

CS162 – Section # 6, 03/04/2003

Rodrigo Fonseca

Announcements

Code due Thursday

- Don't be tempted to use many slip days for this phase!
- Submit 'proj1-final-design' up to two days later!
(no slip for this! Penalty is automatic!)
- Fill in 'Group Evaluation Form' on the website, up to 2 days later!
(mandatory, there will be a penalty for not doing this)

Midterm: everything up to Lecture 13 (in the reader)

- Lecture Notes and Lectures (up to next Monday)
- Related readings (as per *Schedule*)
- Project information

Newsgroup: avoid repeated questions or vacuous posts

Watch out for Midterm Review Session

CVS Quick Reference Card: <http://www.refcards.com/about/cvs.html>

Today: Scheduling and Address Spaces

CPU Scheduling

Goals:

- Response time
- Throughput
- Fairness

Policies:

FIFO:

keep CPU until thread blocks
first come first serve

- + simple
- + optimum for equal size jobs
- may have short jobs wait for long jobs

Round Robin

Time-slice (preemption, external events)

- + fair (equal division of CPU)
- context switch overhead (unnecessary context switches)
- not good when jobs are the same length

Maximize Throughput: get rid of short jobs first

SJF and SRTF (non-preemptive and preemptive)

optimum average response time
becomes FIFO when equal job lengths

- + optimum
- can't predict the future
- can lead to starvation (arriving short jobs versus a long job)

Adaptative Policies

Use past to predict future

Multilevel feedback queues, UNIX (give more priority to hasn't gotten service)

Lottery Scheduler

For the savvy,

http://www.usenix.org/publications/library/proceedings/osdi/full_papers/waldspurger.pdf

Simple

Flexible

No starvation

Can give short jobs more tickets, long jobs less tickets

Example question:

Process Name	Arrival Time	Processing Time
1	0	4
2	1	4
3	4	3
4	8	2

Show when processes will run, and determine average response times for FIFO, RR, SRTF. Assume RR quantum is 1 unit of time, and that arriving jobs get to run first in RR.

Address Spaces

Problem: share memory among several processes

Protection

Controlled communication

Should also be flexible

Protection

Hardware support

Address translation and User-kernel mode

Without HW support

Strong typing

Compiler help, software isolation

Address translation

CPU generates addresses in a 'virtual space'

MMU (hardware) translates every reference to a 'physical address', an actual location in main memory

Protection controlled by the translation box

But need to protect the translation data

Processor runs in two modes: privileged and unprivileged modes

In general: user programs use virtual (translated) addresses

OS uses direct access to physical memory (untranslated)

New issues:

Communicating between processes

Switching between user and kernel modes

Communicating between user and kernel (system calls)