Lecture 17

CprE 308

February 19, 2013

Intro

Review

- Producer Consumer using Semaphores
- Condition Variable

Today's Topics

- Readers/Writer Problem

Review

Review: Producer Consumer using Semaphores

Shared Variables

- count (number of items in buffer)
- buffer
- N (maximum size of buffer)

Semaphores

- Empty - semaphore initialized to N (number of free slots in buffer)
- Full - semaphore initialized to zero (number of items in buffer)

Review: Producer Consumer using Semaphores (Example)

Producer

```
while(TRUE) {
  item = produce();
  down(Empty);
  lock(mutex);
  insert(item,buffer);
  count++;
  unlock(mutex);
  up(Full);
```

Consumer

```
while(TRUE) {
  down(Full);
  lock(mutex);
  item = remove(buffer);
  count--;
  unlock(mutex);
  up(Empty);
  consume(item);
}
```

Review: Condition Variables

- `pthread_cond_t condition_variable`
- `pthread_mutex_t mutex;`

Waiting Thread

```
pthread_mutex_lock(&mutex);
while(!cond. satisfied) {
  pthread_cond_wait(
    &condition_variable,
    &mutex);
}
pthread_mutex_unlock(
  &mutex);
```

Signaling Thread

```
pthread_mutex_lock(&mutex);
/* change variable value */
if(cond. satisfied) {
  pthread_cond_signal(
    &condition_variable);
}
pthread_mutex_unlock(
  &mutex);
```

Review: Solved using condition variables

Global

```
int thread1_done = 0;
pthread_cond_t cv;
pthread_mutex_t mutex;
```

Review: Solved using condition variables

Global

```
int thread1_done = 0;
pthread_cond_t cv;
pthread_mutex_t mutex;
```

Thread 1

```
printf("hello");
pthread_mutex_lock(&mutex);
thread1_done = 1;
pthread_cond_signal(&cv);
pthread_mutex_unlock(
  &mutex);
```

Review: Solved using condition variables

Global

```
int thread1_done = 0;
pthread_cond_t cv;
pthread_mutex_t mutex;
```

Thread 1

```
printf("hello");
pthread_mutex_lock(&mutex);
thread1_done = 1;
pthread_cond_signal(&cv);
pthread_mutex_unlock(
  &mutex);
```

Thread 2

```
pthread_mutex_lock(&mutex);
while(thread1_done == 0) {
  pthread_cond_wait(
    &cv, &mutex);
}
printf(" world\n");
pthread_mutex_unlock(
  &mutex);
```

Reader/Writers Problem

Readers/Writers Problem

- Multiple threads reading/writing a database
- Many threads can read simultaneously
- Only one can be writing at any time
    - When a writer is executing, nobody else can read or write

Readers-Writers Problem

- Database has multiple threads operating
  - Many threads can read simultaneously
  - Only one can be writing at a time
  - When a writer is executing, nobody else can read or write
- Example: Multithreaded web server cache
- Performance Problems with naive solution

Solution Idea

- Readers:
  - First reader locks the database
  - If a reader inside, other readers enter without locking again
  - Checking for readers occurs within a mutex
- Writer:
  - Always lock database before entering

Readers-Writers

Reader

```
while(1) {
  down(&protector)
  rc++;
  if(rc == 1) // first
    down(&database);
  up(&protector);
  read_data();
  down(&protector);
  rc--;
  if(rc == 0) then // last
    up(&database);
  up(&protector);
}
```

Initialized Variables

- Two semaphores:
  - Database (init 1)
  - Protector (init 1)
- rc $= 0$

Readers-Writers

Reader

```
while(1) {
  down(&protector)
  rc++;
  if(rc == 1) // first
    down(&database);
  up(&protector);
  read_data();
  down(&protector);
  rc--;
  if(rc == 0) then // last
    up(&database);
  up(&protector);
}
```

Initialized Variables

- Two semaphores:
  - Database (init 1)
  - Protector (init 1)
- rc = 0

Writer

```
while(1) {
  generate_data();
  down(&database);
  write_data();
  up(&database);
}
```

Writer Starvation

- Readers might continuously enter while a writer waits
- Writer Priority Solution?
    - What does it mean to give writer priority?
- What is a fair solution?
    - Give a specification

Message Passing

Message Passing

- No shared variables
- Two primitives:
    - `send(destination,message)`
    - `receive(destination,message)`
    - Usually blocks till a message arrives
    - Non-blocking version also usually available

Issues

- Across different machines, message passing is the real thing
- Many issues to consider:
    - Marshaling data into messages
    - Provide reliable transmission across unreliable links?
    - Event-driven mode of programming
- Computer Networking: deals with sending messages across machines