CS 620: Homework 13

Carmen St. Jean

December 6, 2012

1. *(2) What is file system mounting?*

   The mounting of a file system makes the directory structure available within the file-system name space. A file system is mounted so it can be available to processes on the system.

2. *(2) Explain the bit-vector approach by which the free-space list is implemented.*

   A free-space list records all free disk blocks. Each block is represented by one bit. If the block is free, then the bit is 1; if the block is allocated, then the bit is 0.

   For example, given ten blocks of memory, say blocks 0, 2, 3, 7, and 8 are free. The rest of the blocks are allocated. Then the free-space bit map would be:

   ```
   1011000110
   ```

   The calculation of the first free block number is:

   (number of bits per word) * (number of 0-value words) + offset of first 1 bit

3. *(2) Which of the following system calls use the free-space list: open(), read(), write(), delete(), close(). Explain.*

   Only the write() and delete() system calls would use the free-space list. Before a file can be written, the free-space list needs to be checked. If the free-space list shows there is enough space for the file, then the file is written and the free-space list must be updated. When a file is deleted, its space should be reused so the free-space list must be updated to show the file's blocks are now free.

4. *(8) Compaction is the process by which fragmentation in file systems is removed. In compaction, files are all written contiguously and a large hole remains. Performance improves since reading a file requires a single seek and a single rotation followed by the transfer at full speed. Writing the file back requires the same work. Assume a seek time of 10ms, a rotational delay of 4ms, a transfer rate of 5MB/sec and an average le size of 10KB, Suppose the le system is fragmented, so compaction is started.*

   (a) *How long does it take to read a file into main memory prior to compaction? Assume that prior to compaction, each file, on average, is fragmented into 10 portions (i.e., 10 portions of the file are randomly distributed on the disk, but the data within each portion are contiguous.)*
   $10 * (10 + 4)$ ms = 140 ms per file

   (b) *How long does it take to write the file contiguously into a new location?*
   14 ms per file = 0.014 s per file

   (c) *Using these numbers, how long would it take to compact half of a 20GB disk?*
   Answer

   (d) *By what percentage does the performance of reading a file improve after compaction?*
   Answer

5. *(2) Disk addresses are in terms of sectors/tracks, but file system addresses are in terms of block numbers. The conversion from block numbers to disk sector/track addresses is done by the disk controller.*

   (a) *Give one advantage of this separation between logical and physical addresses.*

   With this separation, a program sees its addresses with relation to itself without needing to know where it is located physically in memory. This means that programs can easily be moved around without having to recalculate the physical addresses within the program.

   (b) *Give one disadvantage of this separation between logical and physical addresses.*

   The disadvantage is that a mapping is required to go from logical address to physical address or physical address to logical address, so there is some overhead due to the required look-ups.

6. *(2) Instead of storing inodes contiguously at a known location on the disk, suppose each file's inode is stored along with the file's data. This results in scattering inodes all over the disk. Name 1 advantage and 1 disadvantage of this approach.*

   The advantage is that there is no disk read time to access the inodes. The disadvantage is since the inodes are not in one centralized place you must search all over the system to find an inode.

7. *(3) Show the file descriptor table(s), and the inode table after the following process1's code and process2's code are successfully executed: Your diagram should show the relationship between the tables.*

```
Process1                                    Process2
---------                                   ---------
fid1 = open("hw.txt", O_RDONLY);            fid1=open("quiz.txt",O_RDONLY);
fid2 = open("quiz.txt", O_RDONLY);
close(fid1);
fid3 = open("exam.txt", O_RDONLY);
```

Process 1's file descriptor table

0	stdin
1	stdout
2	stderr
3	quiz.txt
4	quiz.txt
5	exam.txt

Inode table

inode for stdin
inode for quiz.txt (open count: 2)
inode for exam.txt (open count: 1)
inode for stdout
inode for stderr

Process 2's file descriptor table

0	stdin
1	stdout
2	stderr
3	quiz.txt

8. *(3) How many disk reads are required to fetch the inode for the* `file/usr/student/cs620/hw/hw1.c`*? Assume that the root directory is in memory, but nothing else along the path is in memory. Also, assume that each directory fits in one disk block. Explain your answer.*

   There will be nine disk reads to fetch the inode for . Here is the process that the operating system would go through to fetch the inode for the file:

   (a) look at \ directory for inode of `usr`

   (b) load inode `usr` into inode table (disk access 1)

   (c) load directory `usr` into main memory (disk access 2)

   (d) load inode `student` into inode table (disk access 3)

   (e) load directory `student` into main memory (disk access 4)

   (f) load inode `cs620` into inode table (disk access 5)

(g) load directory `cs620` into main memory (disk access 6)

(h) load inode `hw` into inode table (disk access 7)

(i) load directory `hw` into main memory (disk access 8)

(j) load inode for `hw1.c`

(k) load file `hw1.c` into inode table (disk access 9)

9. *(1) Give the main advantage of the contiguous file allocation scheme when compared to the linked file allocation scheme.*

   The main advantage is that accessing a file is easy because contiguous memory supports both sequential and direct access of blocks. For sequential access, the file system remembers the disk address of the last block referenced and reads the next block when necessary. For direct acces of block i of a file that starts at block b, we can immediately access $b + i$.

   The linked file allocation scheme only supports sequential access and cannot support direct access. To find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block.