Figure 1. Mutual exclusion using critical regions.

(from Tanenbaum’s Modern Operating Systems, page 122)

What was bad about locks?

- spin
- too low level

How do condition variables help?

- semaphore
- wait()
- signal()

Caveats

- message passing

What is the programmer in charge of figuring out?

- wait
- make progress
- locked/unlocked
- signal

The programmer can cause lots of problems if this isn’t done correctly, such as starvation or deadlock. We’ll focus on those issues in the next set of notes.
What is a condition variable?

An example (from the book) – have parent create child thread and then wait for it:

```c
int done = 0;
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t c = PTHREAD_COND_INITIALIZER;

void thr_exit()
{
    Pthread_mutex_lock(&m);
    done = 1;
    Pthread_cond_signal(&c);
    Pthread_mutex_unlock(&m);
}

void *child(void *arg)
{
    printf("child\n");
    thr_exit();
    return NULL;
}

void thr_join()
{
    Pthread_mutex_lock(&m);
    while (done == 0)
    {
        Pthread_cond_wait(&c, &m);
        Pthread_mutex_unlock(&m);
    }
}

main()
{
    printf("parent: begin\n");
    pthread_t p;
    Pthread_create(&p, NULL, child, NULL);
    thr_join();
    printf("parent: end\n");
    return 0;
}
```