04 Mar 2025 - Operating Systems - II

Monitors ⇝ locks and condition variables in C++.

Fig 30.2 ⇝ problem → spin ⇝ in some cases, it's okay (which cases, see silberschatz)

Monitors → sleep

Main problem: How to wait for a condition

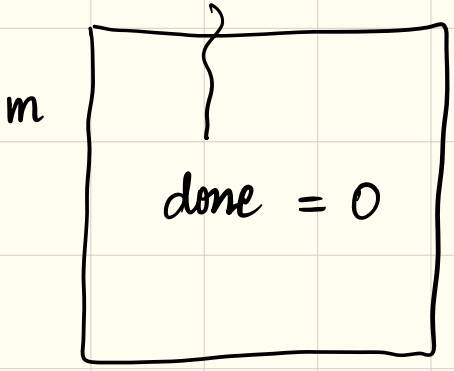
30.3 → pthread_mutex_t m ⇝ lock

thr_join()

```
while (done == 0)        // as long as not done
    pthread_cond_wait (&c, &m);
```

↓ release the lock and put the thread to sleep

Analogy ⇝ monitors




```
thr_join ( ) {
19   pthread_mutex_lock (&m);
20   while (done == 0)
21       pthread_cond_wait (...);
22   pthread_mutex_unlock (&m);
}
```

① Why state variable? ✓ Deadlock

② Why lock? ✓ Race cond^n

③ Why while (?)

child calls cond_signal before main thread waits

Deadlock

needed for more than 2 children

Fig 30.5

Tip: Always hold lock while signaling

Producer / Consumer problem using condition variables.

---

05 Mar 2025

* assert (count == 1) → if not true, abort

How is it different from crashes?

→ graceful exit.

assert (int expression)

↳ if

cond_t ≡ pthreads_cond_t

mutex_t ≡ pthreads_mutex_t

Fig 30.8

count ~ global

while (count == 1)

// p5 → wakes up waiting consumers ← producers wake up and sleep

pthread_cond_signal( ) ⇝ wakes up one thread

Problem in If ⇝ 1 producers, 2 consumers

Debugging tip ⟶ for each thread create a log file.

You are the master of your code.

Documentation is important

* signal_all