CS330 - Quiz 5

Total: 15 marks Duration: 25 mins

open books and notes, no mobile phones, friends disconnected during exam

Be precise, no marks for vague answers. Make reasonable assumptions.

Roll No:

Q1. Select the correct answer(s) for the following questions. No partial marks. (2×4)

(i) Consider a file system using a block layer disk cache with delayed write semantics to store disk blocks in memory to improve application performance. Assume that M_B represents the memory page containing the contents of block B. Given that the file system provides APIs to perform both direct I/O and cached I/O, which of the following statement(s) are true?

(a) M_B must be written back to the disk when the cached block is evicted.

(b) M_B must be written back to disk if it is dirty (modified in memory).

(c) In some cases, M_B is required to be written back to the disk even when it is not modified.

(d) M_B is written back to the disk if and only if it is modified.

(e) None of the above.

Ans: e

(ii) In addition to information provided in (i), consider that B is mapped to K^{th} block of a file F (regular file with no links). Which of the following statement(s) are true?

(a) M_B can be in memory even before K^{th} file block offset is read.

(b) M_B can be in memory even before K^{th} file block offset is read for the first time after mounting the FS.

(c) M_B can be in memory even when the file is closed.

(d) M_B can be in memory even when the file is renamed.

(e) M_B can be in memory even after many write operations on F.

Ans: {a, b, c, d, e}

(iii) Execution of *fsck* utility for a given file system before mount revealed a mismatch between the number of data blocks calculated from the data block bitmap and all inode data pointers. The possible reason(s) *can be* a FS crash

(a) during creation of an empty file.

(b) during creation of an empty directory.

(c) during truncation of a file.

(d) during deletion of a file or directory.

(e) during unmount of the file system.

Ans: {a, b, c, d, e}

(iv) Which of the following statements are true to support a multi-threaded process on a multi-processor system? (Assume gemOS like design on a multi-processor system)

(a) Concurrent execution of expand() system call *must not* be allowed.

(b) The expand() system call handler is required to use locks.

(c) The expand() system call handler is required to use locks and disable interrupts.

(d) Page fault exception handler must use locks.

(e) Page fault exception handler must use locks and disable interrupts.

Ans: {b, d}

Q2. $2 \times 3.5 = 7$

Consider the following code segment executed from two threads.

```
static int var = 0, ctr = 0;
func()
{
    for(ctr=0; ctr<1000; ctr++){
         var++;               // Translates to a single instruction INC
                              // Increments var using three implicit micro-operations
    }
}
```

Assume that, the compiler translates the C statement `var++` to a single instruction—`INC (&var)`. If the code is executed concurrently by two threads to completion, what is the possible minimum and maximum value of `var` in a uniprocessor system? What will be the output in a multiprocessor system?

Uniprocessor system
Minimum: **1000**. Explanation: If both threads set ctr to zero in the beginning and before one checks the condition, the other loops 1000 times, setting var and ctr to 1000 and quit. The first thread's condition will fail and quit. Given that for any thread, ctr++ happens only after the corresponding var++, the 1000 increments to var is inevitable.

Maximum: There are two possible answers assuming the behavior of `ctr++` followed by `ctr<1000`.
Steps: { *1) mov (ctr), R; 2) add 1, R; 3) mov R, (ctr); 4) cmp R, 1000*}
Ans = **501500**. Explanation: Assume that T2 executes the loop 1000 times and gets deschduled before updating ctr and performing the loop condition check. T1 is scheduled and executes the loop only once and gets deschduled before executing the third instruction. T2 resumes execution and reads the value of ctr (which is 0), updates it to 1 and iterates the loop 999 times before getting deschduled. This process continues.
$var = 1000 + 1 + 999 + 1..... + 1 + 1$
$= (1000 * 1003)/2 = 501500$
There is another possibility where the steps are as follows,
Steps: { *1) mov (ctr), R; 2) add 1, R; 3) mov R, (ctr); 4) cmp (ctr), 1000*}
In this case the answer will be: **502500**. You can verify this performing similar analysis

Multiprocessor system
Minimum = **1**. Explanation: T0 (on a very slow core) and T1 (on a highs-peed core) read count into their register, T1 finishes all iterations and sets the value to 1000 (ctr=1000). T0 sets value to 1 and terminates.
Maximum is same as uniprocessor system.
Note that, answers with loop count = 100 will be accepted.