HW - 2.3 - HW 23

From

No unread replies.No replies.

Questions: Tannenbaum & Bos 2.{23,24,30,36}

**23.** Does the busy waiting solution using the *turn* variable (Fig. 2-23) work when the two

processes are running on a shared-memory multiprocessor, that is, two CPUs sharing a

common memory?

No, it violates the 3rd condition to have a good solution. No running process outside its critical region may block any process. Taking turns would have one loop waiting on the other when one process takes more time than another.

**24.** Does Peterson’s solution to the mutual-exclusion problem shown in Fig. 2-24 work

when process scheduling is preemptive? How about when it is nonpreemptive?

With preemptive scheduling it takes turns until a certain amount and frees cpu. In nonpreemptive scheduling, the turn may never change causing an infinite loop.

**30.** Consider the following solution to the mutual-exclusion problem involving two processes

*P0* and *P1*. Assume that the variable turn is initialized to 0. Process *P0*’s code is

presented below.

/\* Other code \*/

while (turn != 0) { } /\* Do nothing and wait. \*/

Critical Section /\* . . . \*/

turn = 0;

/\* Other code \*/

For process *P1*, replace 0 by 1 in above code. Determine if the solution meets *all* the

required conditions for a correct mutual-exclusion solution.

This also violate the 3rd condition. No running process outside its critical region may block any process. The loops alternate entering critical regions can hang up causing infinite loops.

**36.** A fast-food restaurant has four kinds of employees: (1) order takers, who take customers’

orders; (2) cooks, who prepare the food; (3) packaging specialists, who stuff the

food into bags; and (4) cashiers, who give the bags to customers and take their money.

Each employee can be regarded as a communicating sequential process. What form of

interprocess communication do they use? Relate this model to processes in UNIX.

There are two forms of IPC, direct and indirect communication. Each job is like a semaphore design in UNIX. It passes its work along and moves on. This keeps the workers busy instead of just standing there until all the little jobs are complete.