1a)False. The same owner and group users as the owner can read the file

1c)~~True~~. Linked file allocation method involves each file being a linked list of disk blocks that are scattered in the disk. No space is wasted so it may be hard to find a big enough memory to execute a process involving the file

False. Any free physical block can be used in linked allocation method so no external fragmentation

SC2005 Operating Systems

TUTORIAL ELEVEN

File-Systems

False

1b)~~True~~. The open file stores a copy of a file control block so that it can be easily retrieved, improving efficiency

1. Indicate whether the following statements are true or false. Justify your answers.

   a) Assuming the protection bits of a file are “r - x r - x - - x”, only the owner of this file is able to read the file.

   b) Open file table is used to temporarily cache data blocks of a file to improve efficiency.

   c) Linked file allocation method may result in external fragmentation.

a)There may be internal; and external fragmentation. Employ tools to rearrange data blocks on the disk

2. a) Consider a file system where a file can be deleted and its disk space reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

   b) Some systems provide file sharing by maintaining a single copy of a file; other systems maintain several copies, one for each of the users sharing the file. Discuss the relative merits of each approach.

1)Data content: accessing new file wrongly
2)Access protection: accessing new file without checking its access protection

Delete file after all links are deleted

b)Different copies enables different users to have different access permissions and keeps a backup. A single copy makes it easier to sync the data on this file

Single copy:
-concurrent updates can cause race condition
-mutual exclusion to be enforced on file access

Multiple copies:
-storage waste
-inconsistent copies
-performance better

3. Assume that a file system uses Unix-like inodes with six direct pointers and one single-indirect pointer. The file system is block-oriented with both logical and physical block sizes of 1000 bytes. A user executes a program that contains the following code:

```
fd = open(“/usr/ast/mbox”);        // open a file
seek(fd, 5900);  // move the file pointer
read(fd, buf, 200);          // read 200 bytes of data from the file
```

   a) How many disk read operations are required for the first system call (i.e., open) in the above program fragment? Assume that initially, the root directory is in memory, inodes are in disk, and that a directory can fit into a single block.

   b) How many disk read operations are required for the last two system calls (i.e., seek and read) in the above program fragment?

   c) What is the maximum file size the file system can support? Assume that each entry in the indirect block takes 2 bytes.

Five. Load inode of usr
Load data block of usr
Load inode of ast
Load data block of ast
Load inode for mbox

3a)
4 disk read operations:
1 to read through root directory, 1 to read through /usr,1 to read through /usr/ast, 1 to read mbox inode

b)
1 for seek() operation
2 for read() operatiopn

seek requires no disk read operation

4. Some file systems use two block sizes for disk storage allocation, for example, 4-Kbyte and 512-byte blocks. Thus, a 6 Kbytes file can be allocated with a single 4-Kbyte block and four 512-byte blocks. Discuss the advantage of this scheme compared to the file systems that use one block size for disk storage allocation.

This scheme makes it fast to find and transfer blocks without the drawback of wasting space when not the entire last block is occupied.

small block size: reduce fragmentation
large block size: improve throughput

read partially loads blocks 5 and 6 of the file. Three disk block read operations:
-read block 5,
-read indirect block,
-read block 6

from current position pointer = 5900

c)
direct pointers : 6 * 1000 = 6000 Bytes
direct pointers + single indirect pointers: 6000 + ~~2^6~~ 1000/2 *1000 = ~~38~~ 506000 Bytes