Chapter 7

Process Scheduling

Process Scheduler

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

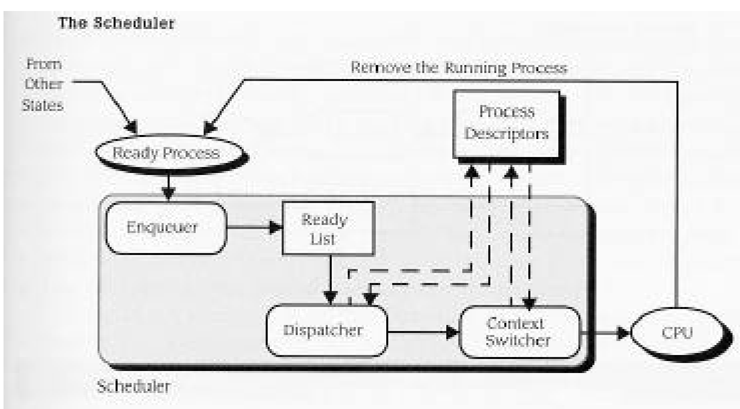
- Scheduling [Policy] addresses...
  - When to remove a process from CPU ?
  - Which ready process to allocate the CPU to ?

Context Switch

- Processes are switched out using <u>Context Switching</u>
- 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
CTX
Dispatcher : selects next process
CTX
Process 2 running

Scheduler



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 τ (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

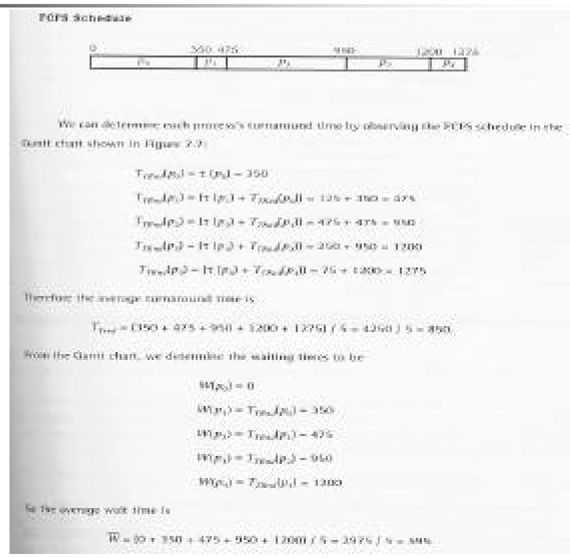
      Recall : Several ways to establish priority

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

FCFS...

i	$\tau(p_i)$
0	350
1	125
2	475
3	250
4	75



FCFS Schedule

We can determine each process's turnaround time by observing the FCFS schedule in the Gantt chart shown in Figure 2.7:

$$T_{TRnd}(p_0) = \tau(p_0) = 350$$
$$T_{TRnd}(p_1) = (\tau(p_1) + T_{TRnd}(p_0)) = 125 + 350 = 475$$
$$T_{TRnd}(p_2) = (\tau(p_2) + T_{TRnd}(p_1)) = 475 + 475 = 950$$
$$T_{TRnd}(p_3) = (\tau(p_3) + T_{TRnd}(p_2)) = 250 + 950 = 1200$$
$$T_{TRnd}(p_4) = (\tau(p_4) + T_{TRnd}(p_3)) = 75 + 1200 = 1275$$

Therefore the average turnaround time is

$$T_{TRnd} = (350 + 475 + 950 + 1200 + 1275) / 5 = 4250 / 5 = 850.$$

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

$$W(p_0) = 0$$
$$W(p_1) = T_{TRnd}(p_0) = 350$$
$$W(p_2) = T_{TRnd}(p_1) = 475$$
$$W(p_3) = T_{TRnd}(p_2) = 950$$
$$W(p_4) = T_{TRnd}(p_3) = 1200$$

So the average wait time is

$$\overline{W} = (0 + 350 + 475 + 950 + 1200) / 5 = 2975 / 5 = 595.$$

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 ???

SJN…..

i	$\tau(p_i)$
0	350
1	125
2	475
3	250
4	75

From the Gantt chart, we compute as follows:

$T_{TRnd}(p_0) = \tau(p_0) + \tau(p_3) + \tau(p_1) + \tau(p_4) = 350 + 250 + 125 + 75 = 800$

$T_{TRnd}(p_1) = \tau(p_1) + \tau(p_4) = 125 + 75 = 200$

$T_{TRnd}(p_2) = \tau(p_2) + \tau(p_0) + \tau(p_3) + \tau(p_1) + \tau(p_4) = 475 + 350 + 250 + 125 + 75 = 1275$

$T_{TRnd}(p_3) = \tau(p_3) + \tau(p_1) + \tau(p_4) = 250 + 125 + 75 = 450$

$T_{TRnd}(p_4) = \tau(p_4) = 75$

Therefore the average turnaround time is

$$T_{TRnd} = (800 + 200 + 1275 + 450 + 75) / 5 = 2800 / 5 = 560.$$

We determine the wait times to be

$$W(p_0) = 450$$
$$W(p_1) = 75$$
$$W(p_2) = 800$$
$$W(p_3) = 200$$
$$W(p_4) = 0.$$

So the average wait time is

$$\overline{W} = (450 + 75 + 800 + 200 + 0) / 5 = 1525 / 5 = 305.$$

Priority Scheduling

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

How might starvation occur ???

Priority Scheduling...

i	$\tau(p_i)$	PRIORITY
0	350	5
1	125	2
2	475	3
3	250	1
4	75	4

$T_{TRnd}(p_0) = \tau(p_0) + \tau(p_4) + \tau(p_2) + \tau(p_1) + \tau(p_3) = 350 + 75 + 475 + 125 + 250 = 1275$

$T_{TRnd}(p_1) = \tau(p_1) + \tau(p_3) = 125 + 250 = 375$

$T_{TRnd}(p_2) = \tau(p_2) + \tau(p_1) + \tau(p_3) = 475 + 125 + 250 = 850$

$T_{TRnd}(p_3) = \tau(p_3) = 250$

$T_{TRnd}(p_4) = \tau(p_4) + \tau(p_2) + \tau(p_1) + \tau(p_3) = 75 + 475 + 125 + 250 = 925$

Therefore the average turnaround time is

$$\bar{T}_{TRnd} = (1275 + 375 + 850 + 250 + 925) / 5 = 3675 / 5 = 735.$$

We determine the waiting times to be

$$W(p_0) = 925$$
$$W(p_1) = 250$$
$$W(p_2) = 375$$
$$W(p_3) = 0$$
$$W(p_4) = 850.$$

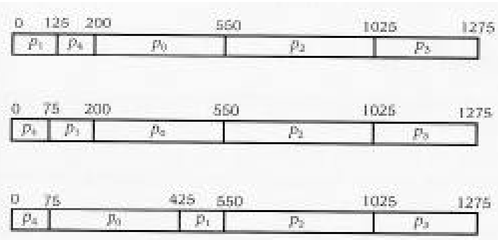
So the average wait time is

$$\bar{W} = (925 + 250 + 375 + 0 + 850) / 5 = 2400 / 5 = 480.$$

Deadline Scheduling

- Deadline Scheduling
  - Processes are scheduled to meet stated deadlines

i	$\tau(p_i)$	DEADLINE
0	350	575
1	125	550
2	475	1050
3	250	(none)
4	75	200



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 $\tau(P_{active}) <= \tau(P_{new})$, 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

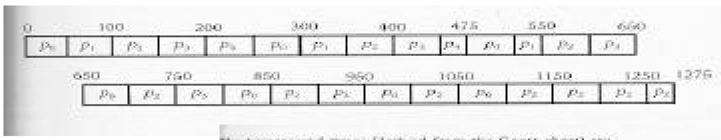
Preemptive Strategies - Round Robin

- 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 loose part of its quantum

Round Robin w/o considering CTX

Quantum = 50

i	$\tau(p_i)$
0	350
1	125
2	475
3	250
4	75



The turnaround times (derived from the Gantt chart) are

$$T_{TRnd}(p_0) = 1100$$
$$T_{TRnd}(p_1) = 550$$
$$T_{TRnd}(p_2) = 1275$$
$$T_{TRnd}(p_3) = 950$$
$$T_{TRnd}(p_4) = 475$$

Therefore the average turnaround time is

$$\overline{T}_{TRnd} = (1100 + 550 + 1275 + 950 + 475) / 5 = 4350 / 5 = 870.$$

Round Robin w/o considering CTX

Quantum = 50

i	$\tau(p_i)$
0	350
1	125
2	475
3	250
4	75

From the Gantt chart, we determine the wait times (the time until the process first acquires the processor) to be

$$W(p_0) = 0$$
$$W(p_1) = 50$$
$$W(p_2) = 100$$
$$W(p_3) = 150$$
$$W(p_4) = 200.$$

So the average wait time is

$$\overline{W} = (0 + 50 + 100 + 150 + 200) / 5 = 500 / 5 = 100.$$

Round Robin w/ considering CTX...

Quantum = 50

CTX = 10

i	$\tau(p_i)$
0	350
1	125
2	475
3	250
4	75

The turnaround times (derived from the Gantt chart) are

$$T_{TRnd}(p_0) = 1320$$
$$T_{TRnd}(p_1) = 660$$
$$T_{TRnd}(p_2) = 1535$$
$$T_{TRnd}(p_3) = 1140$$
$$T_{TRnd}(p_4) = 565.$$

Therefore the average turnaround time is

$$\overline{T}_{TRnd} = (1320 + 660 + 1535 + 1140 + 565) / 5 = 5220 / 5 = 1044.$$

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

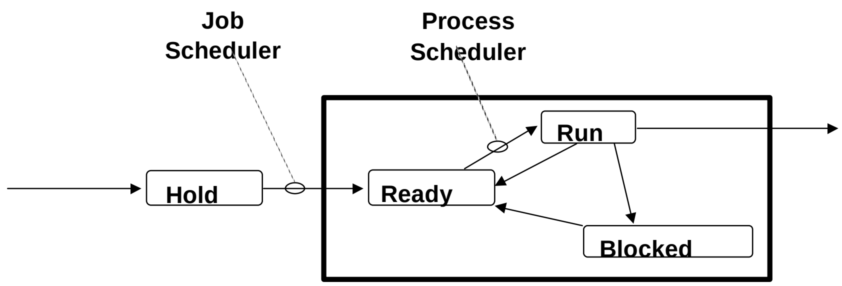
$$W(p_0) = 0$$
$$W(p_1) = 60$$
$$W(p_2) = 120$$
$$W(p_3) = 180$$
$$W(p_4) = 240.$$

So the average wait time is

$$\overline{W} = (0 + 60 + 120 + 180 + 240) / 5 = 600 / 5 = 120.$$

Job & Process Scheduling

Process Life Cycle



Dark square contains fixed, maximum number of processes

Job Scheduler & Process Scheduler

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

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_i be arrival time of i-th job

F_i be finish time of i-th job

Turnaround time for i^{th} job: $\mathbf{T_i = F_i - A_i}$

Average turnaround time for i^{th} job: $\mathbf{T = \Sigma T_i / N}$

Weighted turnaround time for i^{th} job:

$$\mathbf{WT_i = (F_i - A_i) / (Service\text{-}time)_i}$$

Average Weighted Turnaround time:

$$\mathbf{WT = \Sigma WT_i / N}$$

Job & Process Sched: Example 1

Assume

job arrival and runtimes as shown

Non-preemptive **process scheduling** (run to completion)

No I/O or Memory Constraints

Job	Arrives	Run Time
1	10.0	2.0
2	10.1	1.0
3	10.25	0.25

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

1) FIFO

2) Shortest Job First ?

Example 1 - FIFO Solution

Job	Arrives	Start	Finish	T_i	WT_i
1	10.0	___	___	____	_______
2	10.1	___	___	____	_______
3	10.25	___	___	____	_______

Average Turnaround = T = ____

Average Weighted Turnaround = WT = ____

Example 1 - FIFO Solution (completed)

Job	Arrives	Start	Finish	Turnaround
1	10.0	10.0	12.0	2.0
2	10.1	12.0	13.0	2.9
3	10.25	13.0	13.25	3.0
				7.9

Avg Turnaround time T = 2.63

Example 1 - SJF Solution

Job	Arrives	Start	Finish	Turnaround
1	10.0	___	___	____
2	10.1	___	___	____
3	10.25	___	___	____

Average Turnaround time T = ___

Example 1 - SJF Solution

Job	Arrives	Start	Finish	Turnaround
1	10.0	10.0	12.0	2.0
2	10.1	12.25	13.25	3.15
3	10.25	12.0	12.25	2.0

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.

Preemptive Process Sched. Algorithms…

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

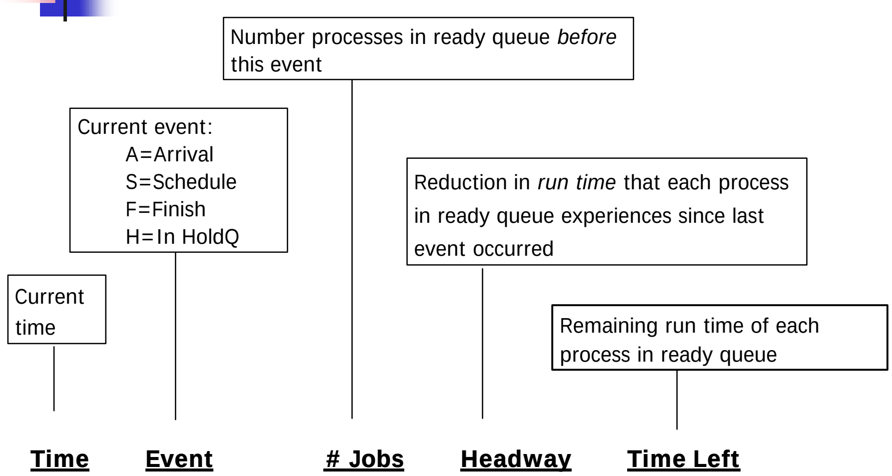
Assume:

Multiprogramming FIFO Job Scheduling

Processor Sharing Process Scheduling

Job	Arrives	Run Time
1	10.0	0.3
2	10.2	0.5
3	10.4	0.1
4	10.5	0.4
5	10.8	0.1

Definitions



Example 2 Continued

Time	Event	# Jobs	Headway	Time Left	
10.0	1 A,S			1	0.3
10.2	2 A,S	1	0.2	1	0.1
				2	0.5
10.4	1 F	2	0.1	2	0.4
	3 A,S			3	0.1
10.5	4 A,S	2	0.05	2	0.35
				3	0.05
				4	0.4
10.65	3 F	3	0.05	2	0.3
				4	0.35

Example 2 Continued...

Time	Event	# Jobs	Headway	Time Left	
10.8	5 A,S	2	0.075	2	0.225
				4	0.275
				5	0.1
11.1	5 F	3	0.1	2	0.125
				4	0.175
11.35	2 F	2	0.125	4	0.05
11.40	4 F	1	0.05		

T and W for Example 2

Job	Run	Start	Finish	Ti	WTi
1	0.3	10.0	10.4	0.4	1.33
2	0.5	10.2	11.35	1.15	2.3
3	0.1	10.4	10.65	0.25	2.5
4	0.4	10.5	11.4	0.9	2.25
5	0.1	10.8	11.1	0.3	3.0
	1.4			**3.0**	**11.38**

T = 0.6 **WT = 2.276**

Check:

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

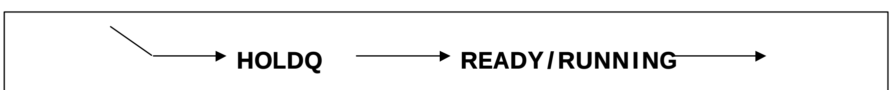
Scheduling Example 3

Assume:

FIFO Job Scheduling 100 K Main Memory

5 Tape Drives Processor Sharing Process Scheduling

Job	Arrives	Run Time	Memory	Tapes
1	**10.0**	**0.3**	**10**	**2**
2	**10.2**	**0.5**	**60**	**1**
3	**10.4**	**0.1**	**50**	**4**
4	**10.5**	**0.4**	**10**	**2**
5	**10.8**	**0.1**	**30**	**3**



Example 3 Continued

Time	Event	# Jobs	Hway	MM	Tapes	Time Left	
10.0	1 A,S			90	3	1	0.3
10.2	2 A,S	1	0.2	30	2	1	0.1
						2	0.5
10.4	1 F	2	0.1	40	4	2	0.4
	3 A,H						
10.5	4 A,S	1	0.1	30	2	2	0.3
						4	0.4
10.8	5 A,H	2	0.15	30	2	2	0.15
						4	0.25

Example 3 Continued ...

Time	Event	# Jobs	HWay	MM	Tapes	Time Left	
11.1	2 F	2	0.15	90	3	4	0.1
	5 S			60	0	5	0.1
11.3	5 F	2	0.1	90	3	3	0.1
	4 F			100	5		
	3 S			50	1		
11.4	3 F	1	0.1	100			

T and W for Example 3

Job	Run	Arrives	Finish	Ti	WTi
1	0.3	10.0	10.4	0.4	1.33
2	0.5	10.2	11.1	0.9	1.8
3	0.1	10.4	11.4	1.0	10.0
4	0.4	10.5	11.3	0.8	2.0
5	0.1	10.8	11.3	0.5	5.0
				3.6	**20.13**

T = 0.72 WT = 4.026

Scheduling Example 4

Assume:

FIFO Job Scheduling 100 K Main Memory

5 Tape Drives Processor Sharing Process Scheduling

Job	Arrives	Run Time	Memory	Tapes
1	1.0	0.5	30	2
2	1.2	1.0	50	1
3	1.3	1.5	50	1
4	1.4	2.0	20	2
5	1.7	0.5	30	3
6	2.1	1.0	30	2

Example 4 Continued

Time	Event	# Jobs	HWay	MM	Tapes	Time Left
1.0	1 A,S			70	3	1 0.5
1.2	2 A,S	1	0.2	20	2	1 0.3
						2 1.0
1.3	3 A,H	2	0.05	20	2	1 0.25
						2 0.95
1.4	4 A,S	2	0.05	0	0	1 0.2
						2 0.9
						4 2.0
1.7	5 A,H	3	0.1	0	0	1 0.1
						2 0.8
						4 1.9
2.0	1 F	3	0.1	30	2	2 0.7
						4 1.8

Example 4 Continued ...

Time	Event	# Jobs	HWay	MM	Tapes	Time Left
2.1	6 A,S	2	0.05	0	0	2 0.65
						4 1.75
						6 1.0
4.05	2 F	3	0.65	50	1	4 1.1
	3 S			0	0	6 0.35
						3 1.5
5.1	6 F	3	0.35	30	2	4 0.75
						3 1.15
6.6	4 F	2	0.75	50	4	3 0.4
	5 S			20	1	5 0.5
7.4	3 F	2	0.4	70	2	5 0.1
7.5	5 F	1	0.1	100	5	

T and W for Example 4

Job	Run	Arrives	Finish	Ti	WTi
1	0.5	1.0	2.0	1.0	2.0
2	1.0	1.2	4.05	2.85	2.85
3	1.5	1.3	7.4	6.1	4.06
4	2.0	1.4	6.6	5.2	2.6
5	0.5	1.7	7.5	5.8	11.6
6	2.1	2.1	5.1	3.0	3.0
				23.95	**26.11**

T = 3.99 WT = 4.35