

# CSCI 1330 Homework Assignment 6

*Due 11:59pm October 22, 2020*

1. Information about files is stored in the following data structures (see Lectures 16 and 17):
  - i. directory entry
  - ii. file-descriptor table
  - iii. file-context structure
  - iv. inode

The directory entry and the inode are stored on disk and brought into memory when necessary; the other two exist only in memory. We know that a single file might be referenced by multiple directory entries as well as be open in multiple processes, perhaps multiple times in some of these processes. Access permissions (the permission vectors described in lecture 17) are checked just when a file is opened, and thus not on each *read* or *write* of the file.

- a. In which of the data structures does it make sense to store the size of a file (i.e., the byte offset of the location just after the last byte of the file)?
  - b. In which of the data structures does it make sense to store the permission vector?
  - c. When a *read* or *write* system call is executed, the location in the file at which the system call will take place is taken from the file-location field of the file-context structure. One option of the *open* system call is `O_APPEND`, which specifies that *writes* to the file always take place at the current end of the file. Explain how this is made to work, in terms of the data structures listed above.
  - d. A directory is represented just like any other file, except that its contents are interpreted specially. In which of the above data structures is it indicated that a file is a directory?
2. Unix's I/O system calls, in particular *read* and *write*, were designed with the idea that the vast majority of I/O is sequential, i.e., starting at the beginning of a file and proceeding in order to the end. However, it is possible to do non-sequential (or random) I/O through the use of the *lseek* system call. This function simply sets the file location in the file-context structure, thus affecting where the next read or write takes place.

A relatively new pair of Unix system calls is *pread* and *pwrite*, which work just like *read* and *write*, except that the file location is specified as an argument and the value in the file-context structure is ignored. This might be considered merely an optimization, allowing random I/O to be done in a single system call rather than two, but it turns out to be necessary in certain situations. What are these situations?