1999 OS Foundations Solution Paper 11

p11q6 JMB

a) WAIT(lock) / SIGNAL(lock) } bracket code executed under mutual exclusion.

WAIT(spaces) in producer ← SIGNAL(spaces) in consumer } condition synchronisation

SIGNAL(items) in producer → WAIT(items) in consumer } condition synchronisation

The system as presented may deadlock since a producer may block on WAIT(spaces) when the buffer is full while holding the lock & the consumer may block on WAIT(items) when the buffer is empty while holding the lock. A process that 8 wants use the buffer & one of these processes will block on lock.

b) Monitor - entry procedures are executed under exclusion
use condition variables
notfull, notempty : condition.

producer calls
```
if count = N then WAIT(notfull);
insert item; count := count + 1;
SIGNAL(notempty)
```

consumer calls
```
if count = 0 then WAIT(notempty)
remove item
count := count - 1
SIGNAL(notfull)
```

- count tested under exclusion so no race condition between test & WAIT.
- implementation frees monitor lock on WAIT(cond-var)
- no value of cond-var so can SIGNAL freely. 12
- implementation ensures one active process in monitor after SIGNAL

(20)