Types of Schedulers

This chart shows the function of each of the three types of schedulers (long-term, short-term, and medium-term) for each of three types of operating systems (batch, interactive, and real-time).

	batch	interactive	real-time
long-term	job admission based on characteristics and resource needs	sessions and processes normally accepted unless capacity reached	processes either permanent or accepted at once
medium-term	usually none—jobs remain in storage until done	processes swapped when necessary	processes never swapped
short-term	processes scheduled by priority; continue until wait voluntarily, request service, or are terminated	processes scheduled on rotating basis; continue until service requested, time quantum expires, or pre-empted	scheduling based on strict priority with immediate preemption; may time-share processes with equal priorities

Process Scheduling Algorithms

In this handout, there are 5 processes:

1. Process *A* arrives at time 0 and takes 10 time units to execute;
2. Process *B* arrives at time 1 and takes 29 time units to execute;
3. Process *C* arrives at time 2 and takes 3 time units to execute;
4. Process *D* arrives at time 3 and takes 7 time units to execute; and
5. Process *E* arrives at time 4 and takes 12 time units to execute.

Each section shows how several process scheduling algorithms would execute the processes.

First Come First Serve (FCFS)

This policy services processes in the order they start.

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	10	10	0	1.0
B	1	29	10	39	38	9	1.3
C	2	3	39	42	40	37	13.3
D	3	7	42	49	46	39	6.6
E	4	12	49	61	57	45	4.8
mean					**38**	**26**	**5.4**

In what follows, the number in parentheses in the comment field is the remaining service time for the process. In order of execution:

time	ready queue	comments
0	*A*	*A*(10) arrives, runs
1	*AB*	*B*(29) arrives, *A*(9) is *not* interrupted and continues to run
2	*ABC*	*C*(3) arrives and is appended to the queue; again, *A*(8) continues to run
3	*ABCD*	*D*(7) arrives and is appended to the queue; again, *A*(7) continues to run
4	*ABCDE*	*E*(12) arrives and is appended to the queue; again, *A*(6) continues to run
10	*BCDE*	*A* finishes, *B*(29) runs
39	*CDE*	*B* finishes, *C*(3) runs
42	*DE*	*C* finishes, *D*(7) runs
49	*E*	*D* finishes, *E*(12) runs
61	—	*E* finishes

Shortest Process Next (SPN)

This policy services the process with the shortest service time next. It is sometimes also called "shortest job next" (SJN).

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	10	10	0	1.0
B	1	29	32	61	60	31	2.1
C	2	3	10	13	11	8	3.7
D	3	7	13	20	17	10	2.4
E	4	12	20	32	28	16	2.3
mean					**25**	**13**	**2.3**

In what follows, the number in parentheses in the comment field is the remaining service time for the process. In order of execution:

time	ready queue	comments
0	*A*	*A*(10) arrives, runs
1	*AB*	*B*(29) arrives, *A*(9) continues to run
2	*ABC*	*C*(3) arrives and is appended to the queue; again, *A*(8) continues to run
3	*ABCD*	*D*(7) arrives and is appended to the queue; again, *A*(7) continues to run
4	*ABCDE*	*E*(12) arrives and is appended to the queue; again, *A*(6) continues to run
10	*BCDE*	*A* finishes; *C*(3) has the shortest service time, so it runs
13	*BDE*	*C* finishes, *D*(7) has the shortest service time, so it runs
20	*BE*	*D* finishes, *E*(12) runs
32	*B*	*E* finishes, *B*(29) runs
61	—	*B* finishes

Preemptive Shortest Process Next (PSPN)

This policy services the process with the shortest service time next. It is sometimes also called "preemptive shortest job next" (PSJN).

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	2	*preempted by C*		
		8	12	20	20	10	2.0
B	1	29	32	61	60	31	2.1
C	2	3	2	5	3	0	1.0
D	3	7	5	12	9	2	1.3
E	4	12	20	32	28	16	2.3
mean					**24**	**12**	**1.7**

In what follows, the number in parentheses in the comment field is the remaining service time for the process. In order of execution:

time	ready queue	comments
0	*A*	*A*(10) arrives, runs
1	*AB*	*B*(29) arrives; as its service time is greater than that of *A*(9), *B* is appended to the queue and *A* continues to run
2	*CAB*	*C*(3) arrives; as its service time is less than that of *A*(8), *C* runs. *A*'s service time is less than that of *B*(29), so it goes before *B* in the queue
3	*CDAB*	*D*(7) arrives; as its service time is greater than that of *C*(2), *D* is placed in the queue at the appropriate place, and *C* continues to run
4	*CDAEB*	*E*(12) arrives; as its service time is greater than that of *C*(1), *E* is placed in the queue at the appropriate place, and *C* continues to run
5	*DAEB*	*C* finishes; *D*(7) has the shortest remaining service time, so it runs
12	*AEB*	*D* finishes, *A*(8) has the shortest remaining service time, so it runs
20	*EB*	*A* finishes, *E*(12) has the shortest remaining service time, so it runs
32	*B*	*E* finishes, *B*(29) has the shortest remaining service time, so it runs
61	—	*B* finishes

Highest Response Ratio Next (HRRN)

This policy services the process with the greatest (highest) response ratio next.

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	10	10	0	1.0
B	1	29	32	61	60	31	2.1
C	2	3	10	13	11	8	3.7
D	3	7	13	20	17	10	2.4
E	4	12	20	32	28	16	2.3
mean					**25**	**13**	**2.3**

In order of execution:

1. At time 0, process *A* runs for 10 time units, then terminates. At this time:
   - Process *B*'s response ratio is $\frac{(10-1)+29}{29} = 1.3$;
   - Process *C*'s response ratio is $\frac{(10-2)+3}{3} = 3.6$;
   - Process *D*'s response ratio is $\frac{(10-3)+7}{7} = 2.0$; and
   - Process *E*'s response ratio is $\frac{(10-4)+12}{12} = 1.5$.

   so process *C* runs.
2. At time 10, process *C* runs for 3 time units, then terminates. At this time:
   - Process *B*'s response ratio is $\frac{(13-1)+29}{29} = 1.4$;
   - Process *D*'s response ratio is $\frac{(13-3)+7}{7} = 2.4$; and
   - Process *E*'s response ratio is $\frac{(13-4)+12}{12} = 1.7$.

   so process *D* runs.
3. At time 13, process *D* runs for 7 time units, then terminates. At this time:
   - Process *B*'s response ratio is $\frac{(20-1)+29}{29} = 1.6$; and
   - Process *E*'s response ratio is $\frac{(20-4)+12}{12} = 2.3$.

   so process *E* runs.
4. At time 20, process *E* runs for 12 time units, then terminates. At this time:
   - Process *B*'s response ratio is $\frac{(32-1)+29}{29} = 2.0$.

   so process *B* runs.
5. At time 32, process *B* runs for 29 time units, then terminates.

Round Robin (RR)

This policy services the processes with a fixed-size quantum, which in this example is 5.

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	5	*end of quantum; B starts*		
		5	23	28	28	18	2.8
B	1	29	5	10	*end of quantum; C starts*		
		24	28	33	*end of quantum; D starts*		
		19	40	45	*end of quantum; E starts*		
		14	47	61	60	31	2.1
C	2	3	10	13	11	8	3.7
D	3	7	13	18	*end of quantum; E starts*		
		2	33	35	32	25	4.6
E	4	12	18	23	*end of quantum; A starts*		
		7	35	40	*end of quantum; B starts*		
		2	45	47	43	31	3.5
mean					**35**	**23**	**3.3**

In what follows, the number in parentheses in the comment field is the remaining service time for the process. In order of execution:

time	ready queue	comments
0	*A*	*A*(10) arrives, runs
1	*AB*	*B*(29) arrives and is appended to the queue, *A*(9) continues to run
2	*ABC*	*C*(3) arrives and is appended to the queue, *A*(8) continues to run
3	*ABCD*	*D*(7) arrives and is appended to the queue, *A*(7) continues to run
4	*ABCDE*	*E*(12) arrives and is appended to the queue, *A*(6) continues to run
5	*BCDEA*	The quantum expires, so *A*(5) moves to the end of the queue and *B*(29) runs
10	*CDEAB*	The quantum expires, so *B*(24) moves to the end of the queue and *C*(3) runs
13	*DEAB*	*C* finishes, so *D*(7) runs
18	*EABD*	The quantum expires, so *D*(2) moves to the end of the queue and *E*(12) runs
23	*ABDE*	The quantum expires, so *E*(7) moves to the end of the queue and *A*(5) runs
28	*BDE*	*A* finishes, so *B*(24) runs
33	*DEB*	The quantum expires, so *B*(19) moves to the end of the queue and *D*(2) runs
35	*EB*	*D* finishes, so *E*(7) runs
40	*BE*	The quantum expires, so *E*(2) moves to the end of the queue and *B*(19) runs
45	*EB*	The quantum expires, so *B*(14) moves to the end of the queue and *E*(2) runs
47	*B*	*E* finishes, so *B*(14) runs
52	*B*	The quantum expires, so *B*(9) moves to the end of the queue and continues to runs
57	*B*	The quantum expires, so *B*(4) moves to the end of the queue and continues to runs
61	—	*B* finishes

Multilevel Feedback Queues (MLFB)

The variant of this class of scheduling algorithms that is shown here uses three levels:

- Processes at level 1 are scheduled round robin; the relevant quantum is 2, and when a quantum expires the job is moved to level 2.
- Processes at level 2 are scheduled round robin; the quantum is 4, and processes are allowed 2 quanta before being moved to level 3.
- Processes at level 3 are serviced first come first serve.

The processes *A*, *B*, *C*, *D*, and *E* have been augmented by *F*, a 1-second job arriving at time 13, and *G*, an 11-second job arriving at time 50. These are to demonstrate that quanta are usually not interrupted.

process name	arrival time	service time	start time	finish time	turnaround time	waiting time	response ratio
A	0	10	0	2	*end of quantum; B starts*		
		8	10	14	*end of quantum; F starts*		
		4	28	32	32	22	3.2
B	1	29	2	4	*end of quantum; C starts*		
		27	15	19	*end of quantum; C starts*		
		23	32	36	*end of quantum; D starts*		
		19	41	60	59	30	2.0
C	2	3	4	6	*end of quantum; D starts*		
		1	19	20	18	15	6.0
D	3	7	6	8	*end of quantum; E starts*		
		5	20	24	*end of quantum; E starts*		
		1	36	37	34	27	4.9
E	4	12	8	10	*end of quantum; A starts*		
		10	24	28	*end of quantum; A starts*		
		6	37	41	*end of quantum; B starts*		
		2	70	72	68	56	5.7
F	13	1	14	15	2	1	2.0
G	50	11	60	70	*end of quantum; E starts*		
		1	72	73	23	12	2.1
mean					**33.7**	**23.3**	**3.7**

In what follows, the number in parentheses in the comment field is the remaining service time for the process. In order of execution:

time	level 1	level 2	level 3	comments
0	*A*	–	–	*A*(10) arrives, runs
1	*AB*	–	–	*B*(29) arrives, *A* continues quantum
2	*BC*	*A*	–	*C*(3) arrives, *A*'s quantum expires (8), moves to level 2, *B* runs
3	*BCD*	*A*	–	*D*(7) arrives, *B* continues quantum
4	*CDE*	*AB*	–	*E*(12)arrives, *B*'s quantum expires (27), moves down, *C* runs
6	*DE*	*ABC*	–	*C*'s quantum expires (1), moves down, *D* runs
8	*E*	*ABCD*	–	*D*'s quantum expires (5), moves down, *E* runs
10	–	*ABCDE*	–	*E*'s quantum expires (10), moves down, *A* runs from level 2 (level 1 is empty)
13	*F*	*ABCDE*	–	*F*(1) arrives, *A*'s quantum continues
14	*F*	*ABCDE*	–	*A*'s quantum expires (4), *F* runs (at level 1)
15	–	*ABCDE*	–	*F* finishes, *B* runs from level 2 (level 1 is empty)
19	–	*ABCDE*	–	*B*'s quantum expires (23), *C* runs
20	–	*ABDE*	–	*C* finishes, *D* runs
28	–	*ABDE*	–	*E*'s quantum expires (6), *A* runs
32	–	*BDE*	–	*A* finishes, *B* runs
36	–	*DE*	*B*	*B*'s quantum expires (19), moves down, *D* runs
37	–	*E*	*B*	*D* finishes, *E* runs
41	–	–	*BE*	*E*'s quantum expires (2), moves down, *B* runs from level 3 (since there is nothing in higher levels)
50	*G*	–	*BE*	*G* arrives(11), *B* continues to run
60	*G*	–	*E*	*B* finishes, *G* runs (since it is in the highest level)
62	–	*G*	*E*	*G*'s quantum expires (9), moves down, *G* runs from level 2
66	–	*G*	*E*	*G*'s quantum expires (5), *G* runs
70	–	–	*EG*	*G*'s quantum expires (1), moves down, *E* runs
72	–	–	*G*	*E* finishes, *G* runs
73	–	–	–	*G* finishes

Fair Share Scheduler

A *fair share scheduler* is used when CPU time is to be divided equally between groups of processes. For this scheduling algorithm, processes are divided into groups based upon external factors. Such factors include the organizational divisions of the owners of the computer, or classes of customers, or other criteria.

For example, suppose group A has 1 process, group B has 2 processes, group C has 3 processes, and group D has 4 processes. Under a regular scheduler, each of the 10 processes would get 10% of the CPU. Under a fair share scheduler, each of the 4 groups would get 25% of the CPU.

Example

Suppose there are 3 processes. Process p_1 is in group A, and processes p_2 and p_3 are in group B. The following formula assigns process p_i a priority P_i:

$$P_i = \frac{p_i\text{'s recent CPU usage}}{2} + \frac{p_i\text{'s group CPU usage}}{2}$$

In addition, a decay function decrements the current CPU usage of all processes. This ”spreads out” the priority of the processes in the ready queue. The decay D_i for p_i is:

$$D_i = \frac{p_i\text{'s recent CPU usage}}{2}$$

In this system, the lower the numerical value of P_i, the higher the priority of process p_i.

The following shows how processes execute, given a quantum of 60 ticks. All arithmetic is integer arithmetic, and the decay function is applied after the most recent CPU time is added in, but before the priorities are computed.

First 60-Tick Interval

At the beginning of this interval, all priorities are equal, so the process to run is chosen randomly. Say p_1 is selected to run. It runs, and at the end of the interval, its CPU usage is updated to 60. The group CPU usage for group A, to which p_1 belongs, also is updated to 60. The decay function is then applied, cutting both to 30. The CPU usage for p_2 and p_3, and for group B, are 0, so the decay function does not change them. The priority P_1 of p_1 becomes

$$P_1 = \frac{p_1\text{'s recent CPU usage}}{2} + \frac{p_1\text{'s group CPU usage}}{2} = \frac{30}{2} + \frac{30}{2} = 15 + 15 = 30$$

Second 60-Tick Interval

At the beginning of this interval, P_2 and P_3 are equal, and both are less than P_1, so either p_2 or p_3 will run. Say p_2 is selected to run. It runs, and at the end of the interval, its CPU usage is updated to 60. The group CPU usage for group B, to which p_2 belongs, also is updated to 60. The decay function is then applied, cutting both to 30. It also cuts the CPU usage of p_1 to 15, and the group CPU usage of group A to 15. The CPU usage for p_3 is 0, so the decay function does not change it. The priorities become

$$\begin{aligned}
P_1 &= \frac{p_1\text{'s recent CPU usage}}{2} + \frac{p_1\text{'s group CPU usage}}{2} = \frac{15}{2} + \frac{15}{2} = 7 + 7 = 14 \\
P_2 &= \frac{p_2\text{'s recent CPU usage}}{2} + \frac{p_2}{2} = \frac{30}{2} + \frac{30}{2} = 15 + 15 = 30 \\
P_3 &= \frac{p_3\text{'s recent CPU usage}}{2} + \frac{p_3\text{'s group CPU usage}}{2} = \frac{0}{2} + \frac{30}{2} = 0 + 15 = 15
\end{aligned}$$

Third 60-Tick Interval

At the beginning of this interval, P_1 is less than P_2 or P_3, so p1 runs. At the end of the interval, its CPU usage is updated to 15 + 60 = 75. The group CPU usage for group A, to which p_1 belongs, is similarly updated to 15 + 60 = 75. The decay function is then applied, cutting both to 37. It also cuts the CPU usage of p_2 to 15, and the group CPU usage of group B to 15. The CPU usage for p_3 is 0, so the decay function does not change it. The priorities become

$$
\begin{aligned}
P_1 &= \frac{p_1\text{'s recent CPU usage}}{2} + \frac{p_1\text{'s group CPU usage}}{2} = \frac{37}{2} + \frac{37}{2} = 18 + 18 = 36 \\
P_2 &= \frac{p_2\text{'s recent CPU usage}}{2} + \frac{p_2\text{'s group CPU usage}}{2} = \frac{15}{2} + \frac{15}{2} = 7 + 7 = 14 \\
P_3 &= \frac{p_3\text{'s recent CPU usage}}{2} + \frac{p_3\text{'s group CPU usage}}{2} = \frac{0}{2} + \frac{15}{2} = 0 + 7 = 7
\end{aligned}
$$

Fourth 60-Tick Interval

At the beginning of this interval, P_3 is less than P_1 or P_2, so p_3 runs. At the end of the interval, its CPU usage is updated to 0 + 60 = 60. The group CPU usage for group B, to which p_2 belongs, is similarly updated to 15 + 60 = 75. The decay function is then applied, cutting p_3's CPU usage to 30 and the group CPU usage to 37. It also cuts the CPU usage of p1 to 18, the CPU usage of p_2 to 7, and the group CPU usage of group A to 18. The priorities become

$$
\begin{aligned}
P_1 &= \frac{p_1\text{'s recent CPU usage}}{2} + \frac{p_1\text{'s group CPU usage}}{2} = \frac{18}{2} + \frac{18}{2} = 9 + 9 = 18 \\
P_2 &= \frac{p_2\text{'s recent CPU usage}}{2} + \frac{p_2\text{'s group CPU usage}}{2} = \frac{7}{2} + \frac{37}{2} = 3 + 18 = 21 \\
P_3 &= \frac{p_3\text{'s recent CPU usage}}{2} + \frac{p_3\text{'s group CPU usage}}{2} = \frac{30}{2} + \frac{37}{2} = 15 + 18 = 33
\end{aligned}
$$

Summary Table

This table summarizes the first 8 seconds. The figures shown are for after the ticks and after the calculations of priorities. The usages are after the decays.

	priorities			CPU usage			group usage		
ticks	P_1	P_2	P_3	p_1	p_2	p_3	A	B	runs
0	0	0	0	0	0	0	0	0	A
60	30	0	0	30	0	0	30	0	B
120	14	30	15	15	30	0	15	30	A
180	36	14	7	37	15	0	37	15	C
240	18	21	33	18	7	30	18	37	A
300	38	10	16	39	3	15	39	18	B
360	18	34	22	19	31	7	19	39	A
420	38	16	10	39	15	3	39	19	C
...	...	...	...	...	...	...	...	...	...