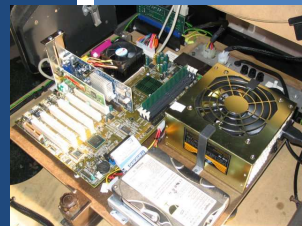


TCSS 422: OPERATING SYSTEMS

Three Easy Pieces: Condition Variables

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma



April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

OBJECTIVES

- Assignment 2 – Matrix Task Processor
- Condition Variables – Ch. 30
- Practice Midterm

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.2


FEEDBACK – 4/23

- Where should locks be implemented?
 - Lock-based data structures

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.3



CHAPTER 30 –
CONDITION VARIABLES

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.4

CONDITION VARIABLES

- There are many cases where a thread wants to wait for another thread before proceeding with execution
- Consider when a precondition must be fulfilled before it is meaningful to proceed ...

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.5

CONDITION VARIABLES - 2

- Support a signaling mechanism to alert threads when preconditions have been satisfied
- Eliminate busy waiting
- Alert one or more threads to “consume” a result, or respond to state changes in the application
- Threads are placed on an **explicit queue** (FIFO) to wait for signals
- **Signal**: wakes one thread
broadcast wakes all (ordering by the OS)



April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.6

CONDITION VARIABLES - 3

■ Condition variable

```
pthread_cond_t c;
```

- Requires initialization

■ Condition API calls

```
pthread_cond_wait(pthread_cond_t *c, pthread_mutex_t *m); // wait()
pthread_cond_signal(pthread_cond_t *c); // signal()
```

■ wait() accepts a mutex parameter

- Releases lock, puts thread to sleep

■ signal()

- Wakes up thread, awakening thread acquires lock

April 30, 2018

TCCS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.7

CONDITION VARIABLES - QUESTIONS

■ Why would we want to put waiting threads on a queue... why not use a stack?

- Queue (FIFO), Stack (LIFO)
- Using condition variables eliminates busy waiting by putting threads to “sleep” and yielding the CPU.

■ Why do we want to not busily wait for the lock to become available?

- A program has 10-threads, where 9 threads are waiting. The working thread finishes and broadcasts that the lock is available. What happens next?

April 30, 2018

TCCS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.8

MATRIX GENERATOR

Matrix generation example

Chapter 30
signal.c

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.9

MATRIX GENERATOR

- The main thread, and worker thread (generates matrices) share a single matrix pointer.
- What would happen if we don't use a condition variable to coordinate exchange of the lock?
- Let's try "nosignal.c"

April 30, 2018

TCSS422: Operating Systems [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.10

