IOWA STATE UNIVERSITY

Department of Electrical and Computer Engineering

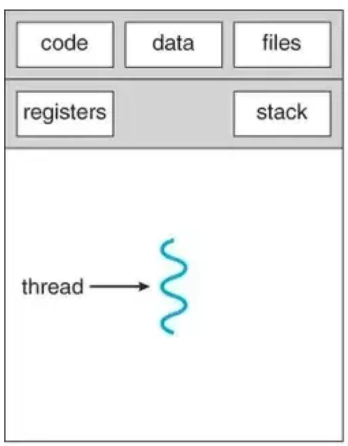
Lecture 09:
CPU Scheduling I

Agenda

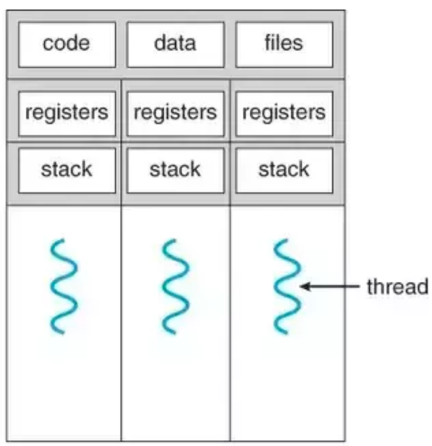
- Recap
- CPU Scheduling I
    - Scheduling Concept
    - POSIX Threads

Recap

- Thread Concept
  - Multiple threads of control *within* a process
    - All threads within a process share the same
      - text, heap, static data segments, open files ...
    - each thread has its own
      - State, PC, registers, stack



single-threaded process



multithreaded process

Per-process items	**Per-thread items**
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

Recap

- POSIX Threads
  - Thread creation

```
#include <pthread.h>

Int pthread_create(   pthread_t*       thread,
                const pthread_attr_t* attr,
                     void*            (*start_routine)(void*),
                     void*            arg);
```

  - Thread join

```
int pthread_join(pthread_t thread, void **value_ptr);
```

Recap

- Example

```
typedef struct __myarg_t {
    int a;
    int b;
} myarg_t;

typedef struct __myret_t {
    int x;
    int y;
} myret_t;

void *mythread(void *arg) {
    myarg_t *m = (myarg_t *) arg;
    printf("%d %d\n", m->a, m->b);
    myret_t *r = malloc(sizeof(myret_t));
    r->x = 1;
    r->y = 2;
    return (void *) r;
}
```

Recap

- Example (cont')

```
int main(int argc, char *argv[]) {
    int rc;
    pthread_t p;
    myret_t *m;

    myarg_t args;
    args.a = 10;
    args.b = 20;
    pthread_create(&p, NULL, mythread, &args);
    pthread_join(p, (void **) &m);
    printf("returned %d %d\n", m->x, m->y);
    free(m);
    return 0;
}
```

Recap

- Be careful with how values are returned from a thread

```
 void *mythread(void *arg) {
     myarg_t *m = (myarg_t *) arg;
     printf("%d %d\n", m->a, m->b);
     myret_t r; // ALLOCATED ON STACK!
     r.x = 1;
     r.y = 2;
     return (void *) &r;
}
```

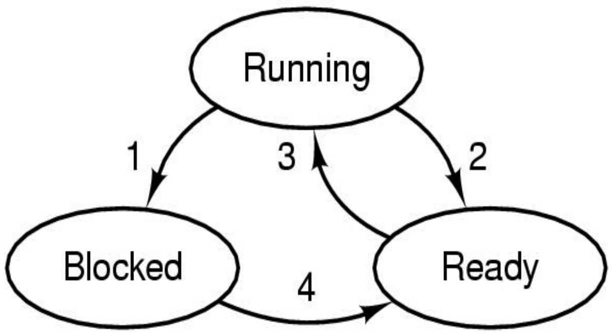
- When the function returns, `r` is automatically de-allocated (i.e., the stack frame is destroyed)
  - pointer to `r` will point to invalid data

Agenda

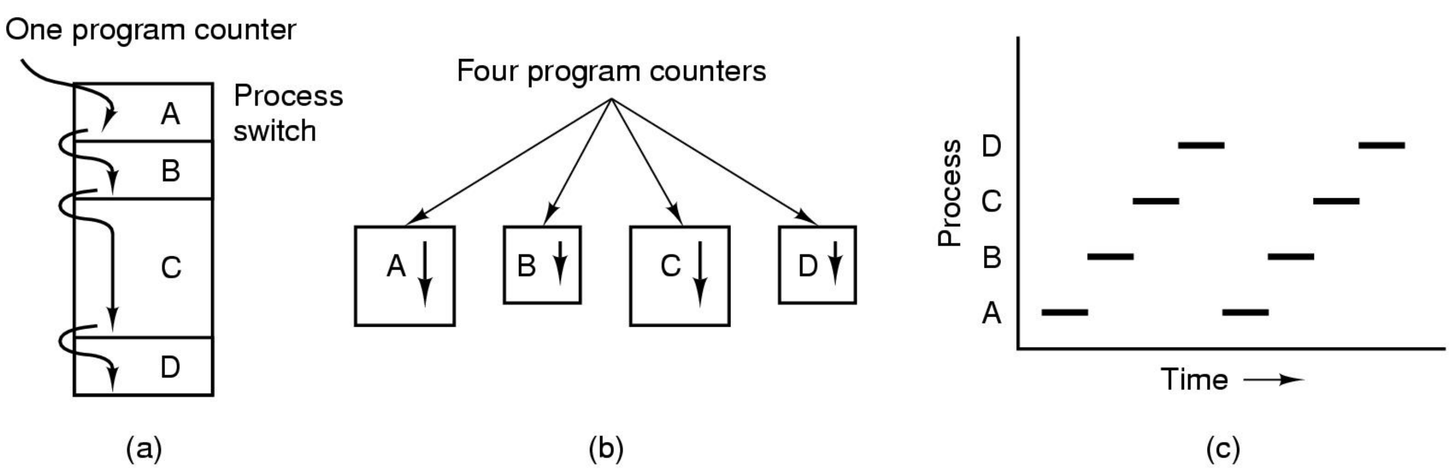
- Recap
- CPU Scheduling I
  - Scheduling Concept
  - Scheduling Algorithms

Scheduling Concept

- Process state & multiprogramming (revisit)



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

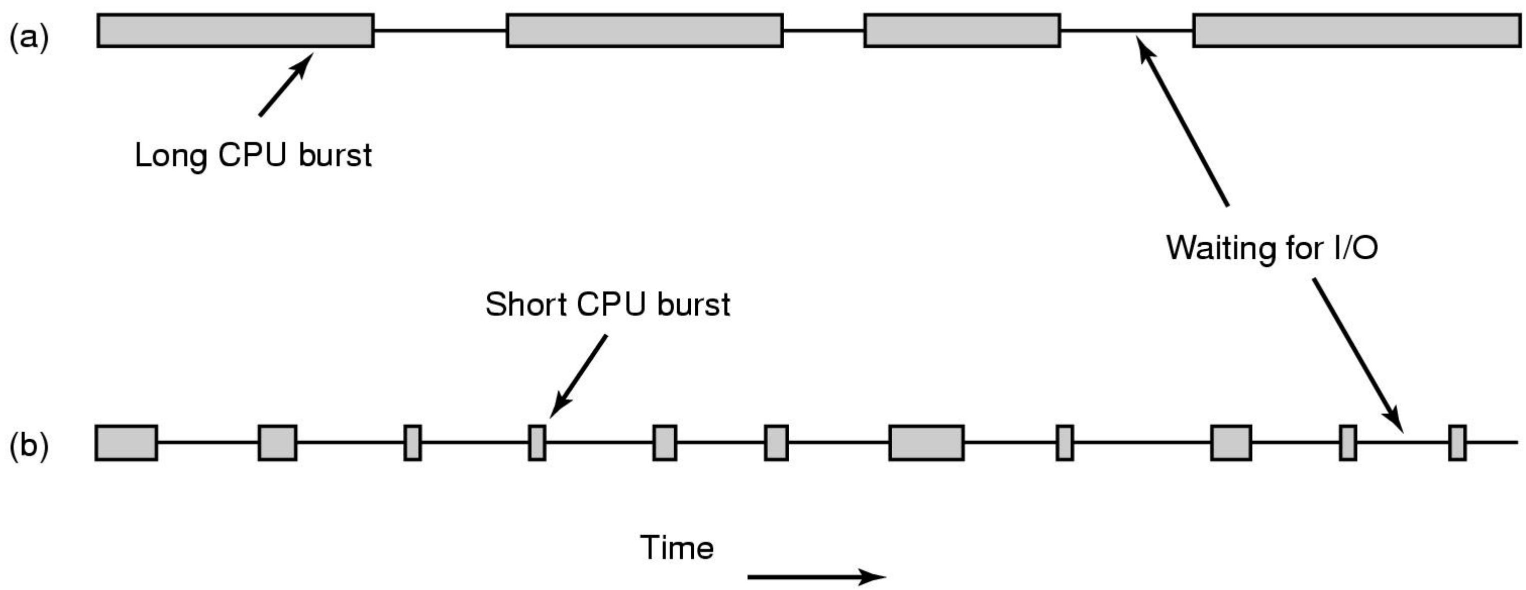


Scheduling Concept

- Process scheduling
  - When to run a different process?
    - New process is created
    - Process terminates
    - A process blocks on I/O etc.
    - interrupt occurs
  - Which one to run?
    - Depends on scheduling algorithms

Scheduling Concept

- Process behavior
  - CPU-bound (compute-bound)
  - I/O-bound



Scheduling Concept

- Non-preemptive V.S. preemptive scheduling
  - Non-preemptive
    - Processes run until they are blocked (for I/O) or voluntarily releases the CPU (e.g., terminates)
  - Preemptive
    - Can forcibly suspend a process and switch to another

Scheduling Concept

- Goals of scheduling algorithms
  - General goals
    - Fairness
      - giving each process a fair share of the CPU
    - Policy enforcement
      - ensuring that desired policy is carried out
    - Balance
      - keeping all parts of the system busy
  - Different systems may have different specific goals

Scheduling Concept

- Goals of scheduling algorithms
  - Specific goals for batch systems
    - Throughput
      - maximize jobs per hour
    - Turnaround time
      - minimize time between submission and termination
    - CPU utilization
      - keep the CPU busy all the time

Scheduling Concept

- Goals of scheduling algorithms
  - Specific goals for batch systems
    - Throughput
      - maximize jobs per hour
    - Turnaround time
      - minimize time between submission and termination
    - CPU utilization
      - keep the CPU busy all the time

Scheduling Concept

- Goals of scheduling algorithms
  - Specific goals for interactive systems
    - Response time
      - respond to user requests quickly
    - Proportionality
      - meet users’ expectations

Scheduling Concept

- Goals of scheduling algorithms
  - Specific goals for real-time systems
    - Meeting deadlines
      - hard real time
        - there are absolute deadlines that must be met
      - soft real time
        - missing an occasional deadline is undesirable, but nevertheless tolerable

Agenda

- Recap
- CPU Scheduling I
  - Scheduling Concept
  - Scheduling Algorithms

Scheduling Algorithms

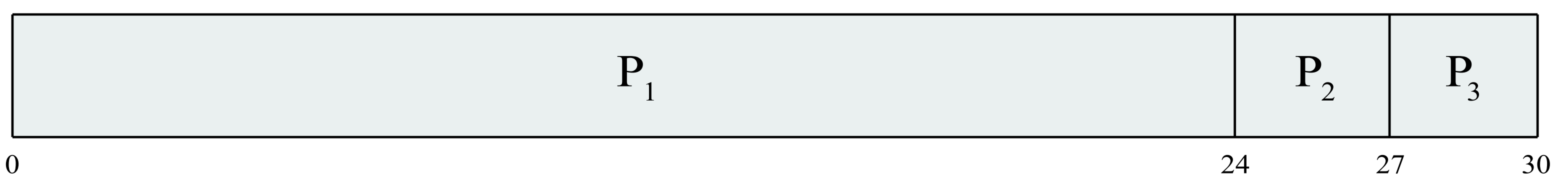
- First-Come, First-Served (FCFS)
  - Scheduling based on the arrival order of processes

Scheduling Algorithms

- First-Come, First-Served (FCFS)
  - Scheduling based on the arrival order of processes

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Assume the processes arrive in the order: P_1 , P_2 , P_3
- The **Gantt Chart** for the schedule is:



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

Scheduling Algorithms

- First-Come, First-Served (FCFS)
  - Scheduling based on the arrival order of processes

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Assume the processes arrive in the order: P_2 , P_3 , P_1
- The Gantt Chart for the schedule is:
- Waiting time:
- Average waiting time:

Scheduling Algorithms

- First-Come, First-Served (FCFS)
  - Scheduling based on the arrival order of processes

Process	Burst Time
P_1	24
P_2	3
P_3	3

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

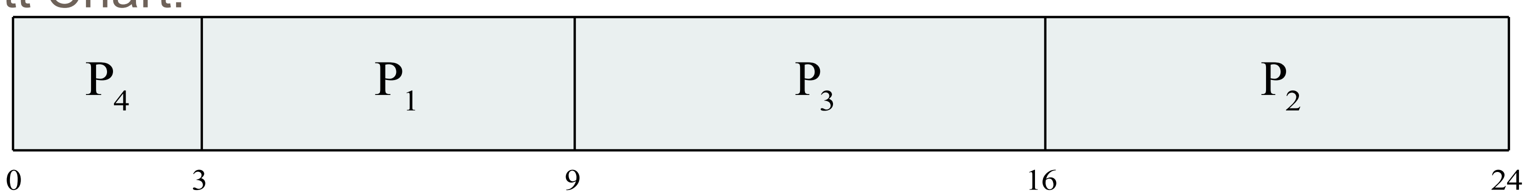
- Waiting time: P_1 = 6; P_2 = 0; P_3 = 3
- Average waiting time: (6 + 0 + 3)/3 = 3

Scheduling Algorithms

- Shortest-Job-First (SJF)
  - Scheduling based on (predicted) CPU burst time

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- Assume all processes are ready at time 0
- Gantt Chart:



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7
- Turnaround time: P1 = 9; P2 = 24; P3 = 16; P4 = 3

Agenda

- Recap
- CPU Scheduling I
    - Scheduling Concept
    - Scheduling Algorithms

Questions?

*acknowledgement: slides include content from “Modern Operating Systems” by A. Tanenbaum, “Operating Systems Concepts” by A. Silberschatz etc., “Operating Systems: Three Easy Pieces” by R. Arpaci-Dusseau etc., and anonymous pictures from internet.