OPERATING SYSTEMS (THEORY)

LECTURE - 3

K.ARIVUSELVAN

Assistant Professor (Senior) – (SITE)

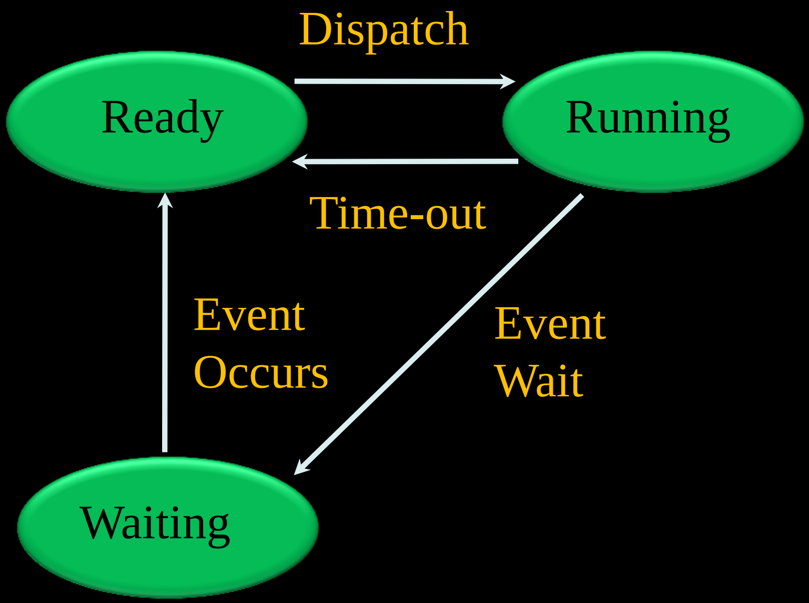
VIT University

PROCESS STATES

=> Three State Process Model

=> Five State Process Model

Three State Process Model



Dispatch

Ready

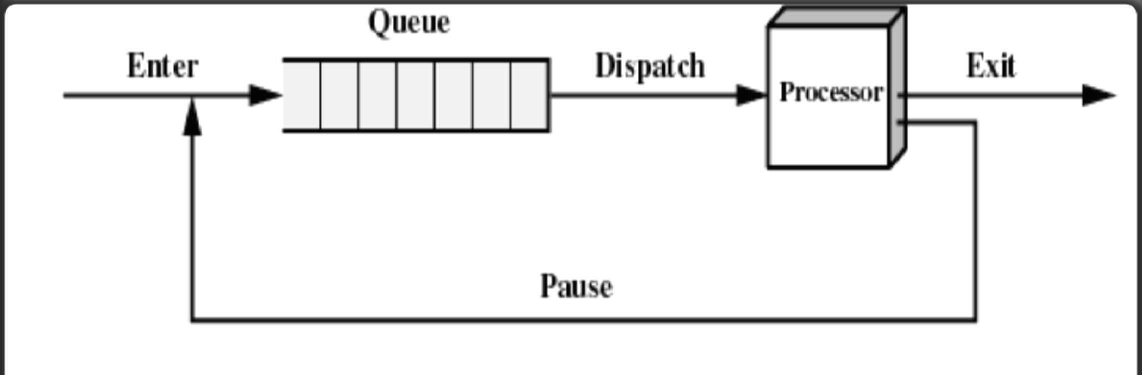
Running

Time-out

Event Occurs

Event Wait

Waiting



(b) Queuing diagram

Ready → Running

- **Dispatcher** selects a **new process** to run

Running → Ready

- Running process has **expired** his time slot

- Running process gets **interrupted** because a **higher priority process** is in the ready state

Running → Waiting

- An access to a **resource not yet available**

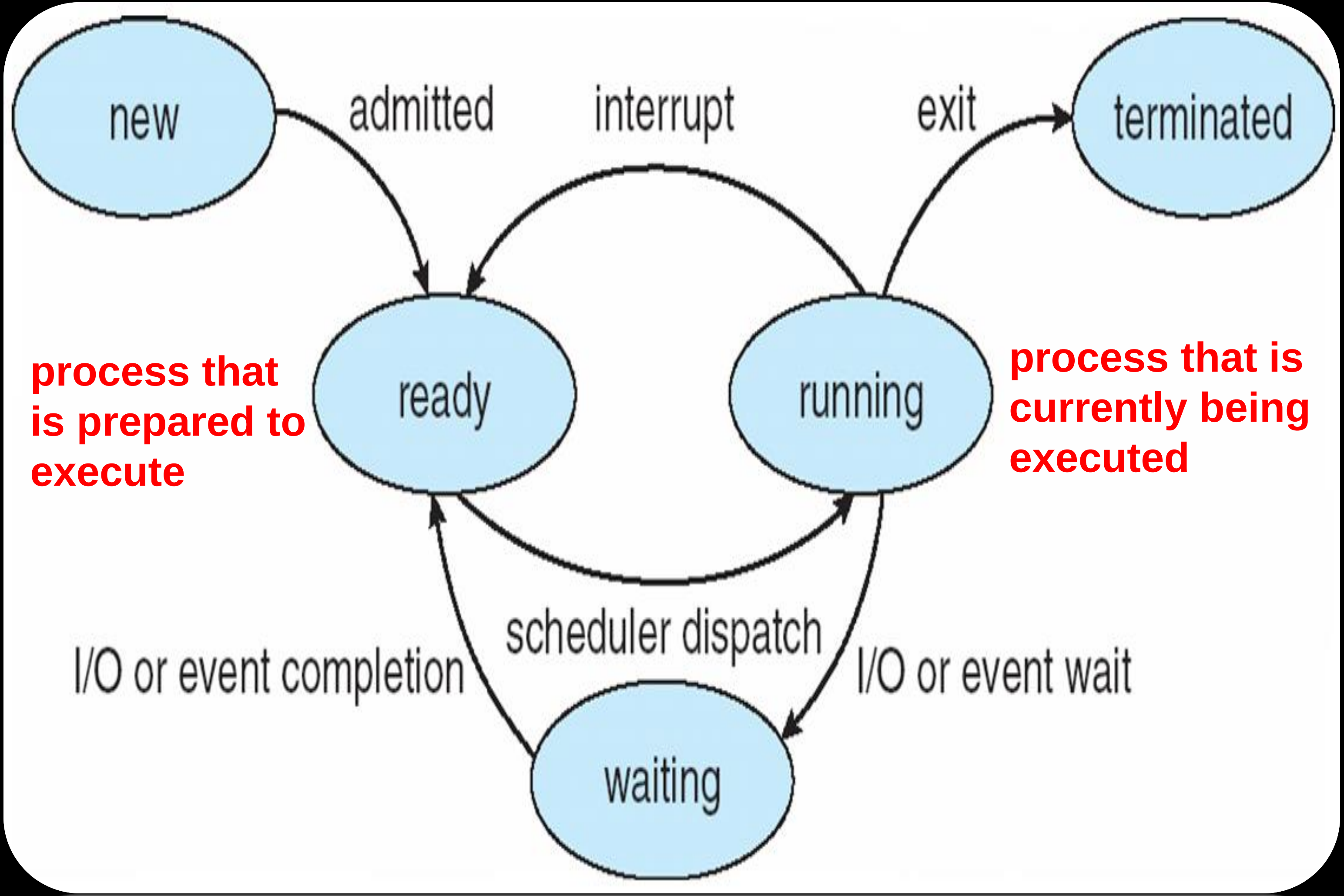
- Waiting for a **process** to **provide input**

Waiting → Ready

- The **event** for which it was waiting **occurs**

Five State Process Model

process that has just been created



process that is prepared to execute

process that is currently being executed

Reason For Process Creation

- Submission of a batch job

- User logs on

- Created by OS to provide a service to a user (e.g., printing a file)

PROCESS TERMINATION

Reasons:

(1) Normal completion

(2) Time Limit Exceeded

Process has run longer than the specified total time

(3) Memory Unavailable

Process require more memory than the system can provide

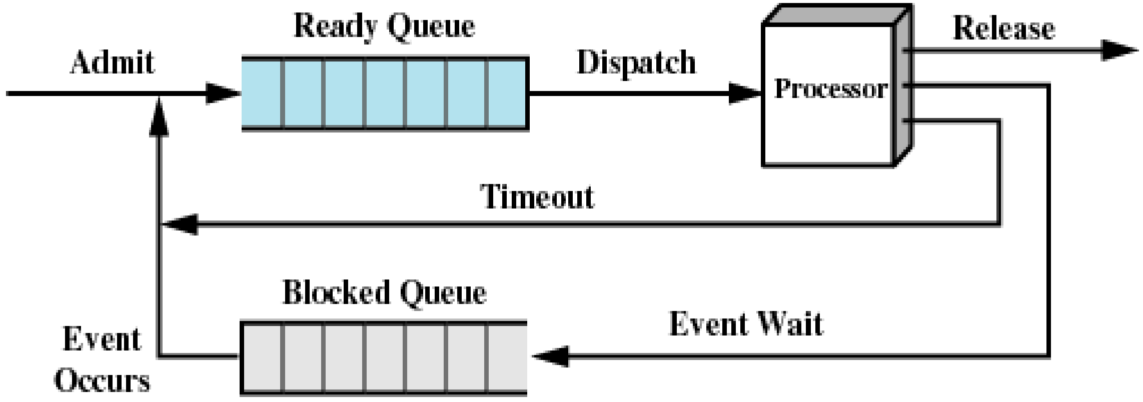
(4) Bounds Violation

Process tries to access a memory location that is not allowed to access

(5) Arithmetic Error

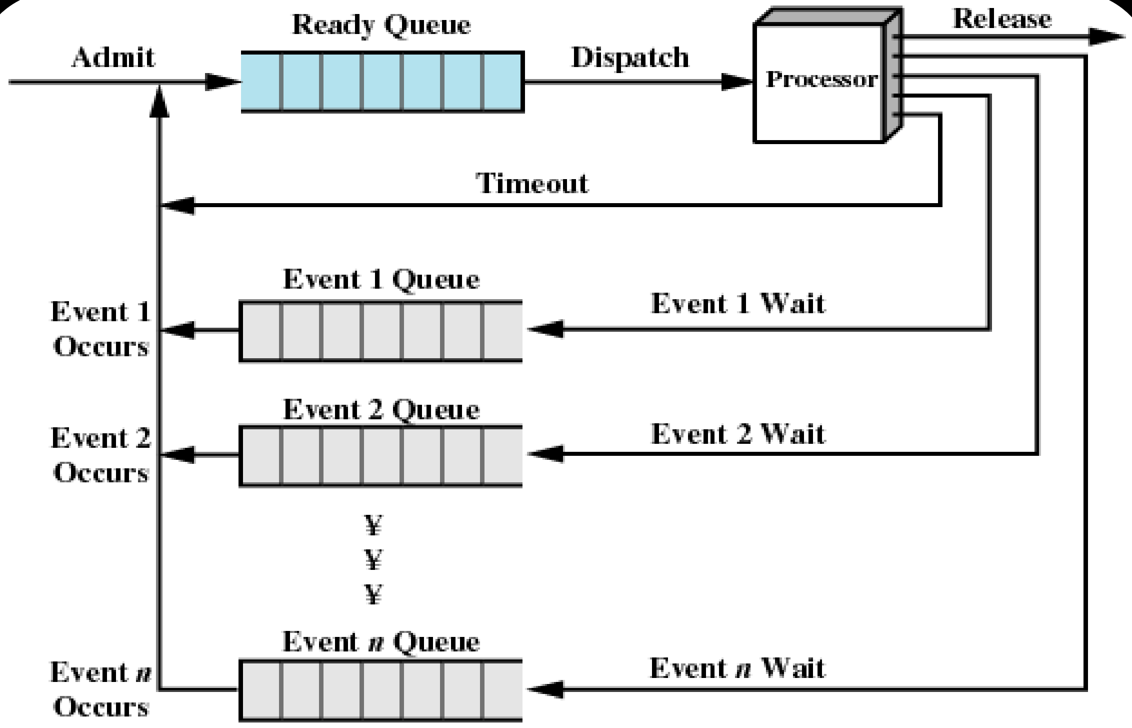
Process tries a prohibited computation, such as divisible by zero

TWO QUEUES



(a) Single blocked queue

Multiple Blocked Queues



(b) Multiple blocked queues

Modes of Execution

2 Modes:

=>**User Mode** (**Less** Privileged Mode)

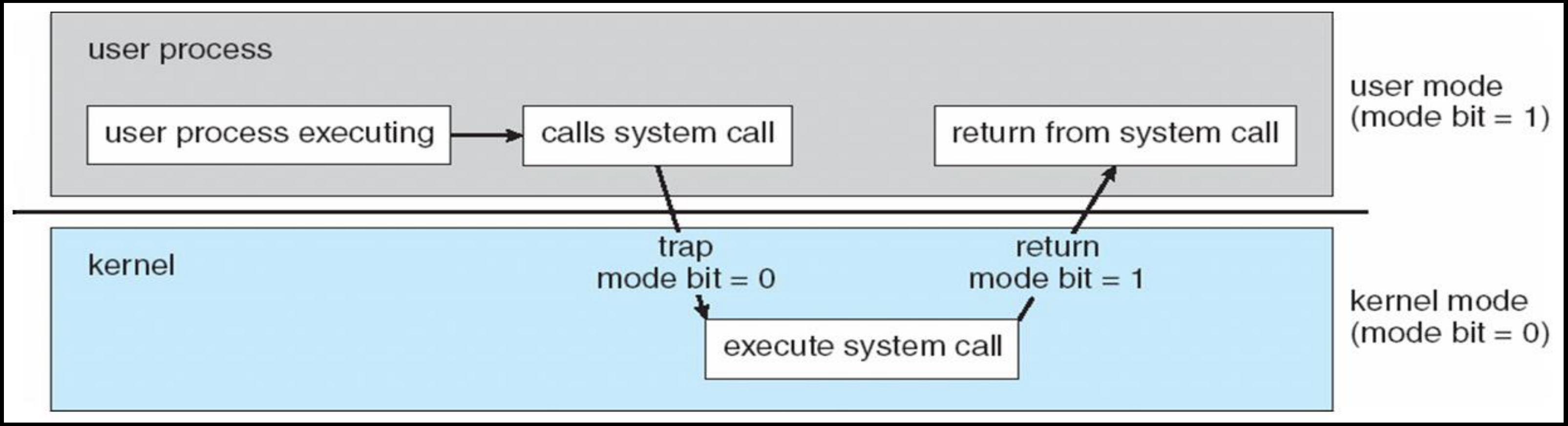
=>**System Mode / Kernel Mode** (**More** Privileged Mode)

Why ?

- **To protect OS programs from interface by User programs**

How ?

- **PSW** indicates the **mode of execution**



Process Switching

When to Switch a Process?

(1) Trap :

An error resulted from the last instruction–(it may cause the process to be moved to terminated state)

(2) Interrupt:

The cause is external to the execution of the current instruction – (control is transferred to Interrupt Handler)

Context Switch

Main Idea:

- The act of swapping a process state on or off the CPU is a 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 of a process represented in the PCB

Linux PCB Structure (task_struct)

```
struct task_struct {
volatile long state;
unsigned long flags;          Execution state
int sigpending;
mm_segment_t addr_limit;
struct exec_domain *exec_domain;
volatile long need_resched;
unsigned long ptrace;
int lock_depth;
unsigned int cpu;
int prio, static_prio;
struct list_head run_list;
prio_array_t *array;
unsigned long sleep_avg;
unsigned long last_run;
unsigned long policy;
unsigned long cpus_allowed;
unsigned int time_slice, first_time_slice;
atomic_t usage;
struct list_head tasks;
struct list_head ptrace_children;
struct list_head ptrace_list;
struct mm_struct *mm, *active_mm;   Memory mgmt info
struct linux_binfmt *binfmt;
int exit_code, exit_signal;
int pdeath_signal;
unsigned long personality;
int did_exec:1;
unsigned task_dumpable:1;
pid_t pid;   Process ID
pid_t pgrp;
pid_t tty_old_pgrp;
pid_t session;
pid_t tgid;
int leader;
struct task_struct *real_parent;
struct task_struct *parent;
struct list_head children;
struct list_head sibling;
struct task_struct *group_leader;
struct pid_link pids[PIDTYPE_MAX];
wait_queue_head_t wait_chldexit;
struct completion *vfork_done;
int *set_child_tid;
int *clear_child_tid;
unsigned long rt_priority;   Priority
```

```
unsigned long it_real_value, it_prof_value, it_virt_value;
unsigned long it_real_incr, it_prof_incr, it_virt_incr;
struct timer_list real_timer;
struct tms times;
struct tms group_times;                             Accounting info
unsigned long start_time;
long per_cpu_utime[NR_CPUS], per_cpu_stime[NR_CPUS];
unsigned long min_flt, maj_flt, nswap, cmin_flt, cmaj_flt,
cnswap;
int swappable:1;
uid_t uid,euid,suid,fsuid;   User ID
gid_t gid,egid,sgid,fsgid;
int ngroups;
gid_t groups[NGROUPS];
kernel_cap_t   cap_effective, cap_inheritable, cap_permitted;
int keep_capabilities:1;
struct user_struct *user;
struct rlimit rlim[RLIM_NLIMITS];
unsigned short used_math;
char comm[16];
int link_count, total_link_count;
struct tty_struct *tty;
unsigned int locks;
struct sem_undo *semundo;
struct sem_queue *semsleeping;
struct thread_struct thread;   CPU state
struct fs_struct *fs;
struct files_struct *files;    Open files
struct namespace *namespace;
struct signal_struct *signal;
struct sighand_struct *sighand;
sigset_t blocked, real_blocked;
struct sigpending pending;
unsigned long sas_ss_sp;
size_t sas_ss_size;
int (*notifier)(void *priv);
void *notifier_data;
sigset_t *notifier_mask;
void *tux_info;
void (*tux_exit)(void);
      u32 parent_exec_id;
      u32 self_exec_id;
spinlock_t alloc_lock;
        spinlock_t switch_lock;
void *journal_info;
unsigned long ptrace_message;
siginfo_t *last_siginfo;
};
```

Steps in Context Switch

- **Save context** of processor including **program counter** and **other registers**

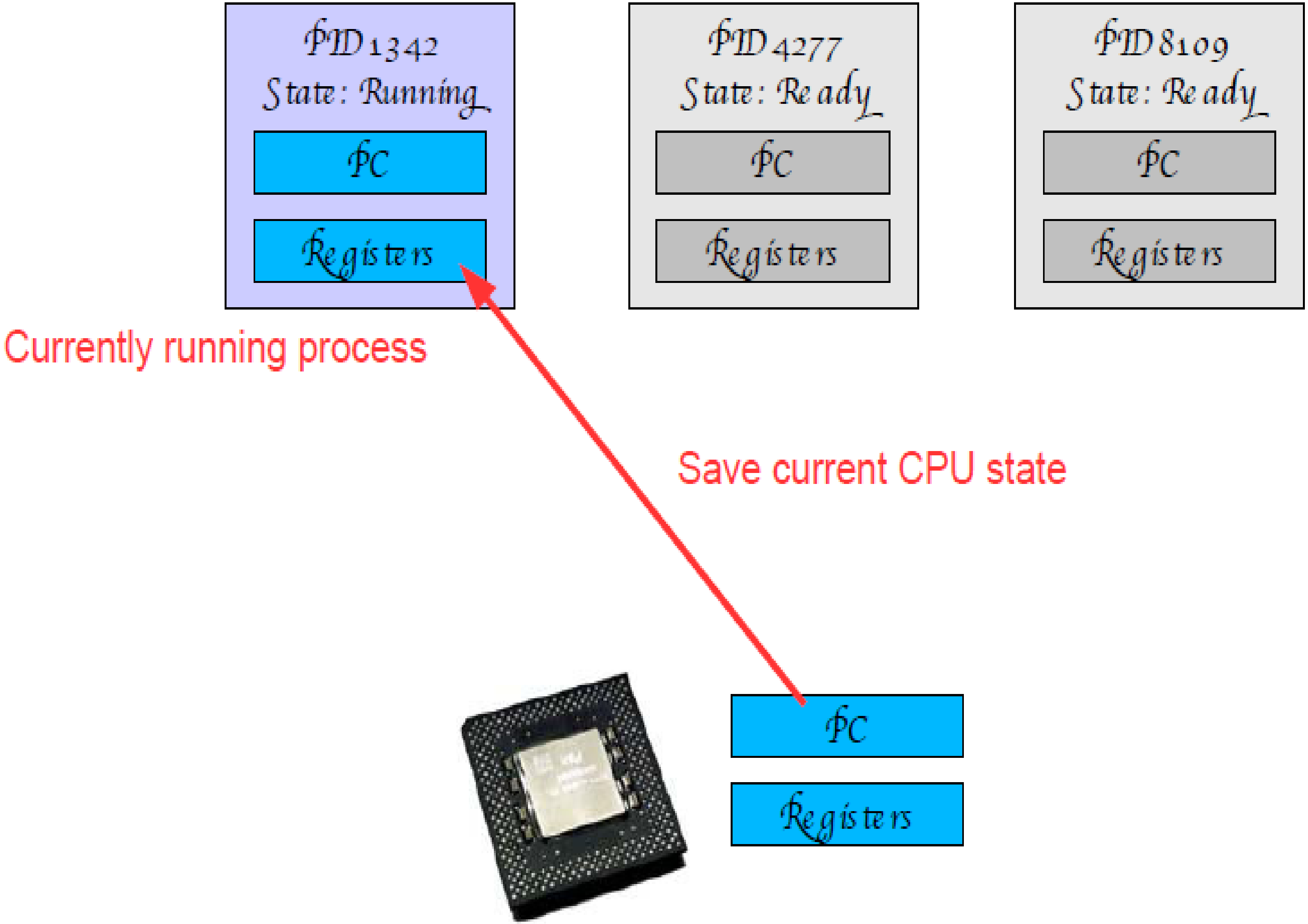
- **Update** the **PCB** of the **running process** with its **new state** and other associate information

- **Move PCB** to appropriate queue – **ready, (or)waiting**

- **Select** another process for **execution.**

- **Update PCB** of the selected process

- **Restore CPU context** from that of the **selected process.**



PID 1342
State: Running
PC
Registers

PID 4277
State: Ready
PC
Registers

PID 8109
State: Ready
PC
Registers

Currently running process

Save current CPU state

PC
Registers

PID 1342
State: Ready

PC

Registers

PID 4277
State: Ready

PC

Registers

PID 8109
State: Ready

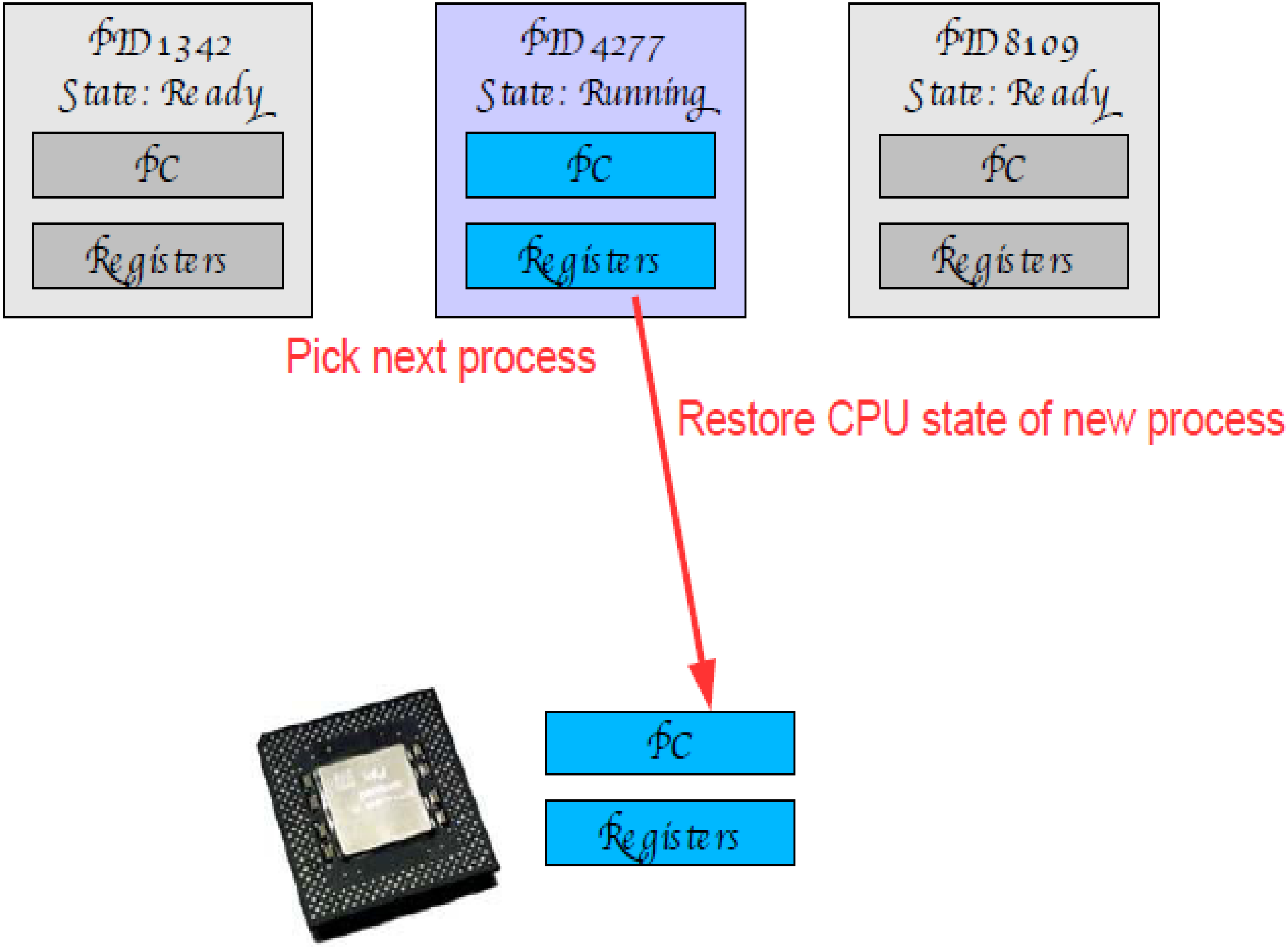
PC

Registers

Suspend process

PC

Registers



PID 1342
State: Ready
PC
Registers

PID 4277
State: Running
PC
Registers

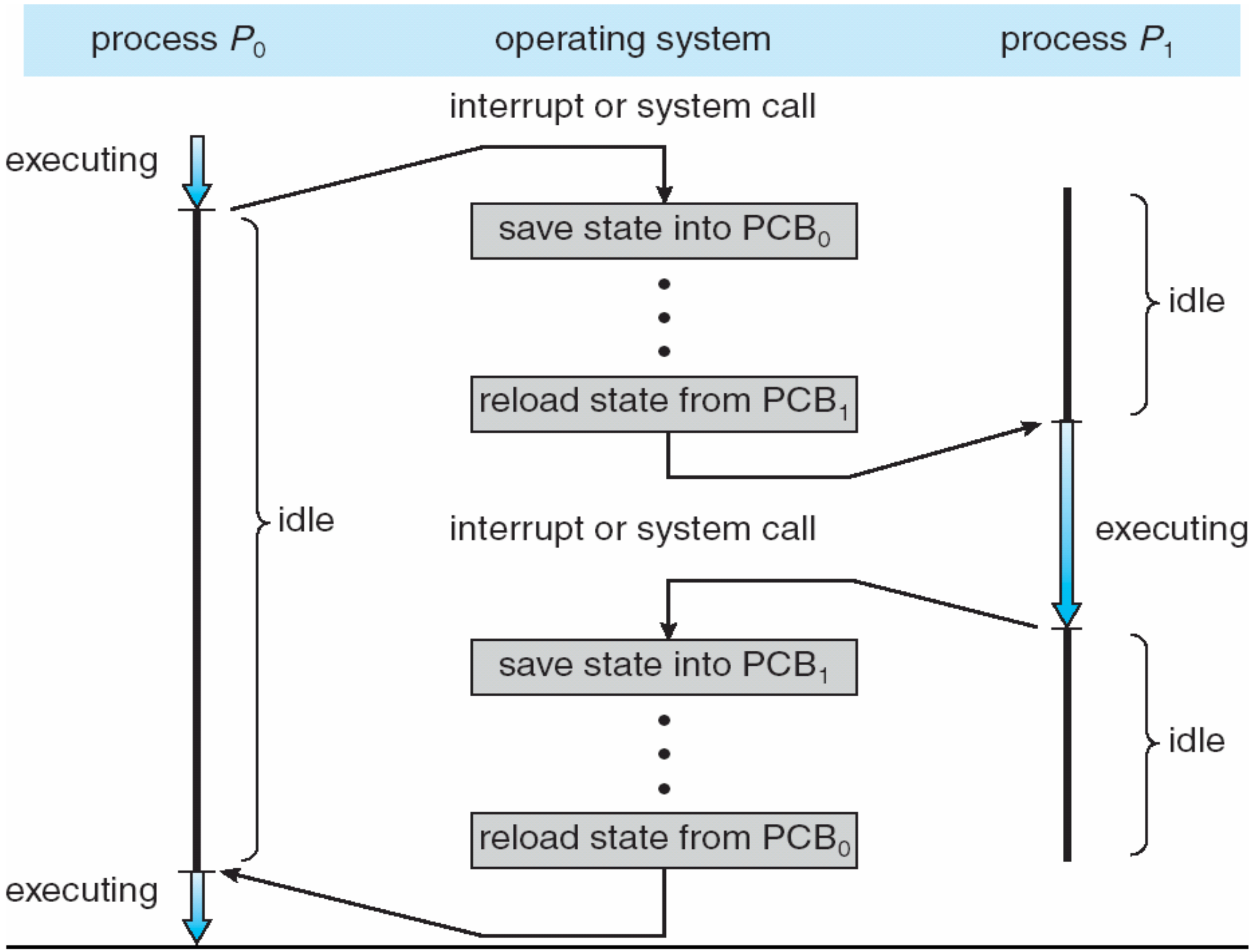
PID 8109
State: Ready
PC
Registers

Pick next process

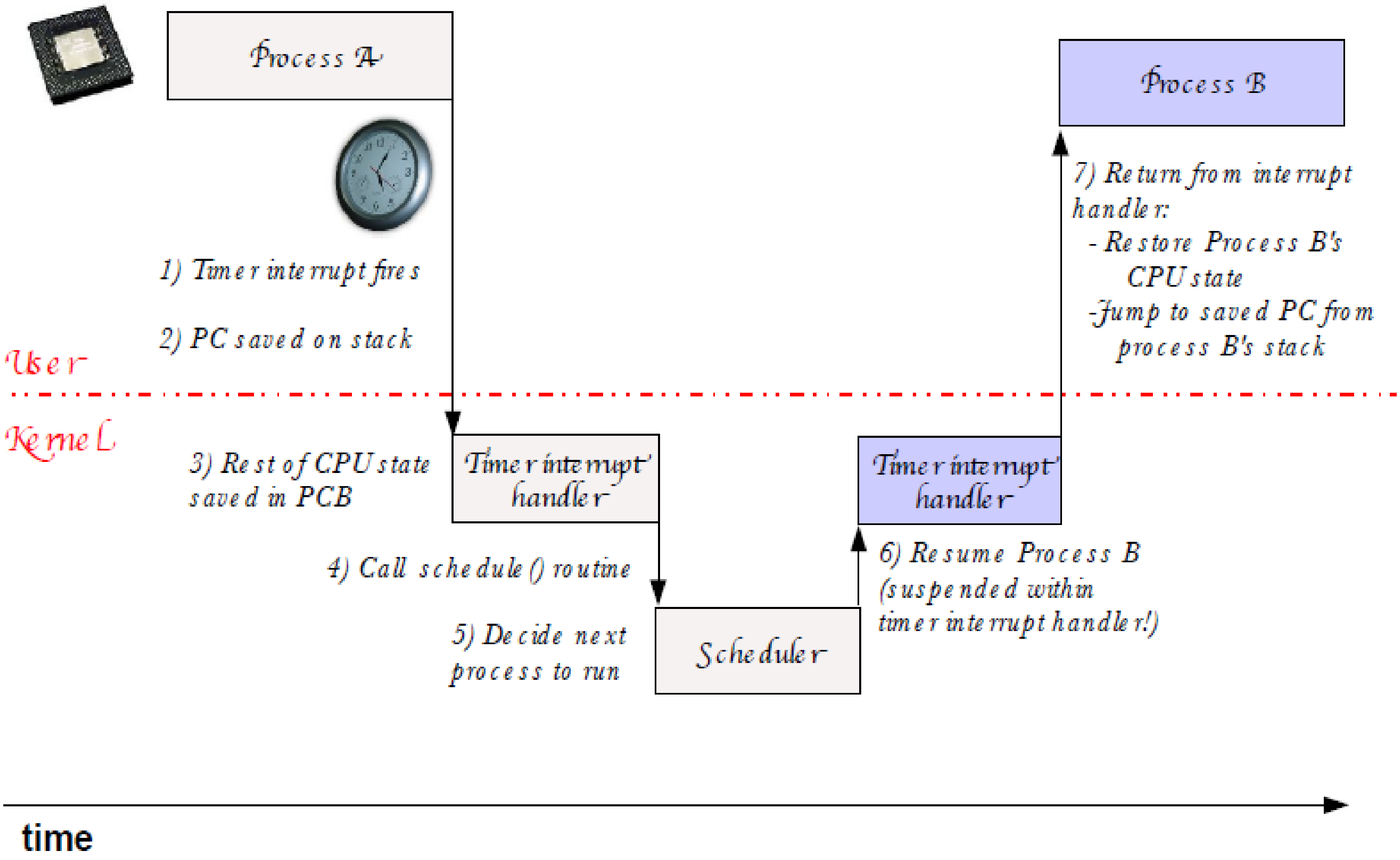
Restore CPU state of new process

PC
Registers

- Context-switch time is overhead; the system does no useful work while switching



Context Switching in Linux



Process A

1) Timer interrupt fires

2) PC saved on stack

User

Kernel

3) Rest of CPU state saved in PCB

Timer interrupt handler

4) Call schedule () routine

5) Decide next process to run

Scheduler

6) Resume Process B (suspended within timer interrupt handler!)

Timer interrupt handler

7) Return from interrupt handler:
- Restore Process B's CPU state
- Jump to saved PC from process B's stack

Process B

time

Revision

- Program
- Process
- Program VS Process
- Process Image
- PCB
- Process States
- Process Creation
- Process Termination
- Process Control Modes
- Process switching
- Context Switching