Operating Systems

(Process Concept)

Chapter 3

These lecture materials are modified from the source lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

Spring, 2020

Outline

- Process concept
- Process scheduling
- Operations on processes
- Inter-process communication

Process concept

- Process – a program in execution; process execution must progress in sequential fashion
- Textbook uses the terms *job, task* and *process* almost interchangeably

Process in memory

Process != Program

Process is more than the program code

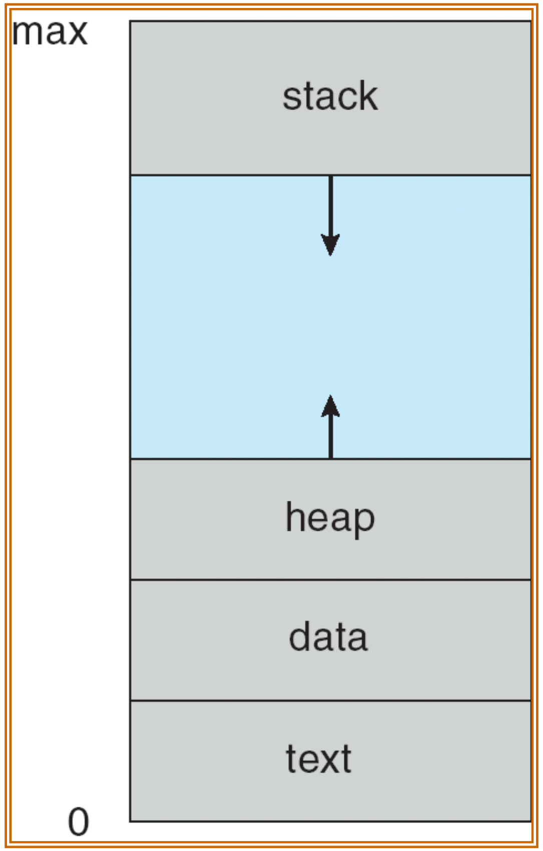
Stack (temporary data function parameters and local variables)

Heap (dynamically allocated memory during process run time)

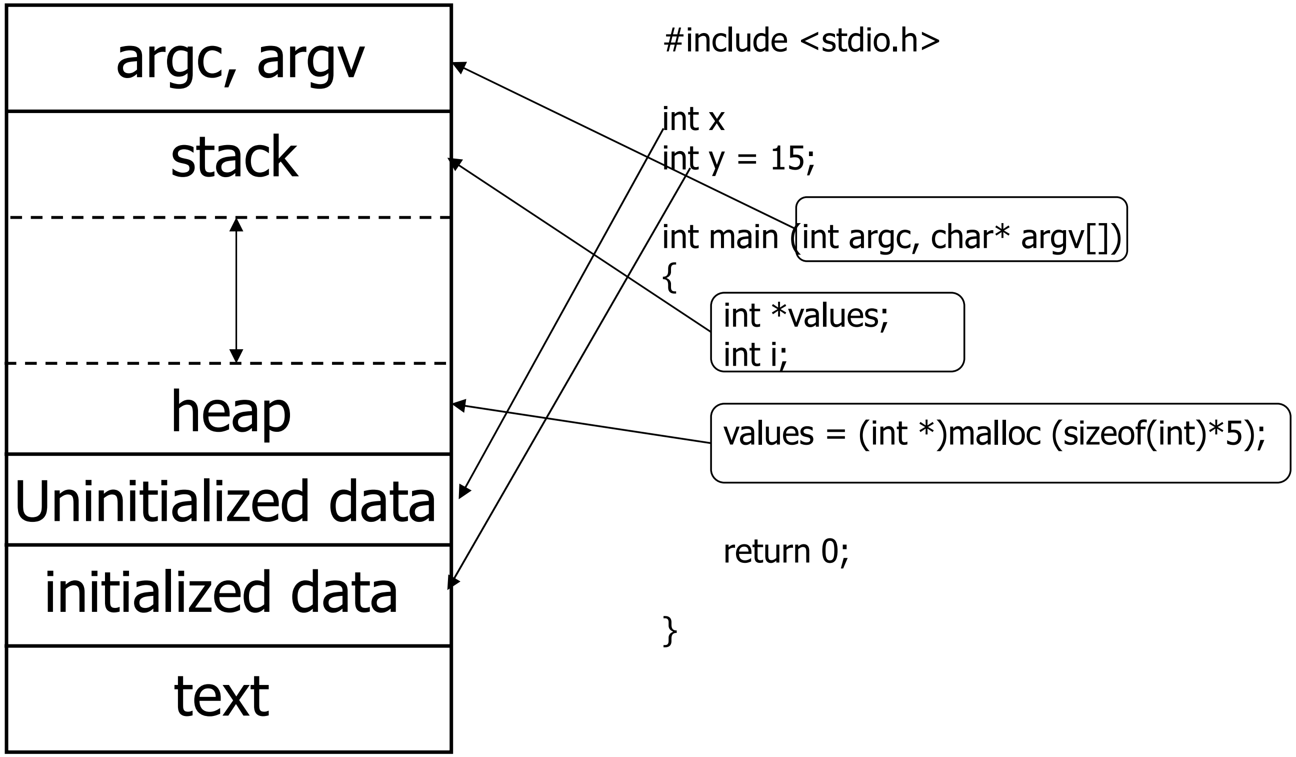
+ (PCB)

=> (Program counter)

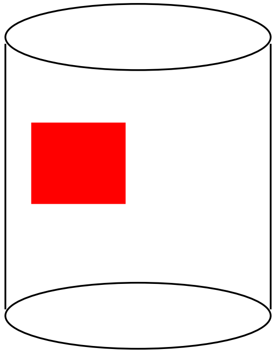
=> (Processor registers)



- Memory layout in C program



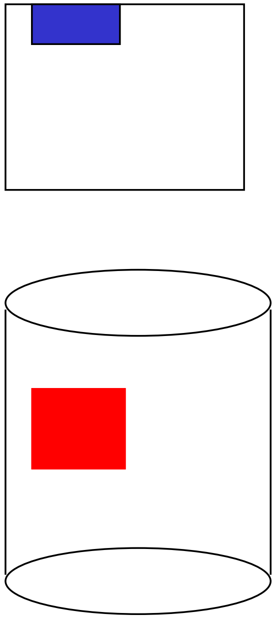
```
#include <stdio.h>

int x
int y = 15;

int main (int argc, char* argv[])
{
    int *values;
    int i;

    values = (int *)malloc (sizeof(int)*5);


    return 0;

}
```



Disk

Program

Memory

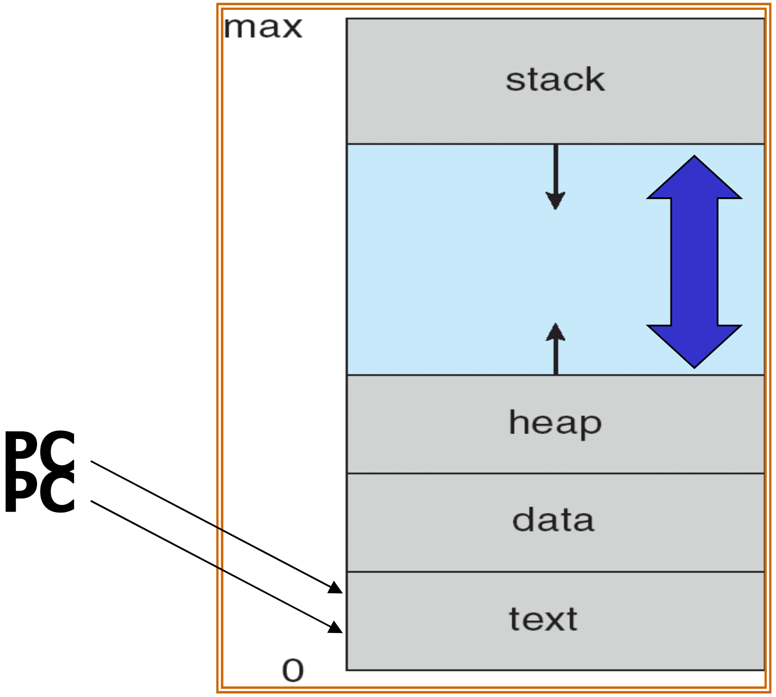


PCB, stack …

Disk

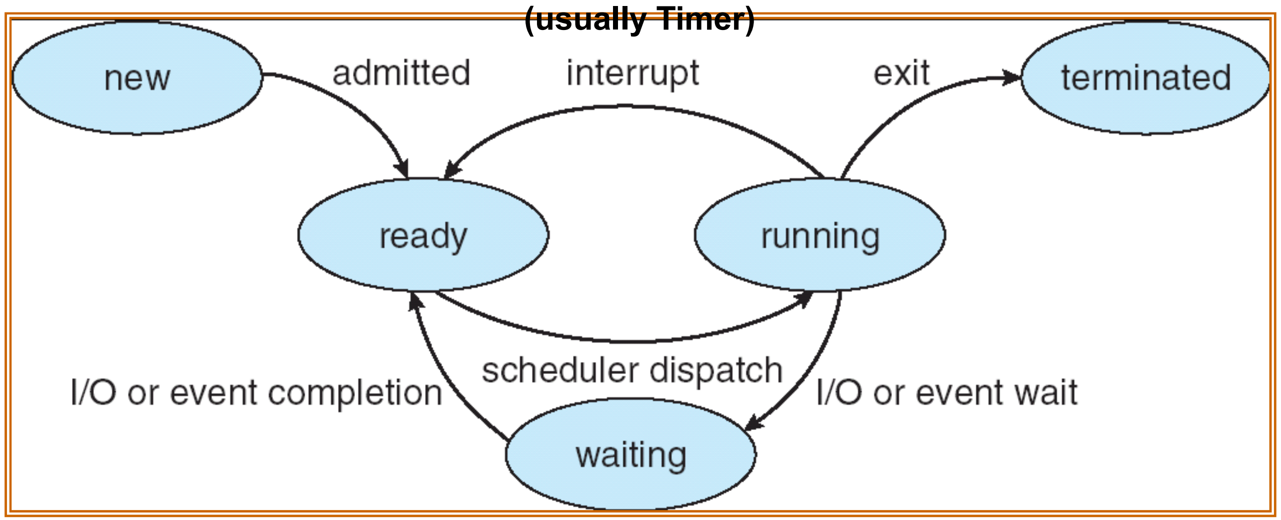
Process

- A program is a passive entity whereas a process is an active entity
  - Why ?

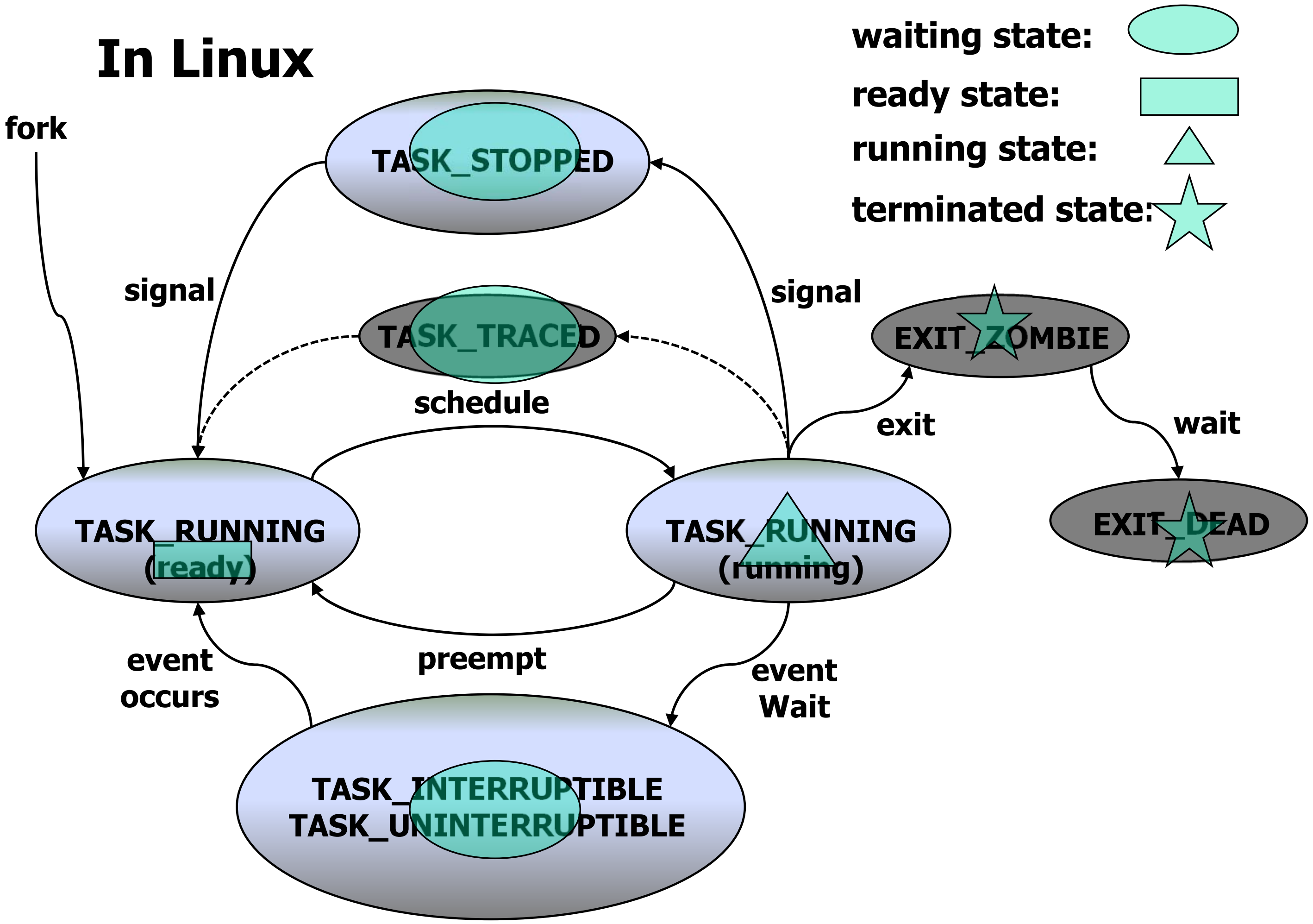


- Process state
  - As a process executes, it changes *state*
    - **new**: The process is being created
    - **running**: Instructions are being executed
    - **waiting**: The process is waiting for some event to occur
    - **ready**: The process is waiting to be assigned to a processor
    - **terminated**: The process has finished execution

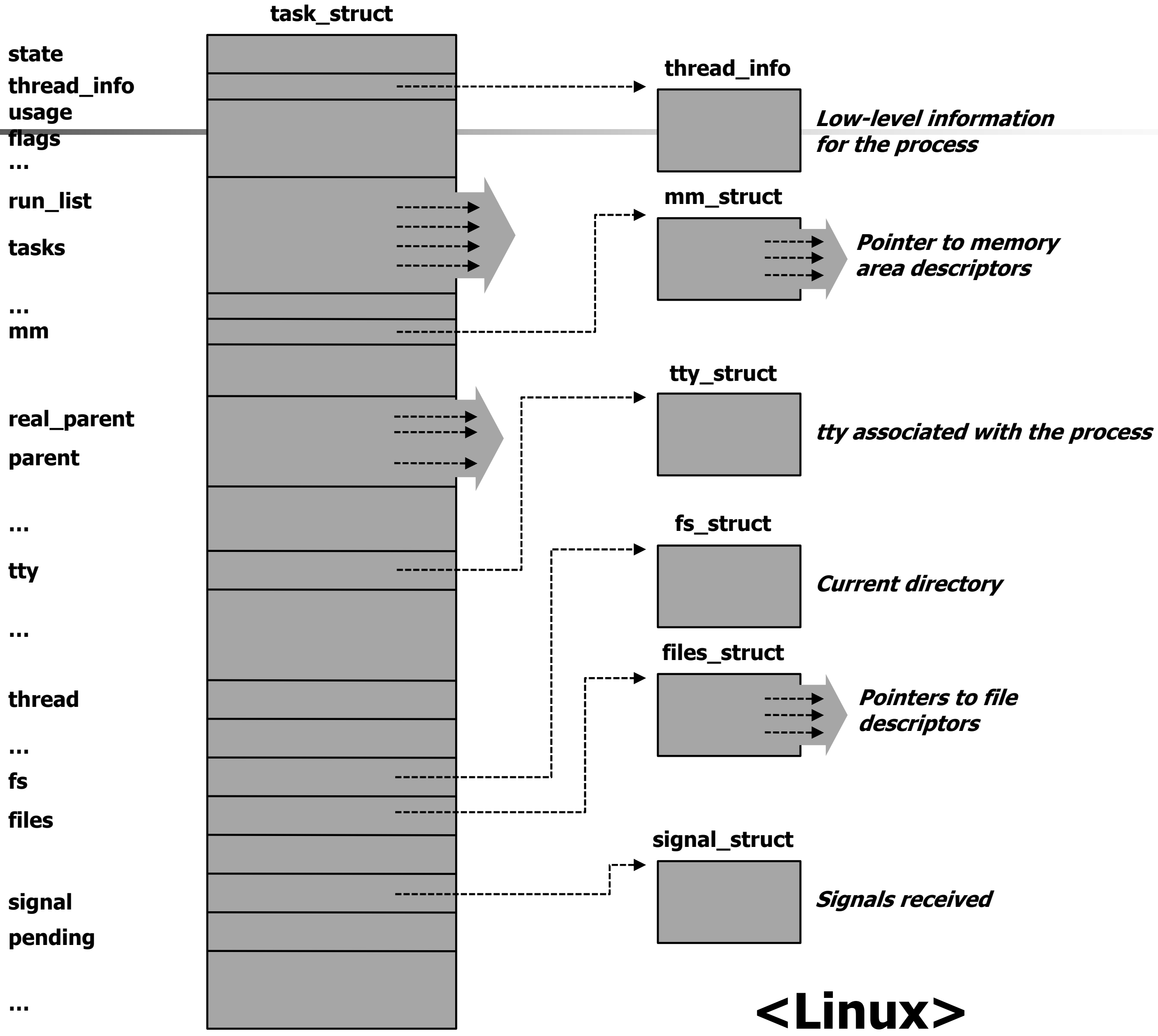
- State diagram



In Linux

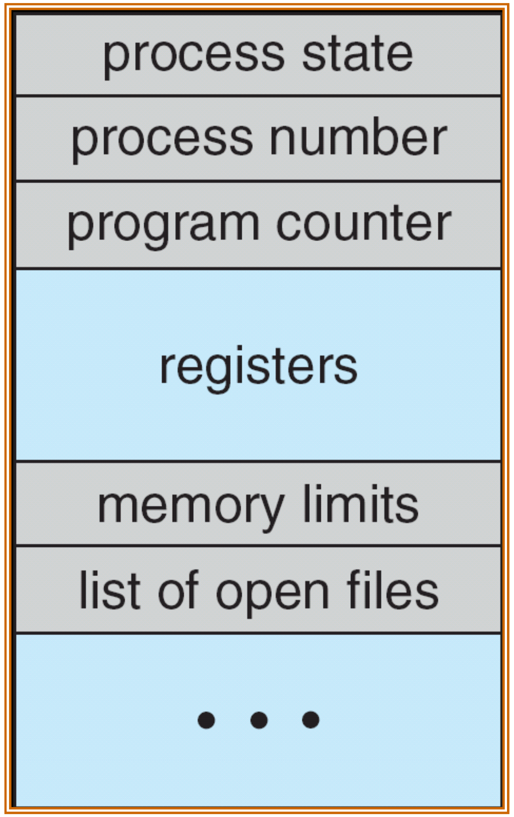

waiting state:
ready state:
running state:
terminated state:
fork
TASK_STOPPED
signal
signal
TASK_TRACED
schedule
EXIT_ZOMBIE
exit
wait
EXIT_DEAD
TASK_RUNNING
(ready)
TASK_RUNNING
(running)
preempt
event
occurs
event
Wait
TASK_INTERRUPTIBLE
TASK_UNINTERRUPTIBLE

- PCB (Process Control Block)
  - Information associated with each process
    - Process state
    - Program counter
      - Next instruction
    - CPU registers
      - General-purpose registers, stack pointer
    - CPU scheduling information
      - Such as process priority information
    - Memory-management information
    - Accounting information
    - I/O status information
      - Opened files


task_struct
state
thread_info
usage
flags
...
run_list
tasks
...
mm
real_parent
parent
...
tty
...
thread
...
fs
files
signal
pending
...
thread_info
Low-level information for the process
mm_struct
Pointer to memory area descriptors
tty_struct
tty associated with the process
fs_struct
Current directory
files_struct
Pointers to file descriptors
signal_struct
Signals received

<Linux>

- PCB



process state

process number

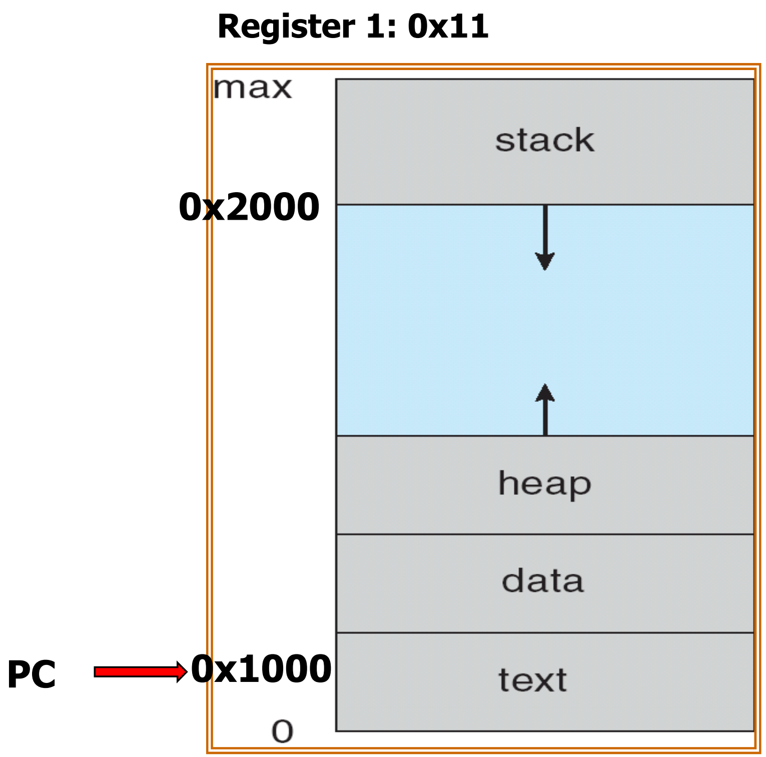
program counter

registers

memory limits

list of open files

• • •



Running
?
?
?
…

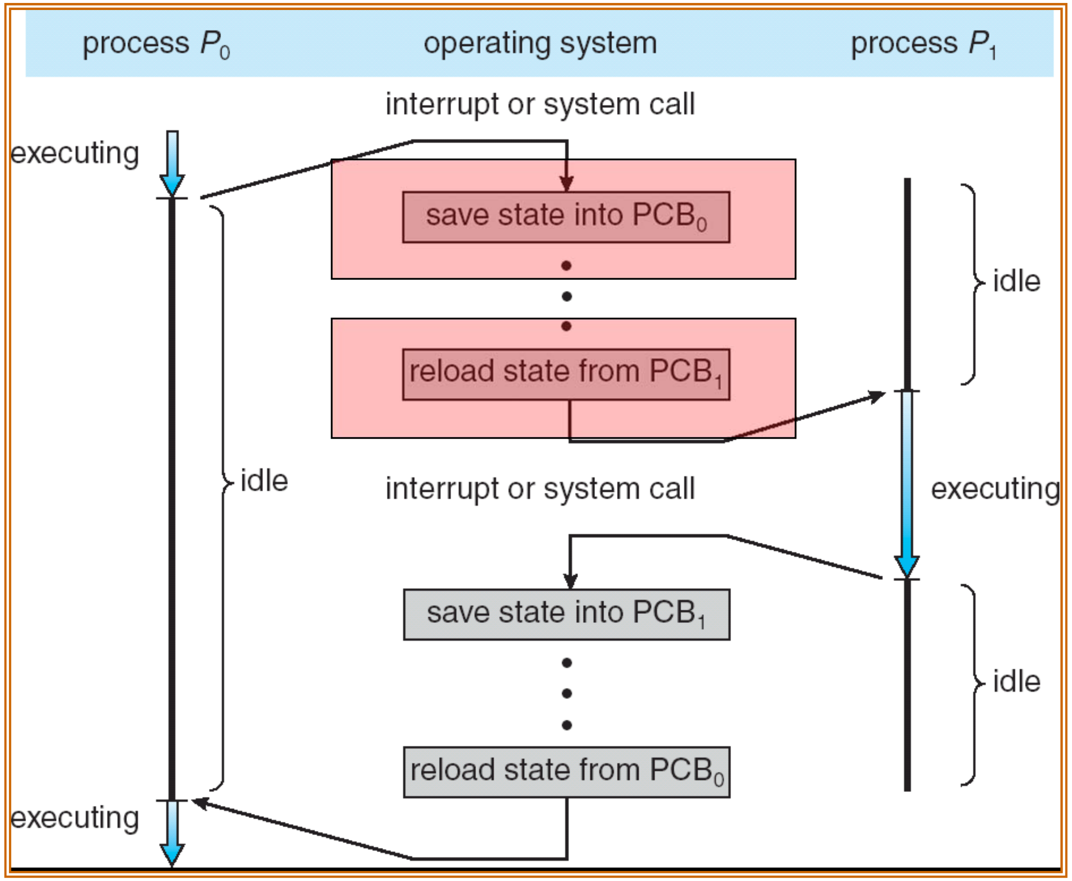
⇩

Ready
0x1000
0x2000
0x11
…

state
PC
SP
Register 1
…

PCB

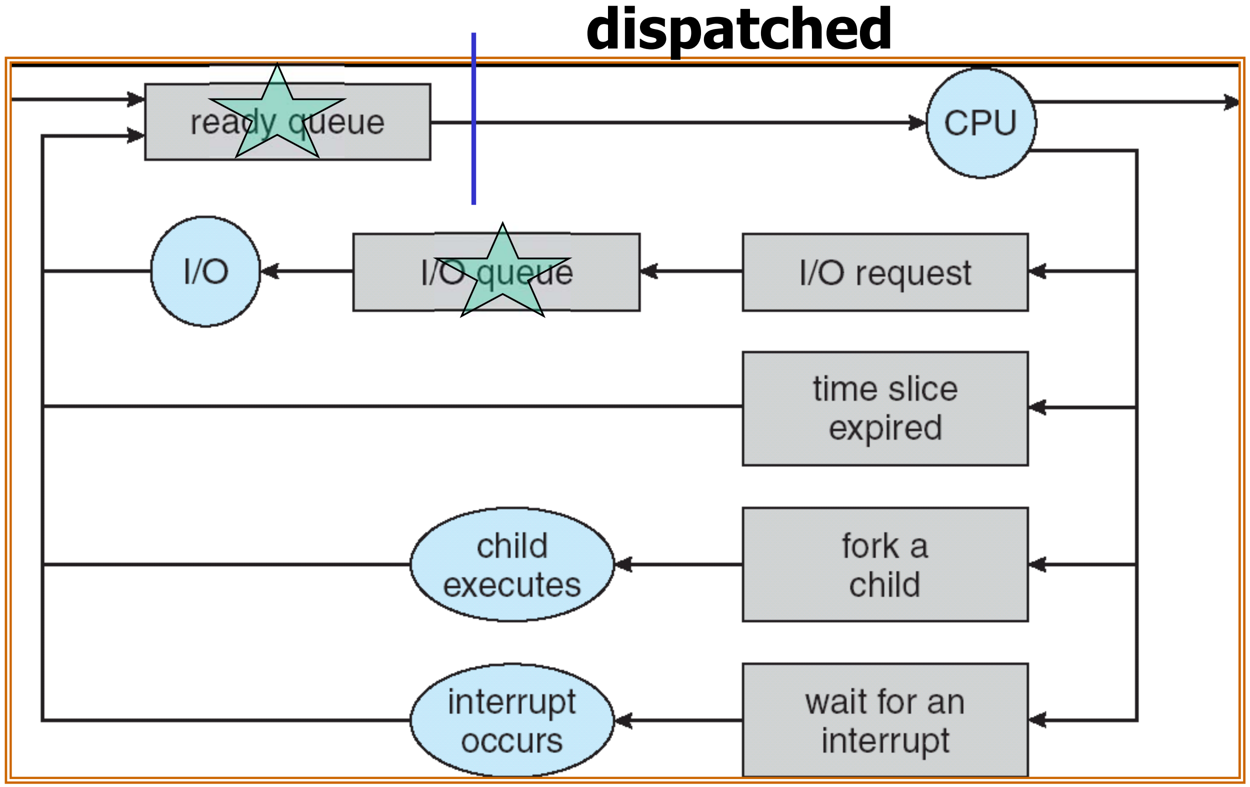
CPU is going to disptach another process

process P_0
operating system
process P_1
interrupt or system call
executing
save state into PCB_0
idle
reload state from PCB_1
idle
interrupt or system call
executing
save state into PCB_1
idle
reload state from PCB_0
executing

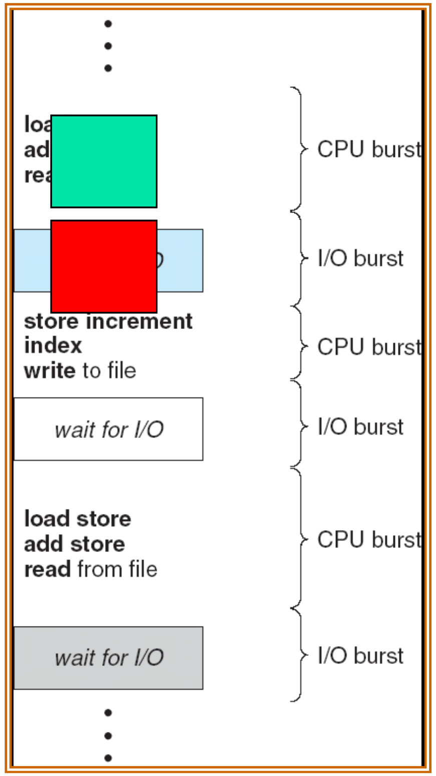
Process scheduling

- Process scheduling queue
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

- Process scheduling queueing diagram



- Processes can be described as:
  - **I/O-bound process**
    - spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process**
    - spends more time doing computations; few very long CPU bursts



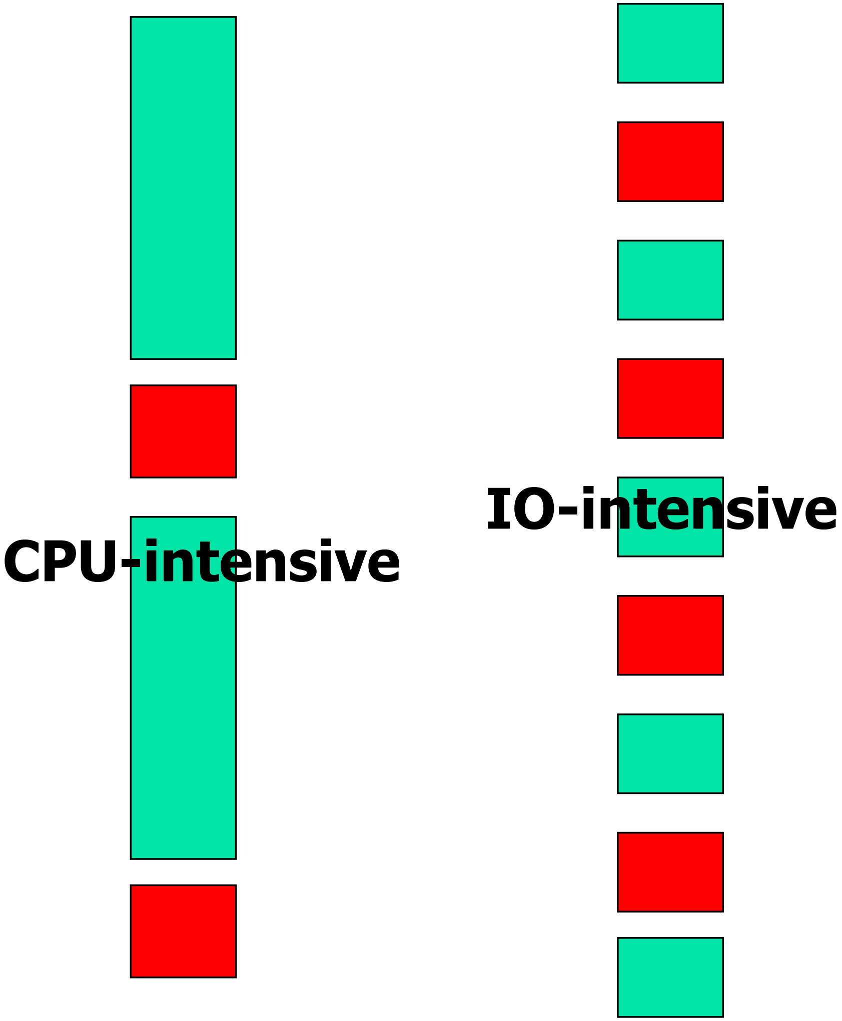
loa
ad
rea

store increment
index
write to file

wait for I/O

load store
add store
read from file

wait for I/O



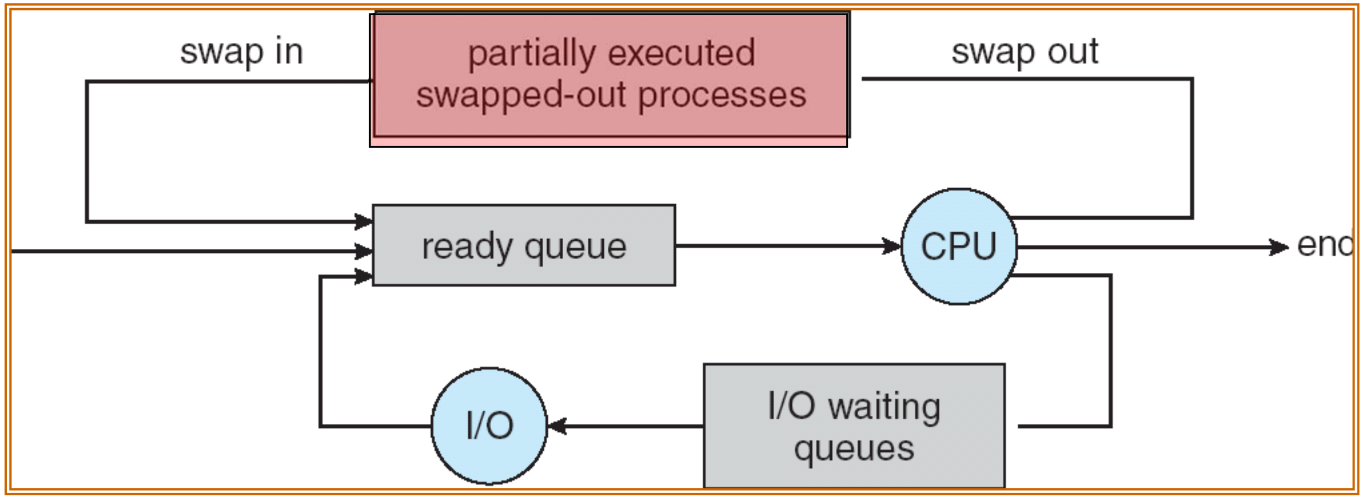
- Schedulers tend to give higher priorities to I/O-bound processes over CPU-bound processes

- Scheduler
  - 1. Long-term scheduler
  - 2. Short-term scheduler
  - 3. Sometimes medium-term scheduler

- 1. Long-term scheduler (job scheduler)
  - Selects which processes should be brought into the ready queue
    - How to allocate memory
  - Invoked very infrequently (seconds, minutes)
  - Controls the *degree of multiprogramming*
    - Degree of multiprogramming
      - Number of processes in memory
    - So, it determines system stability
  - may be absent or minimal
    - Current UNIX and Windows often have no long-term scheduler
    - They simply put every new process in memory

- 2. Short-term scheduler (CPU scheduler)
  - selects which process should be executed next and allocates CPU
    - How to allocate CPU resource
  - invoked very frequently (milliseconds) ⇒ (must be fast)

- 3. Medium-term scheduler



- 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-switch time is a big overhead;
    - The system does no useful work while switching.
    - There may be some mechanisms to reduce such overhead
      - SUN UltraSPARC
        - **Multiple sets of registers**
      - ARM
        - **Multiple store and load instructions**

ARM architecture;

LDMIA R0, {R5-R8}

R0:

0x00001000

R5:

0x11111111

R6:

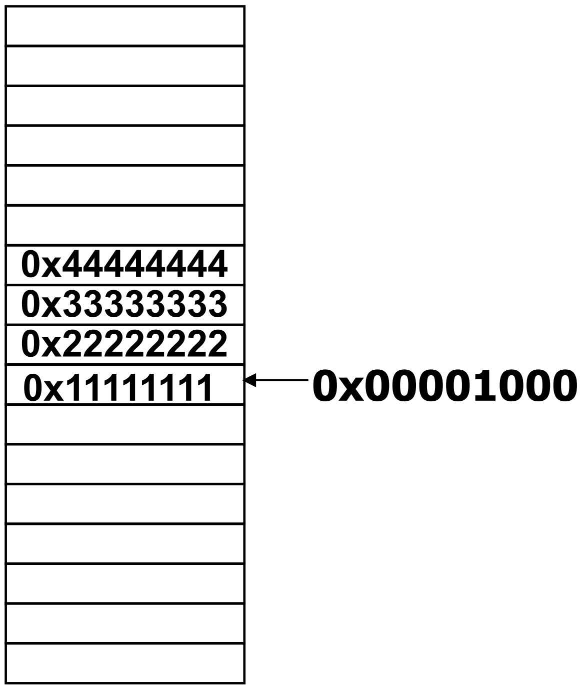
0x22222222

R7:

0x33333333

R8:

0x44444444


0x44444444
0x33333333
0x22222222
0x11111111
0x00001000