HW4.1

Questions: Tannenbaum & Bos 4.{1,3,5,9,11,12,15,17,18,19}

**1.** Give five different path names for the file */etc/passwd*. (*Hint*: Think about the directory

entries ‘‘.’’ and ‘‘..’’.)

./passwd

../passwd

etc/passwd

./etc/passwd

../etc/passwd

/./etc/passwd

/etc/../etc/passwd

**3.** In early UNIX systems, executable files (*a.out* files) began with a very specific magic

number, not one chosen at random. These files began with a header, followed by the

text and data segments. Why do you think a very specific number was chosen for executable

files, whereas other file types had a more-or-less random magic number as the

first word?

The magic number 0 was chosen for executable files. These were loaded directly into memory. The magic number is a branch instruction with a target address just above the header. This avoids trying to execute the header as code. The binary file can now be read directly in new process’ address space without size of the header.

**5.** Systems that support sequential files always have an operation to rewind files. Do systems

that support random-access files need this, too?

No, randomly accessing the byte 0 will read the file again.

**9.** In UNIX and Windows, random access is done by having a special system call that

moves the ‘‘current position’’ pointer associated with a file to a given byte in the file.

Propose an alternative way to do random access without having this system call.

If the read system call had a extra parameter specifying which address to read, which would add overhead and keeping track of the file pointer. This enables the reads to seek within a file if needed.

**11.** Contiguous allocation of files leads to disk fragmentation, as mentioned in the text, because

some space in the last disk block will be wasted in files whose length is not an

integral number of blocks. Is this internal fragmentation or external fragmentation?

Make an analogy with something discussed in the previous chapter.

External fragmentation because the memory wasted is in between files. If the memory was inside of the file it would be internal. This is similar to a swapping system or pure segmentation system having main memory external fragmentation.

**12.** Describe the effects of a corrupted data block for a given file for: (a) contiguous, (b)

linked, and (c) indexed (or table based).

Contiguous has some protection before the corruption. Linked has no protection. Indexed would only the corrupted data block.

**15.** Some digital consumer devices need to store data, for example as files. Name a modern

device that requires file storage and for which contiguous allocation would be a fine

idea.

Digital cameras record in sequence on flash cards. The pictures are saved in order, like a contiguous file system, while space is available. Then, the data is removed and stored on something else, like another computer, a disk, or digital picture frame.

**17.** For a given class, the student records are stored in a file. The records are randomly accessed

and updated. Assume that each student’s record is of fixed size. Which of the

three allocation schemes (contiguous, linked and table/indexed) will be most appropriate?

A table would make accessing and updating fastest.

**18.** Consider a file whose size varies between 4 KB and 4 MB during its lifetime. Which

of the three allocation schemes (contiguous, linked and table/indexed) will be most appropriate?

Linked allocation would be best because the memory needed is different sizes. Linked just needs to know where the memory begins in a linked list of disk blocks. A file can continue to grow with free blocks available. Pointers can be lost or damaged causing bugs, and accessing files directly is a disadvantage.

**19.** It has been suggested that efficiency could be improved and disk space saved by storing

the data of a short file within the i-node. For the i-node of Fig. 4-13, how many

bytes of data could be stored inside the i-node?

A way to determine if address block pointers hold data or pointers is needed. A left over bit from file attributes can be made available for use. The 9 pointers can then be used for data, for how many bytes each. So 9k. Or if no bit is left over from attributes the first disk address becomes invalid to mark the bytes are data and not pointers. Then only 8 pointers are left so 8k bytes available for file.