Quiz

Bookmark this page

Reliability

1/1 point (graded)

Consider a file system implemented on solid state memory.

Which of the following are ways to improve the reliability of the file system? The idea is in what ways could have Lab 5 been changed to decrease the probability of lost data?

- [✓] Change the file system mechanism so important information is stored in multiple places on the disk
- [✓] Add additional storage mechanisms like SDC cards or FRAM so important information is stored on different physical media
- [✓] Add verification software that checks to see if data is properly stored, so that you get errors when the disk is beginning to fail.
- [✓] Include checksum data to each file, and add periodic tasks in the OS that check to see if the files are in a consistent state. This way you can get a warning when the disk is beginning to fail.
- [✓] Assuming each block can be written a finite number of times before it fails, let the OS keep track of the number of times a block has been written and when a block approaches its expected end of life, have the OS mark that block as unusable.

✓

Answer

Correct:
Storing data in multiple places on the disk is good because if one sector is lost, then the data is still on the other places
Storing data on multiple media is good because if the entire solid state disk is lost, then the data is still on the other places
Adding verification is good, because verification can give you warning when only some data is lost, allowing you time to repair/replace the disk

Adding checksums to each file is good because it can give you warning when only some data is lost, allowing you time to repair/replace the disk
Keeping track of the number of times a sector is used is part of a process called wear-leveling

Submit

✓ Correct (1/1 point)

Indexed allocation

1/1 point (graded)

A disk with indexed allocation has 2 mebibytes of storage. Each file has a separate index table, and that index occupies just one sector. The disk sector size is 1024 bytes. Assume the index table uses 16-bit sector addresses.

What is the largest file that can be created? Give your answer in kibibytes

512 ✓

512

Answer

Correct:

There are 2^21/2^10=2048 sectors, so the 11-bit sector address can be stored as a 16-bit number. One can store 1024/2=512 index entries in one 1024-byte sector. So the maximum file size is 512*1024 = 2^9*2^10 = 2^19 = 512 kibibytes.

Submit

✓ Correct (1/1 point)

Consider a file system that uses **contiguous allocation** to define the set of sectors allocated to each file, as shown in Figure Q5.1. In this file system the sector size equals the block size, which is 32 bytes. There are 8192 bytes on this disk made up of 256 sectors. This file system is used to record important “black box” information. Therefore, the file system is initialized to empty when the device is manufactured. Each time the system is turned on, a new file is created. While running important data are stored into that file (create new file, append data at the end, close file). Files are never deleted. Once a file is closed, it can be opened for reading, but it cannot be opened again for writing. Sector 0 contains the directory and not available for data. Each directory entry has three fields: name, sector number of the first sector, and total number of bytes stored. The example in Figure Q5.1 shows file A with 3 allocated sectors (1,2,3 containing 32,32,8 bytes), file B with 2 sectors (4,5 containing 32,32 bytes), and file C with 7 sectors (6,7,8,9,10,11,12 containing 32,32,32,32,32,32,8). All 32 bytes of each data sectors can contain data for the file. However if a file contains more than one sectors, only the last one may be partially used; all the other sectors will have 32 bytes of data.

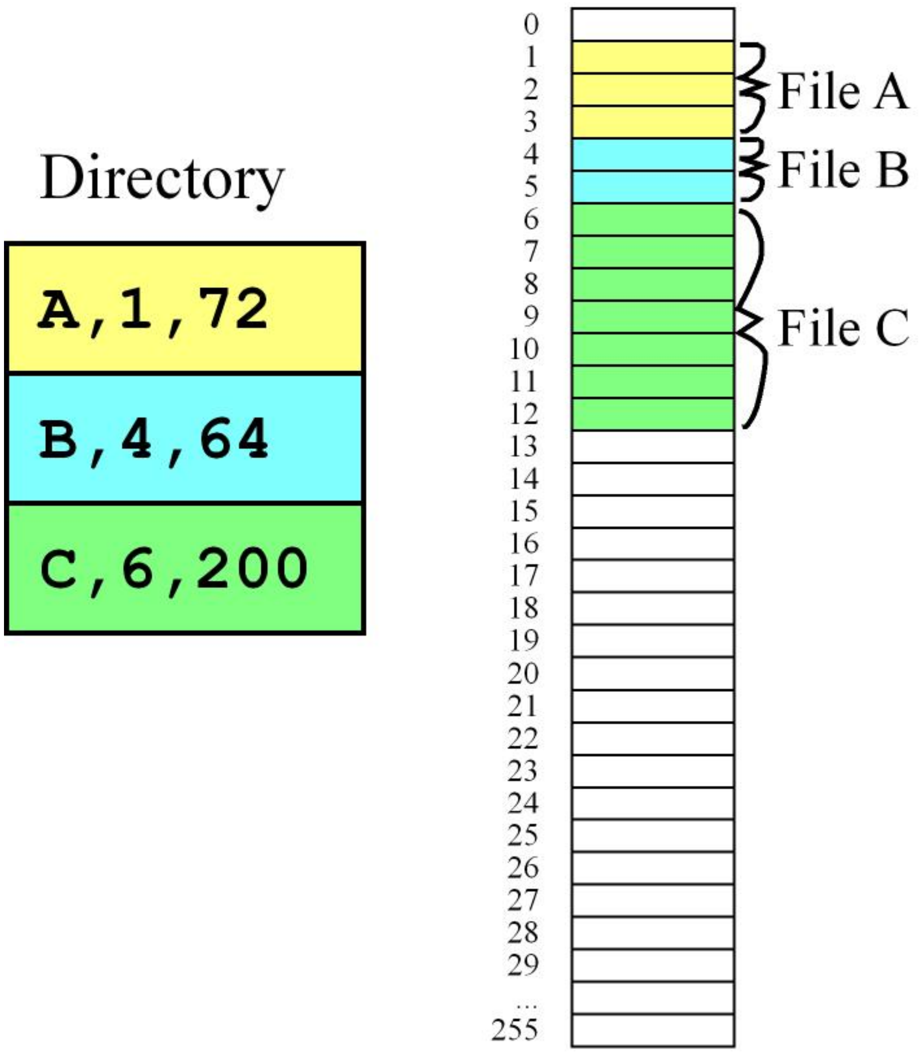

Figure Q5.1. Block diagram showing a simple file system using contiguous allocation and write-once behavior.

Part a)

1/1 point (graded)

Does the above file system have any external fragmentation?

- yes, external fragmentation might occur
- [✓ no, external fragmentation cannot occur] ✓

Answer

Correct:

Free space will always be one large chunk at the end.In this case you can always use all free blocks to create one large file.

Submit

✓ Correct (1/1 point)

Part b)

1/1 point (graded)

Assume a file has n data sectors. It takes one sector read to fetch the directory. On average, how many more sector reads does it take to read a single byte at a random position in the file?

1 ✓

1

Answer

Correct:

Only one more read is required because of contiguous allocation one knows exactly where each byte of each file is located.

Submit

✓ Correct (1/1 point)

Part c)

1/1 point (graded)

Which free space management would be best for this file system? While all of these could work, which is simplest?

- ○ Link all the free sectors in a FAT
- ● Direct calculation looking at just the directory? ✓
- ○ Link all the free sectors with pointers included in with the data
- ○ Bit vector
- ○ Index table of free sectors

Answer

Correct:

Find the last file in the directory; n=start sector of that file, m= the number of sectors in that file; first free sector is the n+m+1

Submit

✓ Correct (1/1 point)

Part d)

1/1 point (graded)

File names are a single character. How many files can be stored?

8 ✓

8

Answer

Correct:
file name is 1 byte, starting sector is 1 byte, length is 2 bytes. Since each entry is 4 bytes, there can be 32/4=8 files on this disk

Submit

✓ Correct (1/1 point)

Part e)

1/1 point (graded)

Assume you have multiple files each with of random size. Quantify the number of wasted bytes due to internal fragmentation.

16 ✓

16

Answer

Correct: On average, each file wastes a half a sector

Submit

✓ Correct (1/1 point)

Consider this file system that uses a **File Allocation Table** (FAT). Each disk sector contains 32 bytes. The directory is in sector zero, the FAT is in sector 1. Each directory entry contains a file name and an index to the first FAT entry for that file. The last entry in the directory contains an index to the first FAT entry for the free space. Entire sectors are allocated to a file, so there is no internal fragmentation.

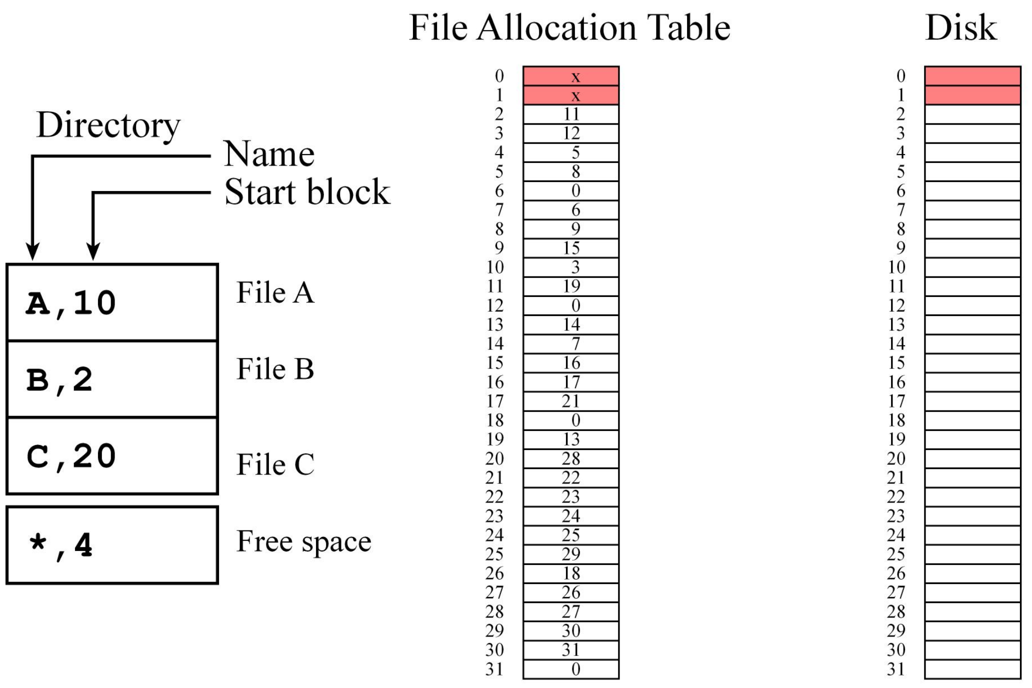

Figure Q5.2. Simple file system using a FAT allocation scheme.

Part a)

1/1 point (graded)

Given the values shown in Figure Q5.2, how many free sectors are there?

15 ✓

15

Answer

Correct:

Start with 4, follow the links and count 15 free sectors. The free sectors are 4,5,8,9,15,16,17,21,22,23,24,25,29,30,31 (0 does not count) .

Submit

✓ Correct (1/1 point)

Part b)

1/1 point (graded)

Does this file system have external fragmentation?

[✓ No, there cannot be external fragmentation] ✓

Yes, there can be external fragmentation

Answer

Correct: All 15 free sectors could be used for one file

Submit

✓ Correct (1/1 point)

Part c)

1/1 point (graded)

Given the situation drawn in Figure Q5.2, how many bytes are saved in File A?

96 ✓

96

Answer

Correct:

Start with 10 and count 3 sectors. File A uses sectors 10,3,12 (0 doesn't count)

Submit

✓ Correct (1/1 point)