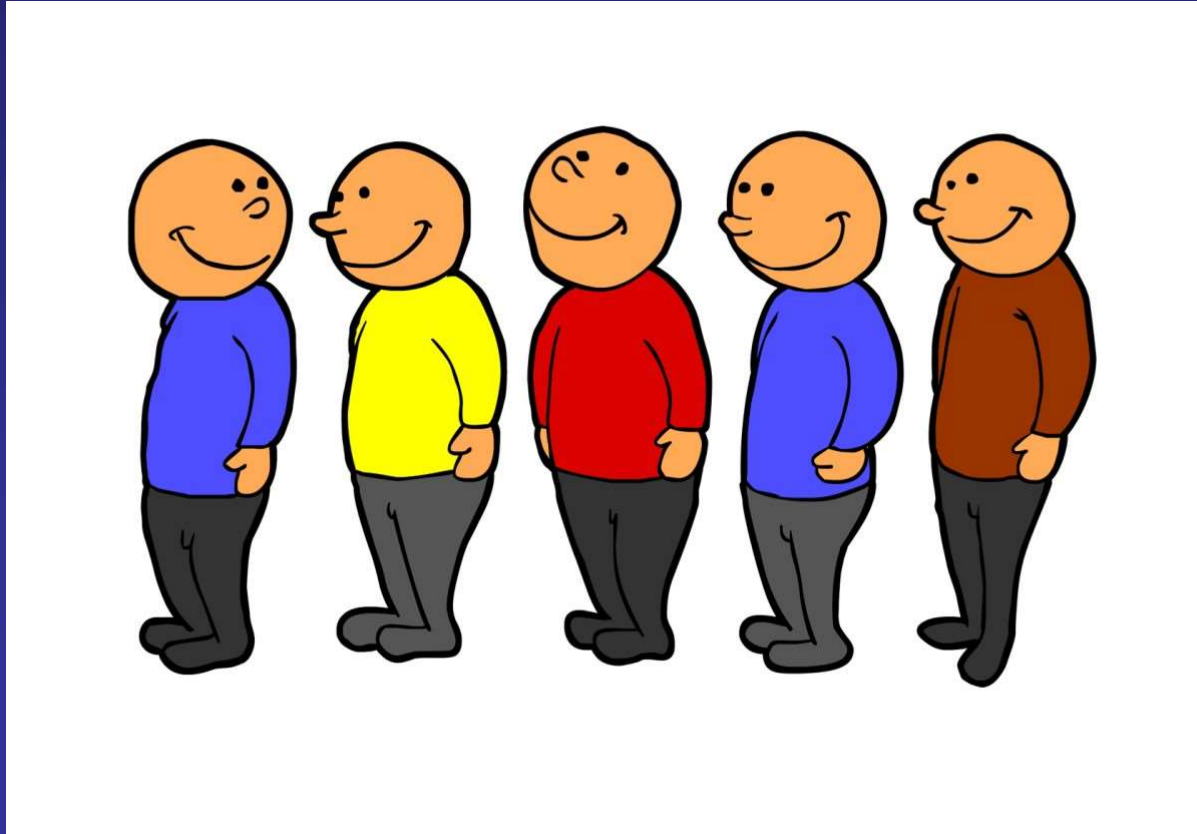


# Question

- Which is correct about non-preemptive scheduler?
  - A. no arc from RUNNING to READY states
  - B. no arc from RUNNING to WAITING states
  - C. no arc from WAITING to READY states
  - D. no arc from READY to RUNNING states

# First comes first served (FCFS)

- Use FIFO queue
- Process at the head of the queue is run first



# First comes first serves (FCFS)

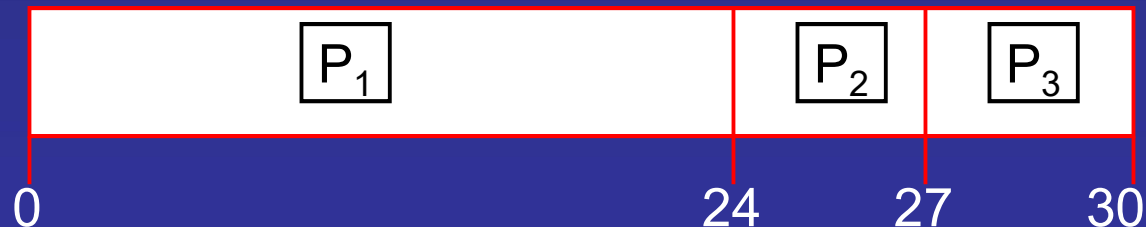
<u>Process</u>	<u>Burst Time</u>
----------------	-------------------

$P_1$	24
-------	----

$P_2$	3
-------	---

$P_3$	3
-------	---

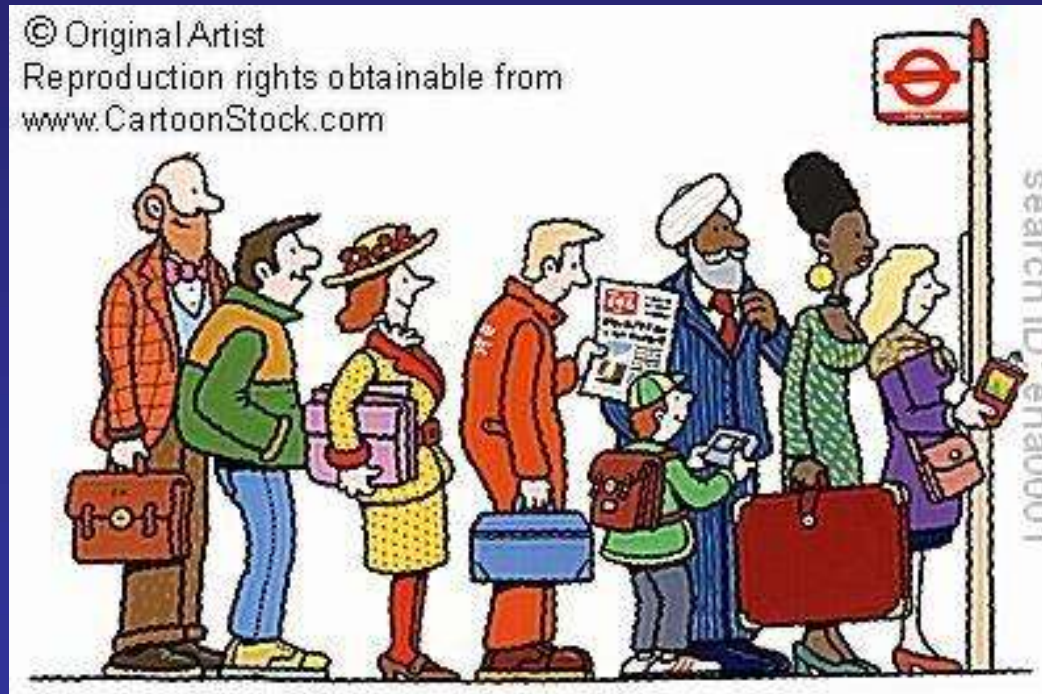
- Suppose that the processes arrive in the order:  $P_1, P_2, P_3$   
The Gantt Chart for the schedule is:



# Shortest Job First (SJF)

- Also called Shortest Job Next (SJN)
- Shortest job in the queue is selected to be run
- There are two flavors
  - Non-preemptive
  - Preemptive (Shortest Remaining Time First – SRTF)

# Shortest Job First (SJF)



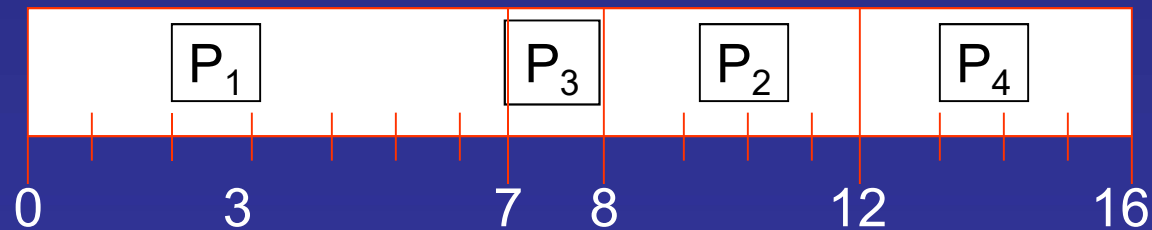


# Shortest Job First (SJF)



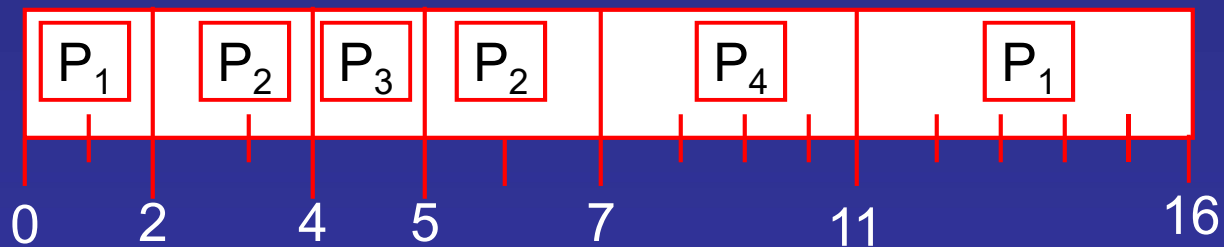
# Example of Non-Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	7
$P_2$	2	4
$P_3$	4	1
$P_4$	5	4



# Example of Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	7
$P_2$	2	4
$P_3$	4	1
$P_4$	5	4



# Next CPU burst estimation

- What if we don't know the length of burst time?
- Can only estimate
- Can be done by using the length of previous CPU bursts, using exponential averaging

1.  $t_n$  = actual length of  $n^{th}$  CPU burst
2.  $\tau_{n+1}$  = predicted value for the next CPU burst
3.  $\alpha, 0 \leq \alpha \leq 1$
4. Define:  $\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$ .

# Examples

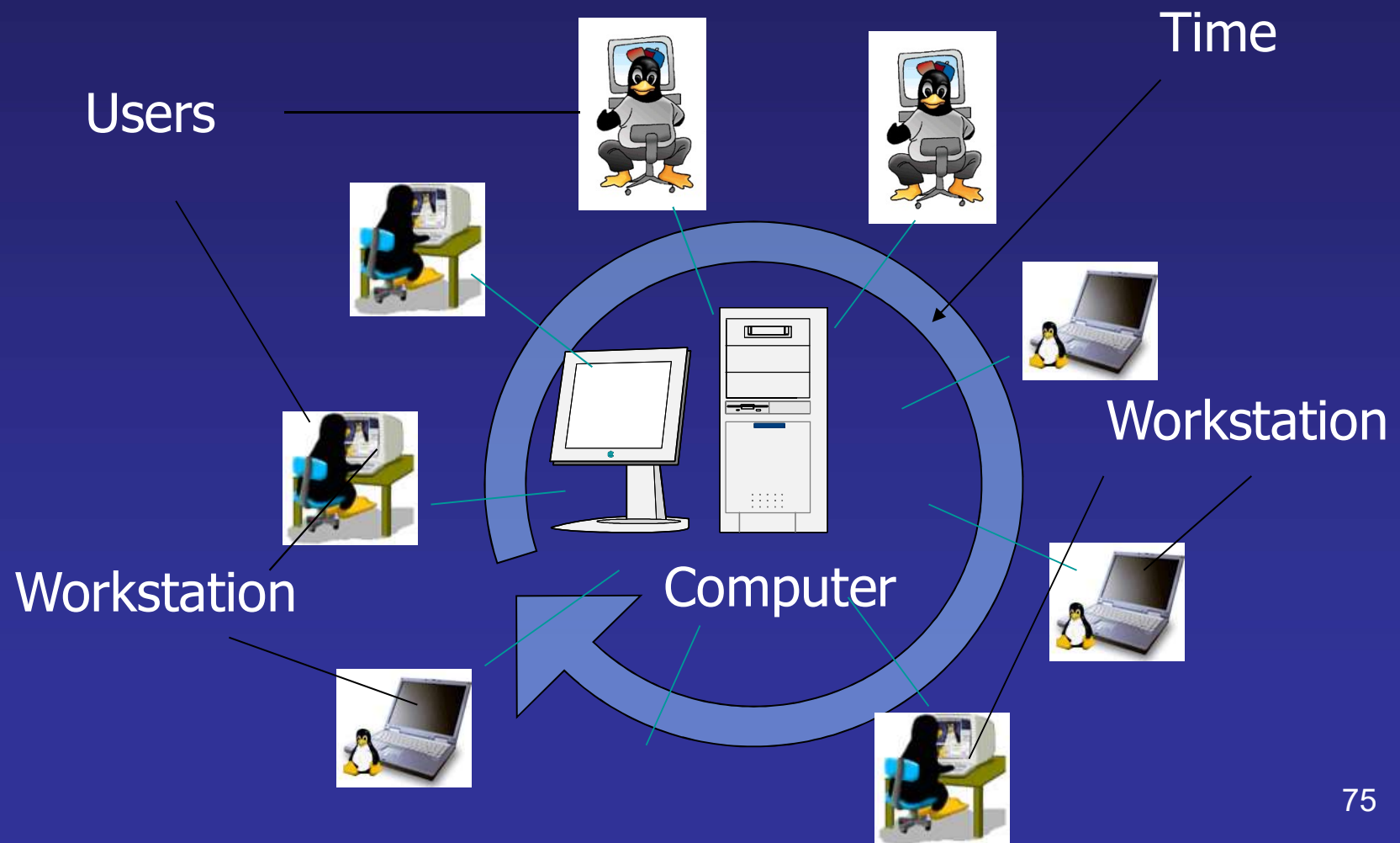
- $\alpha = 0$ 
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- $\alpha = 1$ 
  - $\tau_{n+1} = \alpha t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:
$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \dots$$
$$+ (1 - \alpha)^j \alpha t_{n-j} + \dots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$
- Since both  $\alpha$  and  $(1 - \alpha)$  are less than or equal to 1, each successive term has less weight than its predecessor

# Round Robin (RR)

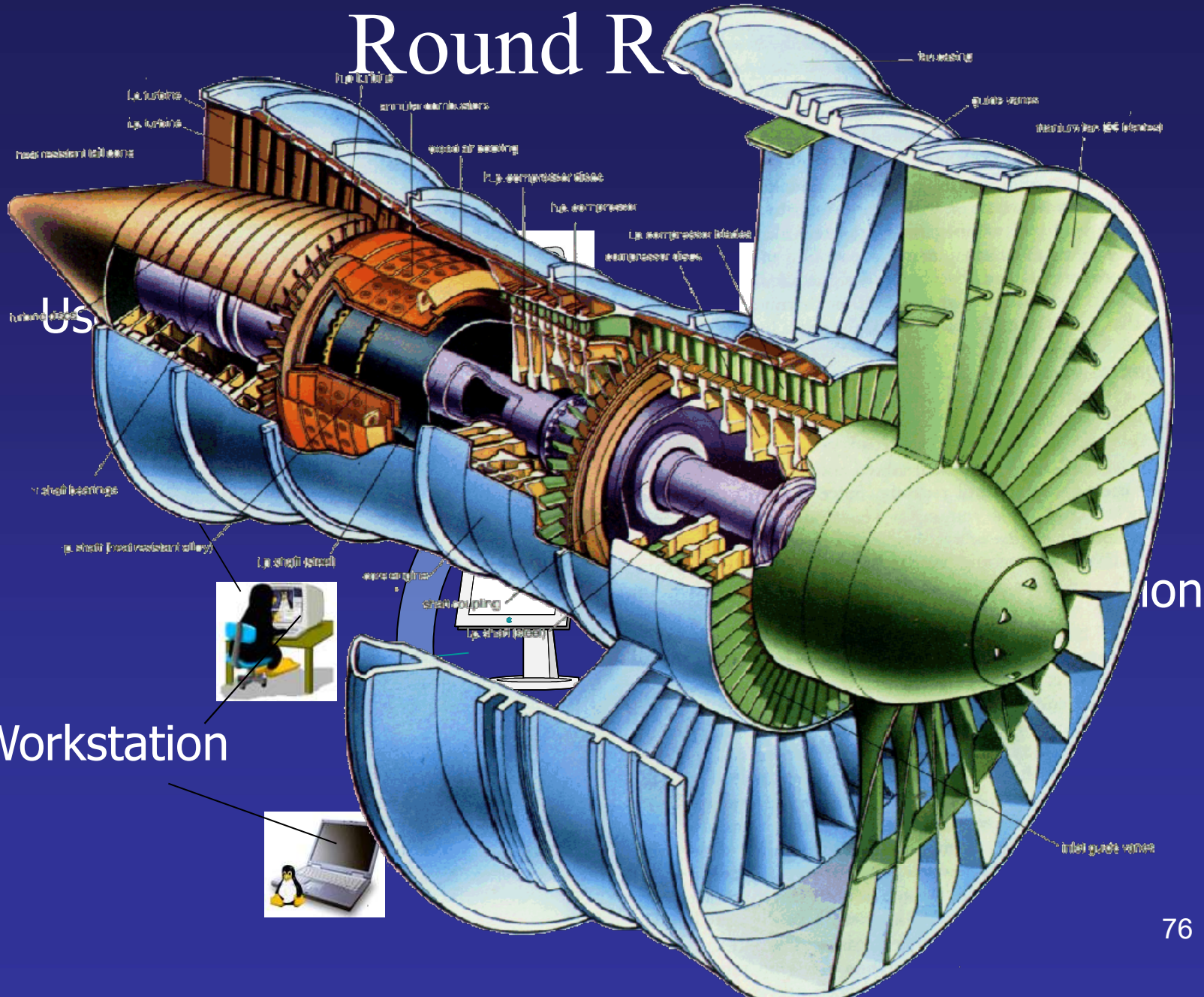
- Each process gets a small unit of CPU time
  - *time quantum* (usually 10-100 milliseconds)
  - After time quantum, the process is preempted and added to the end of the READY queue.
- Performance
  - $q$  large  $\Rightarrow$  FIFO
  - $q$  small  $\Rightarrow q$  must be large with respect to context switch, otherwise overhead is too high



# Round Robin



# Round Re



# Workstation

# Round Ro



ion

Wo



# Example of RR

Process Burst Time

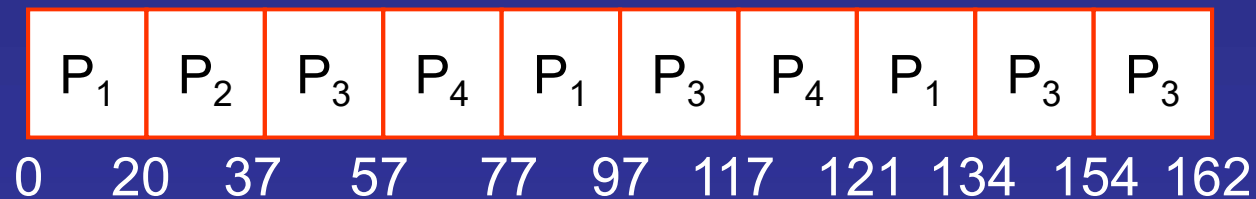
$P_1$  53

$P_2$  17

$P_3$  68

$P_4$  24

- Quantum is 20



# Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer  $\equiv$  highest priority)
  - Preemptive
  - Non-preemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem  $\equiv$  Starvation (low priority processes may never execute)
- Solution  $\equiv$  Aging (as time progresses increase the priority of the process)

# Multilevel Queue

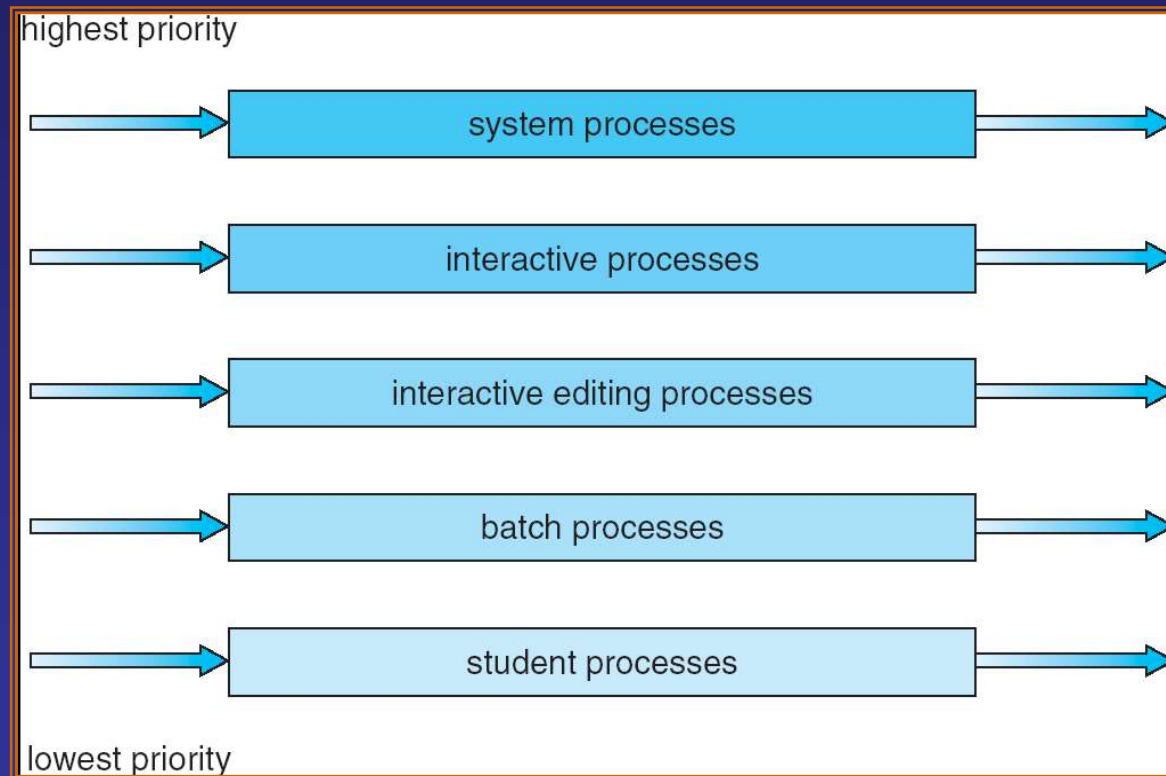
- Ready queue is partitioned into separate queues
  - foreground (interactive)
  - background (batch)
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS



# Multilevel Queue (cont'd)

- Scheduling must be done between the queues
  - Fixed priority scheduling
    - (i.e., serve all from foreground then from background)
    - Possibility of starvation
  - Time slice
    - each queue gets a certain amount of CPU time which it can schedule amongst its processes
      - i.e., 80% to foreground in RR
      - 20% to background in FCFS

# Multilevel Queue Scheduling



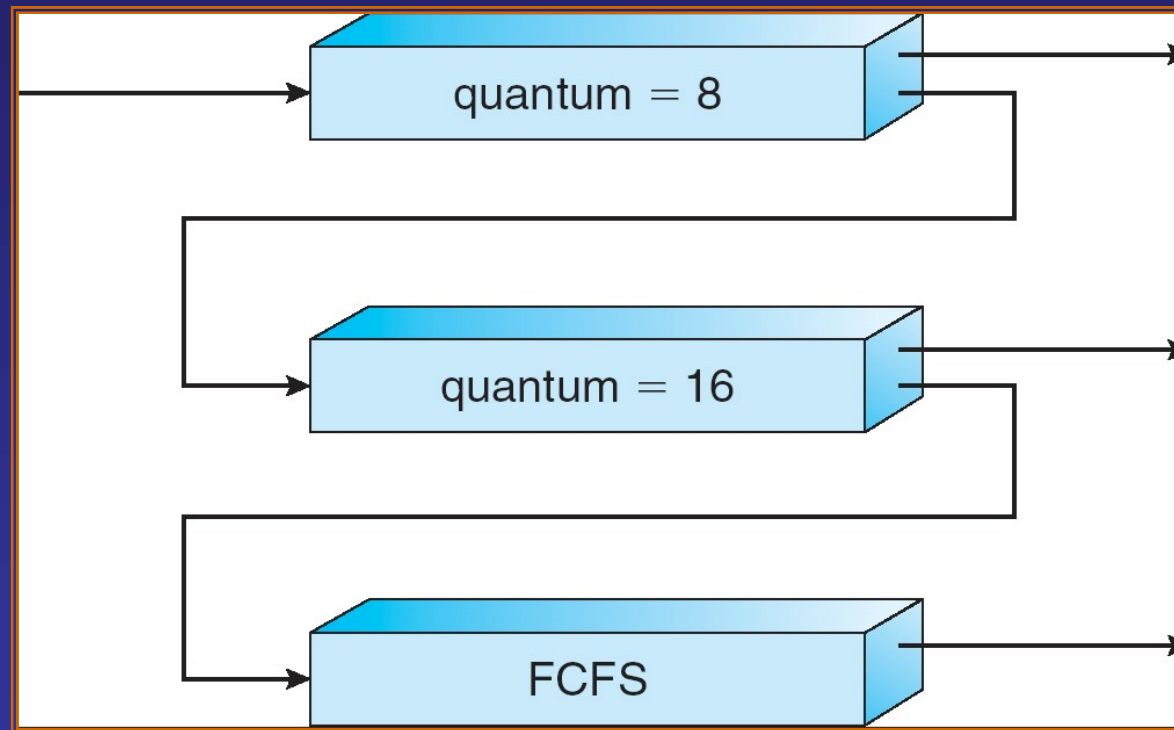
# Multilevel Feedback Queue

- A process can move between the various queues
  - aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

# Example

- Three queues:
  - $Q_0$  – RR with time quantum 8 milliseconds
  - $Q_1$  – RR time quantum 16 milliseconds
  - $Q_2$  – FCFS
- Scheduling
  - A new job enters queue  $Q_0$ . When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue  $Q_1$ .
  - At  $Q_1$  job receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue  $Q_2$ .

# Example



# Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
  - *Homogeneous processors* within a multiprocessor
  - *Load sharing*
- *Asymmetric multiprocessing*
  - only one processor accesses the system data structures, alleviating the need for data sharing



# Scheduling criteria

- CPU utilization
  - keep the CPU as busy as possible
- Throughput
  - # of complete processes per time unit
- Turnaround time
  - amount of time to execute a particular process
- Waiting time
  - amount of time waiting in the ready queue
- Response time
  - amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

# Question

Which is incorrect about scheduling optimization?

- A. Maximize turnaround time
- B. Maximize throughput
- C. Minimize waiting time
- D. Minimize response time

# First comes first serves (FCFS)

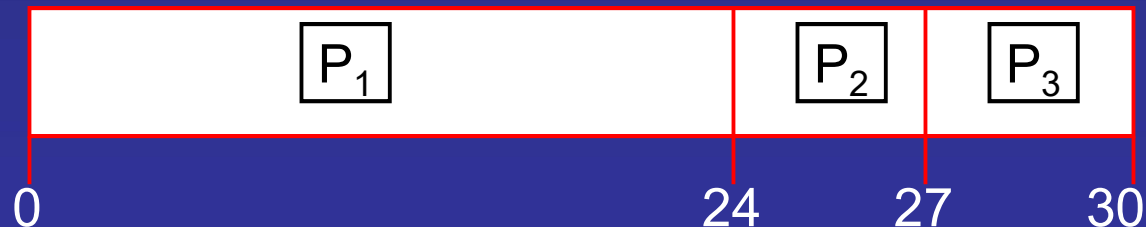
<u>Process</u>	<u>Burst Time</u>
----------------	-------------------

$P_1$	24
-------	----

$P_2$	3
-------	---

$P_3$	3
-------	---

- Suppose that the processes arrive in the order:  $P_1, P_2, P_3$   
The Gantt Chart for the schedule is:



# Question

Which is the total waiting time in FCFS example?

- A. 21
- B. 31
- C. 41
- D. 51

# Question

Which is the average waiting time in FCFS example?

- A. 15
- B. 16
- C. 17
- D. 18

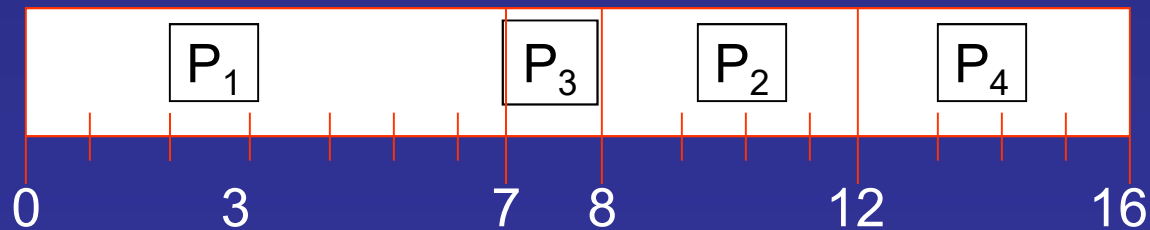
# Question

Which is the throughput in FCFS example?

- A. 0.1
- B. 0.2
- C. 0.3
- D. 0.4

# Example of Non-Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	7
$P_2$	2	4
$P_3$	4	1
$P_4$	5	4





# Question

Which is the total waiting time in **non-preemptive** SJF example?

- A. 15
- B. 16
- C. 17
- D. 18

# Question

Which is the average waiting time in the **non-preemptive** SJF example?

- A. 2
- B. 3
- C. 4
- D. 6

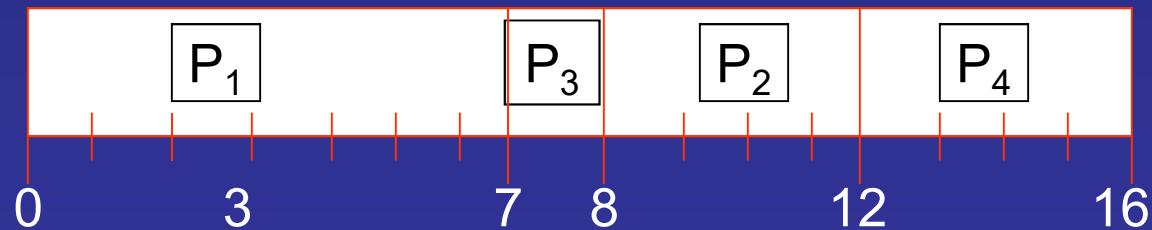
# Question

Which is the throughput in the **non-preemptive** SJF example?

- A. 0.65
- B. 0.25
- C. 0.35
- D. 0.45

# Example of Non-Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	7
$P_2$	2	4
$P_3$	4	1
$P_4$	5	4



# Question

What is the turnaround time of  $P_2$  in the SJF example?

- A. 6
- B. 8
- C. 10
- D. 12

# Question

What is the response time of  $P_2$  in the SJF example?

- A. 6
- B. 8
- C. 4
- D. 0







Question?