Systems and Networking I

Applied Computer Science and Artificial Intelligence

2025-2026

Gabriele Tolomei

Computer Science Department

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA UNIVERSITÀ DI ROMA

Recap from Last Lecture

- Process is the **unit of execution** (running on a single CPU)
- OS keeps track of process-related information using an ad hoc data structure called Process Control Block (PCB)
- Process can be in one of **5 possible states**: **new**, **ready**, **waiting**, **running**, or **terminated**
- Context **switch** to intertwine the execution of multiple processes
- Process communication either via **message passing** or **shared memory**

Today: CPU Scheduling

$\boxed{\text{Policy to establish which process to execute on the CPU}}$

- Basic scheduling concepts
- Scheduling **criteria/metrics**
- Scheduling **algorithms**
- Advanced scheduling concepts

Basic Concepts

- Almost every program has some alternating cycles of CPU computations and I/O waiting
- Even a simple fetch from main memory takes a long time relatively to CPU speed
- <u>**Our assumptions:**</u> Multi-programmed, uni-processor system

Basic Concepts

- In a system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever

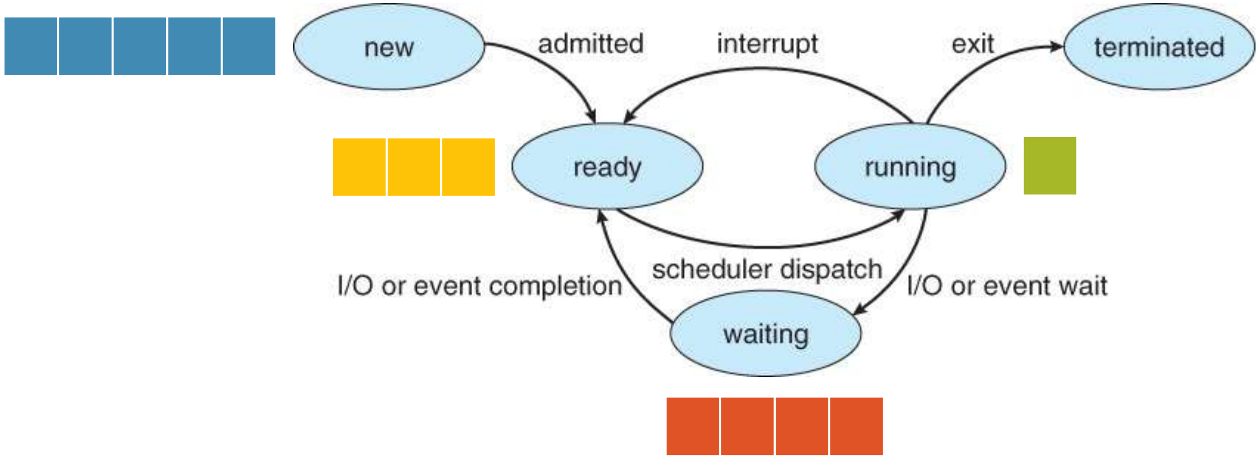
Basic Concepts

- In a system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever
- A scheduling system allows one process to use the CPU while another is waiting for I/O, thereby maximizing system utilization

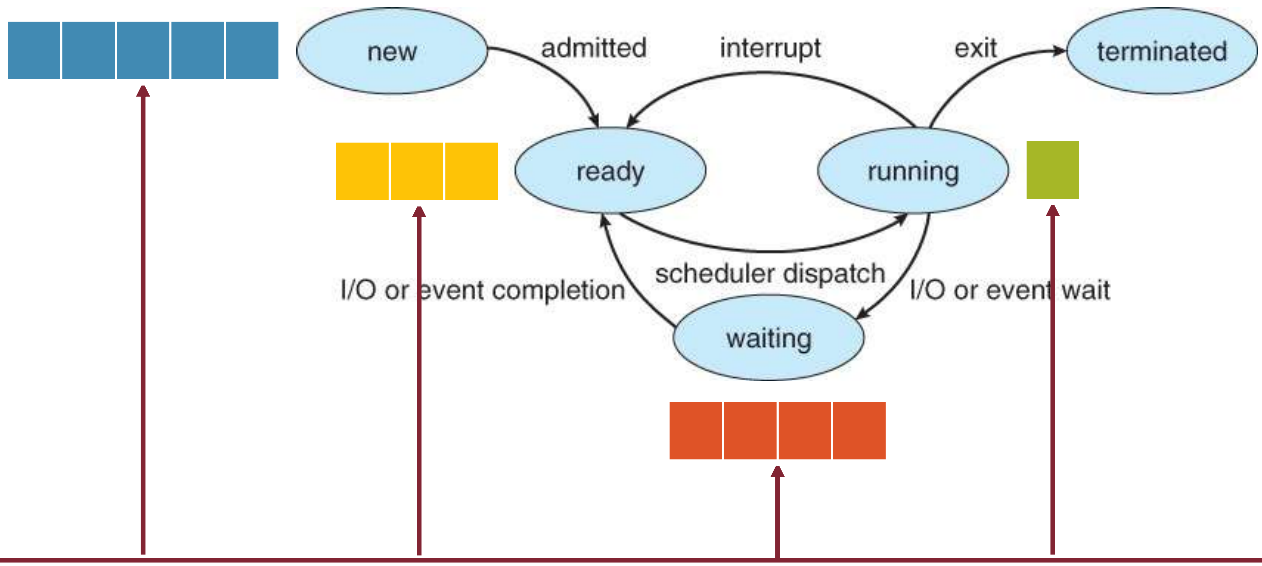
Basic Concepts

- In a system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever
- A scheduling system allows one process to use the CPU while another is waiting for I/O, thereby maximizing system utilization
- **Challenge:** Make the system as "efficient" and "fair" as possible, subject to varying and often dynamic conditions

Process Execution State Diagram

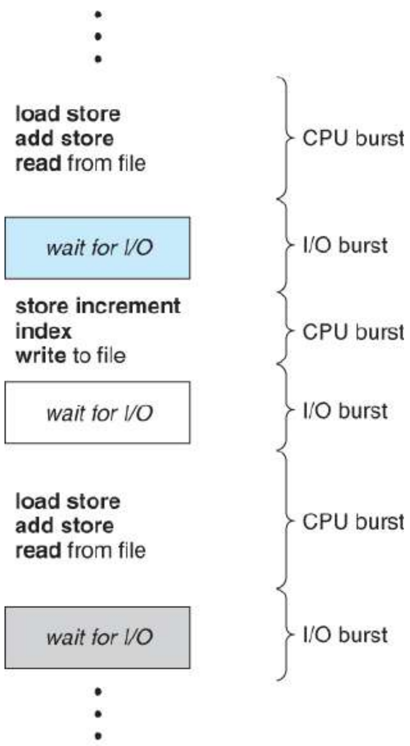


Process Execution State Diagram



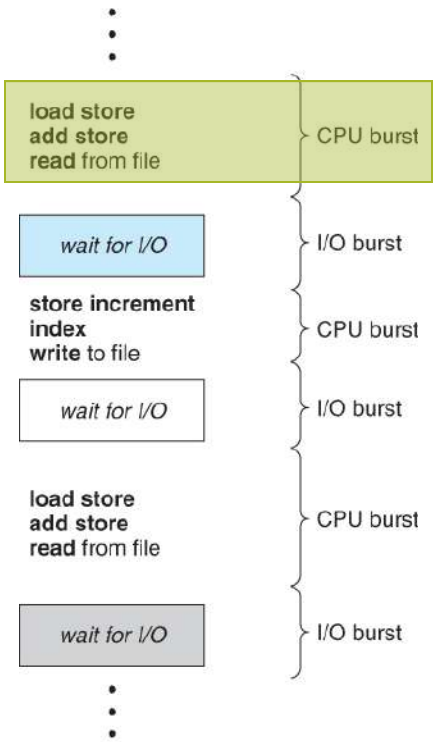
Processes managed by the OS reside in exactly one of the state queues

CPU vs. I/O Burst Cycle



All processes alternate between two states in a continuing cycle: CPU burst and I/O burst

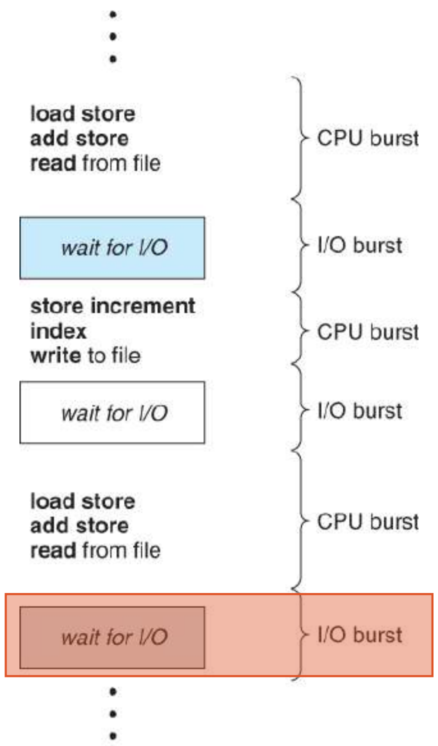
CPU vs. I/O Burst Cycle



All processes alternate between two states in a continuing cycle: CPU burst and I/O burst

CPU burst → performing calculations

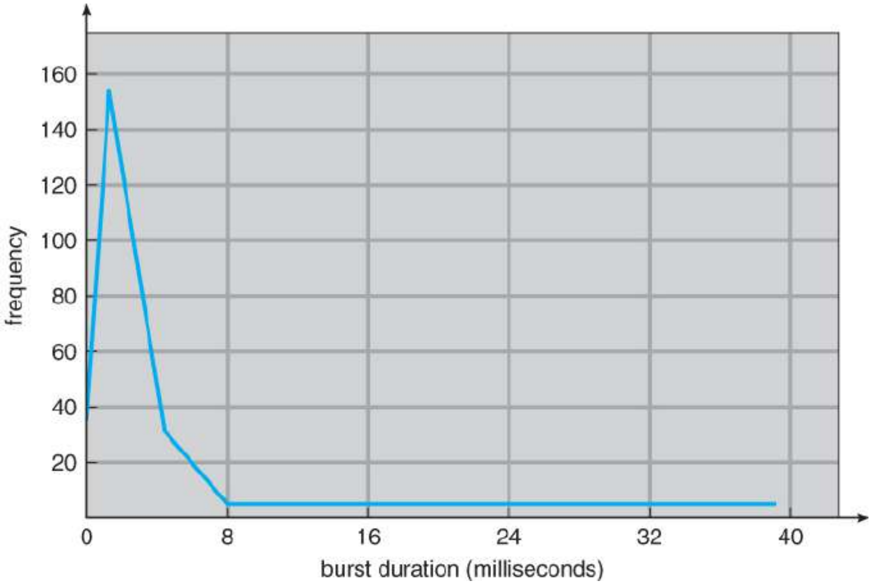
CPU vs. I/O Burst Cycle



All processes alternate between two states in a continuing cycle: CPU burst and I/O burst

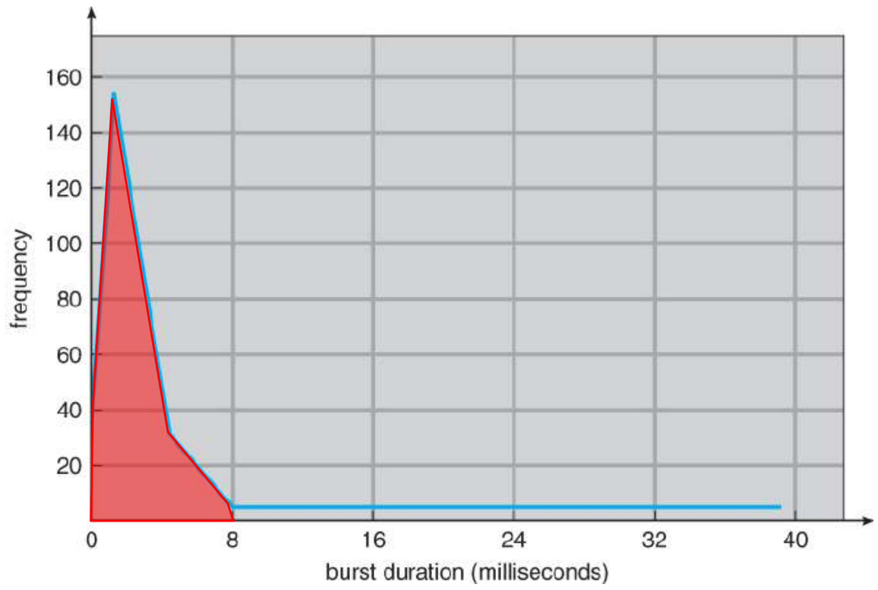
I/O burst → waiting for data transfer in or out of the system

CPU Burst Cycle: Frequency Pattern



Highly skewed distribution

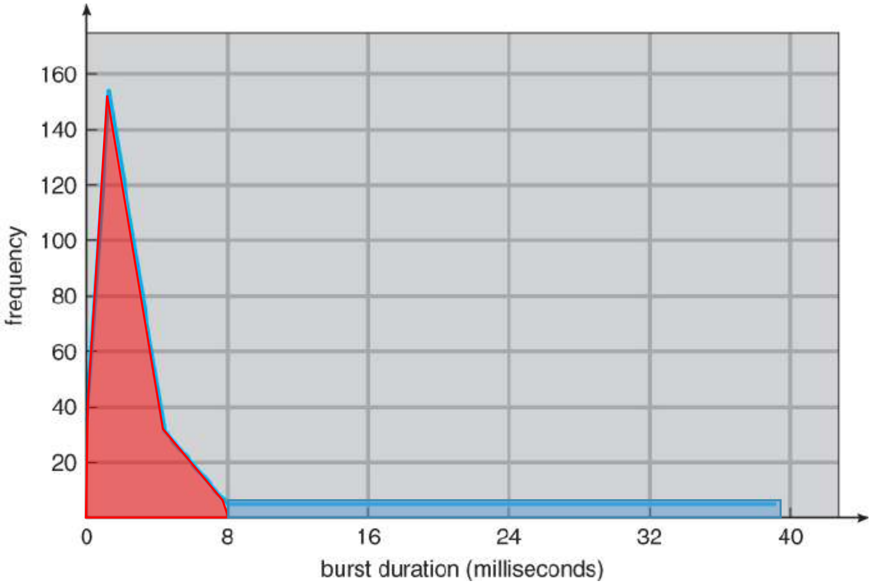
CPU Burst Cycle: Frequency Pattern



Highly skewed distribution

The vast majority of processes have short CPU bursts

CPU Burst Cycle: Frequency Pattern



Highly skewed distribution

The vast majority of processes have short CPU bursts

Few processes exhibit very long CPU bursts

Long- vs. Short-term Scheduling

Long-term scheduling

How does the OS determine the level of multiprogramming (i.e., the number of processes to be loaded in main memory)

Long- vs. Short-term Scheduling

Long-term scheduling

How does the OS determine the level of multiprogramming (i.e., the number of processes to be loaded in main memory)

Short-term scheduling

How does the OS select a process to execute from the ready queue

Long- vs. Short-term Scheduling

Long-term scheduling

How does the OS determine the level of multiprogramming (i.e., the number of processes to be loaded in main memory)

Short-term scheduling

How does the OS select a process to execute from the ready queue

CPU Scheduler

CPU Scheduler

- Whenever the CPU becomes idle, it picks another process from the `ready` queue to execute next
- The data structure for the ready queue and the algorithm used to select the next process are not necessarily based on a FIFO queue

CPU Scheduler

- Whenever the CPU becomes idle, it picks another process from the ready queue to execute next
- The data structure for the ready queue and the algorithm used to select the next process are not necessarily based on a FIFO queue

Policy goals vs. Mechanism implementations

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the **running** state to the **waiting** state

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the **running** state to the **waiting** state

for an I/O request or invocation of the wait system call

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state

in response to an interrupt

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the **waiting** state to the **ready** state

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the **waiting** state to the **ready** state

after I/O completion or a return from wait

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process is created or terminates

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process is created or terminates

No choice!
A new process must be selected

CPU Scheduling: When?

CPU scheduling decisions take place under one of 4 conditions:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process is created or terminates

Either continue with the current process or select a new one

Non-preemptive vs. Preemptive

Non-preemptive scheduling

If it takes place only when there is no choice (i.e., conditions 1 and 4)

Once a process starts it keeps running until it either voluntarily blocks or it finishes

Non-preemptive vs. Preemptive

Non-preemptive scheduling

If it takes place only when there is no choice (i.e., conditions 1 and 4)

Once a process starts it keeps running until it either voluntarily blocks or it finishes

Preemptive scheduling

Whenever scheduling takes also place under conditions 2 and 3

Non-preemptive vs. Preemptive: Examples

	Windows	Mac	UNIX-like
Non-preemptive	up to Win 3.x	up to Mac OS 9.x	–
Preemptive	since Win 95	since Mac OS X	since forever

Preemption: Issues

- Preemption might cause troubles if it occurs while:
  - the kernel is busy implementing a system call (e.g., updating critical kernel data structures)
  - two processes share data, one may get interrupted in the middle of updating shared data structures

Preemption: Issues

- Possible countermeasures:
  - Make the process wait until the system call has either completed or blocked before allowing the preemption

Preemption: Issues

- Possible countermeasures:
  - Make the process wait until the system call has either completed or blocked before allowing the preemption

problematic for real-time systems, as real-time response can no longer be guaranteed

Preemption: Issues

- Possible countermeasures:
  - Make the process wait until the system call has either completed or blocked before allowing the preemption

    problematic for real-time systems, as real-time response can no longer be guaranteed

  - Disable interrupts before entering critical code section and re-enabling immediately afterwards

Preemption: Issues

- Possible countermeasures:
  - Make the process wait until the system call has either completed or blocked before allowing the preemption

    problematic for real-time systems, as real-time response can no longer be guaranteed

  - Disable interrupts before entering critical code section and re-enabling immediately afterwards

    should only be done in rare situations, and only on very short pieces of code that will finish quickly

The Dispatcher

- The module that gives control of the CPU to the process selected by the scheduler

The Dispatcher

- The module that gives control of the CPU to the process selected by the scheduler
- Its functions include:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the newly loaded program

The Dispatcher

- The module that gives control of the CPU to the process selected by the scheduler
- Its functions include:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the newly loaded program
- The dispatcher is run on every context switch therefore the time it consumes (**dispatch latency**) must be as shortest as possible

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue
- **Completion Time:** time at which the process completes its execution

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue
- **Completion Time:** time at which the process completes its execution
- **Burst Time:** time required by a process for CPU execution

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue
- **Completion Time:** time at which the process completes its execution
- **Burst Time:** time required by a process for CPU execution
- **Turnaround Time:** time difference between completion and arrival time

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue
- **Completion Time:** time at which the process completes its execution
- **Burst Time:** time required by a process for CPU execution
- **Turnaround Time:** time difference between completion and arrival time
- **Waiting Time:** time difference between turnaround time and burst time

Useful Definitions

- **Arrival Time:** time at which the process arrives in the ready queue
- **Completion Time:** time at which the process completes its execution
- **Burst Time:** time required by a process for CPU execution
- **Turnaround Time:** time difference between completion and arrival time
- **Waiting Time:** time difference between turnaround time and burst time

NOTE

I/O waiting time is not considered here!

Useful Definitions

$$T^{arrival} = \text{arrival time}$$

$$T^{completion} = \text{completion time}$$

$$T^{burst} = \text{burst time}$$

$$T^{turnaround} = \text{tournaround time} = T^{completion} - T^{arrival}$$

$$T^{waiting} = \text{waiting time} = T^{turnaround} - T^{burst}$$

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - Waiting time
  - Response time

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - **CPU utilization**
  - Throughput
  - Tournaround time
  - Waiting time
  - Response time

Intuitively, the percentage of time the CPU is busy

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - **CPU utilization**
  - Throughput
  - Tournaround time
  - Waiting time
  - Response time

Intuitively, the percentage of time the CPU is busy

Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - **CPU utilization**
  - Throughput
  - Tournaround time
  - Waiting time
  - Response time

Intuitively, the percentage of time the CPU is busy

Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles

On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded)

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - **CPU utilization**
  - Throughput
  - Tournaround time
  - Waiting time
  - Response time

Intuitively, the percentage of time the CPU is busy

Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles

On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded)

maximize

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - **Throughput**
  - Tournaround time
  - Waiting time
  - Response time

The number of processes completed in a unit of time

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - **Throughput**
  - Tournaround time
  - Waiting time
  - Response time

The number of processes completed in a unit of time

May range from 10 per second to 1 per hour

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - **Throughput**
  - Tournaround time
  - Waiting time
  - Response time

The number of processes completed in a unit of time

May range from 10 per second to 1 per hour

maximize

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - **Tournaround time**
  - Waiting time
  - Response time

Time required for a particular process to complete, from submission time to completion

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - **Tournaround time**
  - Waiting time
  - Response time

Time required for a particular process to complete, from submission time to completion

Includes all the waiting time

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - **Tournaround time**
  - Waiting time
  - Response time

Time required for a particular process to complete, from submission time to completion

Includes all the waiting time

minimize

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - **Waiting time**
  - Response time

How much time processes spend in the ready queue waiting their turn to get on the CPU

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - **Waiting time**
  - Response time

How much time processes spend in the ready queue waiting their turn to get on the CPU

Not to be confused with the waiting state!

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - **Waiting time**
  - Response time

How much time processes spend in the ready queue waiting their turn to get on the CPU

Not to be confused with the waiting state!

Processes in the waiting state are not under control of the CPU scheduler (they are just not ready to run!)

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - **Waiting time**
  - Response time

How much time processes spend in the ready queue waiting their turn to get on the CPU

Not to be confused with the waiting state!

Processes in the waiting state are not under control of the CPU scheduler (they are just not ready to run!)

minimize

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - Waiting time
  - **Response time**

The time taken from the issuance of a command to the beginning of a response to that command

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - Waiting time
  - **Response time**

The time taken from the issuance of a command to the beginning of a response to that command

Mostly, for interactive processes

Scheduling Criteria/Metrics

- There are several different criteria to consider when trying to select the "best" scheduling algorithm, including:
  - CPU utilization
  - Throughput
  - Tournaround time
  - Waiting time
  - **Response time**

The time taken from the issuance of a command to the beginning of a response to that command

Mostly, for interactive processes

minimize

Scheduling: Trade-off

- Ideally, choose a CPU scheduler that optimizes all metrics simultaneously
- Generally, the above is impossible and a trade-off is needed!
- **Idea:** Choose a scheduling algorithm based on its ability to satisfy a given policy

Scheduling Policies

- Minimize **average** response time
  - Provide output to the user as quickly as possible

Scheduling Policies

- Minimize **average** response time
  - Provide output to the user as quickly as possible
- Minimize the **maximum** response time
  - Worst-case bound

Scheduling Policies

- Minimize **average** response time
  - Provide output to the user as quickly as possible
- Minimize the **maximum** response time
  - Worst-case bound
- Minimize **variance** of response time
  - Users are more accepting of a consistent predictable system than an inconsistent one

Scheduling Policies

- Minimize **average** response time
  - Provide output to the user as quickly as possible
- Minimize the **maximum** response time
  - Worst-case bound
- Minimize **variance** of response time
  - Users are more accepting of a consistent predictable system than an inconsistent one

Typical of interactive systems

Scheduling Policies

- Maximize throughput translates into:
  - Minimizing overhead (OS context switching)
  - Efficient usage of system resources (CPU and I/O devices)

Scheduling Policies

- Maximize throughput translates into:
  - Minimizing overhead (OS context switching)
  - Efficient usage of system resources (CPU and I/O devices)
- Minimize waiting time
  - Give every process the same amount of time on the CPU
  - Might increase the average response time

Scheduling Policies

- Maximize throughput translates into:
  - Minimizing overhead (OS context switching)
  - Efficient usage of system resources (CPU and I/O devices)
- Minimize waiting time
  - Give every process the same amount of time on the CPU
  - Might increase the average response time

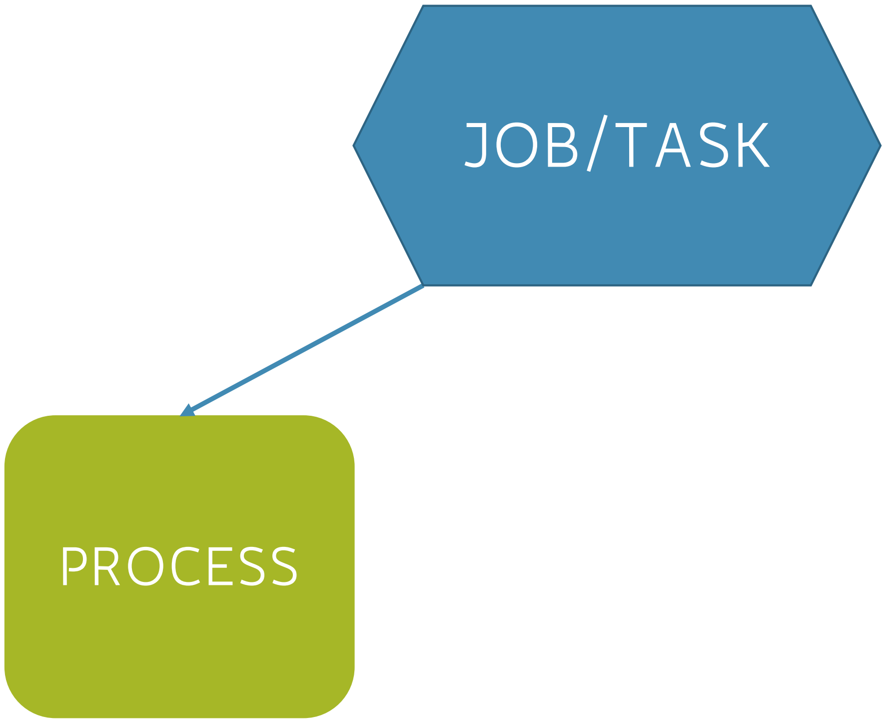
Typical of batch systems

A Quick Note on Terminology

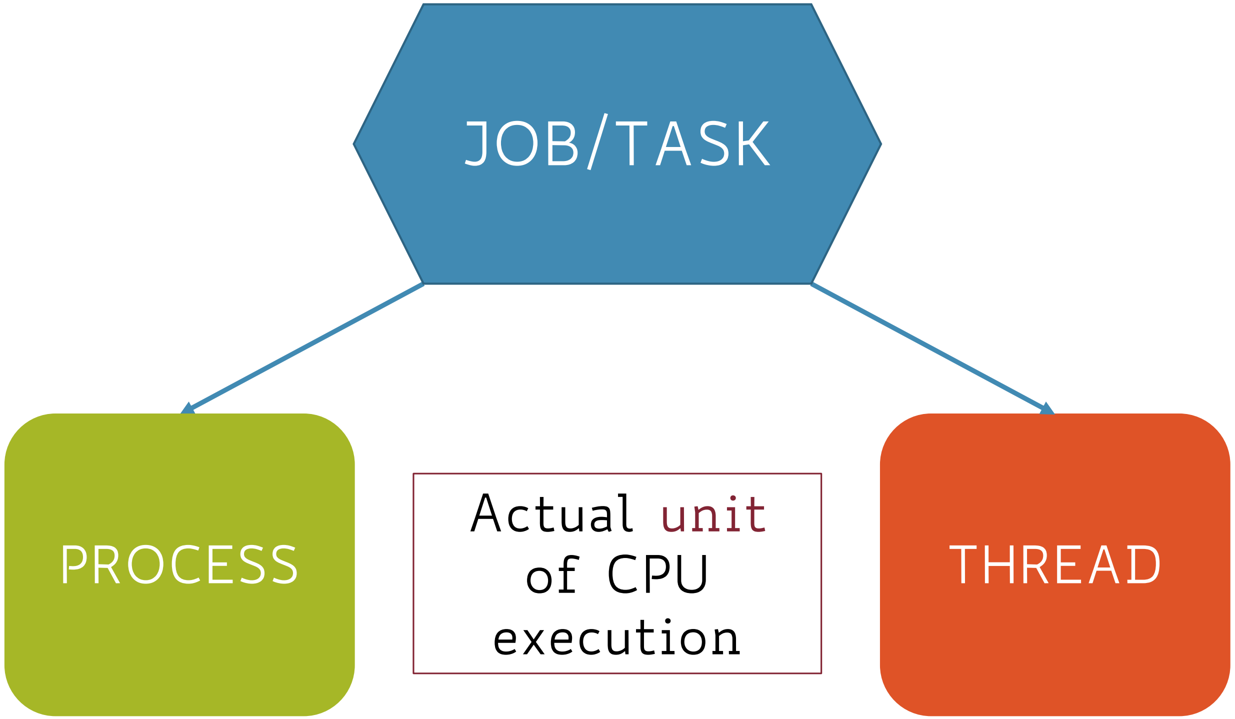
JOB/TASK

General unit of CPU execution

A Quick Note on Terminology



A Quick Note on Terminology



Scheduling Policies: Our Assumptions

- One process per user

Scheduling Policies: Our Assumptions

- One process per user
- Processes are independent from each other (i.e., no communication)

Scheduling Policies: Our Assumptions

- One process per user
- Processes are independent from each other (i.e., no communication)
- One single thread (of execution) per process

Scheduling Policies: Our Assumptions

- One process per user
- Processes are independent from each other (i.e., no communication)
- One single thread (of execution) per process

We will talk about threads very soon but for now most of the things we will be discussing remain valid even on a multi-threaded system

Summary

- Scheduling allows one process to use the CPU while another is waiting for I/O

Summary

- Scheduling allows one process to use the CPU while another is waiting for I/O
- The goal of scheduling is to maximize system utilization

Summary

- Scheduling allows one process to use the CPU while another is waiting for I/O
- The goal of scheduling is to maximize system utilization
- `non-preemptive` vs. `preemptive` scheduler

Summary

- Scheduling allows one process to use the CPU while another is waiting for I/O
- The goal of scheduling is to maximize system utilization
- **non-preemptive** vs. **preemptive** scheduler
- Different scheduling policies optimize different metrics