David Liu

CSS 430

CSS-430 : Operating Systems : HW04-Questions

Assignment Text

Complete the following problems from the OSC book, 10th edition:

- Problem 5.2
- Problem 5.11
- Problem 6.7
- Problem 6.10
- Problem 6.17

5.2

Question: Explain the difference between preemptive and nonpreemptive scheduling.

The difference is that preemptive scheduling if foreseen and was scheduled for limited time while nonpreemptive scheduling has CPU allocated until termination.

5.11

Question: Of these two types of programs:

a. I/O-bound

b. CPU-bound

which is more likely to have voluntary context switches, and which is more likely to have nonvoluntary context switches? Explain your answer.

I/O requires more voluntary context switches since it uses more resources for I/O operations.
CPU bound use non-voluntary context switches for computations within CPU more than the I/O processor.

6.7

Question: The pseudocode of Figure 6.15 illustrates the basic push() and pop() operations of an array-based stack. Assuming that this algorithm could be used in a concurrent environment, answer the following questions:

a. What data have a race condition?
    a. Top will face a race condition since it can be interchangeably affected by push and pop.
b. b. How could the race condition be fixed?
    a. The race condition can be fixed with using semaphores or mutex locks kind of like dead locks in databases.

```
push(item) {
  if (top < SIZE) {
    stack[top] = item;
    top++;
  } else
    ERROR
  }

pop() {
  if (!is empty()) {
    top--;
    return stack[top];
  } else
    ERROR
  }

is.empty() {
  if (top == 0)
    return true;
  else
    return false;
}
```

6.10

Question: The compare and swap() instruction can be used to design lock-free data structures such as stacks, queues, and lists. The program example shown in Figure 6.17 presents a possible solution to a lock-free stack using CAS instructions, where the stack is represented as a linked list of Node elements with top representing the top of the stack. Is this implementation free from race conditions?

No, as far as I can see it will run into the same problem. The summation of elements can be prematurely called to create race conditions. In figure 6.17, we can see that index 7 is used to make a sum 3 times while using it's previous value which could be prematurely called if we don't wait for all the summations to be finished within any given row.

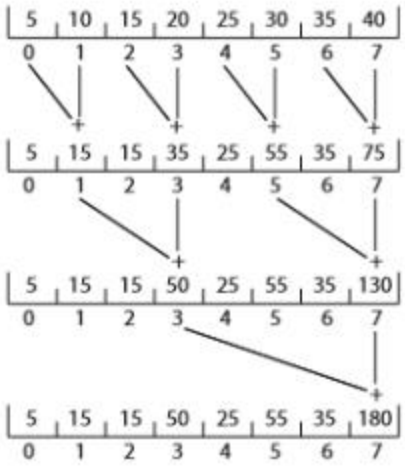

Figure 6.17 Summing an array as a series of partial sums for Exercise 6.14.

6.17

Question: Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.

Interrupts messes with the mutual exclusive access to any given program state and can limit/hinder other processes from executing on that processor.