

5. Process and thread scheduling

5.1 Organization of Schedulers

- Embedded and Autonomous Schedulers
- Priority Scheduling

5.2 Scheduling Methods

- A Framework for Scheduling
- Common Scheduling Algorithms
- Comparison of Methods

5.3 Priority Inversion

5.4 Multiprocessor and Distributed Scheduling

ICS 143

1

Process and thread scheduling

- process scheduling
 - long term scheduling
 - move process to Ready List after creation
 - when and in which order
- dispatching
 - short term scheduling
 - select process from Ready List to run
- we use “scheduling” to refer to both

ICS 143

2

Organization of schedulers

- embedded
 - called as function at end of kernel call
 - runs as part of calling process
- autonomous
 - separate process
 - may have dedicated CPU on a multiprocessor
 - on single-processor run at every quantum:
 - scheduler and other processes alternate

Figure 5-1

ICS 143

3

Priority scheduling

- priority function returns numerical value P for process p :

$$P = \text{Priority}(p)$$

- static priority: unchanged for lifetime of p
- dynamic priority: changes at runtime
- priority divides processes into levels
 - implemented as multi-level RL
 - p at $RL[i]$ run before q at $RL[j]$ if $i > j$
 - p, q at same level ordered using other criteria

ICS 143

4

An embedded scheduler

```
Scheduler() {
do {
  Find highest priority ready_a process p;
  Find a free cpu;
  if (cpu != NIL) Allocate_CPU(p,cpu);
} while (cpu != NIL);
do {
  Find highest priority ready_a process p;
  Find lowest priority running process q;
  if (Priority(p) > Priority(q)) Preempt(p,q);
} while (Priority(p) > Priority(q));
if (self->Status.Type!='running') Preempt(p,self);
}
```

ICS 143

5

Scheduling methods

- when is scheduler invoked?
 - decision mode
 - preemptive: scheduler caller periodically (quantum-oriented) or when system state changes
 - non-preemptive: scheduler caller when process terminates or blocks
- how does it select highest priority process?
 - priority function: $P = \text{Priority}(p)$
 - arbitration rule: break ties
 - random
 - chronological (FIFO)
 - cyclic (round robin)

ICS 143

6

Priority function

- possible parameters:
 - attained service time (a)
 - real time in system (r)
 - total service time (t)
 - period (d)
 - deadline (explicit or implied by period)
 - external priority (e)
 - memory requirements (mostly for batch)
 - system load (not process-specific)

ICS 143

7

Scheduling algorithms

Name, Decision mode, Priority, Arbitration

FIFO: nonpreemptive P = r random

SJF: nonpreemptive P = -t chron./random

SRT: preemptive P = -(t-a) chron./random

RR: preemptive P = 0 cyclic

ML: preemptive P = e cyclic

nonpreemptive P = e chronological

- n fixed priority levels
- level P is serviced when n through P+1 empty

ICS 143

8

Scheduling algorithms

MLF:

- similar to ML but priority changes dynamically
- every process enters at highest level n
- each level P prescribes maximum time t_p
- t_p increases as P decreases
- typically:

$$t_n = T \quad (\text{constant})$$

$$t_p = 2 \times t_{p+1}$$

Figure 5-3

ICS 143

9

Scheduling algorithms

MLF priority function:

find $P = n-i$ for given a :

priority	attained time
n	$a < T$
$n-1$	$a < T+2T$
$n-2$	$a < T+2T+4T$
\dots	\dots
$n-i$	$a < (2^{i+1}-1)T$

- find smallest i such that $a < (2^{i+1}-1)T$:
- solve for i : $i = \lfloor \lg_2(a/T+1) \rfloor$
- $P = n-i = n - \lfloor \lg_2(a/T+1) \rfloor$

ICS 143

10

Scheduling algorithms

RM:

- intended for periodic (real-time) processes
- preemptive
- highest priority: shortest period: $P = -d$

EDF:

- intended for periodic (real-time) processes
- preemptive
- highest priority: shortest time to next deadline
 - $r + d$ number of completed periods
 - $r \% d$ time in current period
 - $d - r \% d$ time remaining in current period
 - $P = -(d - r \% d)$

ICS 143

11

Comparison of methods

• FIFO, SJF, SRT:

- primarily for batch systems
 - FIFO simplest
 - SJF/SRT have better average turnaround times
- $$(r_1+r_2+\dots+r_m)/n$$
- $r = \text{waiting} + t$

Example: (Figure 5-2)

- FIFO: $((0+5) + (3+2))/2 = 5$
- SRT: $((2+5) + (2+0))/2 = 4.5$

ICS 143

12

Comparison of methods

- time-sharing systems
 - response time is critical
 - RR or MLF with RR within each queue are suitable
 - choice of quantum determines overhead
 - when $q \rightarrow \infty$, RR approaches FIFO
 - when $q \rightarrow 0$, context switch overhead $\rightarrow 100\%$
 - when $q \gg$ context switch overhead, n processes run concurrently at $1/n$ CPU speed

ICS 143

13

Comparison of methods

- real-time systems
 - feasible schedule: all deadline are met
 - CPU utilization is defined as:
$$U = \sum_{i=1}^n t_i/d_i$$
 - schedule is feasible if $U \leq 1$
 - EDF always yields feasible schedule (if $U \leq 1$)
 - RM yields feasible schedule if $U \leq 0.7$
- Example: Figure 5-9

ICS 143

14

Priority inversion problem

Figure 5-10

- unrelated process p2 may hold up a higher-priority process p1 indefinitely

ICS 143

15

Priority inversion problem

- solutions
 - always run CS at priority of highest process p that shares the CS
 - problem: p cannot interrupt lower-priority process inside CS -- a different form of priority inversion
 - dynamic priority inheritance:
 - p3 is in CS
 - $\text{priority}(p1) > \text{priority}(p3)$
 - p1 attempts to enter CS
 - ⇒ p3 inherits p1's priority for the duration of CS

Figure 5-11

ICS 143

16
