OPERATING SYSTEMS

Professor: Hungwen Li

HOMEWORK 4

Name: Jeyanthh Venkatachari Ravikumar SJSU ID: 010720528

5.6 Binary Semaphores uses only two states, either 1 or 0 which can signify either locked or unlocked.
Mutual exclusion among n processes can be achieved using binary semaphores using the wait and signal commands. Wait () being executed before entering the critical section in case it is currently used. And Signal() used to signal that the thread is out of the critical section.
S for each process entering the critical section, it checks the state and allows to enter the critical section, later it signals its completion.
So when n processes share a semaphore, mutex is initialized to 1. Each process Pi…n is organized as follows:

```
do {
wait(mutex);
/ critical section /
signal(mutex);
/ remainder section /
} while (true);
```

5.7 consider the following scenario

Step 1: Register 1: current balance amount (100)
Step 2: Register 2: withdraw amount (100)
Step 3: Register 3: deposit account (100)
Step 4: Register 2 : Register 2- Register 1 =0(In case of amount withdrawal)

Step 5: Register 3: Register 3 + Register 1=100 (In case of amount deposit)

Here In step 4, the local value of husband (register 2) will be zero, but before the transaction is committed, if the deposit takes place, the local value will be 100.

Thus causing ambiguity. To avoid this. The execution of statements must be atomic.

5.10

If the user programs in the single processor systems are given the privilege to disable interrupts,

It will be possible for the user program to disable the timer interrupt which is processor.

The system clock is updated based on timer interrupts. The timer interrupts are necessary if there are scheduling mechanisms in place to context switch between one process and other based on the quantum time allocated, disabling them will lead to loss of data corresponding to the to the quantum time used by each process leading to a loss of efficiency in context switching.

5.11

Disabling interrupts in an multi-processor system is time consuming as message should passed to each processor in the system, which will lead to delays in a process trying to enter an critical section reducing its efficiency.

Also disabling interrupts in a processor will lead to restriction of other processes in an multi-processor system in accessing the processor , but we are not aware what are the other processes being executed, thus mutual exclusion may or may not be satisfied.

5.13 A race condition exists when several process try to access and modify a set of code, but the outcome depends on the order in which the processes are executed.

Example 1: consider there is process accessing I/O and a new process waiting for I/O device, waiting for the current running process to be context switched out.But during the time at which the process in the I/O has finished running , and another new process is requesting a I/O, this will lead to race condition between the waiting two process in the I/O queue.

Example 2: Consider a data structure maintaining a list open files in a system and the list is modified when a new file is opened or closed. In this case if two files where opened at the same time the separate updates to the list could result in race conditions.