Questions and Answers A3

1.

Consider a Unix like file system as on page 11 of lecture 18 where the number of direct blocks is 16. Each block is 4KiB and a block number consists of 4 bytes.

For all questions show your working. If you wrote a program to help calculate these answers you may submit this program. Ensure that the program is clear with good comments (and your login name).

If there is no working take off 0.5 marks per part. Similarly take of 0.5 marks per part if the answer is not in decimal bytes.

Give the answers from a) to d) in decimal bytes e.g. 65,536 rather than 2^16.

a)

What is the maximum size for a file which does not use an indirect block?

Ans:

16 * 4KiB = 64KiB = 2^16 = 65,536 bytes

b)

What is the maximum size for a file which uses the single indirect block? (not the double or triple).

Ans:

Indirect block holds 2^12 / 2^2 = 2^10 or 1024 block numbers

2^10 * 2^12 + 64KiB = 2^22 + 64KiB = 4,194,304 + 65,536 = 4,259,840 bytes

c)

What is the maximum size for a file which uses the double indirect block? (not the triple).

Ans:

2^10 * 2^10 * 2^12 + 4259840 = 4,294,967,296 + 4,259,840 = 4,299,227,136 bytes

d)

What is the maximum size for a file in this system?

Ans:

2^10 * 2^10 * 2^10 * 2^12 + 4299227136 = 4,398,046,511,104 + 4,299,227,136 = 4,402,345,738,240 bytes

e)

How many levels of indirection would be needed if we were to extend this file system to accommodate files up to 1 exbibyte (1024^6) in size?

Ans:

5

The extra bytes from the previous level are always less than 0.1% of the current level so there needs to be 5 levels of indirection.

2^10^5 * 2^12 = 2^62 (5 levels of indirection gives files bigger than this)

1 exbibyte - 2^10^6 = 2^60

For questions f) to j) you have to specify the number of block accesses which are required to complete the task. *The number of reads and writes is enough for the marks but hopefully they will have explained why.*

f)

Reading 100 bytes starting at byte position 4050 of the file.

Ans:

2 reads. The data is in direct blocks but spans two blocks

g)

Reading 100 bytes starting at byte position 4,259,820.

Ans:

This spans the last block in the single indirect level with the first block in the double indirect level.

To access the first block 2 reads.

To access the next block 3 reads. So 5 accesses in total.

h)

Reading 100 bytes starting at byte position 4,263,900

Ans:

This spans the first and second double indirect data blocks.

3 reads the first block (index, index, data), 1 read the second because the index blocks are now cached. So 4 accesses in total.

i)

Writing 100 bytes starting at byte position 4,259,820 when the file length is 5,000,000.

Ans:

Have to read both blocks as in g). So 5 read accesses. 2 write accesses to the blocks which are modified. 1 write for the inode information because of the change in the modified time (the length hasn't changed). 8 accesses all together. *Half a mark if the inode write is omitted.*

j)

Writing 100 bytes starting at byte position 4,259,820 when the file is currently 4,259,820 bytes long.

Ans:

Read the current block - 2 accesses (index, data). Write the current block - 1 access. Allocate next block, this is into the double indirect area, so allocate double indirect block, single indirect block and data block - 3 writes (blocks are new so don't need to be read).

inode needs updating - length and modified time as well as the double indirect block pointer - 1 access to write (already read in memory).

Total of 7 block accesses.

2.

a)

There was a simple file system which used indexed allocation to store location information for its files. There was a pointer in the file directory entry called the key block. If a file was small enough to fit in one block the key block pointed to that one data block and the file was called a seedling file. If the file was too large to fit in one block then the key block pointed to an index block. This one index block pointed to as many data blocks as could fit in the space of the block. Files using one index block were called sapling files. If the file was too large to have just one index block, the key block pointed to a block consisting of pointers to index blocks. Files with these two levels of index block were known as tree files.

Given a block size of 512 bytes, and the space for a pointer to a block of 2 bytes, what are the minimum and maximum sizes of seedling, sapling and tree files? Give the answers in bytes. *In this case because the question didn't say decimal bytes they could leave the answer as a power of 2.*

Ans:

seeding: min = 0, max = 512

sapling: min = 513, max = 2^9/2 * 2^9 = 2^17 = 131,072

tree: min = 131,073, max = 2^9/2 * 2^9/2 * 2^9 = 2^25 = 33,554,432

b)

Unlike the question above in the actual file system this was based on the first level index block for a tree file only held 128 index block pointers. Give a reason why this might be so. What was the name of this file system?

Ans:

The answer is to do with the amount of storage required to store the length of the file. Half a mark for anything which refers to this. The other half mark if they explain that the maximum length of a tree file with this restriction is 2^7 * 2^8 * 2^9 = 2^24 which can be stored in 3 bytes, and that without the restriction 4 bytes is required. (Extra to explain why - this is not necessary for the mark. This was important because the file attributes were designed to fit in a carefully chosen size for the directory information. This meant a specific number of directory entries fitted into a disk block. Also the disk maximum size was 32MiBs and restricting the maximum file size to 16MiBs seemed perfectly acceptable.)

Extra mark: ProDOS (or SOS)

c)

When a seedling file grows into a sapling file (by adding one extra byte to the file length) how many WRITES need to be made to the directory entry and any other blocks on the disk to store the changed information? Describe each of the writes e.g. change to bitmap block. *Lose half a mark if the writes are not described.*

Ans:

1 write to newly allocated block for the one byte of extra data

1 write to newly allocated index block

1 write to directory entry - updated key block, storage type and file length

Either 1 or 2 writes to bitmap block (because we have allocated 2 new blocks)

So 4 or 5 writes (full marks for either, as long as they are properly described)

d)

Answer the same as part c) for a sapling file growing into a tree file.

Ans:

1 write to newly allocated block for the one byte of extra data

1 write to newly allocated index block

1 write to newly allocated first level index block

1 write to directory entry - updated key block, storage type and file length

Either 1, 2 or 3 writes to bitmap block (because we have allocated 3 new blocks)

So 5, 6 or 7 writes (full marks for any one, as long as they are properly described)

3.

Produce frame usage tables for the specified algorithms.

The page reference string is along the top row.

Each row after the top row represents one frame. So there are 4 frames of memory.

A zero entry indicates the frame is empty. An entry of "=" means the frame holds the same page as in the previous column.

	1	2	3	4	5	4	3	2	1	6	7	1	2	3	7	5	1
0	1																
0	=																
0	=																
0	=																

f*For each of these 1.5 marks for the table and 0.5 for the number of page faults. If they go wrong after getting the first half of the table correct give 0.5/1.5 for the table.*

a) FIFO - First In First Out

Ans:

	1	2	3	4	5	4	3	2	1	6	7	1	2	3	7	5	1
0	1	=	=	=	5	=	=	=	=	=	=	=	2	=	=	=	=
0	=	2	=	=	=	=	=	=	1	=	=	=	=	3	=	=	=
0	=	=	3	=	=	=	=	=	=	6	=	=	=	=	=	5	=
0	=	=	=	4	=	=	=	=	=	=	7	=	=	=	=	=	1

Page faults = 12

b) LRU - Least Recently Used

Ans:

	1	2	3	4	5	4	3	2	1	6	7	1	2	3	7	5	1
0	1	=	=	=	5	=	=	=	1	=	=	=	=	=	=	5	=
0	=	2	=	=	=	=	=	=	=	=	=	=	=	=	=	=	1
0	=	=	3	=	=	=	=	=	=	=	7	=	=	=	=	=	=
0	=	=	=	4	=	=	=	=	=	6	=	=	=	3	=	=	=

Page faults = 11

c) Optimal

Ans:

	1	2	3	4	5	4	3	2	1	6	7	1	2	3	7	5	1
0	1	=	=	=	5	=	=	=	=	6	7	=	=	=	=	=	=
0	=	2	=	=	=	=	=	=	=	=	=	=	=	=	=	5	=
0	=	=	3	=	=	=	=	=	=	=	=	=	=	=	=	=	=
0	=	=	=	4	=	=	=	=	1	=	=	=	=	=	=	=	=

Page faults = 9

(The last 5 can be inserted in any row except the first because 2, 3 and 4 never occur again.)

4.

Summarise, in your own words, five significant changes made to the memory management system in the transition from Windows 7 to Windows 8 and 10. Also explain why those changes were made. Write no more than a paragraph about each change.

Ans:

You can argue that not all of these changes are to the memory management system, but they are general changes which help the memory system be more efficient. I intended the question to mean these as well.

One mark for each change, one mark for each explanation. There only need to be 5 altogether, not 10.

There may be more than these as well.

Windows 8

When required not only pages can be trimmed from working sets but entire Universal Windows Platform (UWP) applications can be swapped out. UWP apps do not run in the background so if they are not being used they can be swapped out completely without penalty. Doing this also keeps the swapped image contiguous which makes swapping the app back in more efficient.

Memory Combining: The memory management system scans pages checking for duplicates. If a page has the same content as another page already in memory then one of them is released. Of course if one of the pages is modified then a new page has to be provided for the modified version. This reduces the number of pages needed in memory especially in situations such as running a

number of VMs with the same OS, in this case there are many pages the same across multiple processes, but these pages are not otherwise shared.

Service changes and reductions: Some processes in Windows 7 which started at boot time then hung around forever even though they were only used occasionally. The changes mean that some don't start immediately, leading to improved boot times, and some don't start until required and then are removed from memory. This reduces the memory requirements.

Low priority memory requests: Processes can flag memory requests as low priority, usually for pages that are used for a short period of time then no longer needed. This way these pages are released or paged out before other more important pages when memory pressure occurs.

Data structure rearrangement: Some large OS data structures have areas which are needed more often than others. If the frequently accessed areas are moved into the same pages then they can stay in memory while the less frequently accessed data can be paged out. This frees pages which would otherwise have to stay in memory because a little of the data in each page is being used.

Windows 10

Pages are compressed rather than written to disk if possible. If more memory space is required then the compressed pages are written to disk. This saves unnecessary writes and reads from the page file. Also writing compressed pages to the page file reduces the write and read amounts when moving the page back into real memory. Decompression is done using multiple cores and is faster than reading uncompressed pages from the page file.