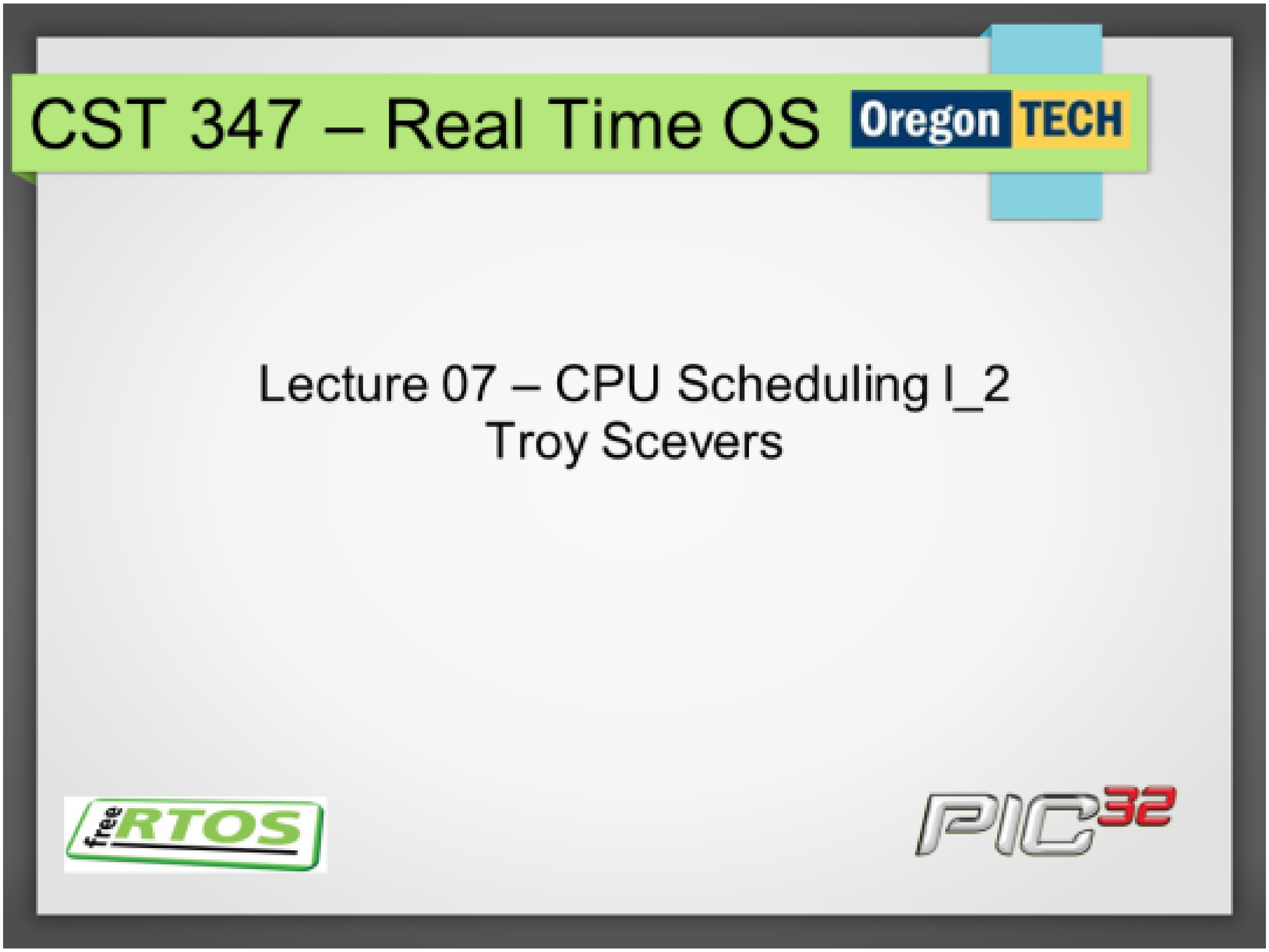
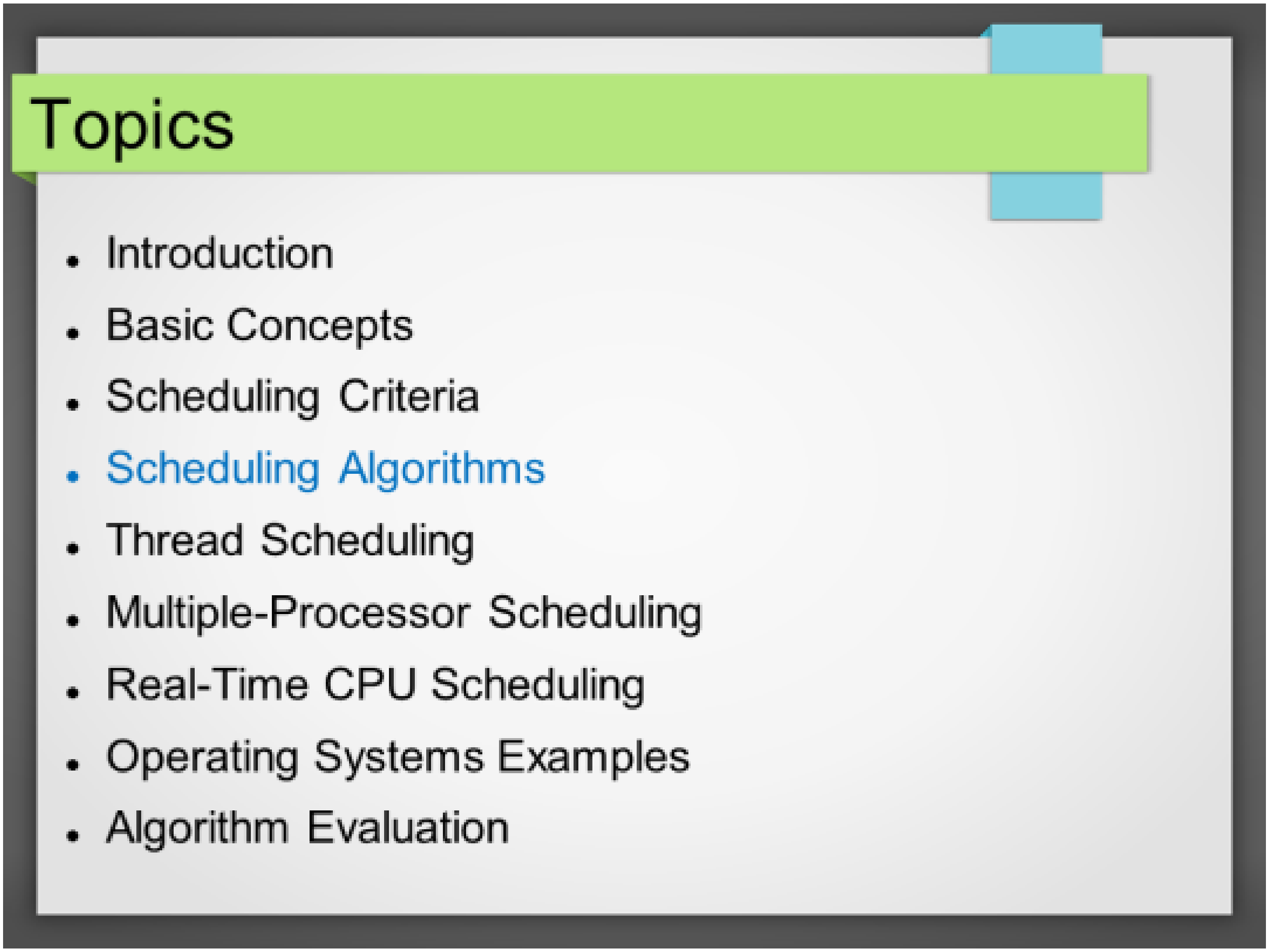
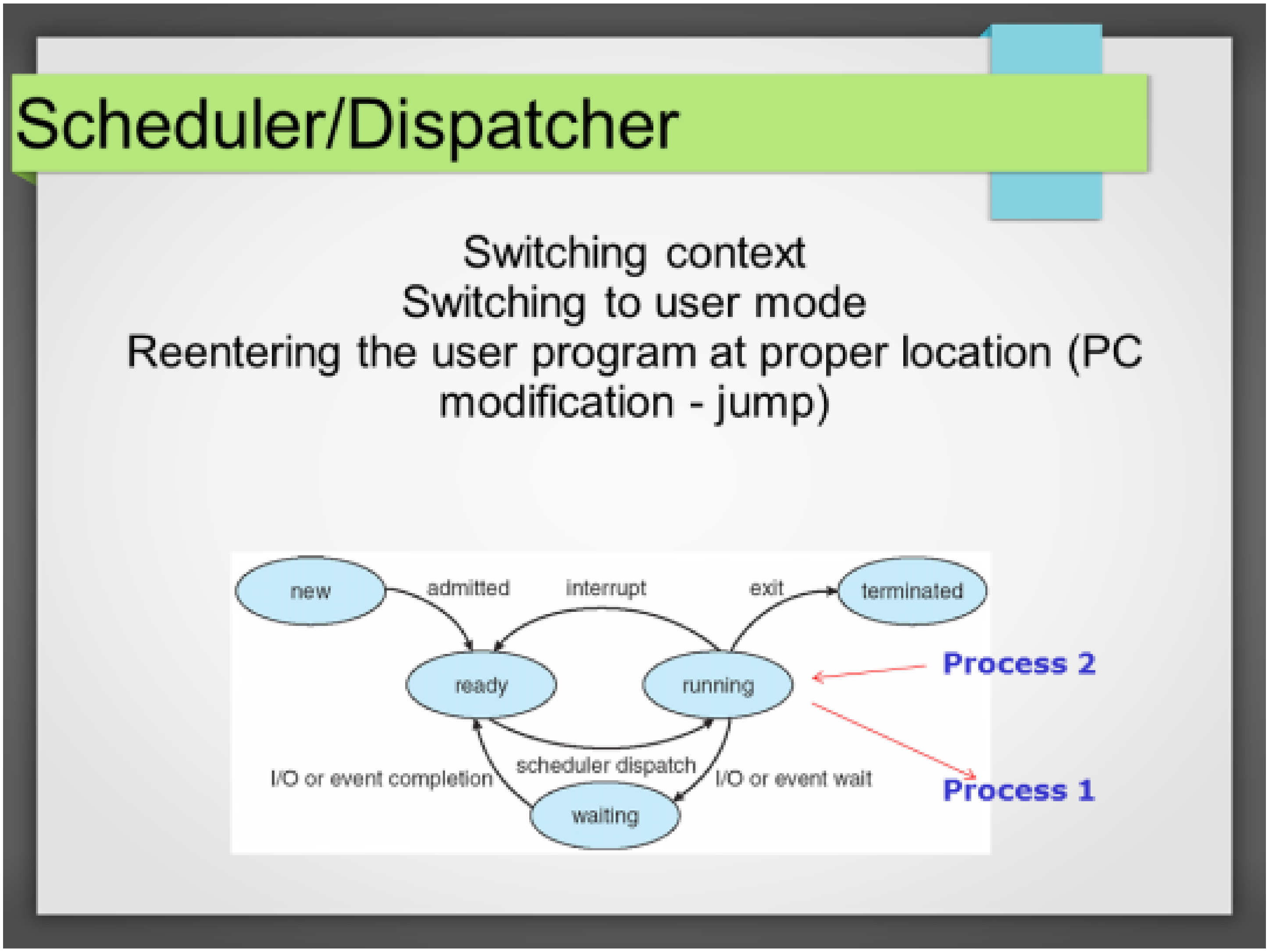
CST 347 ?? Real Time OS



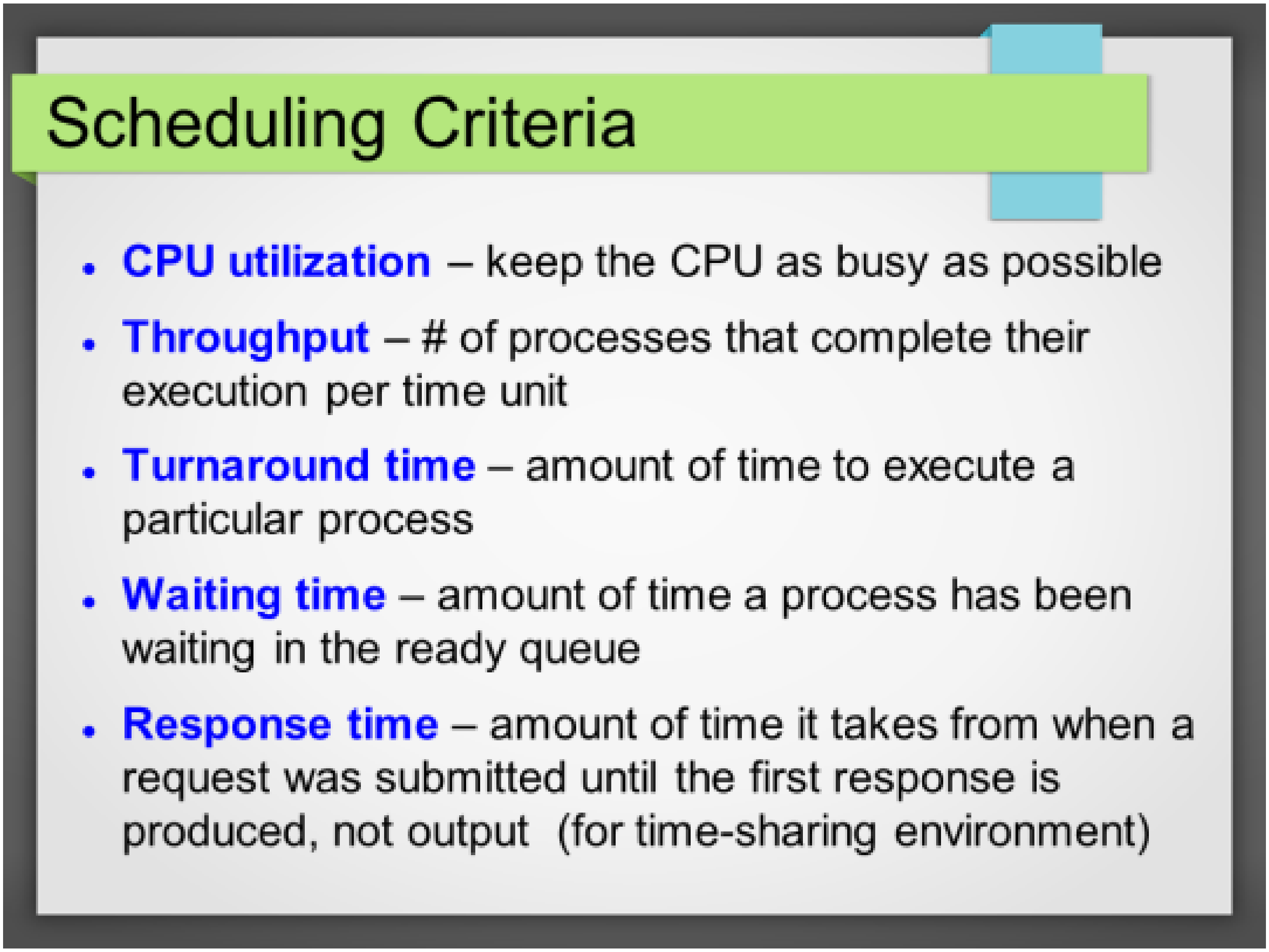
Topics


Topics
Introduction
Basic Concepts
Scheduling Criteria
Scheduling Algorithms
Thread Scheduling
Multiple-Processor Scheduling
Real-Time CPU Scheduling
Operating Systems Examples
Algorithm Evaluation

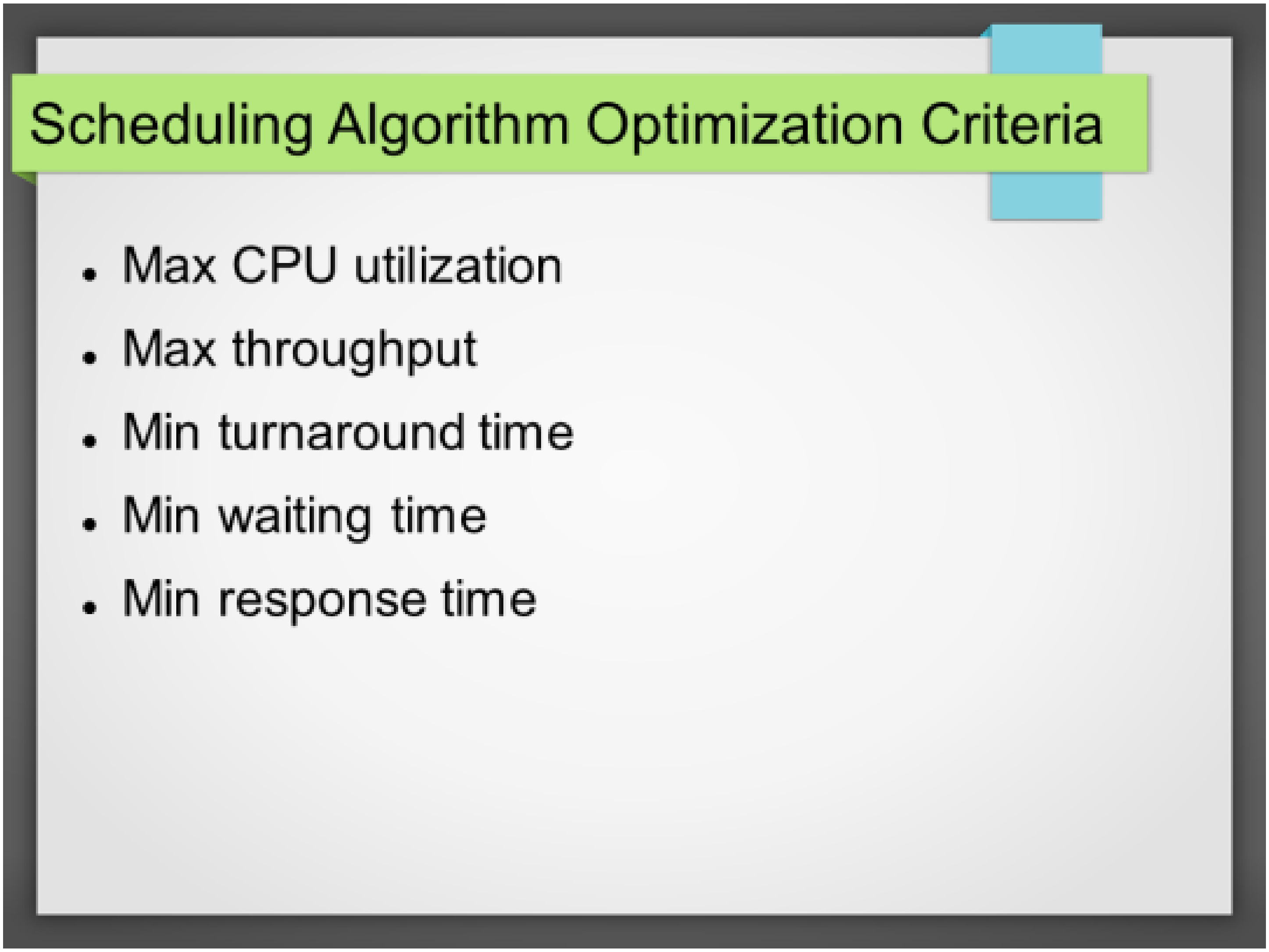
Scheduler/Dispatcher



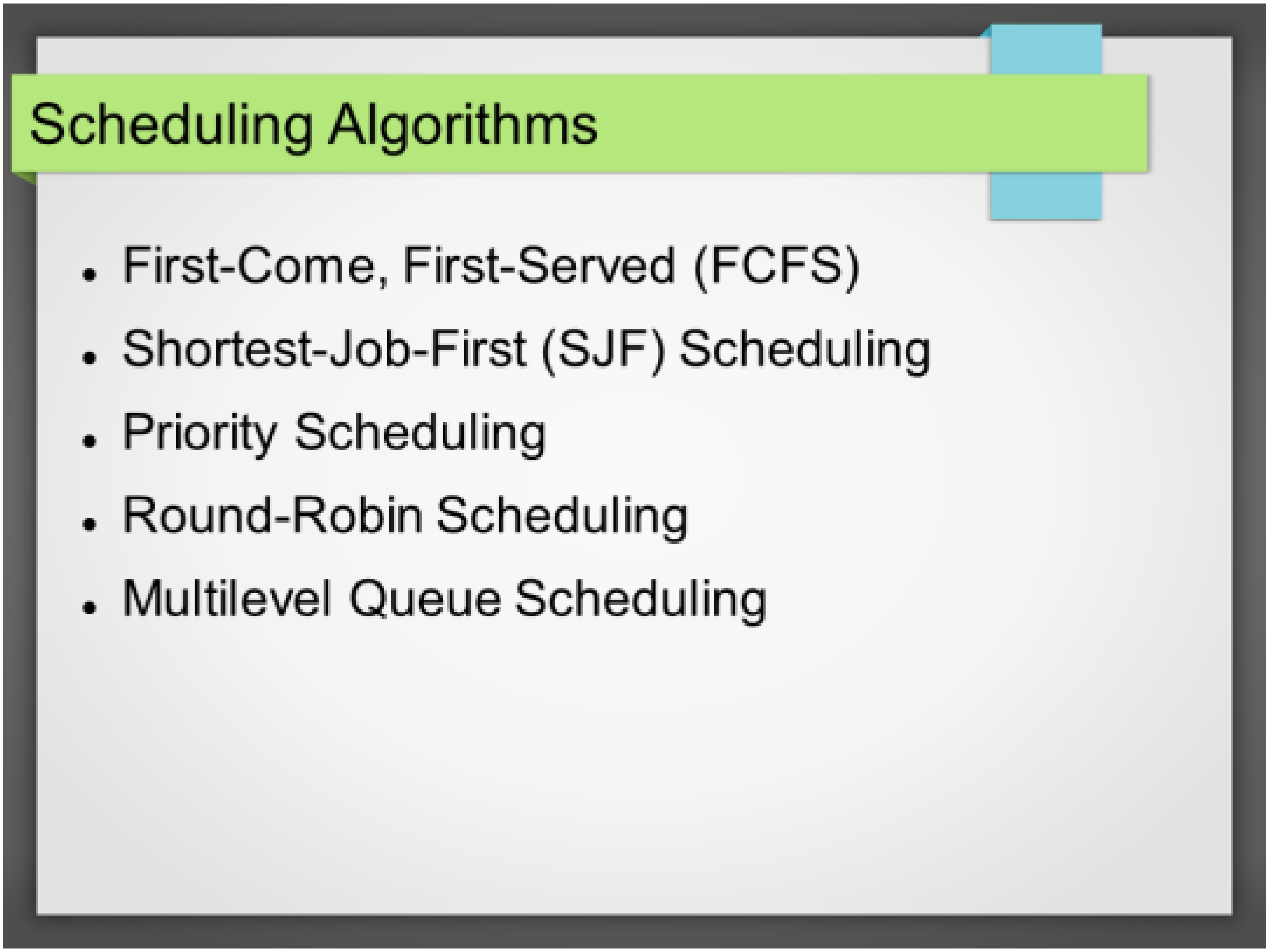
Scheduling Criteria



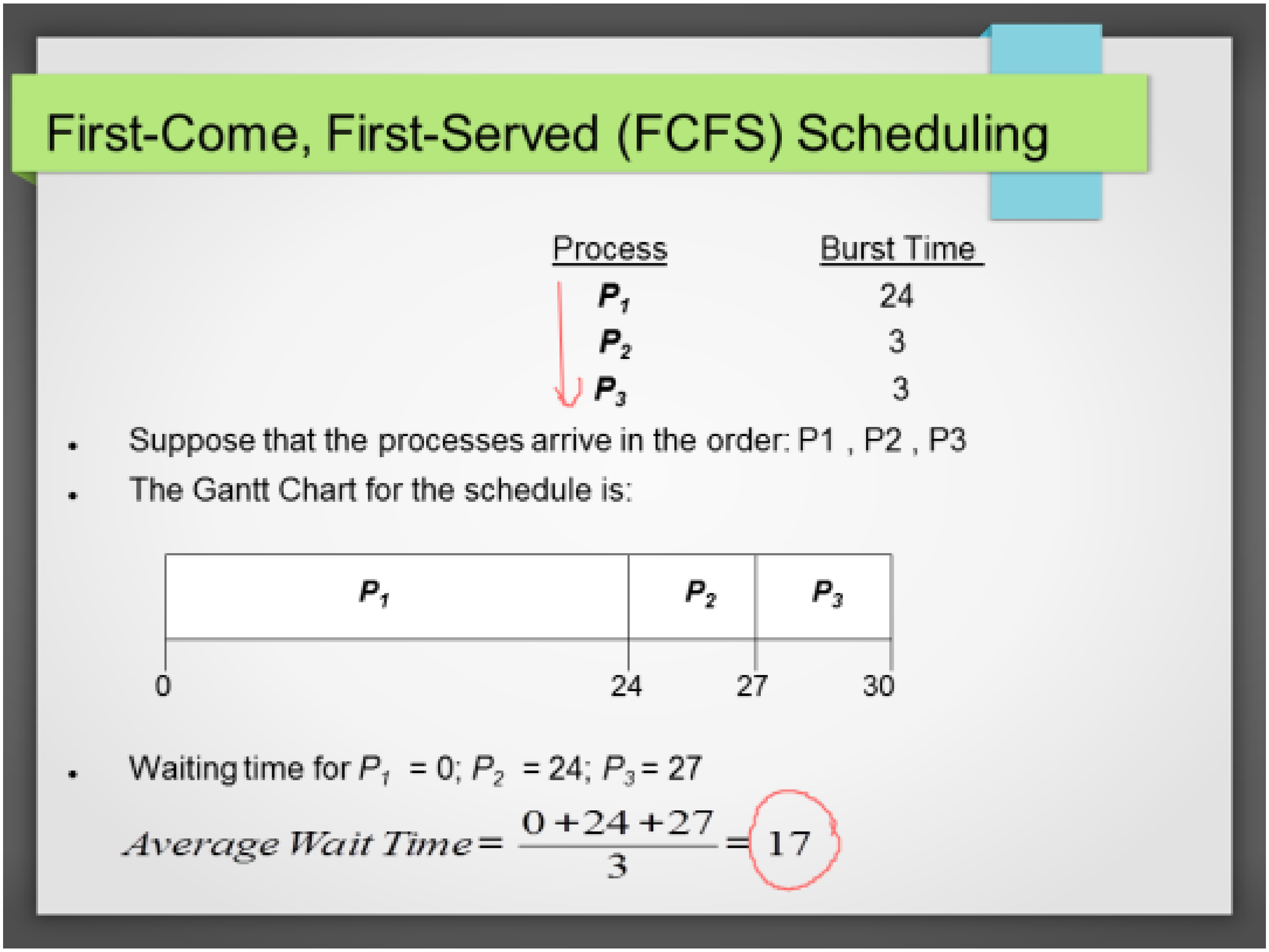
Scheduling Algorithm Optimization Criteria



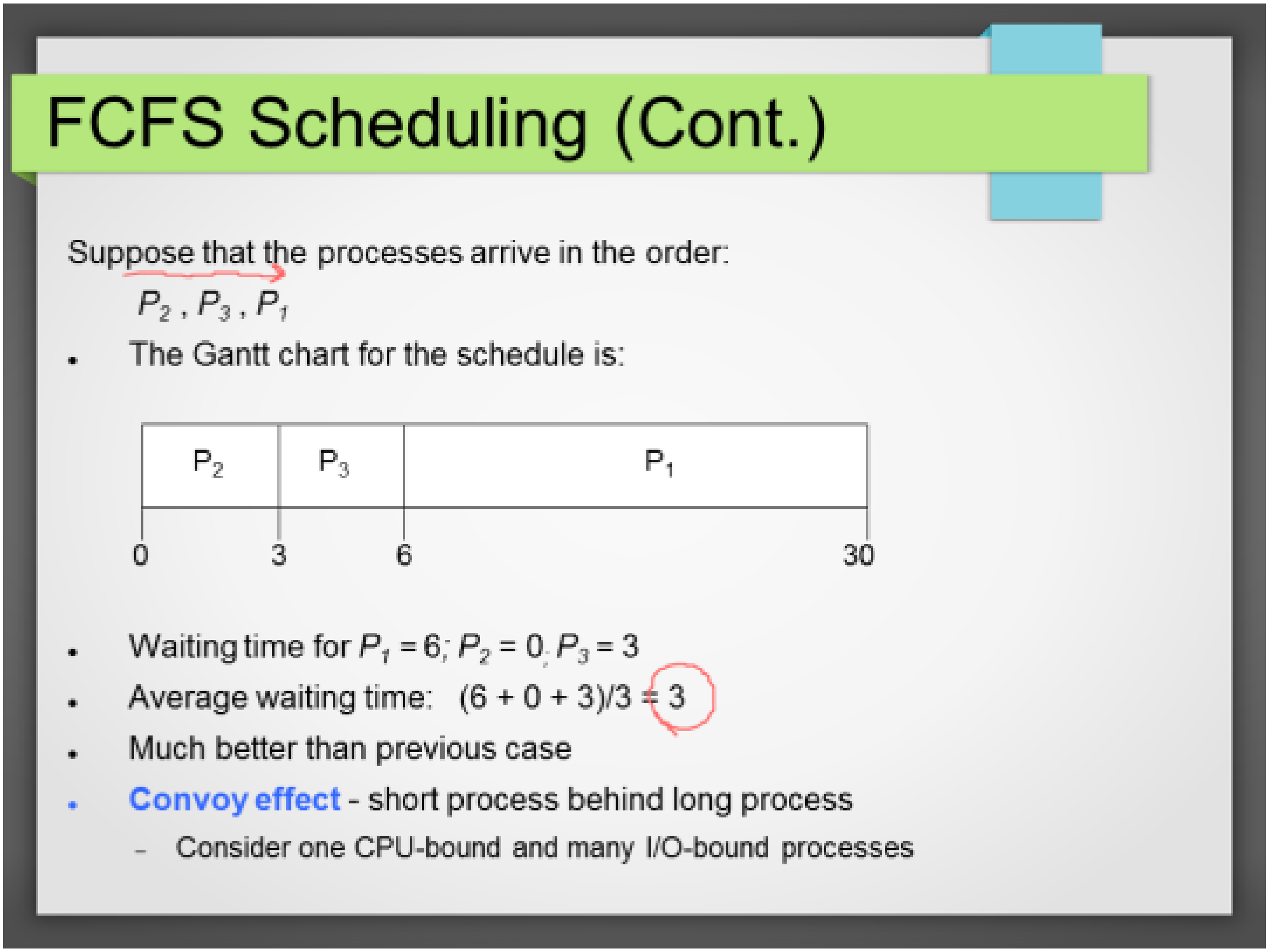
Scheduling Algorithms



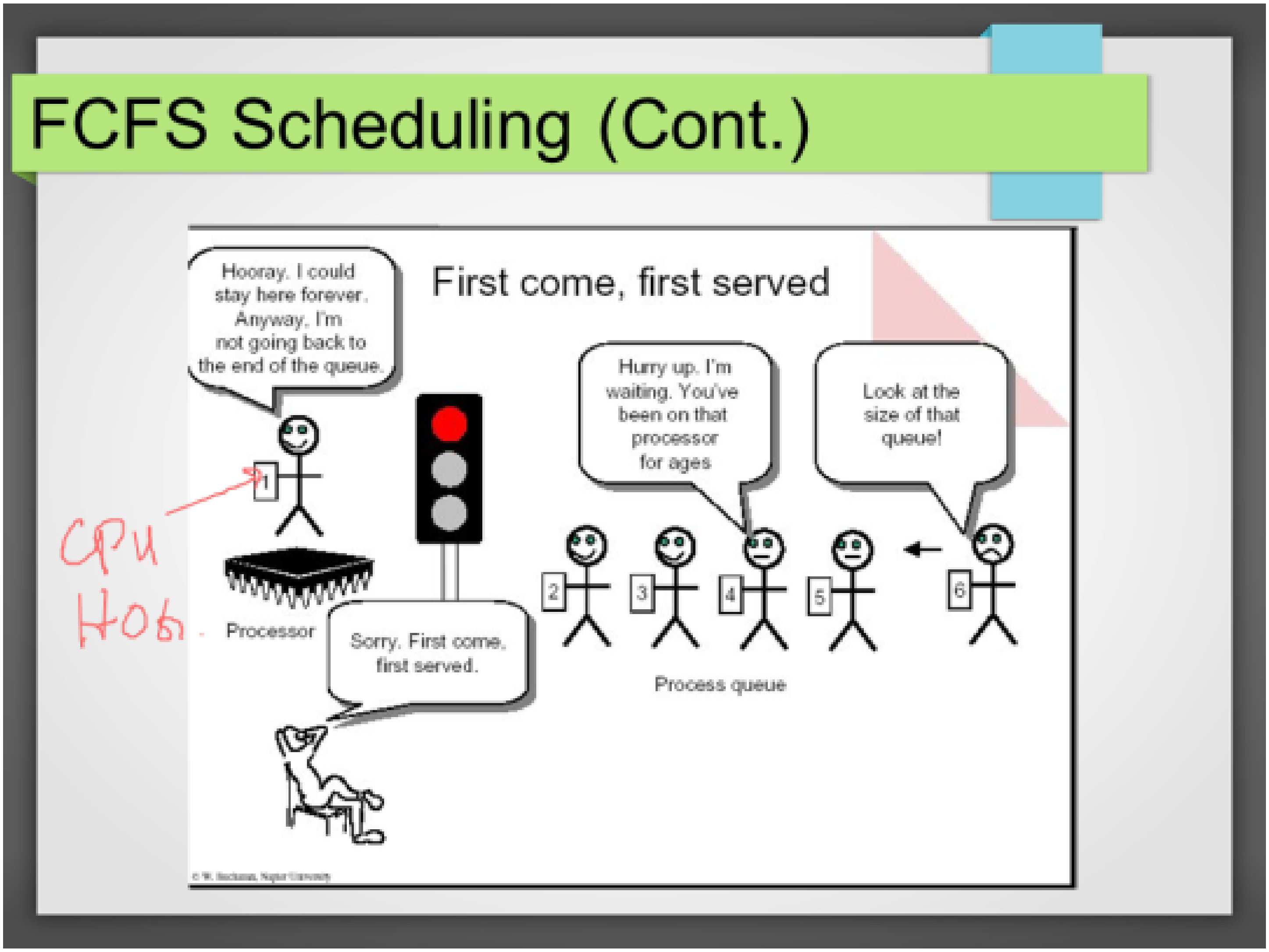
First-Come, First-Served (FCFS) Scheduling



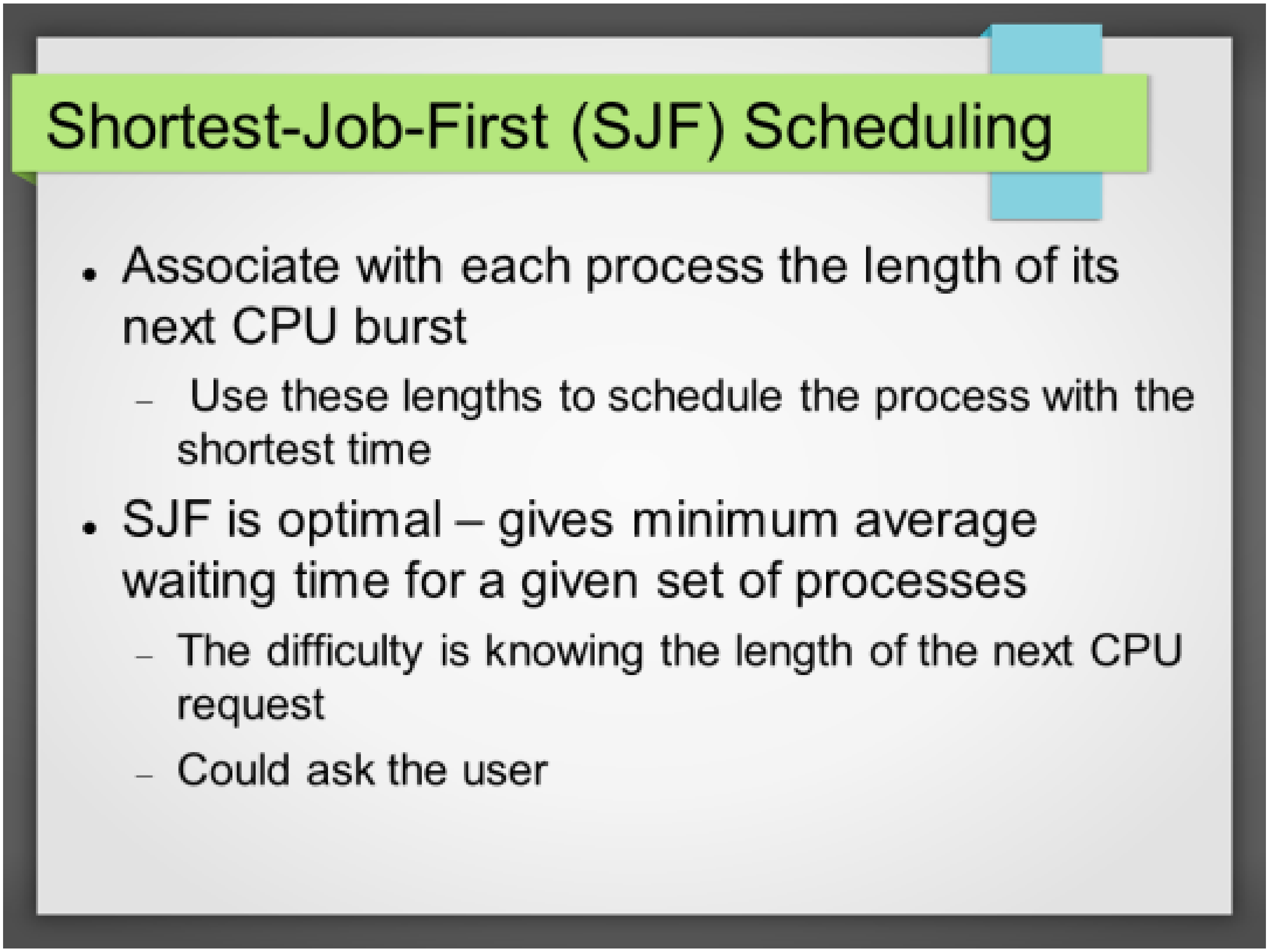
FCFS Scheduling (Cont.)



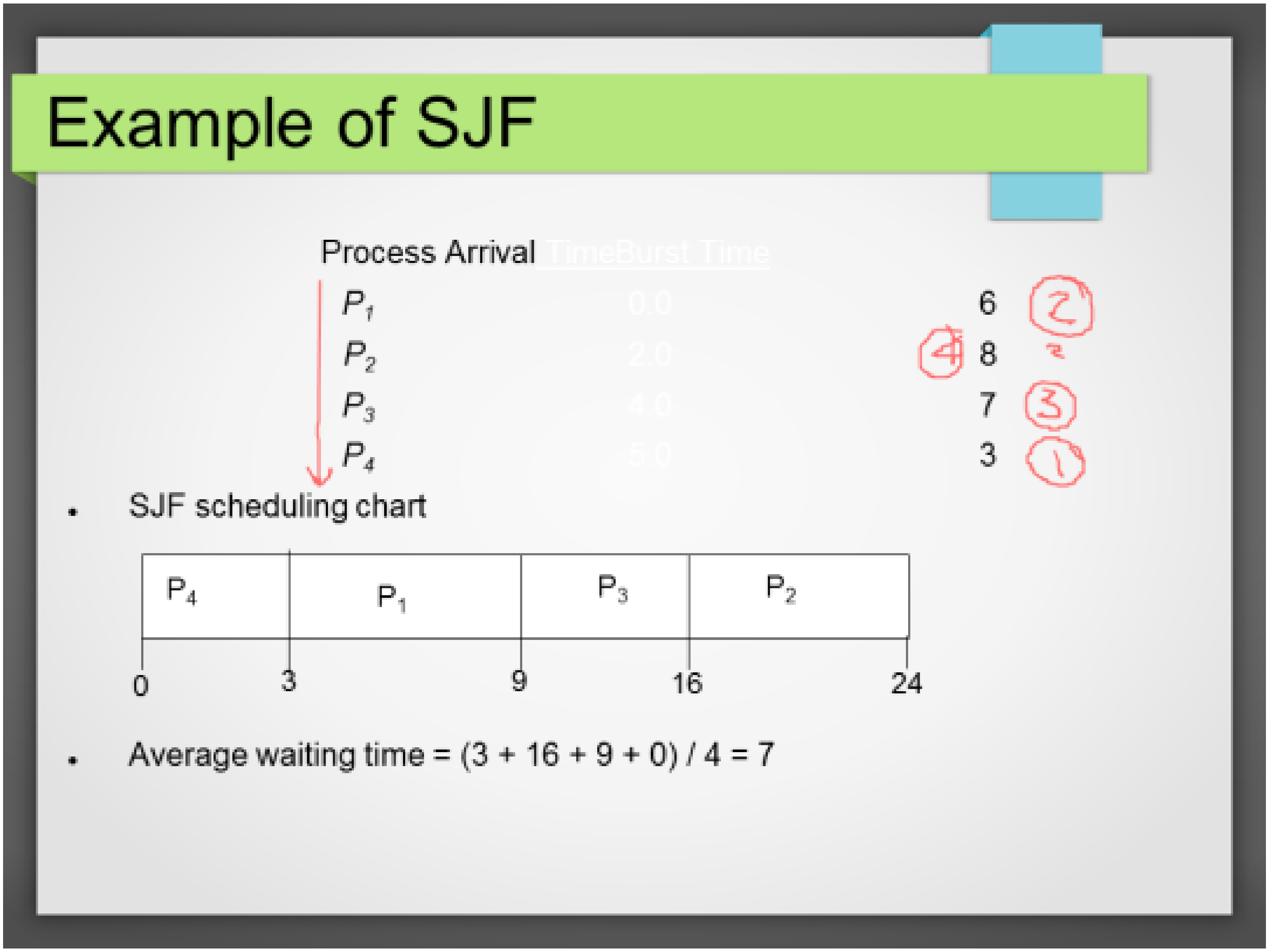
FCFS Scheduling (Cont.)



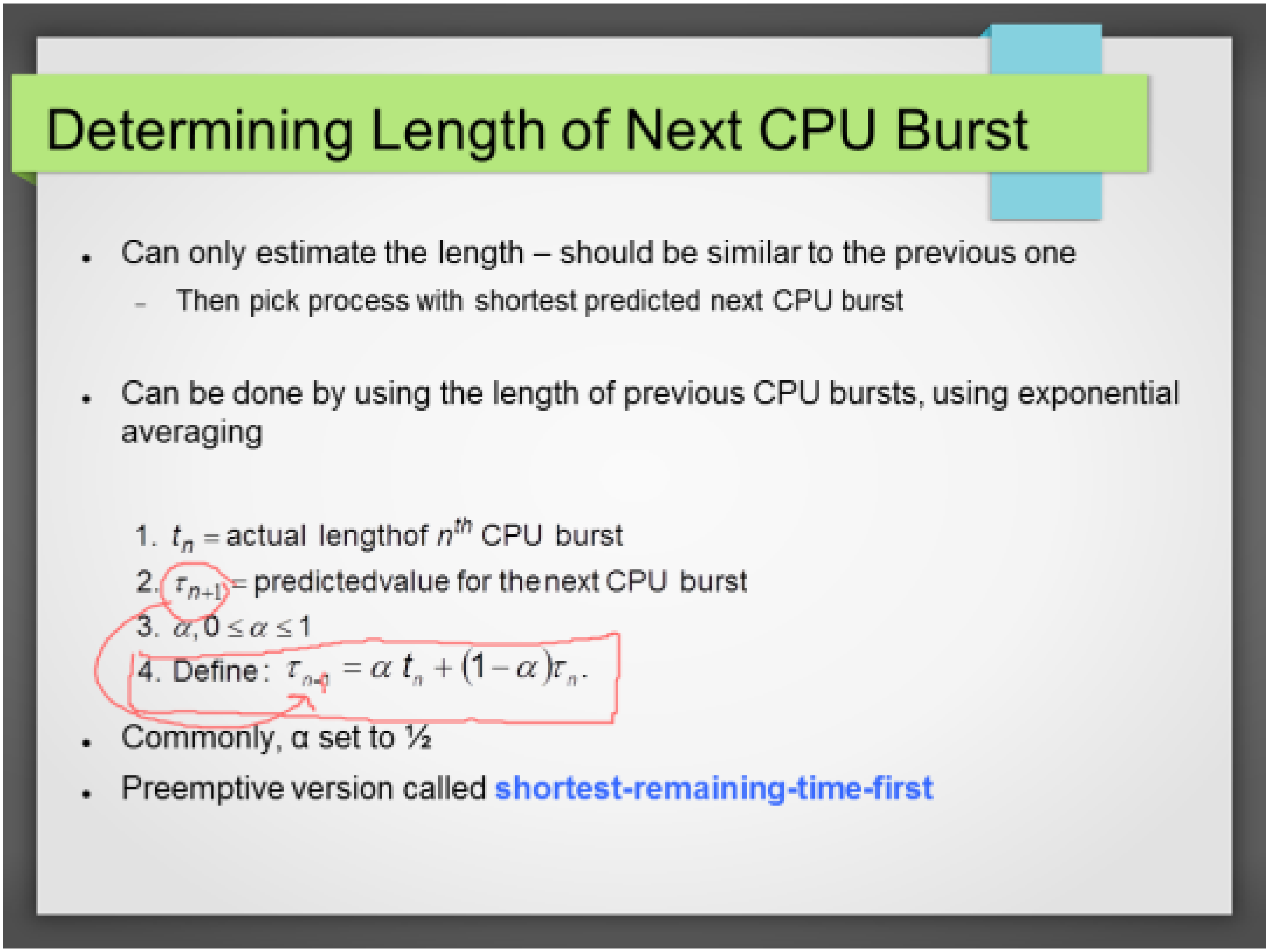
Shortest-Job-First (SJF) Scheduling



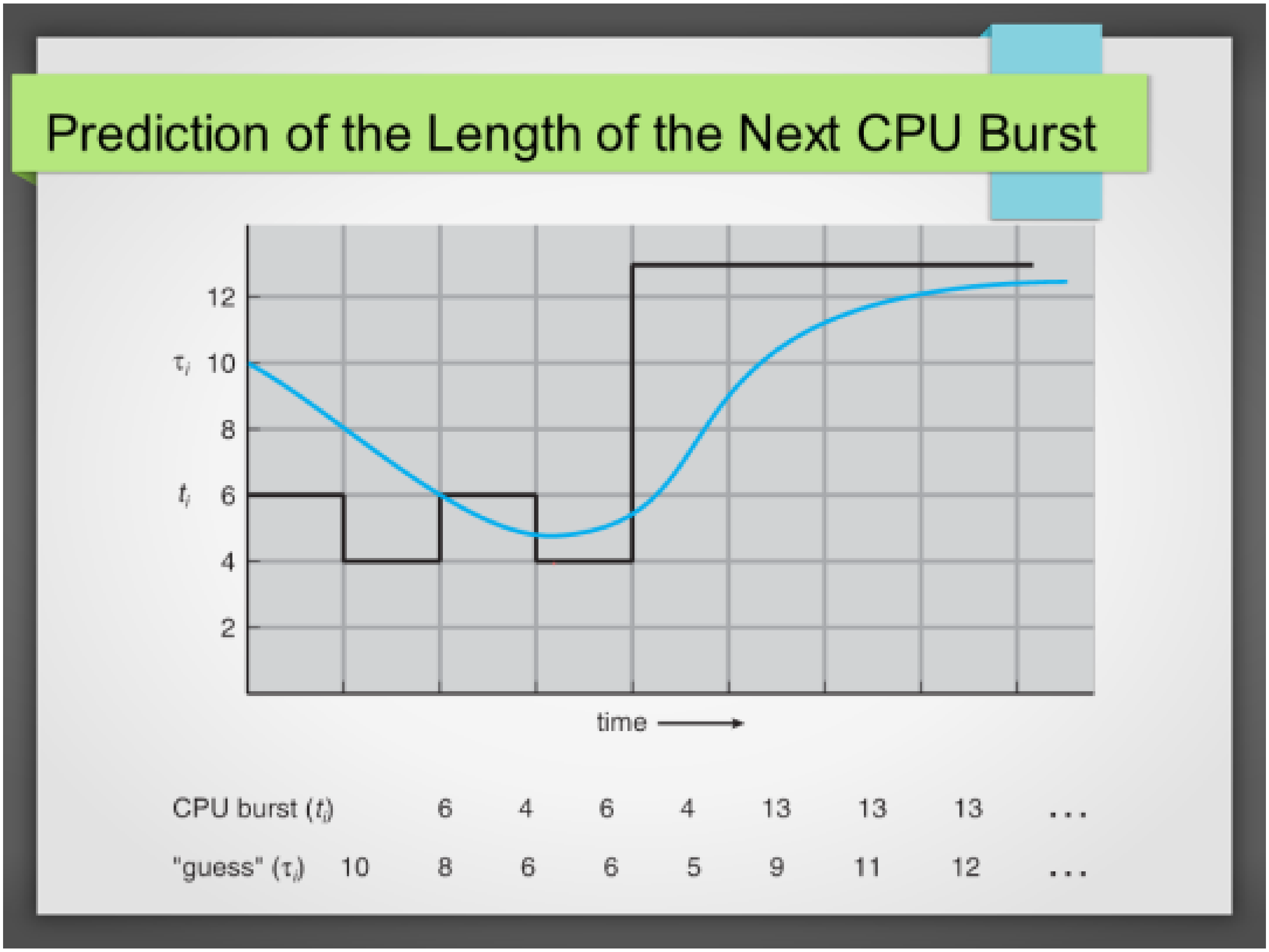
Example of SJF



Determining Length of Next CPU Burst



Prediction of the Length of the Next CPU Burst



Examples of Exponential Averaging

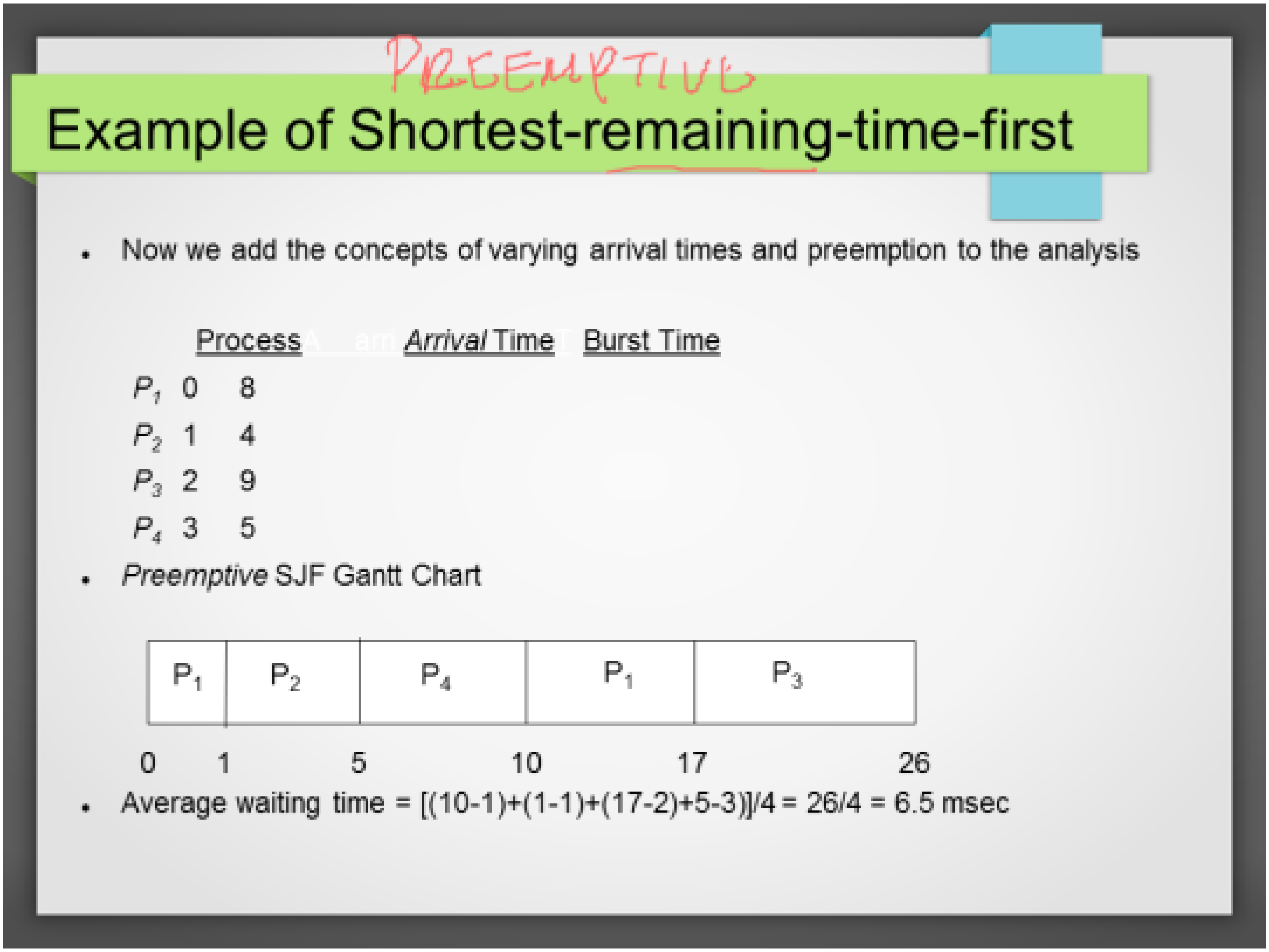
Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - 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:

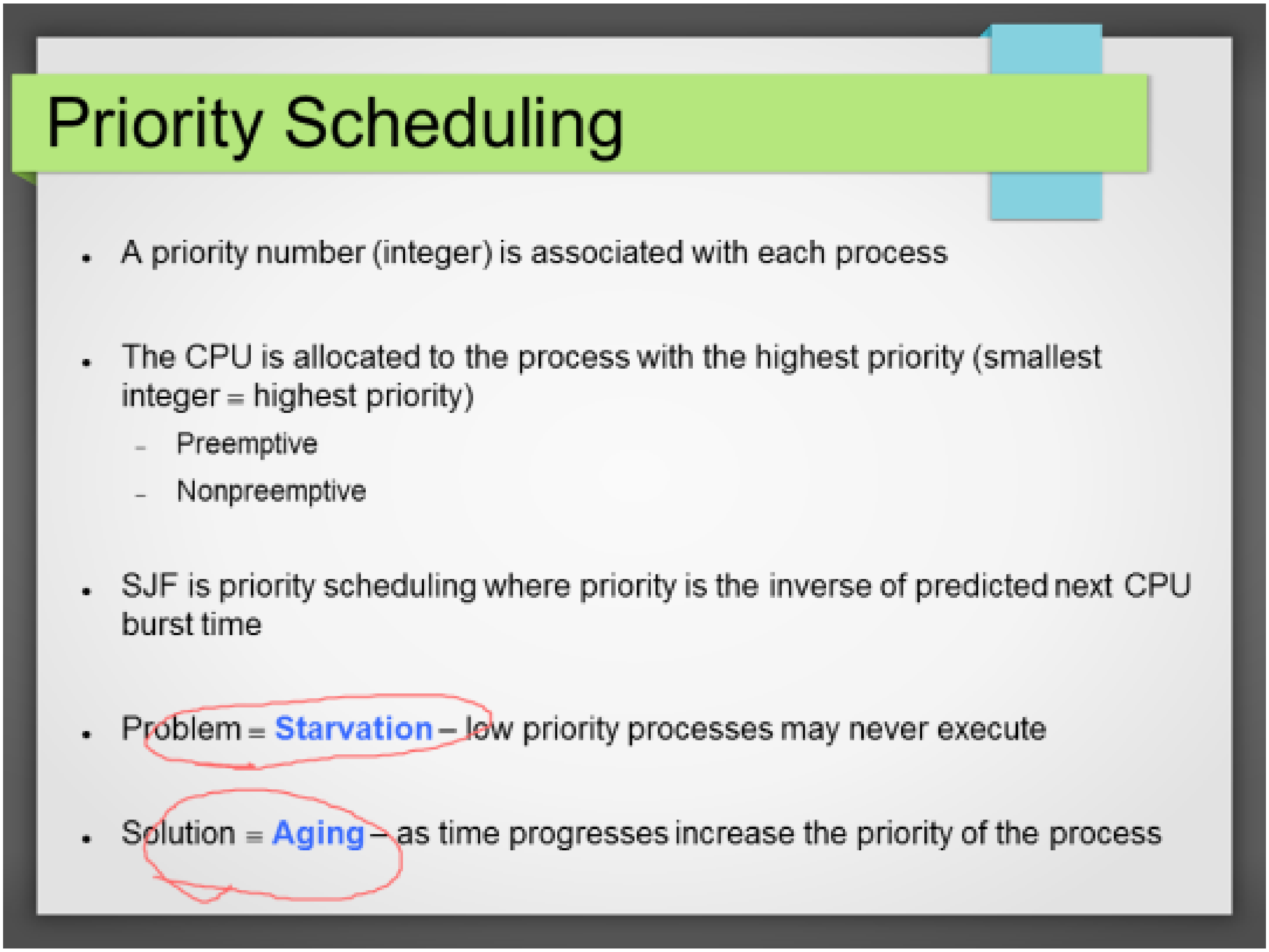
$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_n - 1 + \ldots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

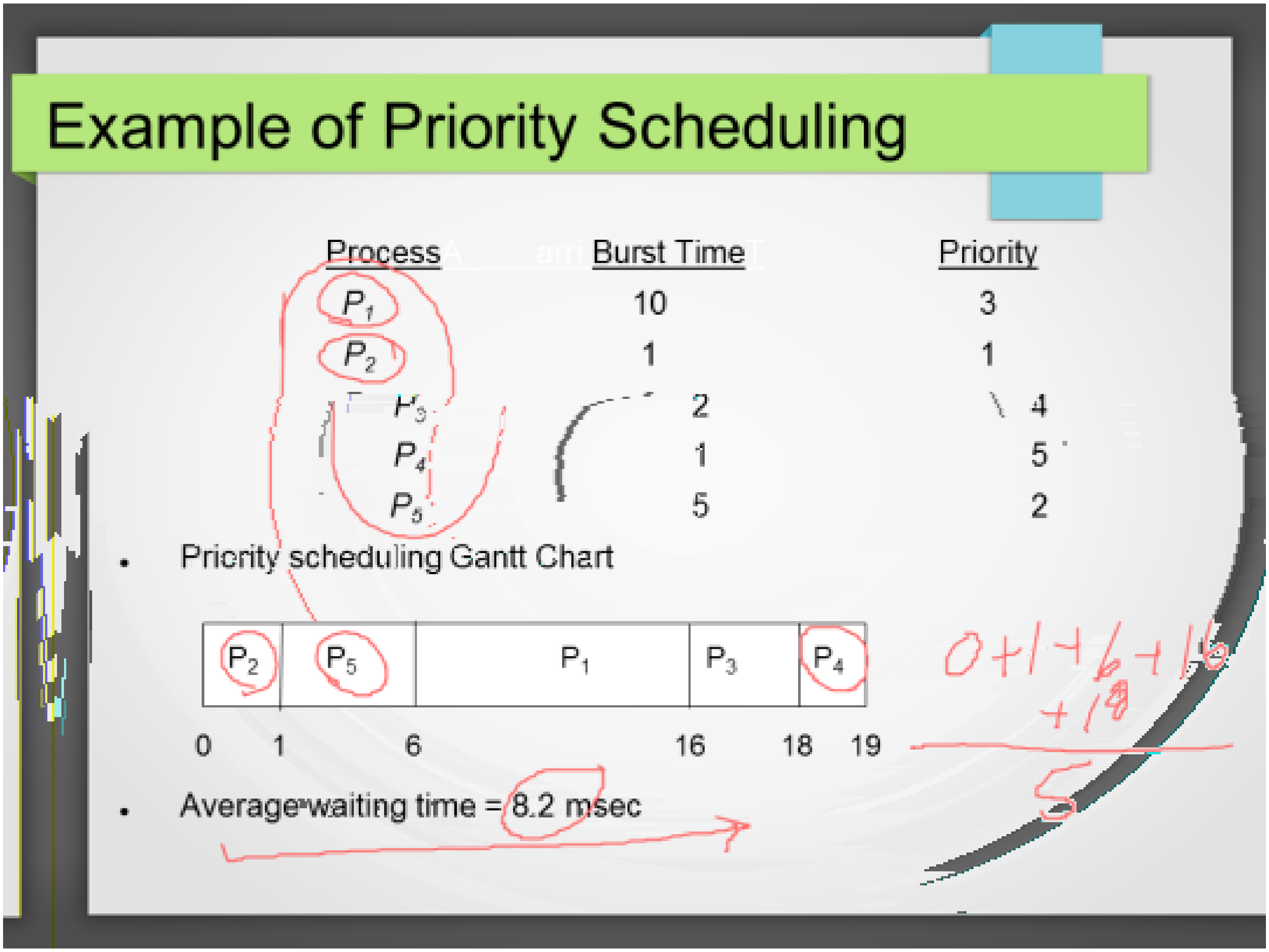
Example of Shortest-remaining-time-first



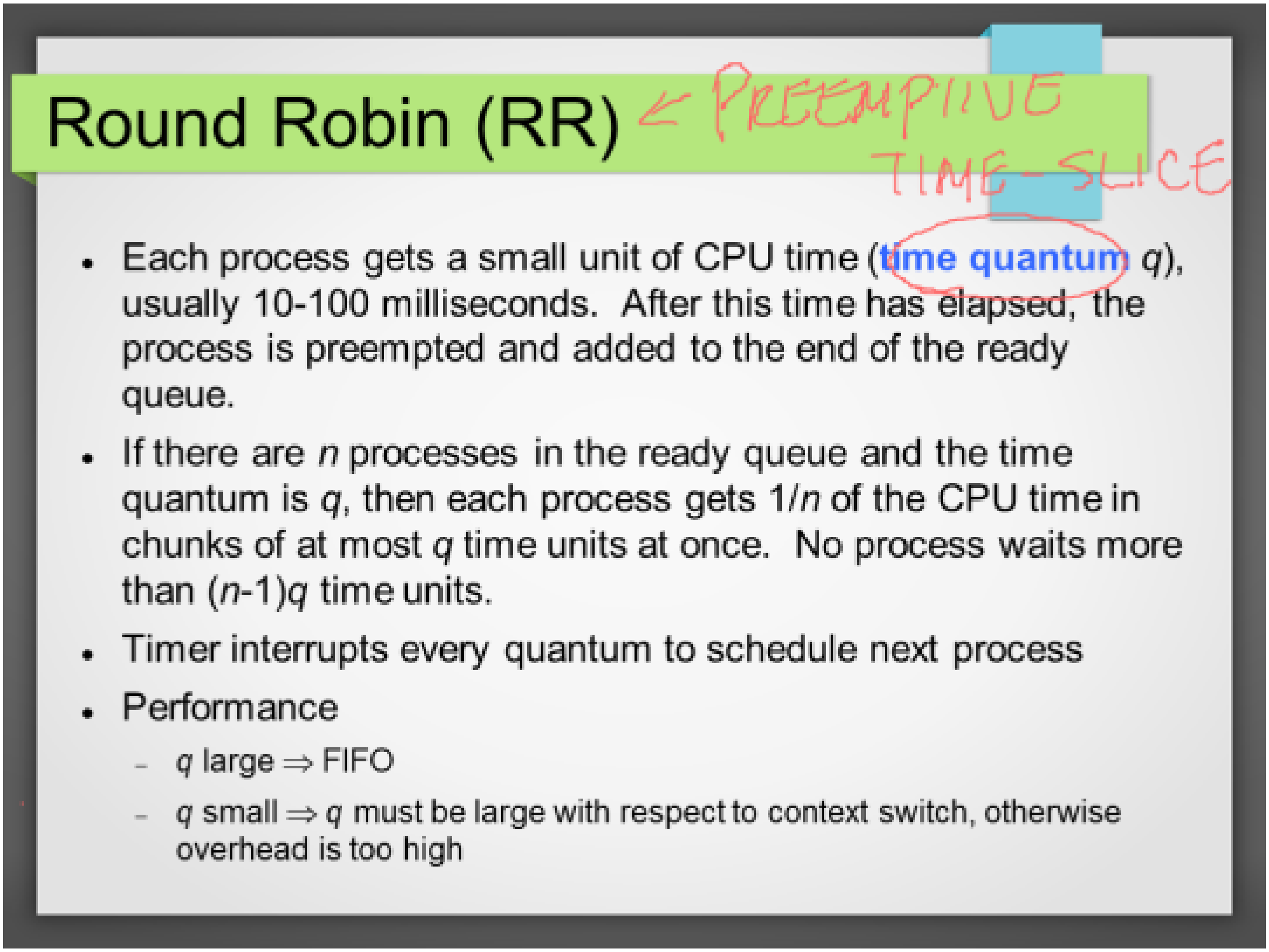
Priority Scheduling



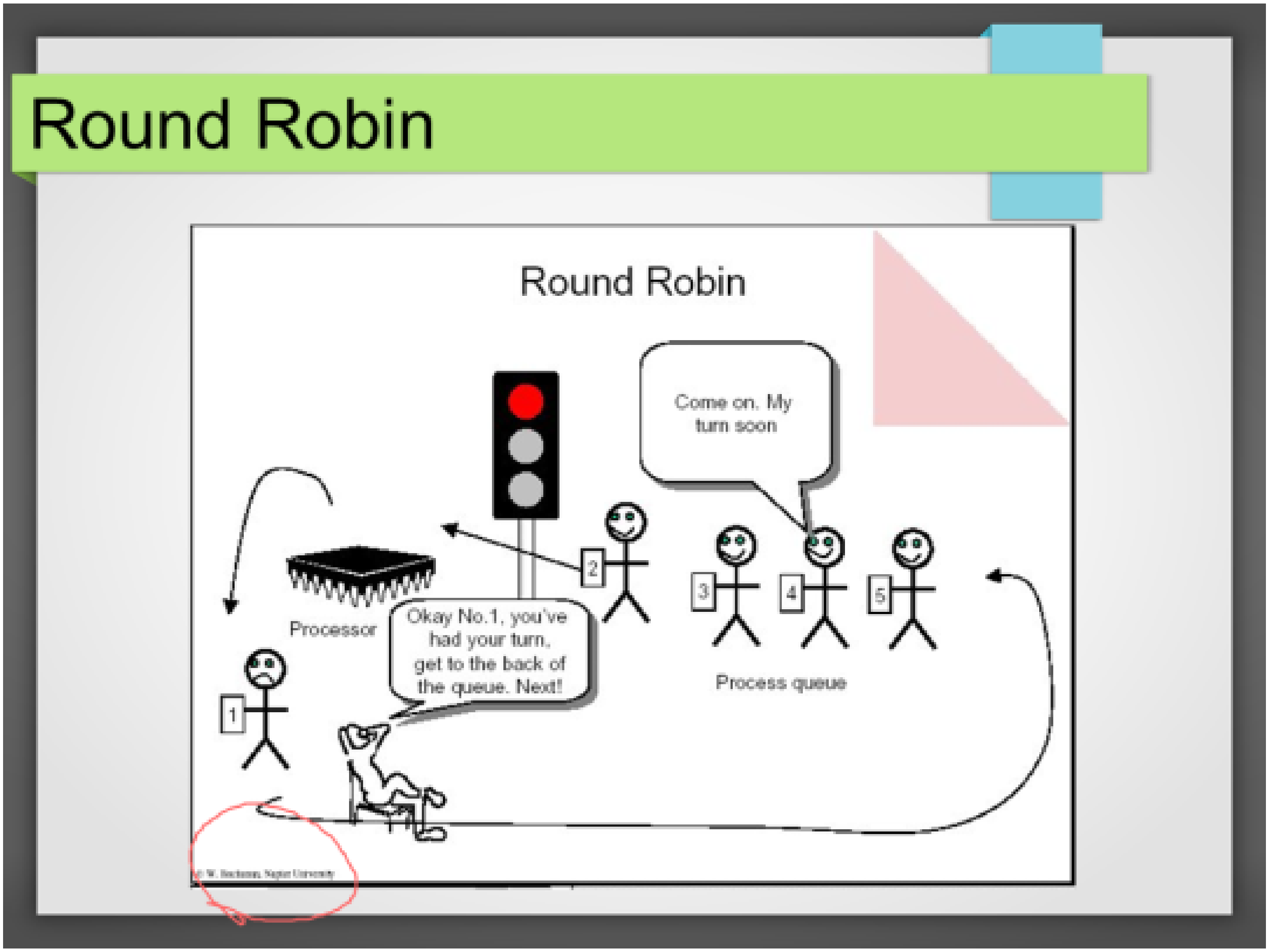
Example of Priority Scheduling



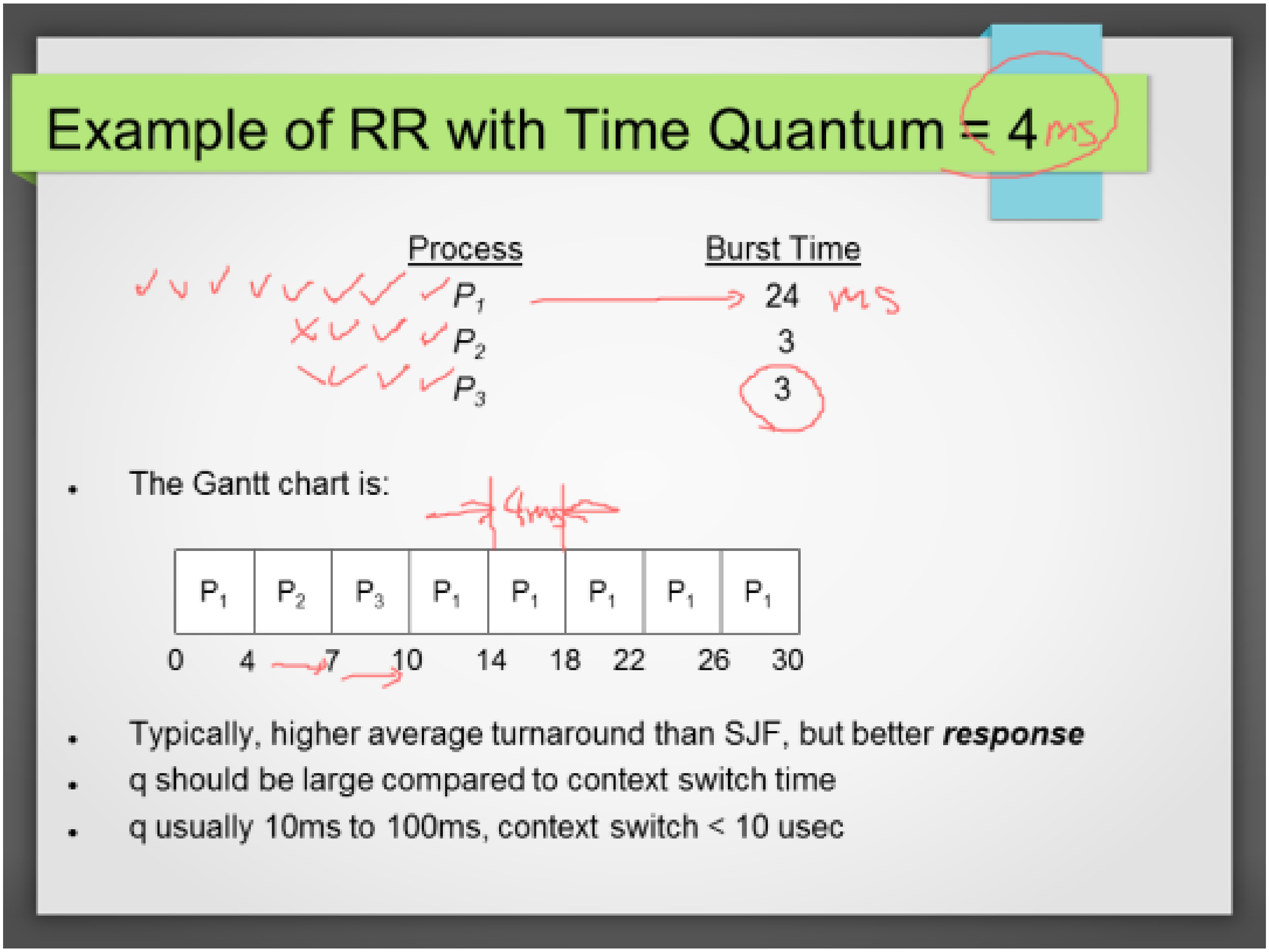
Round Robin (RR)



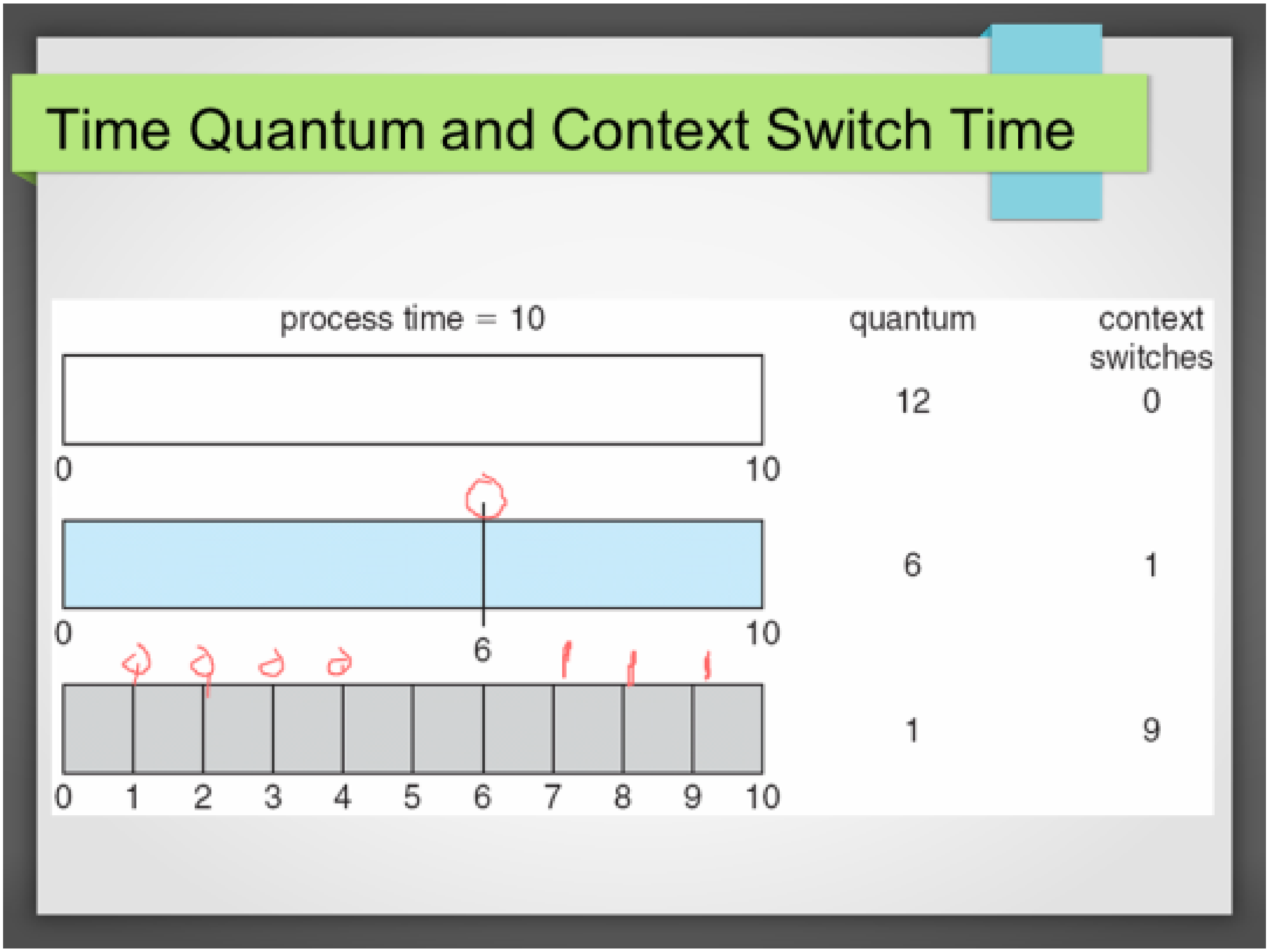
Round Robin


Round Robin
Round Robin
Come on. My turn soon
Processor
Okay No.1, you've had your turn, get to the back of the queue. Next!
1
2
3
4
5
Process queue

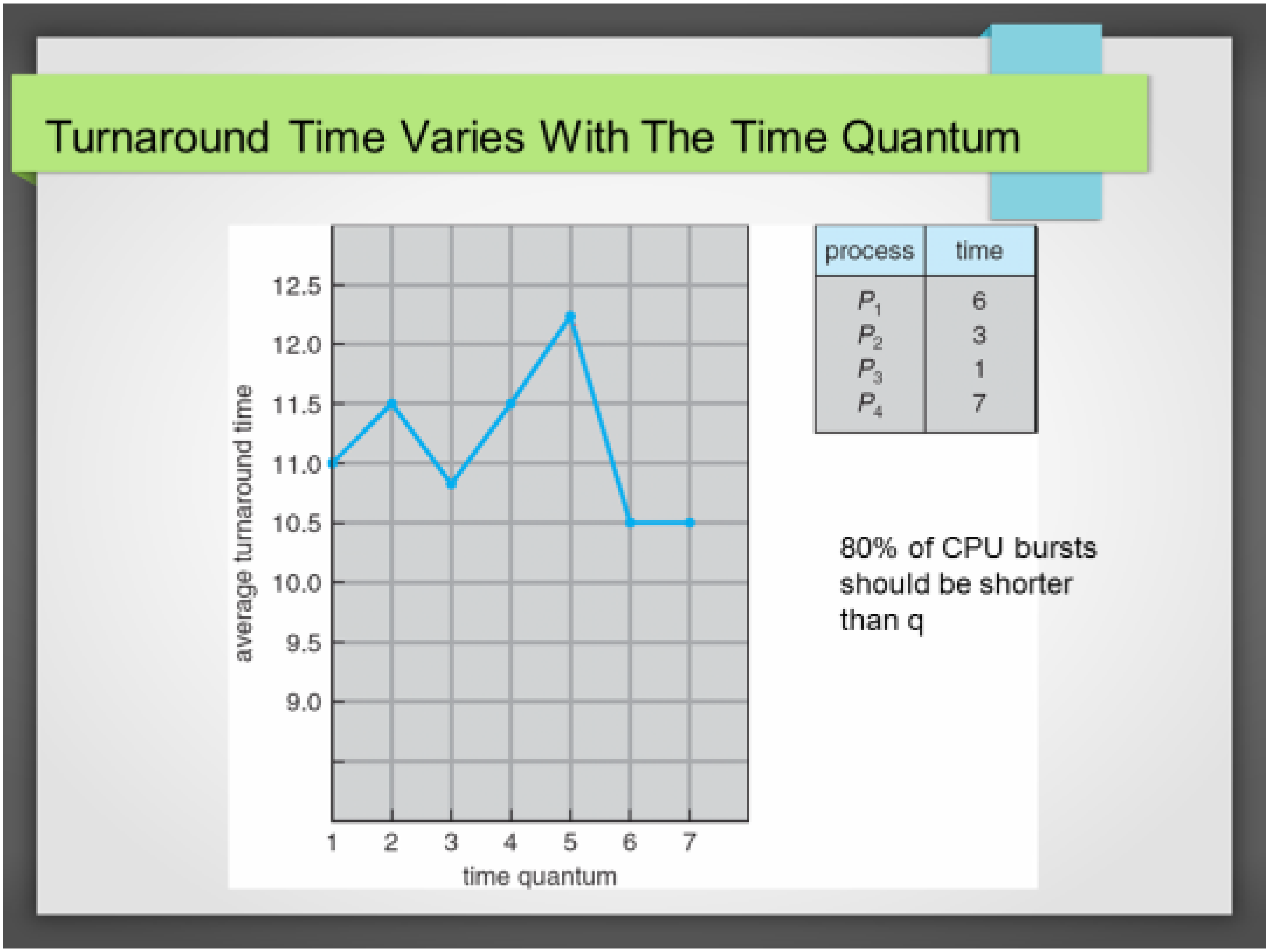
Example of RR with Time Quantum = 4



Time Quantum and Context Switch Time

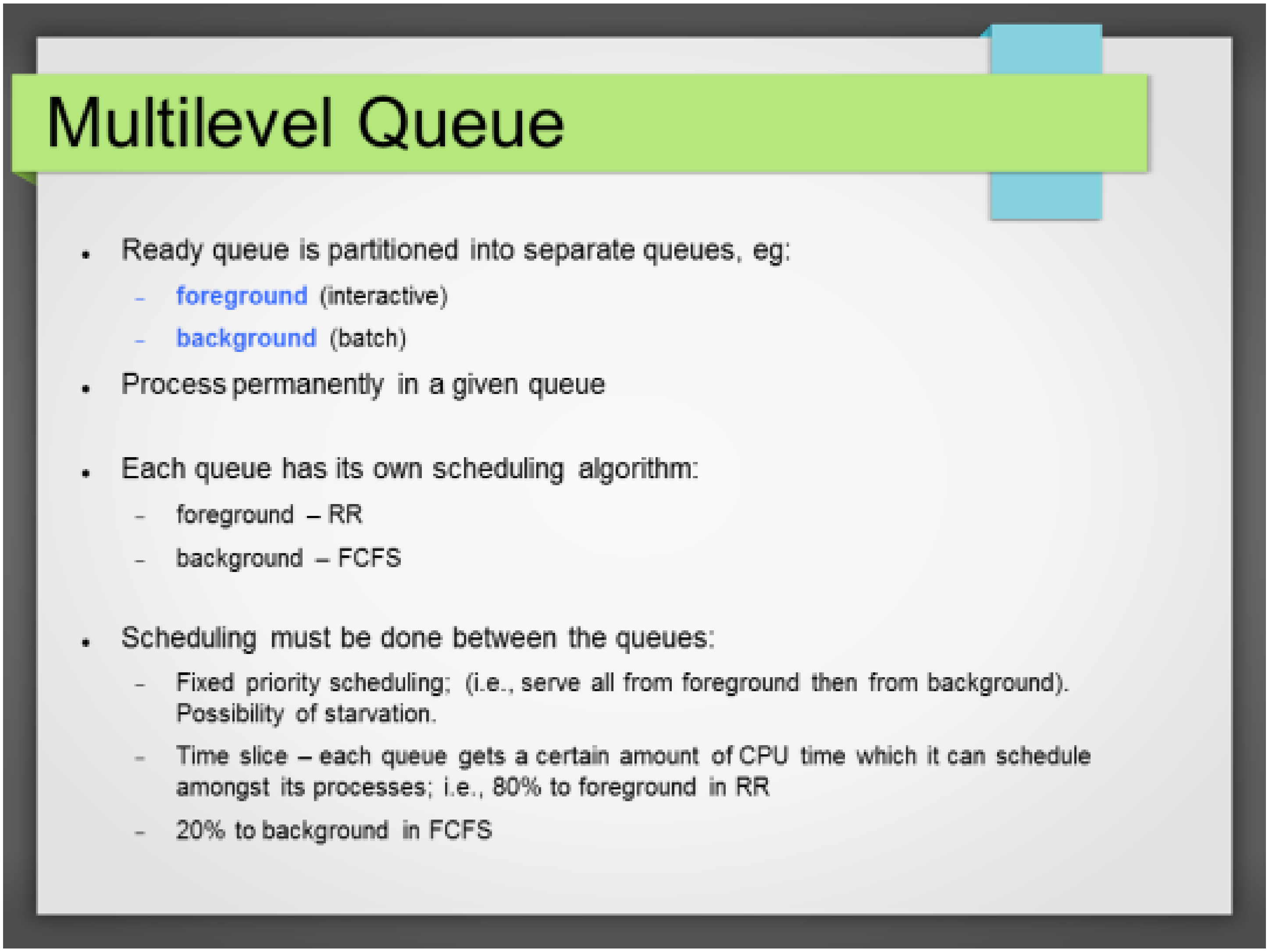


Turnaround Time Varies With The Time Quantum

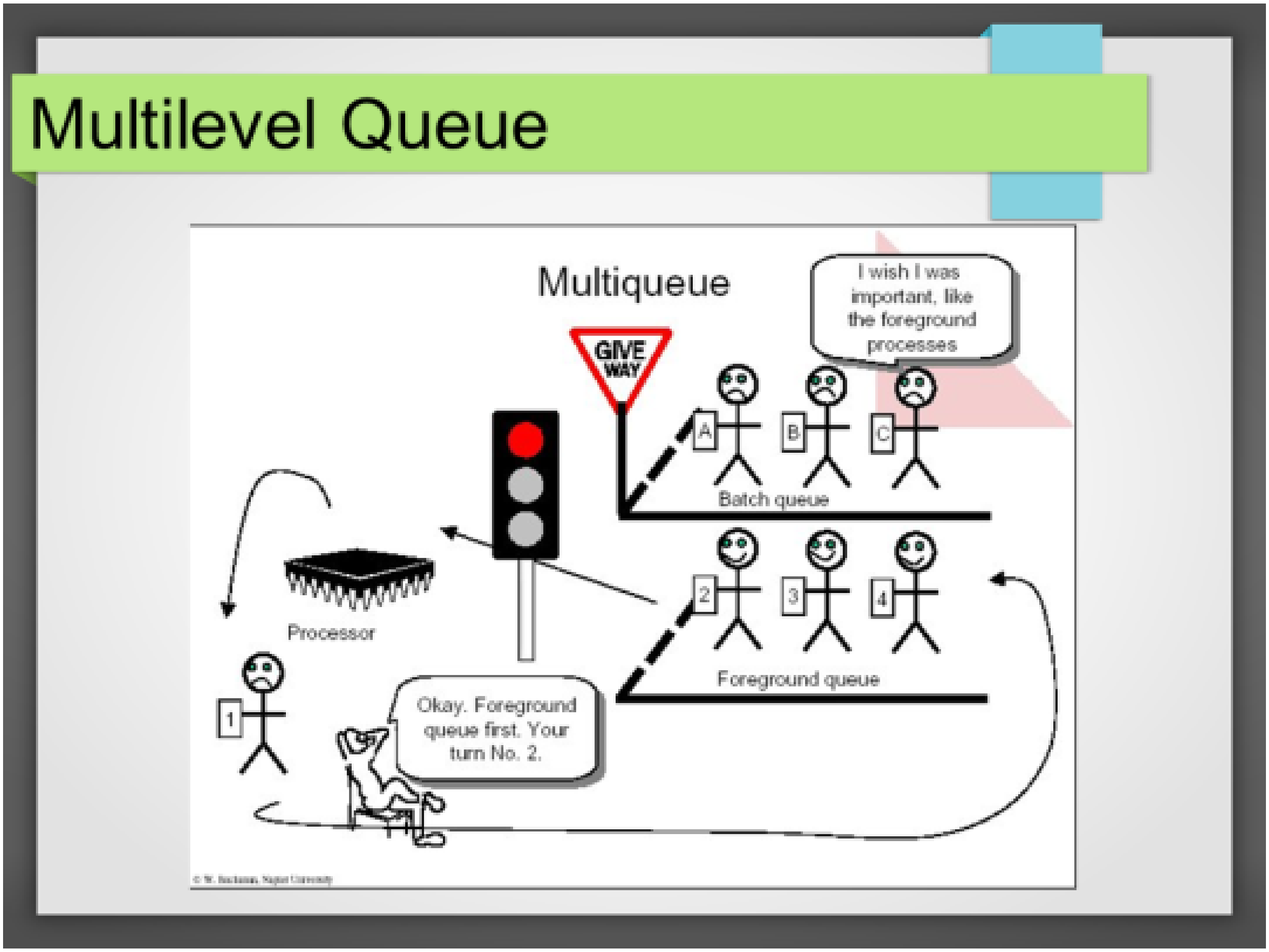


process	time
P_1	6
P_2	3
P_3	1
P_4	7

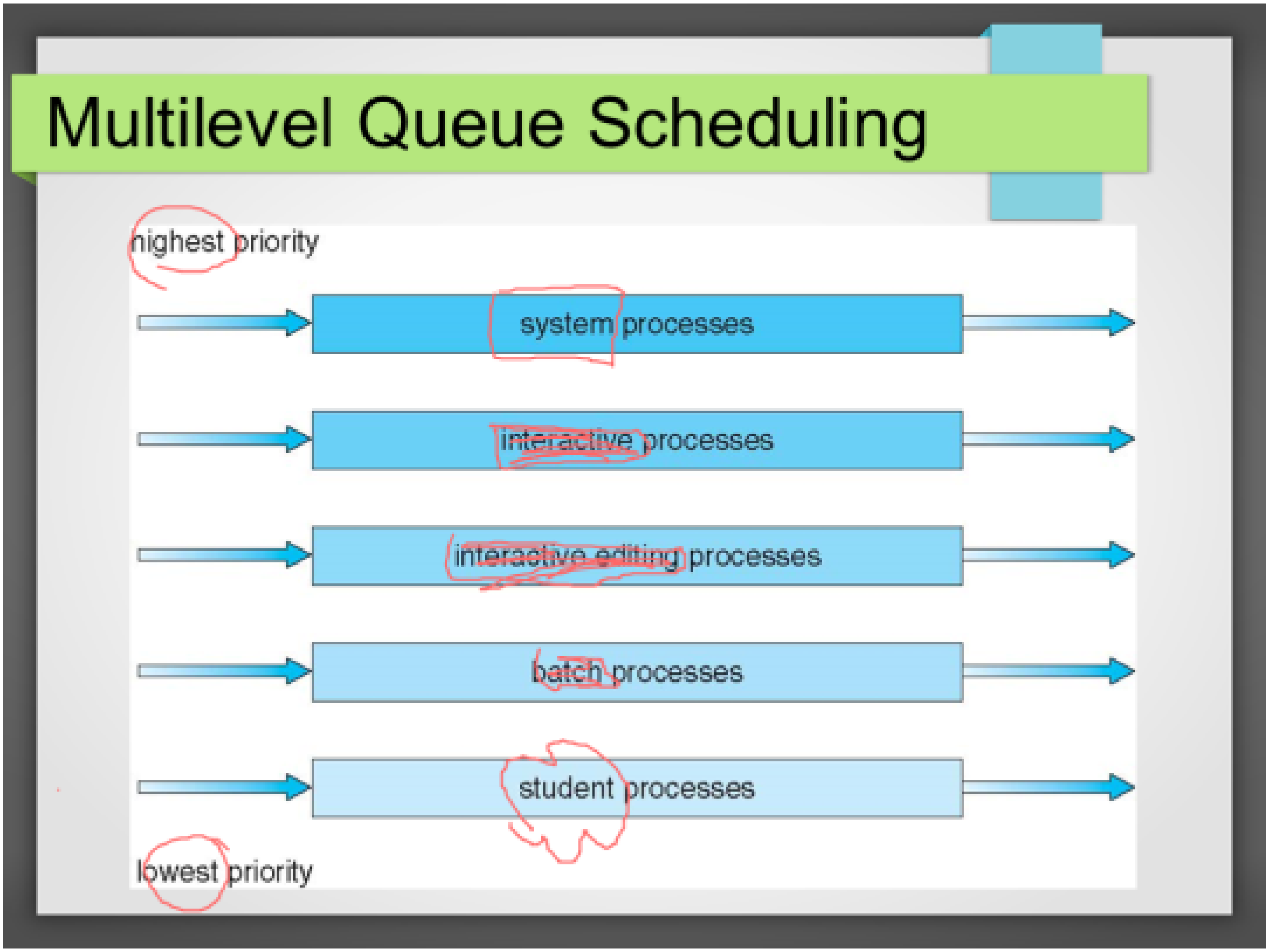
Multilevel Queue



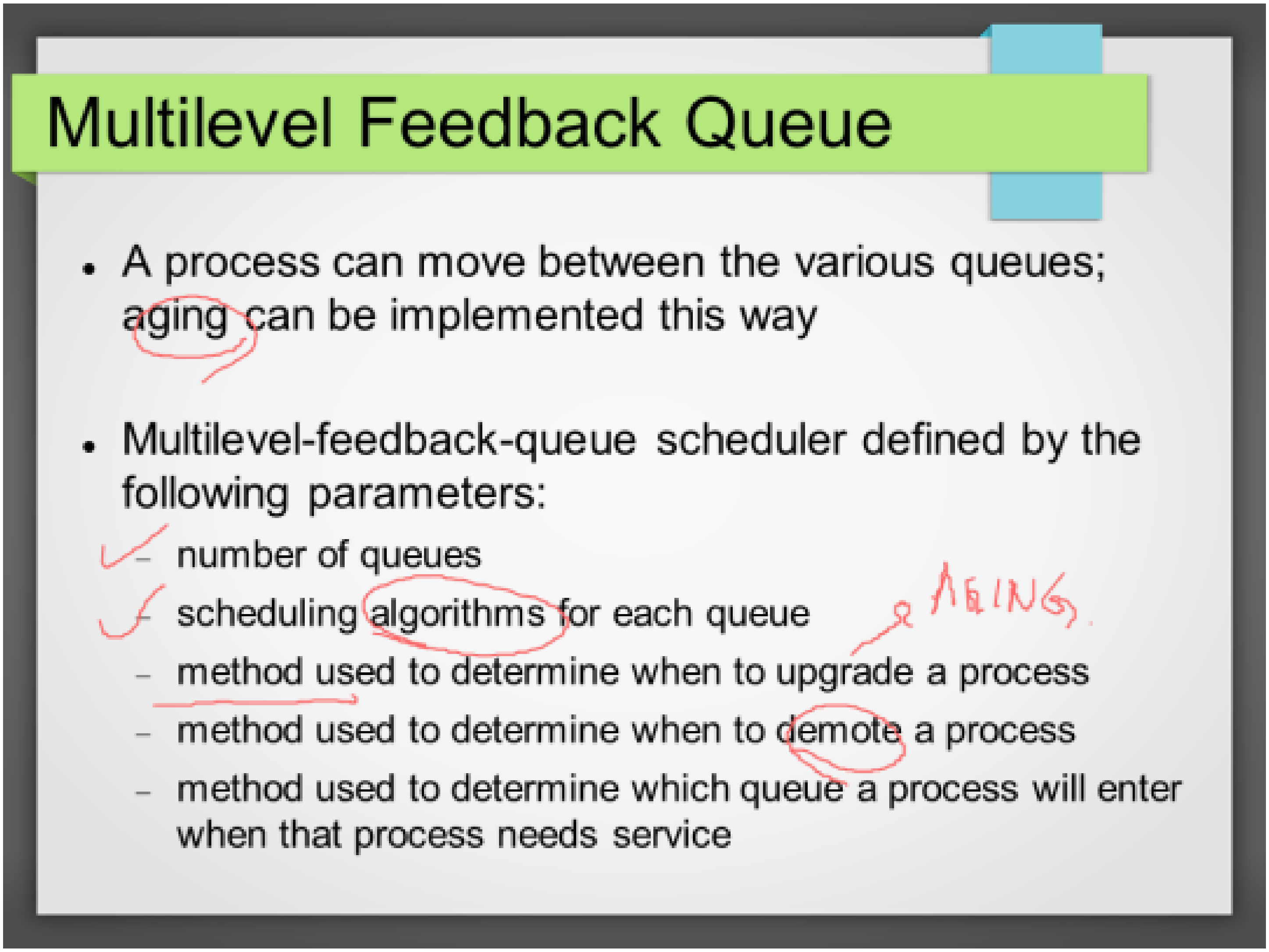
Multilevel Queue



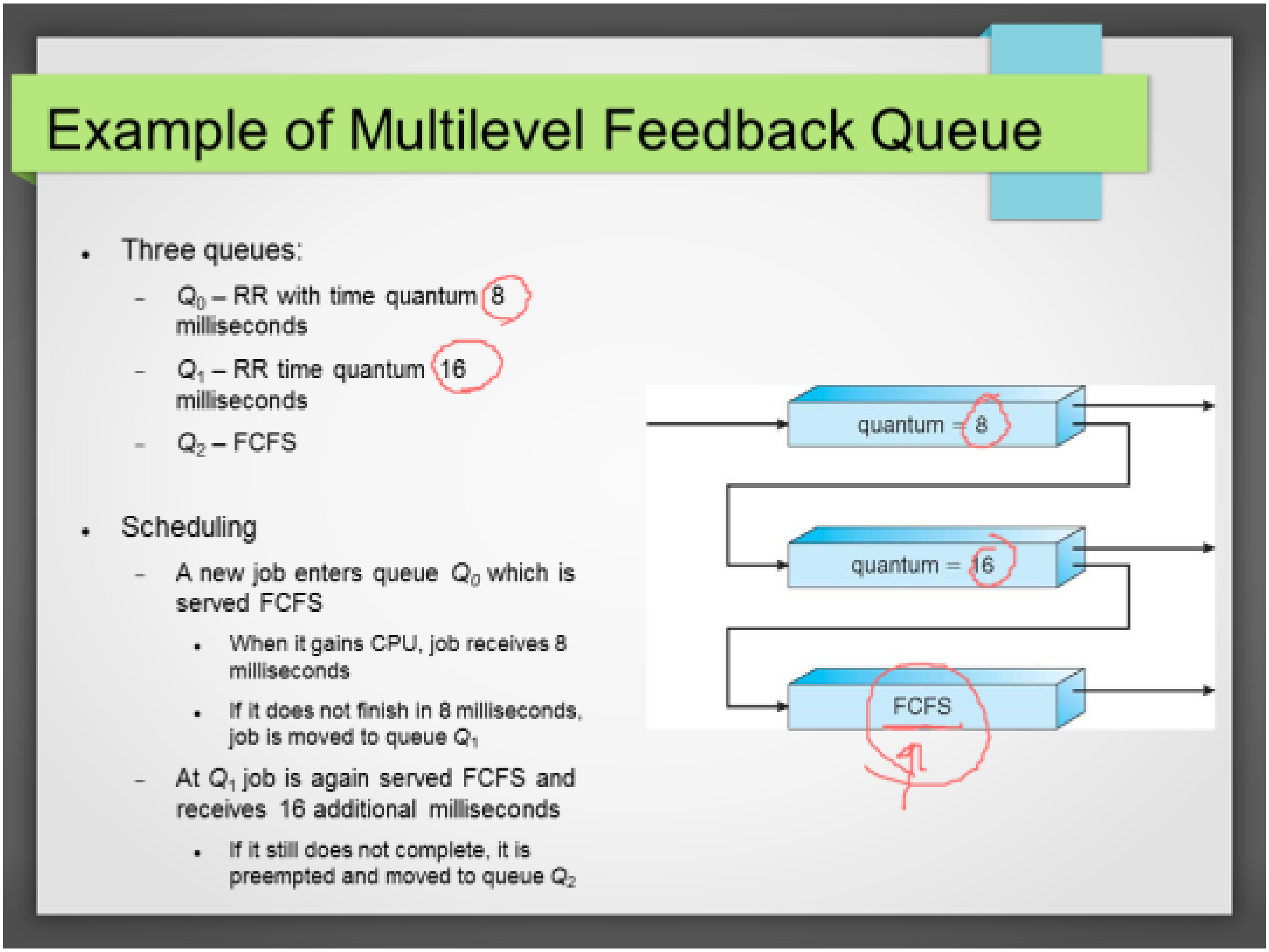
Multilevel Queue Scheduling



Multilevel Feedback Queue

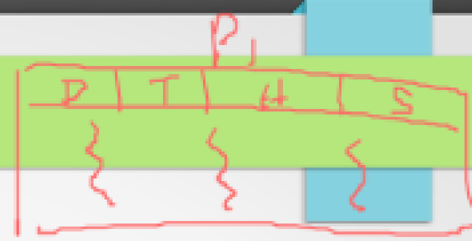


Example of Multilevel Feedback Queue



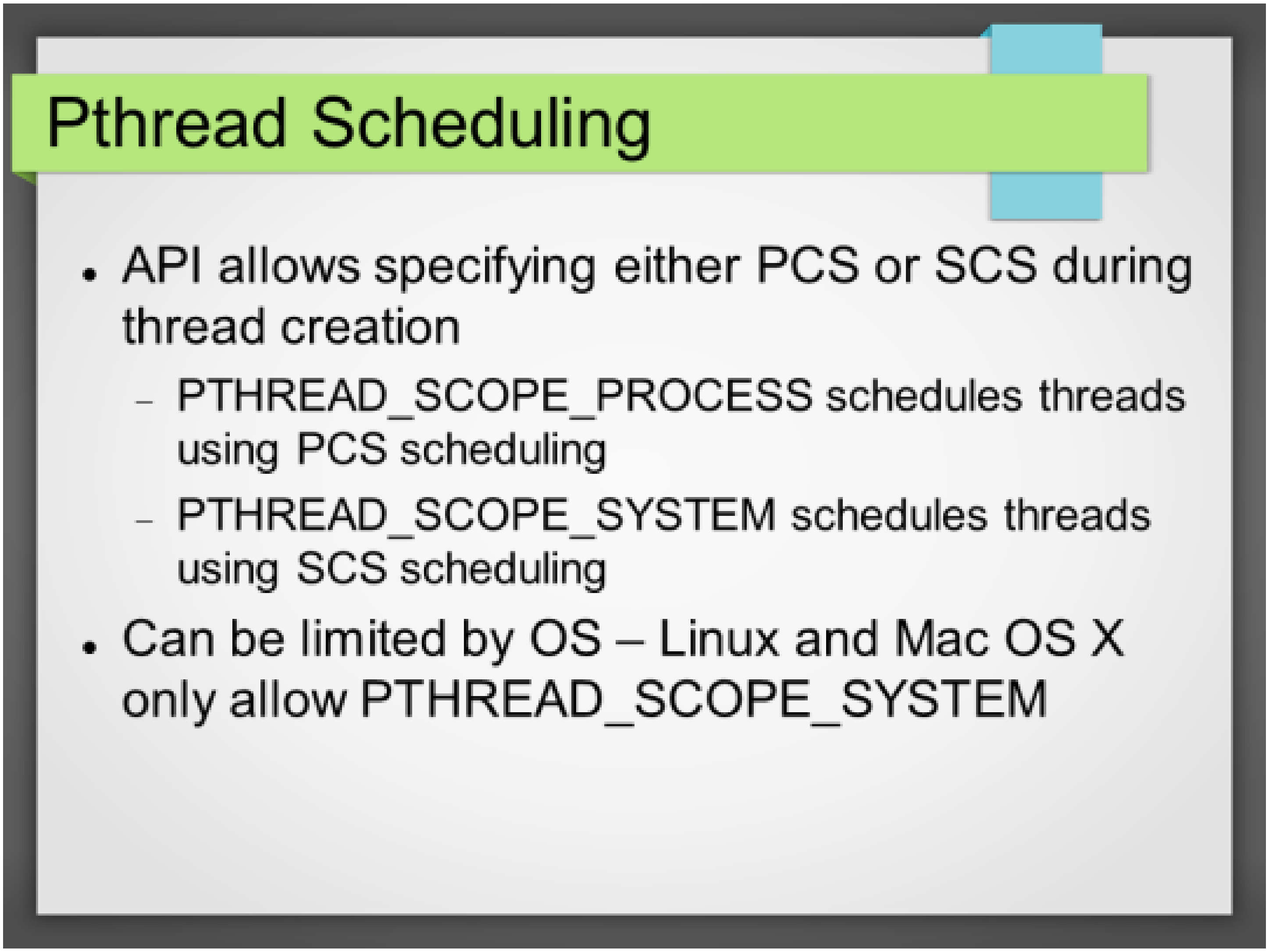
Thread Scheduling

Thread Scheduling



- Distinction between user-level and kernel-level threads
- When threads supported, threads scheduled, not processes
- Many-to-one and many-to-many models, thread library schedules user-level threads to run on LWP
  - Known as **process-contention scope** (**PCS**) since scheduling competition is within the process
  - Typically done via priority set by programmer
- Kernel thread scheduled onto available CPU is **system-contention scope** (**SCS**) – competition among all threads in system

Pthread Scheduling

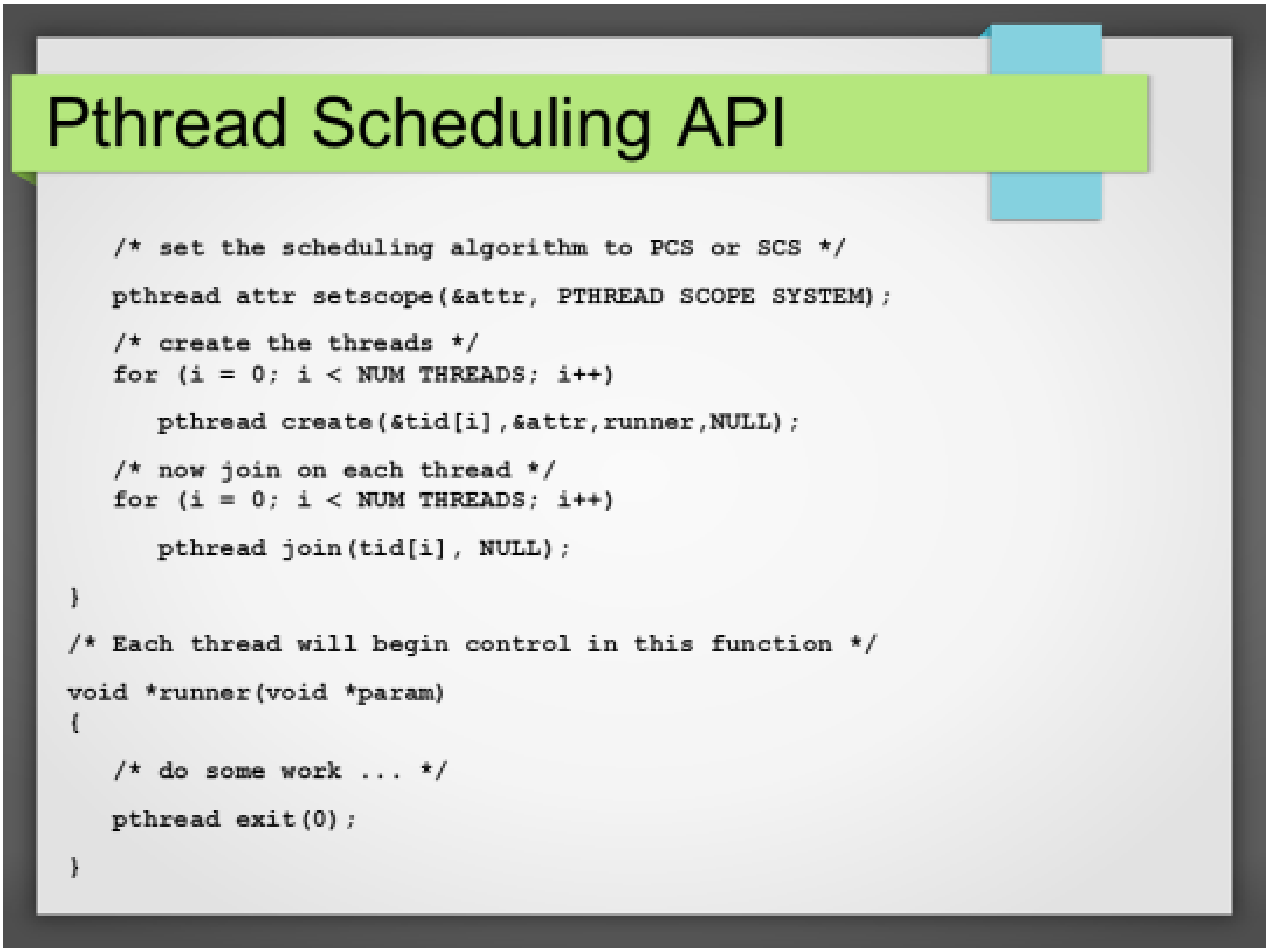


Pthread Scheduling API

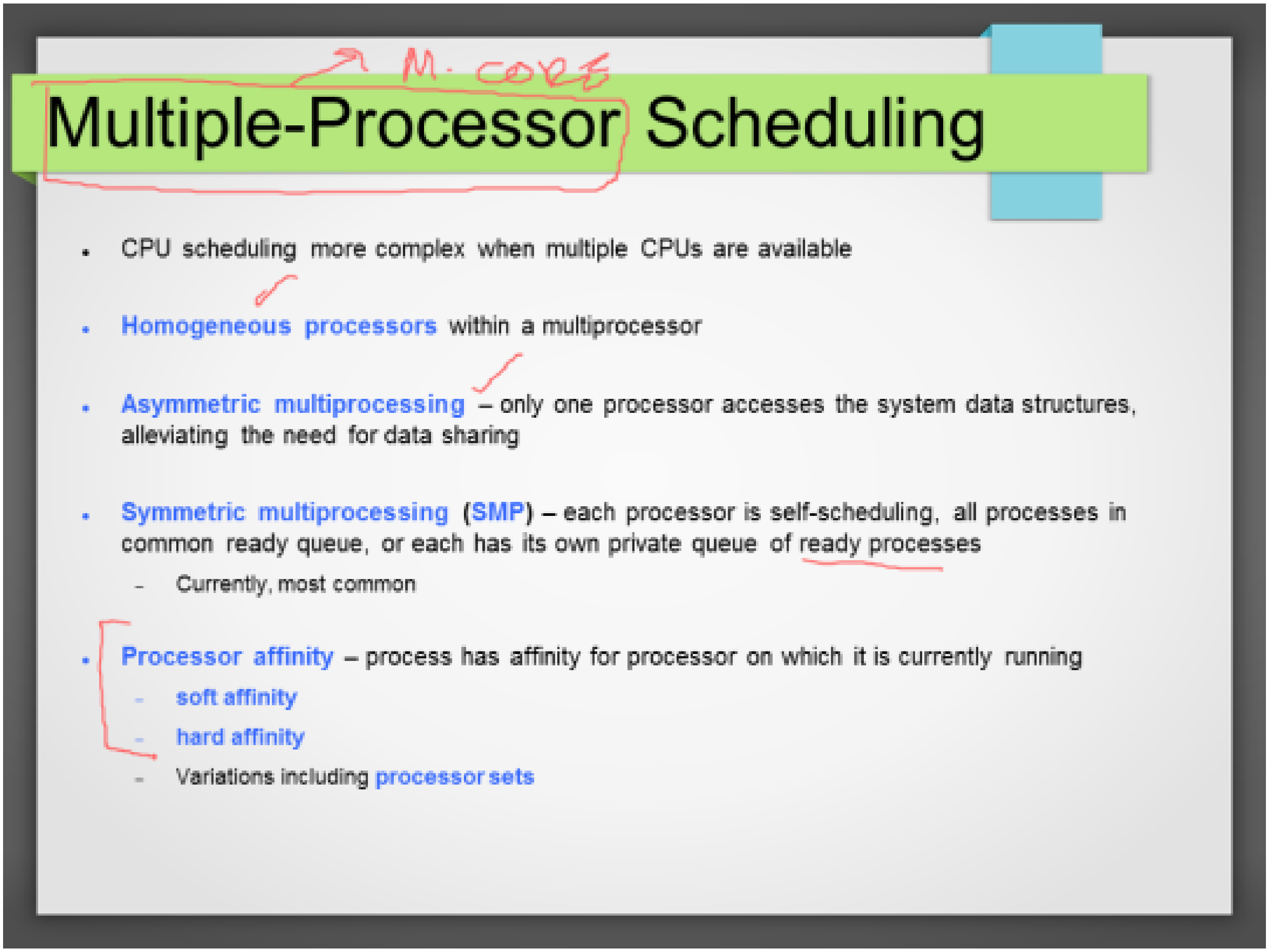
Pthread Scheduling API

```
#include <pthread.h>
#include <stdio.h>
#define NUM THREADS 5
int main(int argc, char *argv[]) {
   int i, scope;
   pthread t tid[NUM THREADS];
   pthread attr t attr;
   /* get the default attributes */
   pthread attr init(&attr);
   /* first inquire on the current scope */
   if (pthread attr getscope(&attr, &scope) != 0)
      fprintf(stderr, "Unable to get scheduling scope\n");
   else {
      if (scope == PTHREAD SCOPE PROCESS)
         printf("PTHREAD SCOPE PROCESS");
      else if (scope == PTHREAD SCOPE SYSTEM)
         printf("PTHREAD SCOPE SYSTEM");
      else
         fprintf(stderr, "Illegal scope value.\n");
   }
```

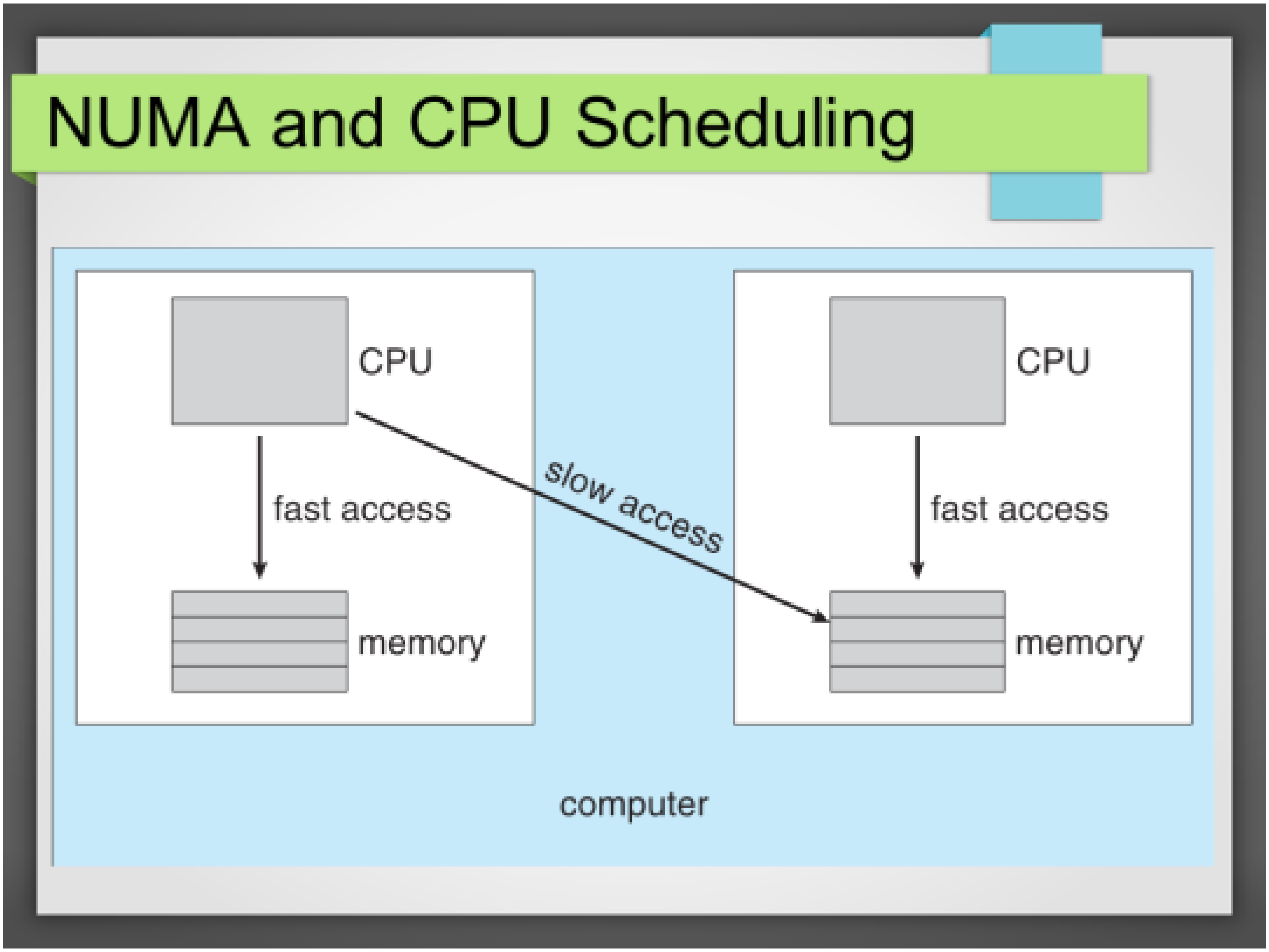
Pthread Scheduling API



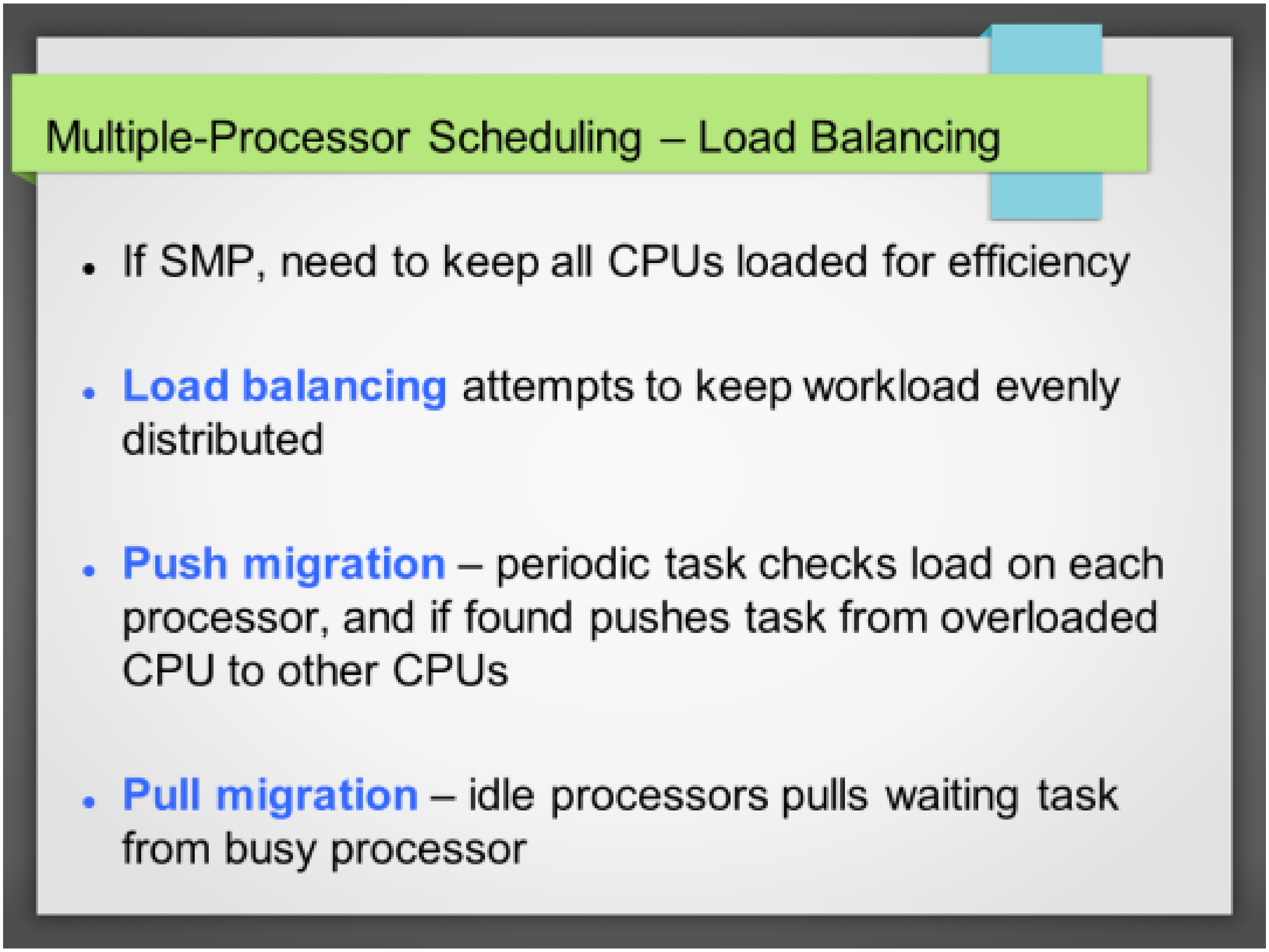
Multiple-Processor Scheduling



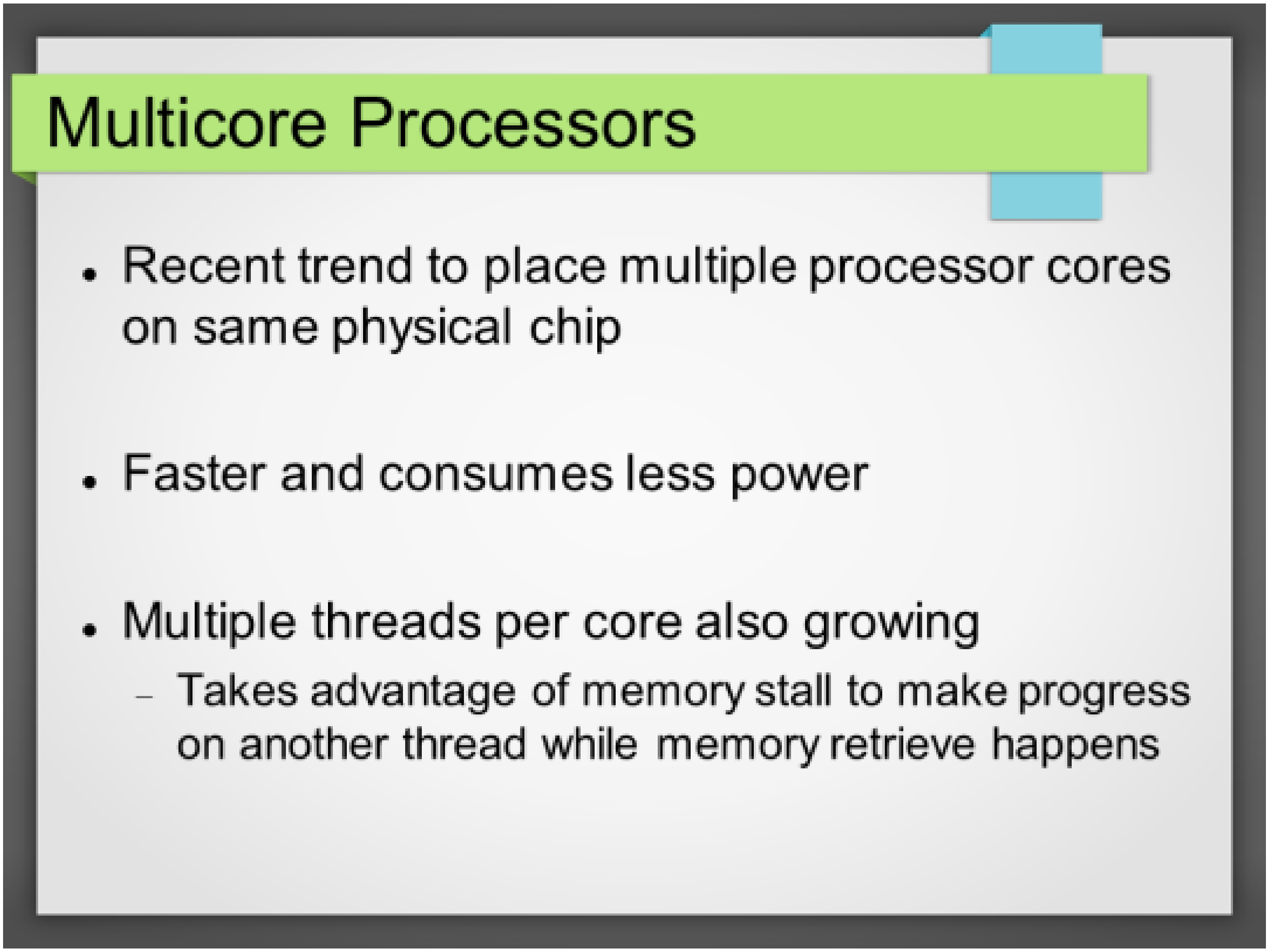
NUMA and CPU Scheduling



Multiple-Processor Scheduling ?? Load Balancing



Multicore Processors



Multithreaded Multicore System