Unit 13 - Week 12

Course outline

How does an NPTEL online course work?

Week 1

Week 2

Week 3

Week 4

Week 5

Week 6

Week 7

Week 8

Week 9

Week 10

Week 11

Week 12

- ● Cash Recovery and Logging
- ○ Logging in Linux ext3 filesystem
- ● Protection and Security
- ○ Scheduling Policies
- ● Lock-free multiprocessor coordination, Read-Copy-Update
- ○ Microkernel, Exokernel, Multikernel
- ○ **Quiz : Assignment 12**
- ○ Week 12 Feedback Form

Assignment Solution

Download Videos

Text Transcripts

Assignment 12

The due date for submitting this assignment has passed.
As per our records you have not submitted this assignment.

Due on 2020-04-22, 23:59 IST.

1) xsyncfs (external synchrony) is useless when used with a Journaling Filesystem like ext3. *2 points*
True/False

○ True
○ False

No, the answer is incorrect.
Score: 0

Accepted Answers:
False

2) What is **dangling pointer** in C language? *2 points*

○ if pointer is pointing to a memory location from where variable has been deleted
○ if pointer is assigned to more than one variable
○ if pointer is not defined properly
○ none of above

No, the answer is incorrect.
Score: 0

Accepted Answers:
if pointer is pointing to a memory location from where variable has been deleted

3) Consider a simple checkpointing protocol and the following set of operations in the log. *2 points*

```
(start, T4); (write, T4, y, 2, 3); (start, T1); (commit, T4); (write, T1, z, 5, 7);
(checkpoint);
(start, T2); (write, T2, x, 1, 9); (commit, T2); (start, T3); (write, T3, z, 7, 2);
```

If a crash happens now and the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list

○ Undo: T3, T1; Redo: T2
○ Undo: T3, T1; Redo: T2, T4
○ Undo: none; Redo: T2, T4, T3; T1
○ Undo: T3, T1, T4; Redo: T2

No, the answer is incorrect.
Score: 0

Accepted Answers:
Undo: T3, T1; Redo: T2

4) During crash recovery (e.g., fsck), we can be sure that the inconsistencies in the filesystem data structure can only be of certain types. How is this ensured in the following situations *2 points*

□ Situation: Write-through buffer cache
Solution: Ordering of writes. E.g., during create, the child should be created before updating parent pointer.
□ Situation: Write-back buffer cache
Solution: Ordering of writes with soft updates. In this case, the cache is flushed to disk in update order as before. In case of cyclic dependency, one of the operations is rolled back to remove cycle and applied later.
□ Situation: Write-through buffer cache
Solution: Ordering of writes with soft updates. In this case, the cache is flushed to disk in update order as before. In case of cyclic dependency, one of the operations is rolled back to remove cycle and applied later.
□ Situation: Write-back buffer cache
Solution: Ordering of writes. E.g., during create, the child should be created before updating parent pointer.

No, the answer is incorrect.
Score: 0

Accepted Answers:
Situation: Write-through buffer cache
Solution: Ordering of writes. E.g., during create, the child should be created before updating parent pointer.
Situation: Write-back buffer cache
Solution: Ordering of writes with soft updates. In this case, the cache is flushed to disk in update order as before. In case of cyclic dependency, one of the operations is rolled back to remove cycle and applied later.

5) On x86, the TLB gets flushed on each context-switch between two distinct VMs (on a VMM). *2 points*
True/False

○ True
○ False

No, the answer is incorrect.
Score: 0

Accepted Answers:
True