Operating Systems Coursework

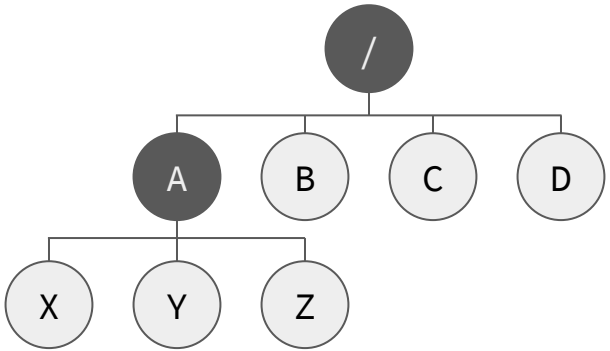
Task 3

TAR File System Driver

DUE: Thursday 30th March @ 4PM GMT

File Systems

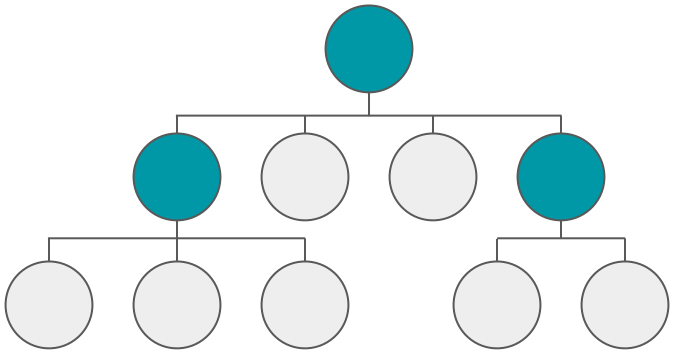
- Used for the organised storage of data.
- Typically hierarchical/tree-based, consisting of directories (nodes) and files (leaves).
- Directories contains files and other directories.



InfOS Virtual File-system Switch

The **virtual file-system switch (VFS)** presents a single, uniform file-system hierarchy, that comprises multiple individual file-systems mounted within, giving the appearance of one big file-system.

These “sub-file-systems” can be “**real**” or “**virtual**”.

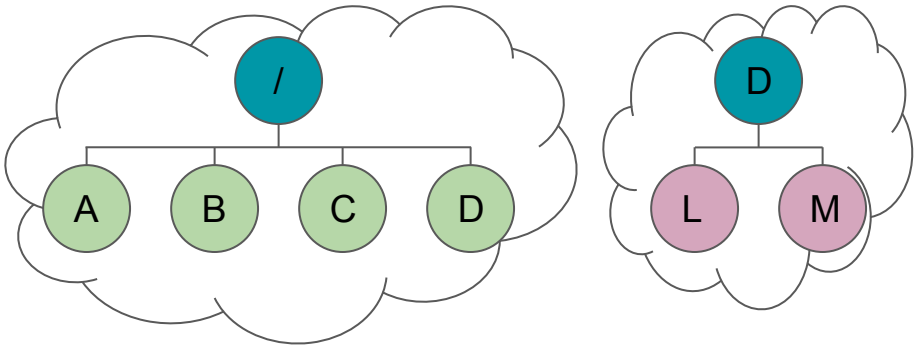


PFSNodes and VFSNodes

PFSNode is a **Physical Filesystem Node**

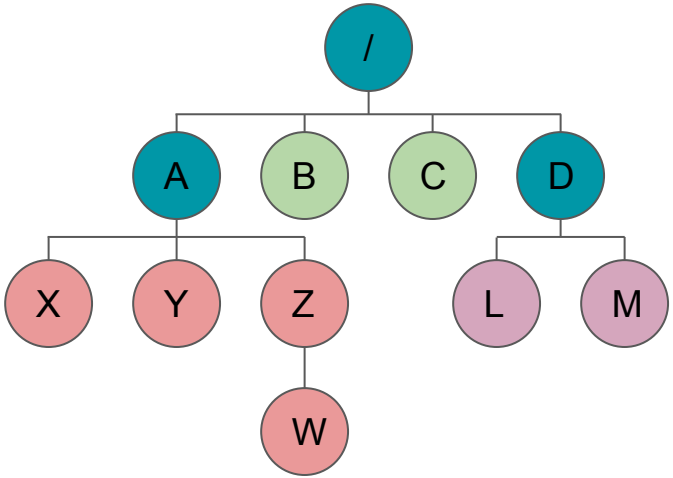
These nodes represent individual files or directories in a real filesystem.

Individual file-systems subclass PFSNodes with their own implementation, e.g. TarFSNode subclasses PFSNode.



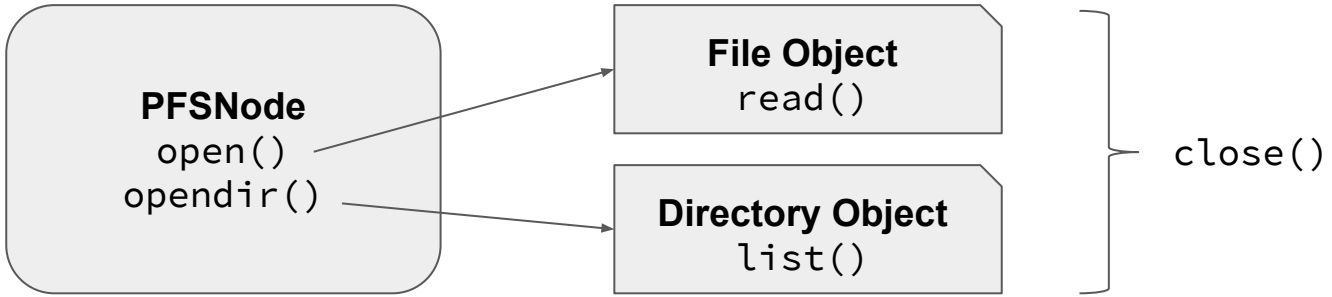
VFSNode is a **Virtual Filesystem Node**

These nodes represent PFSNodes, but in the uniform VFS tree. A VFSNode may reference a “root” PFSNode, and so is said to be a mount point.

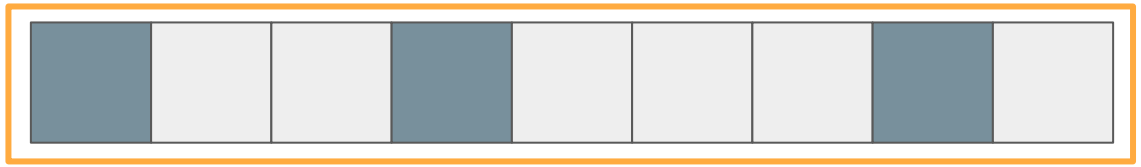


Files and Directories

- A PFSNode represents the metadata of a file or directory in the physical file-system.
- To access data, or list a directory, the PFSNode must be **opened**.
- Opening a PFSNode returns a File or Directory object.
  - The File object can be used to read data from the file.
  - The Directory object can be used to list the contents of the directory.
- When you have finished with the File or Directory, you **close** it.



TAR File Format



- "Tape Archive"
  - Back when cassette tapes were used for archival and storage.
  - Modern systems use it as a convenient format for archiving files,
  - Typically, TAR files are then compressed with e.g. gzip, bzip.
- Optimised for sequential and streaming access.
- Divided into 512-byte blocks.
- Each file starts with a header block
  - Contains details such as filename (including path)
  - File size
  - File attributes
- File data follows, padded out to 512-byte block size
- Archive ends with two zero blocks

https://www.gnu.org/software/tar/manual/html_node/Standard.html

TAR File Format

- Files are stored **sequentially**, not necessarily in any order.
- The **directory hierarchy** is not described in the archive..
- The header contains the **filename**, which contains directory information.
    - Path components are separated by slashes (**/**)
    - This **implies** the hierarchy, and you must build a hierarchy of TarFSNodes to represent this.
- Headers may not necessarily refer to files -- you must check what type they are!

Block Device

- InfOS provides an abstract way to access storage devices on a per-**block** granularity.
- Fits in really well with the TAR file format, as the block device is configured to have the same block size as a TAR file
- Really simple operations:
  - `block_size()` - returns the size (in bytes) of a block.
  - `read_blocks(buffer, offset, count)` - reads **`count`** blocks into **`buffer`**, starting from block **`offset`**.
  - **Important:** buffer **must** be big enough to hold the data you are reading, e.g. it must be at least **`block_size() * count`** bytes in size.
- InfOS will automatically give your filesystem a block device corresponding to the TAR file containing the userspace filesystem.

Coursework Skeleton

- `tarfs.cpp`
  - Contains the source-code for your implementation.
- `tarfs.h`
  - Contains the class prototypes and definitions for your implementation.
- As before, you can make any changes you want to these skeletons.
- You are encouraged to implement "helper" methods to make coding easier.
  - This is why you need to submit your header file. If you add methods to the C++ class, then you will also need to modify the header, where the class prototype lives.

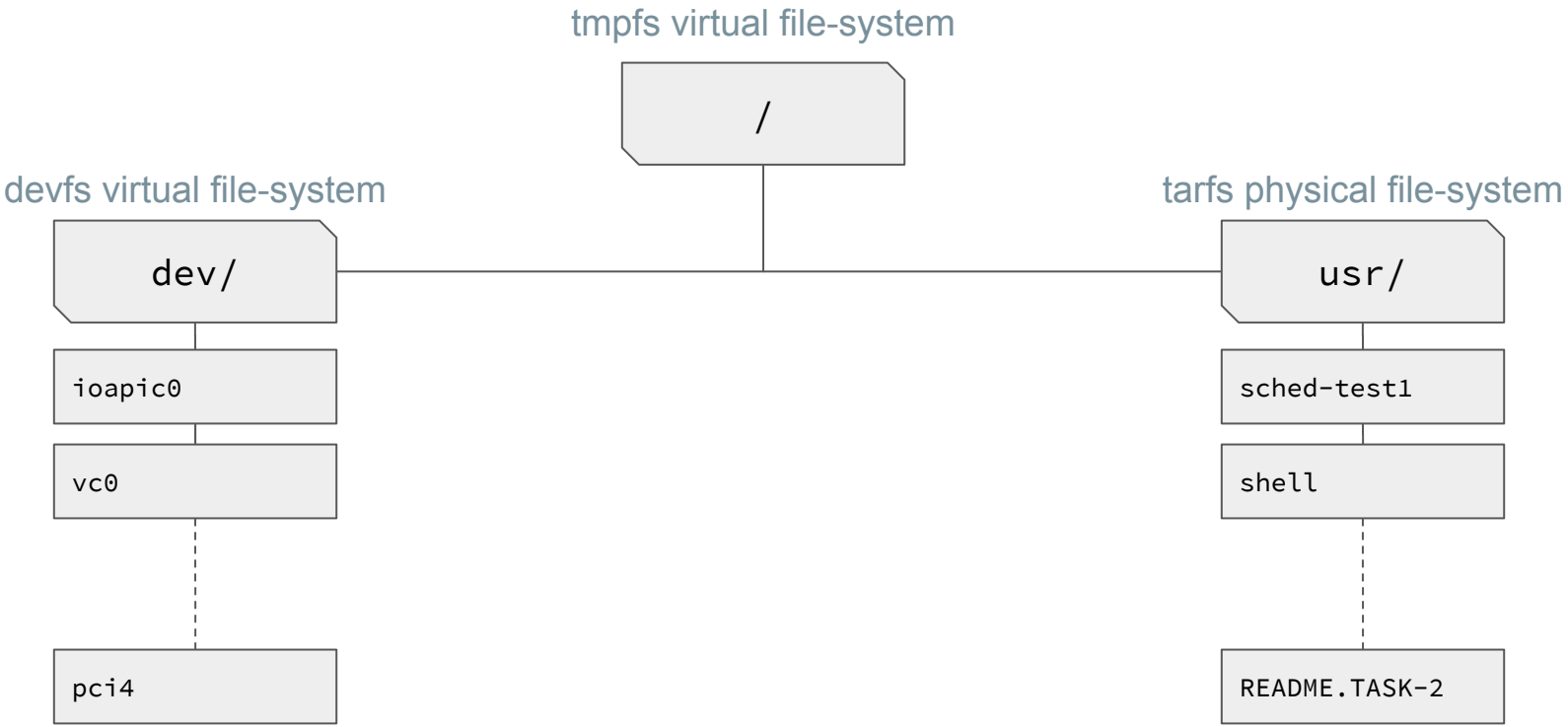
Implementation

Three functions across **two** classes:

- `TarFSNode *TarFS::build_tree()`
  - Called at file-system mount time to build the in-memory representation of the file-system tree.
  - Need to iterate over the TAR file, and build a tree of TarFSNodes.
- `int TarFSFile::pread(void* buffer, size_t size, off_t off)`
  - Called by the VFS core to read out a portion of a file.
- `unsigned int TarFSFile::size() const`
  - Called by the VFS core to determine the size (in bytes) of a file

Testing

- Does the system boot?
- Can you list directories? Do the file-sizes match?
  - `/usr/ls /`
  - `/usr/ls /dev`
  - **`/usr/ls /usr`** **(this is the important one!)**
- Can you read files?
  - `/usr/cat /usr/README`
  - `/usr/cat /usr/README.TASK-3`
- Anything under the `/usr` directory comes from the TAR filesystem driver.
  - `/` and `/dev` are "virtual filesystems"
- Writing files is not supported, as you'd need to re-generate the entire TAR file to handle this.
  - TAR files are not amenable to "in place" editing.



InfOS File-system Hierarchy

Built-in File-system Types

- `tmpfs`
  - A temporary “in-memory” file-system. Changes are not saved, and live only in memory.
- `devfs`
  - A virtual file-system that presents “devices” as files in the directory listing.
- `rootfs`
  - A blank file-system that serves as a placeholder for mounting the root filesystem onto.

A rootfs is mounted in the root first. Then, a tmpfs is overlayed on top, and the directories “dev” and “usr” are created. Next, a devfs filesystem is mounted onto “dev”, and finally the userspace file-system is mounted into “usr”.