# Filesystem Fundamentals
# Activity Instructions

(format borrowed from TIDES and FIC workshops:)

---

All the file layouts we saw had important caveats.  So… let's create a new one!

**Part 1**

What type of data is an on-disk pointer? A string? A char? Something else?
Compute how many pointers-to-blocks we can store in a single indirect block if a pointer is 4 bytes and a block is 4096 bytes. Call this number N.

**Part 2**

Using the file allocation techniques you have learned, fill out the following mad lib. You should use multilevel-indexed allocation for at least one of them. You may repeat allocation techniques if you think they will work well (for example, we already know direct allocation will not work for files larger than N.)

For files between 0 and 10 blocks, use _____ allocation
For the next N blocks past that, use _____ allocation
If we need more blocks than that, use _____ allocation

**Part 3**

Analyze your new filesystem! Here are some questions you can try to answer:

- How many accesses do we need to access small files sequentially? Randomly?
- How many accesses do we need to access large files sequentially? Randomly?
- Is there a maximum file size? If so, what is it?
- What is the minimum number of blocks we need to corrupt in order to break the file? (inode doesn't count--any system breaks when you corrupt the inode)
- Is fragmentation an issue in this filesystem? Only at certain file sizes? Not at all?
- Can you easily grow files? Only at certain file sizes? Not at all?
- How hard do you think it would be to implement this system?

**Part 4**

Now consider the characteristics of systems that might use your file system.

- In what environments will your filesystem do best?  Explain.
- As you designed your file system, did you build in any assumptions about where it might be operating or what sorts of files it might be storing?  Explain.
- If you learned your file system would be operating in a system that experiences mysterious errors on its stable storage devices, what would you change? Explain.