Chapter 7

Process Scheduling

Process Scheduler

- Why do we even need to a process scheduler?
  - In simplest form, CPU must be shared by
    - OS
    - Application
  - In reality, [multiprogramming]
    - OS: many separate pieces (processes)
    - Many Applications

- Scheduling [Policy] addresses...
  - When to remove a process from CPU?
  - Which ready process to allocate the CPU to?
**Context Switch**

- Processes are switched out using **Context Switching**
- Context Switch:
  - **Save** pertinent info for current process
    - PC, Register, Status, etc.
  - **Update** PC, Register, Status, etc.
    - with info for process selected to run

- **Switching User Process**
  - 2 Context switches (CTX)
  - Process 1 running
  - Dispatcher selects next process
  - Process 2 running

---

**Scheduler**

[Diagram of the Scheduler]

**The Scheduler**

- From Other States
- Remove the Running Process
- Process Descriptors
- Ready Process
- Enqueue
  - Ready List
- Dispatcher
  - Context Switcher
  - CPU
Selection Strategies

Motivation
- To “optimize” some aspect of system behavior

Considerations
- Priority of process
  - External: assigned
  - Internal: aging
- Fairness: no starvation
- Overall Resource Utilization
  ...

Selection Strategies...

Considerations...
- Turnaround time
  - Average time / job
- Throughput
  - Jobs / time unit
- Response time
- System availability
- Deadlines
Definition & Terms

- **Time Quantum**
  - Amount of time between timer interrupts
  - Also called Time Slice

- **Service Time** $\tau (P_i)$
  - Amount of time process needs to be in Running state (acquired CPU) before it is completed

- **Wait Time** $W (P_i)$
  - Time a process spends waiting in the Ready state before its first transition to the Running state

Definition & Terms...

- **Turnaround Time** $T (P_i)$
  - Amount of time between moment process first enters Ready state and the moment the process exits Running state for the last time (completed)

- Service time, Wait time & Turnaround time are measurable metrics used to compare scheduling algorithms
Classes of Scheduling Algorithms

- 2 major classes
  - Non-preemptive
    - Run to completion
  - Preemptive
    - Process with highest priority always gets CPU
      
      Recall: Several ways to establish priority

Non-preemptive Strategies - FCFS

- FCFS - First-Come, First-Serve
  - Processes are assigned the CPU in the order they arrive
  - FIFO structure (queue)
  - Ignores service time and any other criteria that may influence performance w.r.t.
    - Turnaround time
    - Waiting time
Non-preemptive Strategies - SJN

- **SJN** - Shortest Job Next
  - Assumes service time known a priori
  - Realistically, can make estimated based on
    - Past experience history
    - Size of input
    - User estimate
  - Algorithm chooses a next process that one which has shortest service time
    => Minimizes average waiting time
    => Maximizes throughput
    => Can penalize processes with high service time

How might starvation occur???
Priority Scheduling

- Priority Scheduling
  - Schedule based on externally assigned priorities
  - Highest priority job always gets CPU next

How might starvation occur ??
Priority Scheduling...

<table>
<thead>
<tr>
<th>(i)</th>
<th>value</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
T_{\text{total}}(P_0) = t(p_0) + t(p_1) + t(p_2) + t(p_3) + t(p_4) = 350 + 128 + 475 + 250 + 75 = 1275
\]

\[
T_{\text{total}}(P_1) = t(p_0) + t(p_1) + t(p_2) + t(p_3) + t(p_4) = 350 + 128 + 250 = 728
\]

\[
T_{\text{total}}(P_2) = t(p_0) + t(p_1) + t(p_2) + t(p_3) + t(p_4) = 350 + 128 + 475 = 953
\]

\[
T_{\text{total}}(P_3) = t(p_0) + t(p_1) + t(p_2) + t(p_3) + t(p_4) = 350 + 128 + 75 = 553
\]

\[
T_{\text{total}}(P_4) = t(p_0) + t(p_1) + t(p_2) + t(p_3) + t(p_4) = 350 + 128 + 250 = 728
\]

Therefore, the average turnaround time is

\[
T_{\text{avg}} = \frac{1275 + 728 + 953 + 553}{4} = 711
\]

We determine the waiting times to be:

\[
\begin{align*}
W_{P_0} &= 925 \\
W_{P_1} &= 250 \\
W_{P_2} &= 375 \\
W_{P_3} &= 0 \\
W_{P_4} &= 0
\end{align*}
\]

So the average wait time is

\[
W = \frac{925 + 250 + 375 + 0 + 0}{5} = 480
\]

Deadline Scheduling

- Deadline Scheduling
  - Processes are scheduled to meet stated deadlines

<table>
<thead>
<tr>
<th>(i)</th>
<th>value</th>
<th>DEADLINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>975</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>690</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>1020</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>8505</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>300</td>
</tr>
</tbody>
</table>

Homework: Compute avg. turnaround time and avg. wait time for each of the possibilities
Preemptive Strategies

- Highest priority among all processes in Ready state is allocated CPU
- If a lower priority process is executing when a higher priority process arrives in Ready queue
  => Lower priority process will be interrupted and replaced with higher priority process
- Depending on scheduling algorithm
  - Provides quick response to higher priority process
  - Provides a fair share of CPU for all processes (esp. when Round Robin is used)

Preemptive Strategies - SJN

- SJN - Shortest Job Next
  - [Initial] selection based on shortest service time
  - When new process arrives in Ready queue, need only compare $\tau(P_{active})$ with $\tau(P_{new})$
    - If $\tau(P_{active}) \leq \tau(P_{new})$, nothing happens
    - If $\tau(P_{active}) > \tau(P_{active})$, interrupt, CTX
  => Service time used to determine priorities
Preemptive Strategies - Priority Scheduling

- Priority Scheduling
  - Externally assigned priorities used to determine
    - Who is (initially) selected to run
    - If currently running process is interrupted in favor of newly arrived process

- Note: With preemptive scheduling, CTX can have significant impact

Preemptive Strategies - Round Robin

- RR – Round Robin
  - Most widely used
  - Each process will receive some time slice or quantum quantum << service time of $P_i$
  - User interrupts timer
  - Scheduler continuously cycles through Ready queue giving each process 1 quantum of CPU time
  - I/O request can cause process to lose part of its quantum
Round Robin w/o considering CTX

Quantum = 50

The turnaround times (derived from the Gantt chart) are:

- $T_{	ext{run}}(p_1) = 1100$
- $T_{	ext{run}}(p_2) = 950$
- $T_{	ext{run}}(p_3) = 1275$
- $T_{	ext{run}}(p_4) = 950$
- $T_{	ext{run}}(p_5) = 475$

Therefore, the average turnaround time is:

$$T_{\text{avg}} = \frac{1100 + 950 + 1275 + 950 + 475}{5} = \frac{5435}{5} = 1087.$$
Quantum = 50  
CTX = 10  

<table>
<thead>
<tr>
<th>i</th>
<th>t[µs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>280</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

The turnaround times (derived from the Gantt chart) are:

- \( T_{p0}(p_0) = 1320 \)
- \( T_{p1}(p_1) = 660 \)
- \( T_{p2}(p_2) = 1545 \)
- \( T_{p3}(p_3) = 1140 \)
- \( T_{p4}(p_4) = 990 \)

Therefore the average turnaround time is:

\[
\overline{T_{\text{avg}}} = \frac{1320 + 660 + 1545 + 1140 + 990}{5} = \frac{5655}{5} = 1131
\]

From the Gantt chart, we determine the waiting times to be:

- \( W_{p0}(p_0) = 0 \)
- \( W_{p1}(p_1) = 850 \)
- \( W_{p2}(p_2) = 120 \)
- \( W_{p3}(p_3) = 180 \)
- \( W_{p4}(p_4) = 240 \)

So the average wait time is:

\[
\overline{W} = \frac{0 + 850 + 120 + 180 + 240}{5} = \frac{1390}{5} = 278
\]

Job & Process Scheduling
Process Life Cycle

Job Scheduler
- Controls when jobs will be allowed to contend the CPU
- Most popular techniques
  - FIFO: First in, first out
  - SJF: Shortest job first

Process Scheduler
- Controls when individual jobs (processes) will actually get the CPU
- Only interesting in multi-programming
- Most popular technique is Round Robin
  - Give each process one time slice in turn until complete

Job Scheduler & Process Scheduler

Dark square contains fixed, maximum number of processes
Interrupts

- An interrupt is generated to inform the system of completion of a particular event, such as the completion of I/O, the expiration of a time slice, etc.

- Interrupts can be generated by hardware or software

- The code which handles an interrupt is called the Interrupt Service Routine

Interrupt Handling

- The OS gains control
- The OS saves the state of the interrupted process
  - Utilizes process PCB
- The OS
  - Analyzes the interrupt
  - Passes control to the appropriate Interrupt Handler (IH)
- When IH is done, CPU is dispatched to either
  - Process that was running prior to interrupt, OR
  - Process at the head of the Ready queue
Scheduling Objectives

• Be Fair
• Maximize Throughput
• Minimize Response time delay
• Balance Resource Usage
• Enforce Priorities
• Give Preference to Processes holding Key Resources

Job Scheduling: SJF -: Shortest Job First

Scheduling based on estimated run time.
(Estimating run time is, however, normally impossible!)

• Favors short jobs over long ones
• Tends to
  • reduce number of jobs running, but
  • increases turnaround time for long jobs
• Usually paired with non-preemptive (run-to-completion) process scheduling
  • average turnaround time is less than or equal to any other nonpreemptive discipline (including FIFO)
Turnaround and Weighted Turnaround Time

Let:  
N be number of jobs  
A<sub>i</sub> be arrival time of i-th job  
F<sub>i</sub> be finish time of i-th job

Turnaround time for i<sup>th</sup> job:  
\[ T_i = F_i - A_i \]

Average turnaround time for i<sup>th</sup> job:  
\[ T = \frac{\sum T_i}{N} \]

Weighted turnaround time for i<sup>th</sup> job:  
\[ WT_i = \frac{(F_i - A_i)}{(Service-time)_i} \]

Average Weighted Turnaround time:  
\[ WT = \frac{\sum WT_i}{N} \]

---

Job & Process Sched: Example 1

Assume job arrival and runtimes as shown

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>10.1</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>10.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Non-preemptive process scheduling (run to completion)  
No I/O or Memory Constraints

When would the jobs finish given that the job scheduling algorithm was:

1) **FIFO**
2) **Shortest Job First**?
Example 1 - FIFO Solution

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Start</th>
<th>Finish</th>
<th>(T_i)</th>
<th>(WT_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>___</td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>2</td>
<td>10.1</td>
<td>___</td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td>3</td>
<td>10.25</td>
<td>___</td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
</tbody>
</table>

Average Turnaround = \(T = \) ____

Average Weighted Turnaround = \(WT = \) ____

---

Example 1 - FIFO Solution (completed)

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Start</th>
<th>Finish</th>
<th>Turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>10.0</td>
<td>12.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>10.1</td>
<td>12.0</td>
<td>13.0</td>
<td>2.9</td>
</tr>
<tr>
<td>3</td>
<td>10.25</td>
<td>13.0</td>
<td>13.25</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[ \text{Avg Turnaround time} \quad T = 2.63 \]
### Example 1 - SJF Solution

#### Job Arrives Start Finish Turnaround

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Start</th>
<th>Finish</th>
<th>Turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>10.0</td>
<td>12.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>10.1</td>
<td>12.25</td>
<td>13.25</td>
<td>3.15</td>
</tr>
<tr>
<td>3</td>
<td>10.25</td>
<td>12.0</td>
<td>12.25</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Average Turnaround time**  
\[ T = 2.38 \]
Preemptive Process Sched. Algorithms

1) **Round Robin (RR)**
   - Each process runs either until
     - its time quantum expires, or
     - it blocks to perform I/O.

2) **Priority**
   - Each process is statically assigned a priority; run high before low priority (RR within Priority for equal priorities)

3) **Dynamic Priority**
   - Same as #2, except priority level of each process can change dynamically.

4. **Processor Sharing (PS)**
   - Limit of RR as time quantum goes to zero.
   - (Like giving each CPU cycle to a different process, in round robin fashion.)
   - N processes scheduled by PS = each job runs on dedicated N-fold slower CPU. Thus, READY = RUNNING.
   - CPU Time “shared” equally among processes
Scheduling Example 2

Assume:
Multiprogramming  FIFO Job Scheduling
Processor Sharing Process Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>10.2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>10.4</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>10.5</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>10.8</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Current event:
A=Arrival
S=Schedule
F=Finish
H=In HoldQ

Definitions

Number processes in ready queue before this event

Current event:
A=Arrival
S=Schedule
F=Finish
H=In HoldQ

Reduction in run time that each process in ready queue experiences since last event occurred

Remaining run time of each process in ready queue
## Example 2 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>1 A,S</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>10.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>0.1, 0.5</td>
</tr>
<tr>
<td>10.4</td>
<td>1 F</td>
<td>2</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>3 A,S</td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>10.5</td>
<td>4 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05, 0.4</td>
</tr>
<tr>
<td>10.65</td>
<td>3 F</td>
<td>3</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.35</td>
</tr>
</tbody>
</table>

### Example 2 Continued...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.8</td>
<td>5 A,S</td>
<td>2</td>
<td>0.075</td>
<td>0.225, 0.275</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>11.1</td>
<td>5 F</td>
<td>3</td>
<td>0.1</td>
<td>0.125, 0.175</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>11.35</td>
<td>2 F</td>
<td>2</td>
<td>0.125</td>
<td>0.05</td>
</tr>
<tr>
<td>11.40</td>
<td>4 F</td>
<td>1</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>
**T and W for Example 2**

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Start</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>10.0</td>
<td>10.4</td>
<td>0.4</td>
<td>1.33</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>10.2</td>
<td>11.35</td>
<td>1.15</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>10.4</td>
<td>10.65</td>
<td>0.25</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>10.5</td>
<td>11.4</td>
<td>0.9</td>
<td>2.25</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>10.8</td>
<td>11.1</td>
<td>0.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[ T = 0.6 \quad WT = 2.276 \]

Check:
Because CPU was never idle, \(1.4 + 10.0\) must equal time of last event (11.4)

---

**Scheduling Example 3**

Assume:

- FIFO Job Scheduling
- 100 K Main Memory
- 5 Tape Drives
- Processor Sharing Process Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
<th>Memory</th>
<th>Tapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>0.3</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10.2</td>
<td>0.5</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>10.4</td>
<td>0.1</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>10.5</td>
<td>0.4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>10.8</td>
<td>0.1</td>
<td>30</td>
<td>3</td>
</tr>
</tbody>
</table>
Example 3 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Hway</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>1 A,S</td>
<td>90</td>
<td>3</td>
<td>1</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>10.2</td>
<td>2 A,S</td>
<td>30</td>
<td>2</td>
<td>1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>10.4</td>
<td>1 F</td>
<td>40</td>
<td>4</td>
<td>2</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>10.5</td>
<td>4 A,S</td>
<td>30</td>
<td>2</td>
<td>2</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>10.8</td>
<td>5 A,H</td>
<td>30</td>
<td>2</td>
<td>2</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Example 3 Continued ...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>2 F</td>
<td>2</td>
<td>0.15</td>
<td>90</td>
<td>3</td>
<td>4 0.1</td>
</tr>
<tr>
<td></td>
<td>5 S</td>
<td></td>
<td>60</td>
<td>0</td>
<td>5</td>
<td>0.1</td>
</tr>
<tr>
<td>11.3</td>
<td>5 F</td>
<td>2</td>
<td>0.1</td>
<td>90</td>
<td>3</td>
<td>3 0.1</td>
</tr>
<tr>
<td></td>
<td>4 F</td>
<td></td>
<td>100</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 S</td>
<td></td>
<td>50</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.4</td>
<td>3 F</td>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### T and W for Example 3

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Arrives</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>10.0</td>
<td>10.4</td>
<td>0.4</td>
<td>1.33</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>10.2</td>
<td>11.1</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>10.4</td>
<td>11.4</td>
<td>1.0</td>
<td>10.0</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>10.5</td>
<td>11.3</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>10.8</td>
<td>11.3</td>
<td>0.5</td>
<td>5.0</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
T &= 0.72 \\
WT &= 4.026
\end{align*} \]

---

### Scheduling Example 4

Assume:

- FIFO Job Scheduling
- 100 K Main Memory
- 5 Tape Drives
- Processor Sharing
- Process Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
<th>Memory</th>
<th>Tapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.5</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>1.0</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1.3</td>
<td>1.5</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>2.0</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.7</td>
<td>0.5</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>1.0</td>
<td>30</td>
<td>2</td>
</tr>
</tbody>
</table>
### Example 4 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1 A,S</td>
<td>70</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>1.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>20</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>1.3</td>
<td>3 A,H</td>
<td>2</td>
<td>0.05</td>
<td>20</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>1.4</td>
<td>4 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>1.7</td>
<td>5 A,H</td>
<td>3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>2.0</td>
<td>1 F</td>
<td>3</td>
<td>0.1</td>
<td>30</td>
<td>2</td>
<td>2.0</td>
</tr>
</tbody>
</table>

### Example 4 Continued ...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>6 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>2.0</td>
</tr>
<tr>
<td>4.05</td>
<td>2 F</td>
<td>3</td>
<td>0.65</td>
<td>50</td>
<td>1</td>
<td>4.05</td>
</tr>
<tr>
<td>5.1</td>
<td>6 F</td>
<td>3</td>
<td>0.35</td>
<td>30</td>
<td>2</td>
<td>5.1</td>
</tr>
<tr>
<td>6.6</td>
<td>4 F</td>
<td>2</td>
<td>0.75</td>
<td>50</td>
<td>4</td>
<td>6.6</td>
</tr>
<tr>
<td>7.4</td>
<td>3 F</td>
<td>2</td>
<td>0.4</td>
<td>70</td>
<td>2</td>
<td>7.4</td>
</tr>
<tr>
<td>7.5</td>
<td>5 F</td>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td>5</td>
<td>7.5</td>
</tr>
</tbody>
</table>
### T and W for Example 4

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Arrives</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.2</td>
<td>4.05</td>
<td>2.85</td>
<td>2.85</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.3</td>
<td>7.4</td>
<td>6.1</td>
<td>4.06</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>1.4</td>
<td>6.6</td>
<td>5.2</td>
<td>2.6</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>1.7</td>
<td>7.5</td>
<td>5.8</td>
<td>11.6</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>2.1</td>
<td>5.1</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**T = 3.99  WT = 4.35**

T = \[3.99\]  WT = 4.35