

ITEE



## What's New:

There are fewer questions this time, since some of you may want to focus on your assignment.

Learning objectives for this week:

- Understand the major concepts we've covered, including:
  - files and file systems
  - directory structures
  - security
- Apply these concepts to reasoning about alternative file system designs
- Apply these concepts to reasoning about alternative directory structure designs
- Understand where and how security is an issue

As before, you should aim to answer all of these questions by the end of the course. Being able to answer them by the end of week 10 is a useful goal.

## 1. Concepts

- a. Why is it necessary to store files noncontiguously?
- b. Why are directories necessary?
- c. Why are access lists more general than the standard UNIX protection scheme?

## 2. File Access and Storage

- a. Explain why it's easy to fake a sequential file on a direct-access file, but not vice-versa.
- b. In the original Macintosh file system, files were structured as follows. Each file had a *resource fork* and a *data fork*, either of which could be empty. A resource fork contained a structured collection of code, user interface elements and data useful for initializing a run of the program. A directory entry included a file's type and creator (represented as 4-byte codes, usually representing 4 capital letters. File names were limited to 31 characters, but could contain any characters except a ":" (which was used as a path separator). Compare this arrangement with the naming and structuring conventions of UNIX – what is made easier or harder in the Macintosh approach?
- c. Compare the FAT approach (p 425) to the UNIX approach (p 428).
  - i. Is there any difference in the total overhead (extra space needed) to store a small file, e.g., 1 disk block? Explain your answer.
  - ii. For a very large file, e.g., 20GB, which scheme is better? Explain your answer.
  - iii. Which of the two schemes in general is likely to result in more disk accesses?
- d. Explain why managing free space is important from the point of view of performance.

## 3. Directory Structures and Mounting

- a. Explain why a tree-structured directory does not meet all of the following design goals for naming files:
  - it should be possible to name a file only considering others used in a like context
  - it should be possible to reuse names for different purposes
  - it should be possible to give more than one name to the same thing
  - names should be convenient for the user
- b. What mechanism solves any problems you have identified in (a) – for example, in UNIX?
- c. Explain how NFS supports the concept of using any machine on a network as if it was your own machine.

## 4. Security and Dependability

- a. If you would like Tom, Don and Sue all to have read-only access to a file, but Don and Sue should have write access to another file, which Tom should not be able to read or write, how would you achieve this using:
  - i. UNIX-style permissions?
  - ii. an access list?
- b. In general, what are points for and against the two mechanisms for access control?
- c. What performance overheads would you expect of a journaling file system? Do you consider those overheads to be justified?

## 5. Work on the assignment, and make sure you can do all of last week's questions.