CS 241 Section Week #4 (02/18/10)

Topics This Section

- MP #2
- MP #3
- 5-state Model
- Review of Scheduling
- Problems

MP #2

MP #2

- qsort()
  - Define your own comparison function
  - int compare (const void *e1, const void *e2)

```
{
    return (-1) *
        strcasecmp(*(const char**)e1, *(const char**)e2);
}
```

  - Sorts string in reverse order.

MP #2

- Merge
  - Consider two sorted lists (integers for example):
    - { 2, 8, 11, 17, 32 }
    - { 4, 12, 17, 34, 57}
  - It's possible to merge them together (and remove duplicates) in one algorithm.
    - Start a pointer at beginning of both lists
    - Compare the current element of the lists
      - If one element is less than the other, print out the desired output and advance the pointer on that list.
      - If the elements are equal, print out either element and advance the pointer of both lists.
    - O(n), rather than O(n lg(n)) of qsort()

MP #2

- pthreads
  - Running threads in parallel was required:
    - INCORRECT:

```
for (i = 1; i < argc; i++)
{
    pthread_create(…);
    pthread_join(…)
}
```

    - CORRECT:

```
for (i = 1; i < argc; i++)
    pthread_create(…);

for (i = 1; i < argc; i++)
    pthread_join(…)
```

MP #3

MP3 Forward

In MP3, you will add code to a simulator for a CPU scheduler.

- We provide you with the code for the simulator.
  - You don't need to understand this code to understand this MP.
  - You should consider the simulator a 'black box'
- You need to implement these algorithms:
  fcfs, sjf, psjf, pri, ppri, rr#

MP3 Forward

- You need to fill in 3 scheduling functions:
  - scheduler_new_job()
  - scheduler_job_finished()
  - scheduler_quantum_expired()

  Note that these are the only times that the scheduler needs to make a decision!
- A clean_up() function to clean up any memory your program may've allocated
- A show_queue() function to help you debug your program
- You need to create your own job queue

MP3 Forward

- You also need to fill in 3 statistics functions:
  - float scheduler_average_response_time()
  - float scheduler_average_waiting_time()
  - float scheduler_average_turnaround_time()

  These are called at the end of the simulation.

MP3 Forward

- How the functions are called…
  - The simulator runs on discrete time units. Every time unit will always execute in the following way:
    - (1): Is there a new job arriving at the current time?
      - If so, call _new_job()
    - (2): Run the job currently scheduled
    - (3): Did the current job finish scheduling?
      - If so, call _job_finished()
    - (4): If the job did not finish, are we in Round Robin and has the quantum expired?
      - If so, call _quantum_expired()

MP3 Forward

- You can find sample input/output file in the examples/ directory.
  - Input File: proc1.csv
  - Example Output:
    - proc1-fcfs.out
    - proc1-sjf.out
    - …
- Total of two sample outputs, covers many test cases, possibly not all (try out some of your own test cases!).

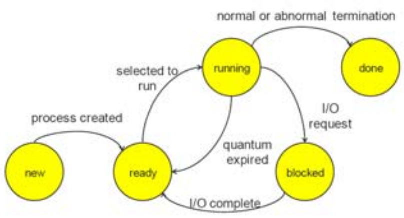
MP3 Forward

- For success on this MP:
  - Carefully read README.txt for details!
  - Look at the example runs and compare your results (e.g. using 'diff')!
- This MP is harder than all previous MPs!!
- Requires a good understanding of data structures, scheduling, and pointers all in one MP!

Good luck!

Five State Model

5-State Model - Transitions



Review of Scheduling

Scheduling

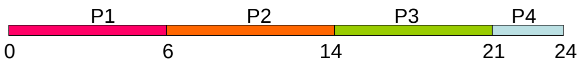
- The CPU Scheduler decides which thread should be in the running state. It is called when:
  - A thread is created or finishes
  - A clock interrupt occurs
  - An I/O interrupt occurs
  - A thread yields

Scheduling

- The algorithms that we usually talk about are:
  - First-Come First-Serve (FCFS)
  - Shortest Job First (SJF)
  - Priority
  - Round Robin (RR)

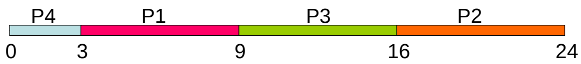
FCFS Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



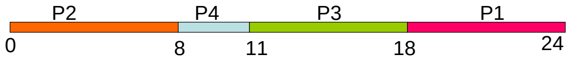
SJF Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



Priority Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



RR(1) Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0

Quanta = 1 time unit

Scheduling

- Scheduling algorithms can be preemptive or non-preemptive
  - Non-preemptive: each thread chooses when to yield to the processor (e.g. when done or system call)
  - Preemptive: scheduler forces the thread to yield (e.g. time quantum expires in RR)

Scheduling

- Metrics for a single job
  - Response Time = time from job submission until it's running for the first time
  - Waiting Time = total time that the job is not running but queued
  - Turnaround Time = time between the job's entry and completion

Problems

Problem #1

Job	Duration	Priority #
J1	6	2
J2	4	1
J3	5	1

These three jobs are going to arrive at our scheduler 1 time unit apart from each other (i.e. one job at time 0, one at time 1, and one at time 2), but the order hasn't been decided yet.

Problem #1

We want to guarantee that the jobs finish in the order

J1 then J2 then J3

Problem #1

Which arrival order(s) guarantee this if the scheduler uses:

1) FCFS?
2) non-premptive SJF?
3) preemptive SJF? (use remaining time, and ties are broken by arrival time)
4) RR-1? (arriving jobs are placed on ready queue immediately)
5) non-preemptive priority?
6) preemptive priority?

Problem #1 Solution

Job Time Priority
1 6 2
2 4 1
3 5 1

1. FCFS: 1 2 3

2. n-p-SJF: 1 2 3 (1 needs to arrive first, then 2 will
1 3 2 beat 3)

3. p-SJF: 1 3 2 (1 needs to arrive first, and tie breaks
will give it control; then 2 beats 3)

4. RR-1: 1 3 2 (1 needs to run the longest, then 3, then 2,
and it barely works out)

5. n-p-Prio: 1 2 3 (1 must come first because of low prio,
and then 2 and 3 must follow in order)
6. p-Prio: none (2 or 3 will always preempt 1)

Problem #2

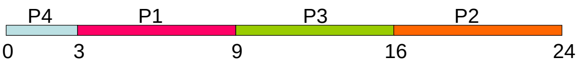
For the SJF and RR examples, calculate:

1) Average response time
2) Average waiting time
3) Average turnaround time

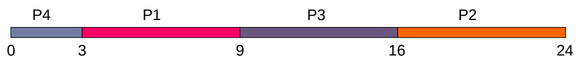
Are either of these clearly better?
When would you use each?

SJF Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



Metrics for Non-preemptive:Shortest Job First



P4 waiting time: 0
P1 waiting time: 3
P3 waiting time: 9
P2 waiting time: 16

Average response time (ART):
(0+3+9+16)/4 = 7
Average waiting time (AWT):
(0+3+9+16)/4 = 7
Average turnaround time (ATT):
(3+9+16+24)/4 = 13

P4 turnaround time: 3
P1 turnaround time: 9
P3 turnaround time: 16
P2 turnaround time: 24

RR(1) Example

Process	Duration	Priority	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0

Quanta = 1 time unit

Metrics for Round Robin Example

P1 waiting time: 13
P2 waiting time: 16
P3 waiting time: 16
P4 waiting time: 9

Average response time (ART):
(0 + 1 + 2 + 3) / 4 = 1.5
Average waiting time (AWT):
(13+16+16+9)/4 = 13.5
Average turnaround time (ATT):
(19+24+23+12)/4 = 19.5

P1 turnaround time: 19
P2 turnaround time: 24
P3 turnaround time: 23
P4 turnaround time: 12