COMP3231/9201/3891/9283 Operating Systems 2020/T1

UNSW

Tutorial Week 7

Questions and Answers

Files and file systems

1. Consider a file currently consisting of 100 records of 400 bytes. The filesystem uses *fixed blocking*, i.e. one 400 byte record is stored per 512 byte block. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one record, the following conditions hold. In the contiguous-allocation case, assume that there is no room to grow at the beginning, but there is room to grow at the end of the file. Assume that the record information to be added is stored in memory.

    a. The record is added at the beginning.
    b. The record is added in the middle.
    c. The record is added at the end.
    d. The record is removed from the beginning.
    e. The record is removed from the middle.
    f. The record is removed from the end.

	Contiguous	Linked	Indexed
a.	100r/101w	0r/1w	0r/1w
b.	50r/51w	50r/2w	0r/1w
c.	0r/1w	100r/2w	0r/1w
d.	0r/0w	1r/0w	0r/0w
e.	49r/49w	50r/1w 51r/1w	0r/0w
f.	0r/0w	99r/1w	0r/0w

---

2. Old versions of UNIX allowed you to write to directories. Newer ones do not even allow the superuser to write to them? Why? Note that many unices allow you read directories.

    To prevent total corruption of the fs. eg `cat /dev/zero > /`

---

3. Why is there VFS Layer in Unix?

    - It provides a framework to support multiple file system types concurrently without requiring each file system to be aware of other file system types.

- Provides transparent access to all supported file systems including network file systems (e.g. NFS, CODA)
- It provides a clean interface between the file system independent kernel code and the file system specific kernel code.
- Provide support for *special* file system types like `/proc`.

---

4. How does choice of block size affect file system performance. You should consider both sequential and random access.

   - Sequential Access

     The larger the block size, the fewer I/O operations required and the more contiguous the disk accesses. Compare loading a single 16K block with loading 32 512-byte blocks.

   - Random Access

     The larger the block size, the more unrelated data loaded. Spatial locality of access can improve the situation.

---

5. Why does Linux pre-allocate up to 8 blocks on a write to a file.

   Pre-allocating provides better locality when many writes to independent files are interleaved.

---

6. Linux uses a *buffer cache* to improve performance. What is the drawback of such a cache? In what scenario is it problematic? What alternative would be more appropriate where a buffer cache is inappropriate?

   The buffering writes in the buffer cache provides the opportunity for data to be lost if the system stops prior to the cache being flushed.

   Removable storage devices are particular problematic if users don't "unmount" them first.

   Robustness can be improved by using a write-through cache at the expense of poor write performance.

---

7. What is the structure of the contents of a directory? Does it contain attributes such as creation times of files? If not, where might this information be stored?

   - See lecture slides.
   - No, directories only have a name-to-inode mapping
   - Attributes of the file are stored in the inode itself.

---

8. The Unix inode structure contains a reference count. What is the reference count for? Why can't we just remove the inode without checking the reference count when a file is deleted?

   Inodes contain a reference count due to hard links. The reference count is equal to the number of directory entries that reference the inode. For hard-linked files, multiple directory entries reference a single inode. The inode must not be removed until no directory entries are left (ie, the reference count is 0) to ensure that the filesystem remains consistent.

---

9. Inode-based filesystems typically divide a file system partition into *block groups*. Each block group consists of a number of contiguous physical disk blocks. Inodes for a given block group are stored in the same physical location as the block groups. What are the advantages of this scheme? Are they any disadvantages?

    - Each group contains a redundant superblock. This make the file system more robust to disk block failures.
    - Block groups keep the inodes physically closer to the files they refer to than they would be (on average) on a system without block groups. Since accessing and updating files also involves accessing or updating its inode, having the inode and the file's block close together reduces disk seek time, and thus improves performance. The OS must take care that all blocks remain within the block group of their inode.

---

10. Assume an inode with 10 direct blocks, as well as single, double and triple indirect block pointers. Taking into account creation and accounting of the indirect blocks themselves, what is the largest possible number of block reads and writes in order to:

    a. Read 1 byte
    b. Write 1 byte

    Assume the inode is cached in memory.

    a. To write 1 byte, in the worst case:

        - 4 writes: create single indirect block, create double indirect block, create triple indirect block, write data block.
        - 3 reads, 2 writes: read single indirect, read double indirect, read triple indirect, write triple indirect, write data block
        - Other combinations are possible

    b. To read 1 byte, in the worst case:

        - 4 reads: read single indirect, read double indirect, read triple indirect, read data block

---

11. Assume you have an inode-based filesystem. The filesystem has 512 byte blocks. Each inode has 10 direct, 1 single indirect, 1 double indirect, and 1 triple indirect block pointer. Block pointers are 4 bytes each. Assume the inode and any block free list is always in memory. Blocks are not cached.

    a. What is the maximum file size that can be stored before
        1. the single indirect pointer is needed?
        2. the double indirect pointer is needed?
        3. the triple indirect pointer is needed?
    b. What is the maximum file size supported?
    c. What is the number of disk block reads required to read 1 byte from a file
        1. in the best case?
        2. in the worst case?
    d. What is the number of disk block reads and writes required to write 1 byte to a file
        1. in the best case?
        2. in the worst case?

    a. 1. 5K
        2. 69K

3. 8261K

b. 1056837K

c.
1. 1
2. 4

d. What is the number of disk block reads and writes required to write 1 byte to a file
1. 1w
2. 4r/1w

---

12. A typical UNIX inode stores both the file's size and the number of blocks currently used to store the file. Why store both? Should not blocks = size / block size?

    Blocks used to store the file are only indirectly related to file size.

    - The blocks used to store a file includes and indirect blocks used by the filesystem to keep track of the file data blocks themselves.
    - File systems only store blocks that actually contain file data. Sparsely populated files can have large regions that are unused within a file.

---

13. How can deleting a file leave a inode-based file system (like ext2fs in Linux) inconsistent in the presence of a power failure.

    Deleting a file consists of three separate modifications to the disk:

    - Mark disk blocks as free.
    - Remove the directory entry.
    - Mark the i-node as free.

    If the system only completes a subset of the operations (due to power failures or the like), the file system is no longer consistent. See lecture slide for example of things that can go wrong.

---

14. How does adding journalling to a file system avoid corruption in the presence of unexpected power failures.

    Simply speaking, adding a journal addresses the issue by grouping file system updates into transactions that should either completely fail or succeed. These transactions are logged prior to manipulating the file system. In the presence of failure the transaction can be completed by replaying the updates remaining in the log.

Page last modified: 12:26pm on Monday, 1st of April, 2019

Screen Version