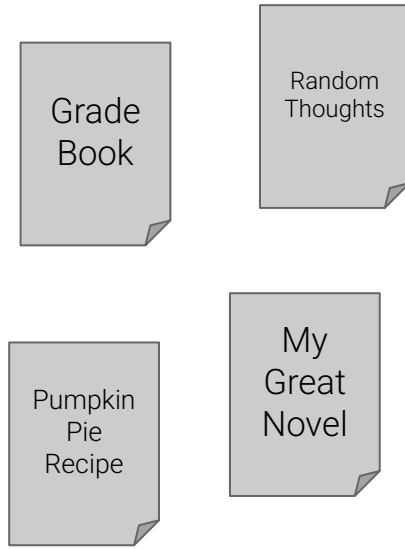


Linux Luminarium

The File System
pwn.college

Files

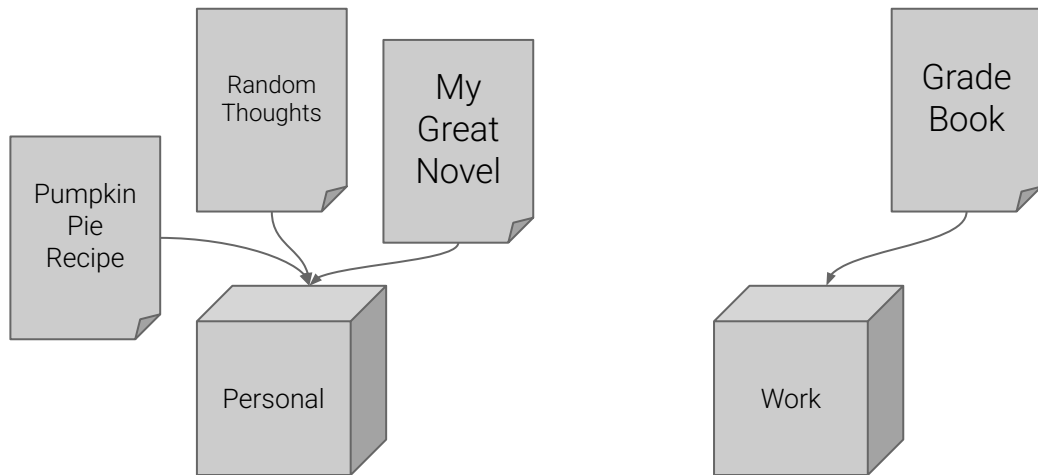
Computers store data in *files*.



Directories

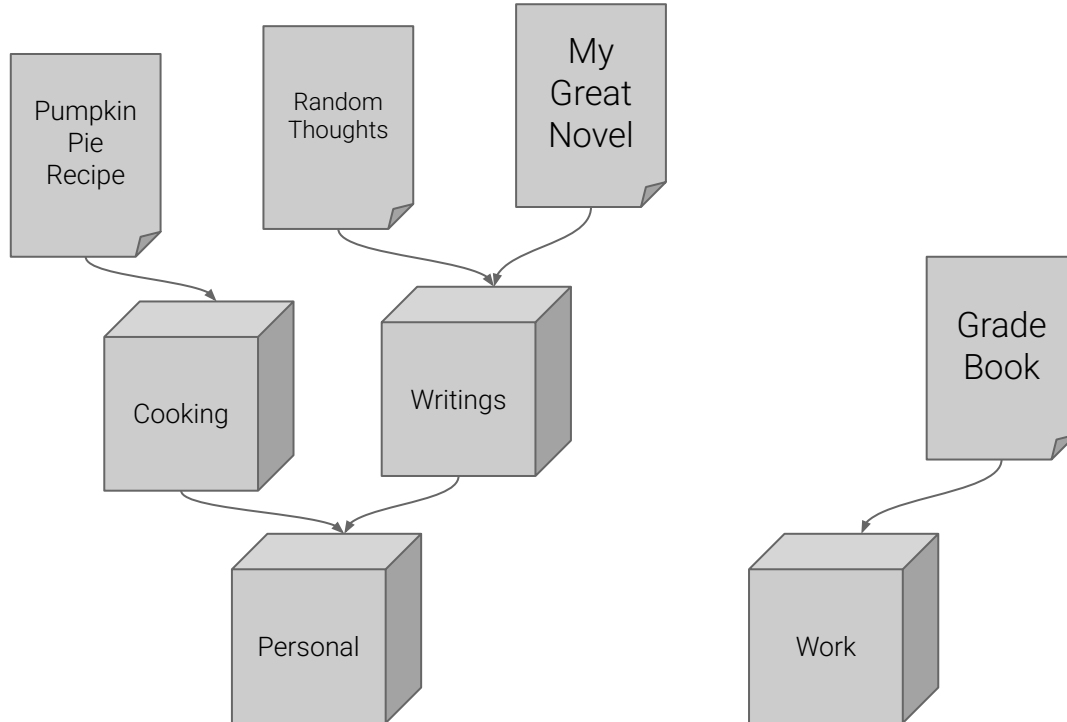
You can't just have files laying around randomly... You'd never be able to find anything!

Computers store files in *directories* in the filesystem. Each directory can contain many files, like a box.



Nesting Directories

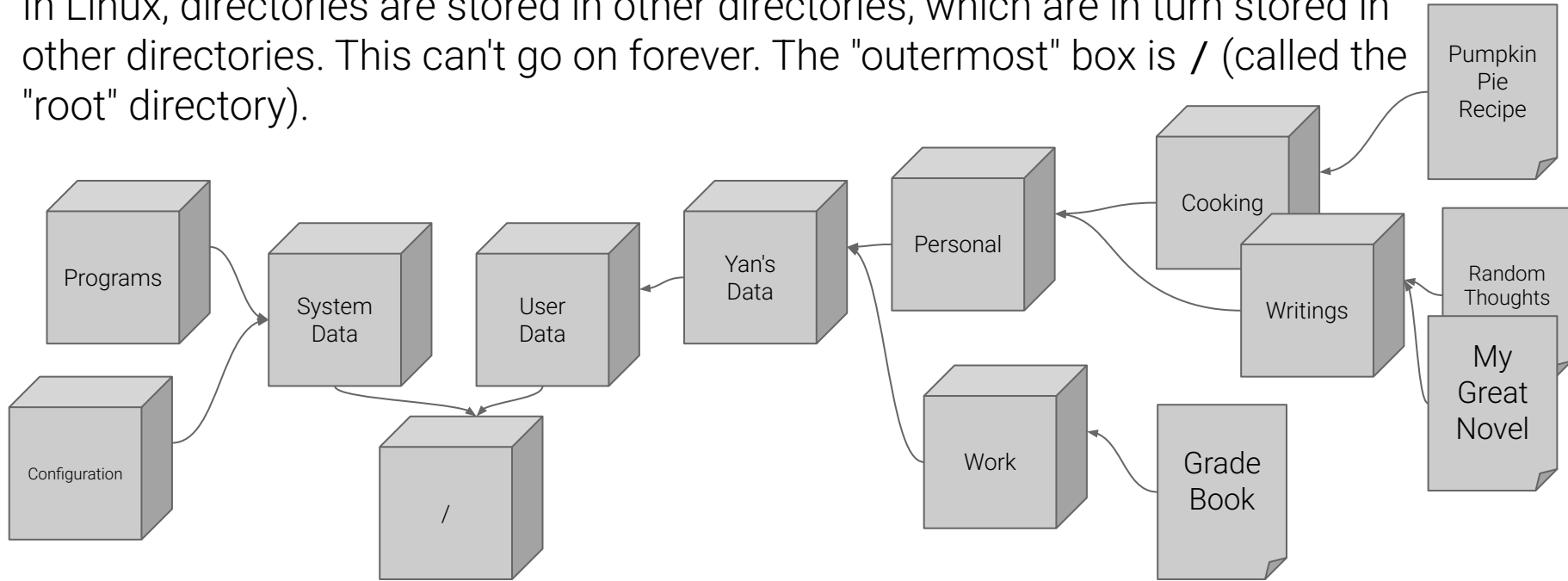
For better organization, directories can be stored in other directories!



The "root" Directory

"There's always a bigger box"... until there isn't.

In Linux, directories are stored in other directories, which are in turn stored in other directories. This can't go on forever. The "outermost" box is / (called the "root" directory).



A "path"

The, well, path through these nested directories that the computer should take to find your file or directory is called a "path".

For example:

/UserData/YansData/Personal/Cooking/PumpkinPieRecipe

Of course, on Linux, typical names of directories are not so descriptive...

The Typical File System Layout

Linux has a standardized file system layout. Here are some examples:

<code>/</code>	The "anchor" of the filesystem. Pronounced "root".
<code>/usr</code>	The Unix System Resource. Contains all the system files.
<code>/usr/bin</code>	Executable files for programs installed on the computer.
<code>/usr/lib</code>	Shared libraries for use by programs on the computer.
<code>/usr/share</code>	Program resources (icons, art assets, etc).
<code>/etc</code>	System configuration.
<code>/var</code>	Logs, caches, etc.
<code>/home</code>	User-owned data.
<code>/home/hacker</code>	Data owned by you in the pwn.college infrastructure.
<code>/proc</code>	Runtime process data.
<code>/tmp</code>	Temporary data storage.

If you're used to Windows (which traditionally has different file systems at different anchor points `C:\`, `D:\`, `E:\`, etc.), this can be a bit of an adjustment.

Navigating The File System

Each process has a "current working directory": the directory it's looking into at the moment. You can view it with the **pwd** command (and it usually shows in your prompt) and change it with the **cd** command.

```
yans@asdf:~$ cd /  
yans@asdf:/$ cd home  
yans@asdf:/home$ pwd  
/home  
yans@asdf:/home$ cd hacker  
yans@asdf:/home/hacker$ pwd  
/home/hacker  
yans@asdf:/home/hacker$
```

You can list the files in a directory using the **ls** command. With no arguments, it will list the files in the current directory.

```
yans@asdf:/$ ls  
bin boot challenge dev etc flag home lib lib32 lib64 libx32 media mnt nix opt proc root run sbin srv sys tmp usr var  
yans@asdf:/$ ls /home  
hacker  
yans@asdf:/$
```

Specifying paths...

There are two ways to specify paths:

Absolute Paths start with `/`, such as `/usr`, `/home/yans/flags/TOPSECRET`, etc.

Relative Paths *don't* start with `/`, and are relative to the current working directory.

```
yans@asdf ~ $ ls /home/yans/flags
TOPSECRET
yans@asdf ~ $ pwd
/home/yans
yans@asdf ~ $ cat TOPSECRET
cat: TOPSECRET: No such file or directory
yans@asdf ~ $ cd flags
yans@asdf ~/flags $ pwd
/home/yans/flags
yans@asdf ~/flags $ cat TOPSECRET
pwn_college{1337}
yans@asdf ~/flags $
```

Absolute path!

Relative path!

Relative path!

Relative path!

Closer look: Paths

A "path" contains:

- Possible leading "/" to specify that the path is absolute (starts at the root).
- Directory names, followed by "/" to reference resources "inside" a directory.
- A ".", signifying "current directory".
- A "..", signifying "the directory that the current directory lives in".
- A file name at the end of the path, referencing a file with that name.

```
yans@asdf /usr/bin $ cat /home/yans/flags/TOPSECRET
pwn_college{1337}
yans@asdf /usr/bin $ cat ../../home/yans/flags/TOPSECRET
pwn_college{1337}
yans@asdf /usr/bin $ cat ../../usr/../lib/../bin/../../home/yans/flags/TOPSECRET
pwn_college{1337}
yans@asdf /usr/bin $
```