Lecture 1B

Introduction to the File System

Eric Brauer

Introduction to the File System

History Lesson II

Recall that Linux descended from an operating system designed for mainframes. That is, a single computer being used by many users simultaneously. In the 1980s, personal computers (PCs) became available, which as you can imagine was a revolutionary leap forward! In this case, you had one user working on one computer. Eventually DOS became the most common operating systems for these PCs, and Microsoft Windows was built on (and eventually replaced) DOS.

A Useful Analogy

- Windows = a single family home
- Linux (especially Matrix) = an apartment building

On Matrix, you have a space which is 'yours' for as long as you are a student. You have control over this space. On the other hand, there is a lot that you *don't* have control over. The plumbing, the heat, the HVAC, and so on. There is an administrator/superintendant that has authority over these spaces, and who has permission to make changes. The super has authority to create, modify and remove user accounts.

Some Useful Features of the Shell

These aren't anything you'll be tested on, but they will absolutely save you time. **Please remember this:** the shell is a place where accuracy is crucial. A typo will cause your command to not work, or to do something you didn't want it to. The *Tab* key is an excellent way of troubleshooting. Here's how it works:

Let's go back to the previous example. You are in **/home/student**. You type in `cd D` and then press tab once. You won't see anything occur. You press tab *twice*. You should see a list of all the matching patterns, in this case both **Desktop** and **Documents** will be listed since they both contain **D**. You continue by typing in `o` and press tab once again. Your path will be autocompleted, and you should see the complete command `cd Documents`. Now you can press Enter to complete the command.

Please note that the auto-complete doesn't unfortunately work when you're working on Assignments. :(But in the real world, using Tab will speed up your work and also provide a 'sanity check' on your commands. If something should be auto-completing and it isn't, 90% of the time it's because you've made a mistake!

> - **Ctrl+a**: Move to start of command
> - **Ctrl+e**: Move to end of command
> - **Up & Down**: Command History
> - **Tab**: Autocomplete a Path (your first sanity check!)
> - **Ctrl+c**: Cancel
> - **Ctrl+d**: Logout
> - **Ctrl+r**: Search your command history
> - **Ctrl+l**: Clears the screen

Explore Your "Apartment"

- Use `ls` to view your home folder.

- Use `cd public_html` to change your current location to `public_html`.
- Use `cd ..` to return to your home.
- Use `pwd` to see your current location.

Explore the "Common Space"

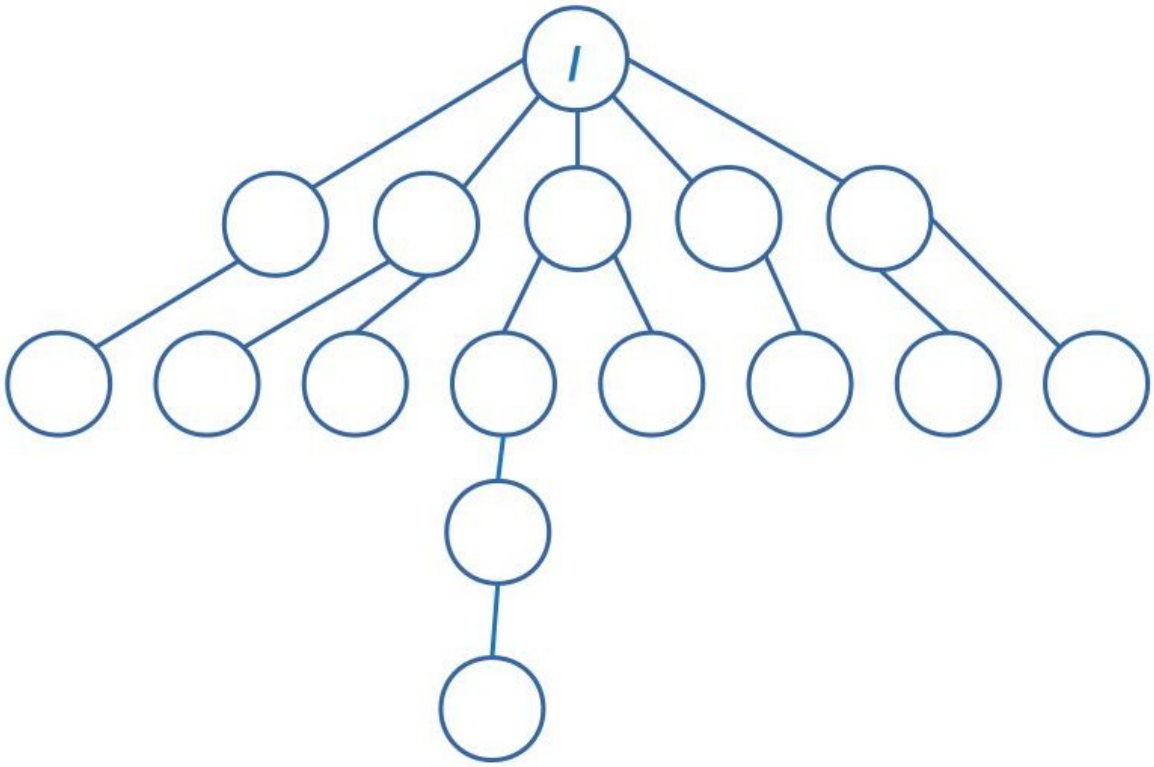
The places *outside* your home folder are common spaces. You can see these places, but you can't modify them.

- Use `cd /bin` to travel to one such location.
- Type `ls` to view its contents.
- Find `ls` in the contents of this folder.

`ls` is a program. And its location is in `/bin`! So what would happen if a malicious person decided to modify `ls` for everyone?

Navigating the Linux Filesystem

You don't know it yet, but we've just demonstrated two different types of filepaths. To see how this works, let's use a different type of diagram.



tree diagram

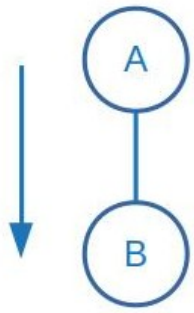
Here are Some Rules

The rules for this 'game' are simple.

- Every dot is a *directory*. Moving along any line will take one 'turn.'
- We want to *minimize* the number of turns we take.
- Every directory has *one parent*. (Except for the very top).
- Directories don't have *unique* names. So there might be *many* directories with the name `bin`.
- We need to be able to move from *anywhere* in the tree to *anywhere else*.
- We need to always be *specific* when we are describing our moves.
- When choosing our path, we can start from our *current location*, or from the *top* of the tree.

Moving Into a Subdirectory

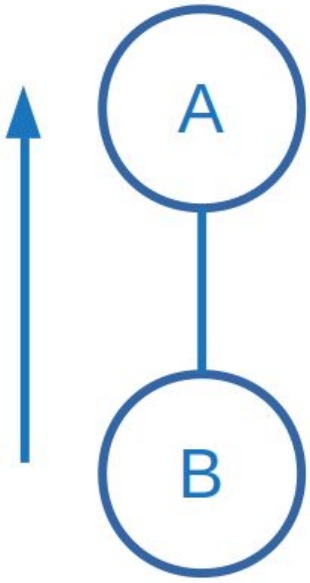
- Assume we are moving from point **A** to point **B**. This movement takes one "turn."
- The command to do this is: `cd B` .



to subdirectory

Moving Into a Parent Directory

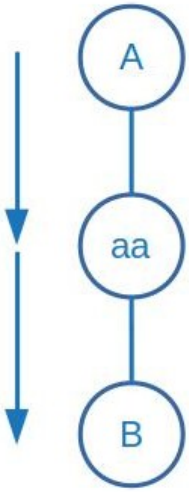
- Assume we are moving from point **B** to point **A** again. This movement takes one turn.
- Since `B` only has one parent, we can save ourselves some typing by using **..**
- The command to do this is: `cd ..` .



to parent

Moving Into a Sub-Subdirectory

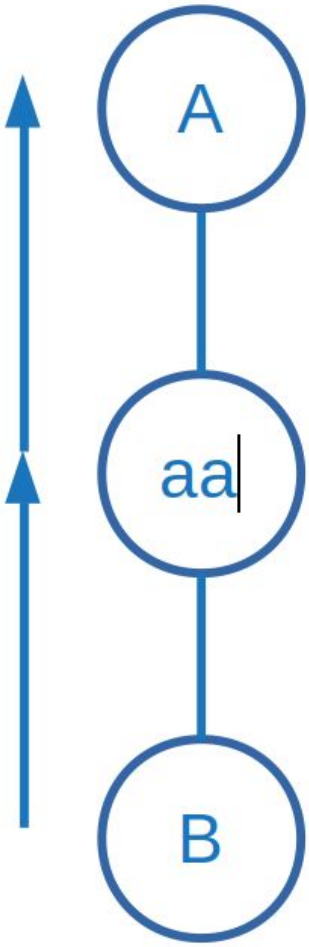
- Assume we are moving from point **A** to point **B** again. This movement takes two turns.
- You could enter the command `cd aa` , and then press enter, and then enter `cd B` . But this is a waste of time. Instead, we can enter: `cd aa/B` .
- The **/** here is indicating that there are two turns: moving into directory `aa` and then moving into directory `B` from `aa` .



to subdirectory2

Moving Into the Parent of a Parent Directory

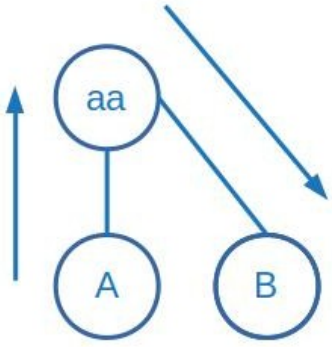
- Once again, we are moving from point **B** to point **A**. We are combining two turns in one command.
- The command is: `cd ../..`.



to parent2

Lateral Movement

- In this case, **A** and **B** have a common parent. We must move up to the directory `aa` before moving to **B**. This movement takes two turns.
- The command is: `cd ../B` .

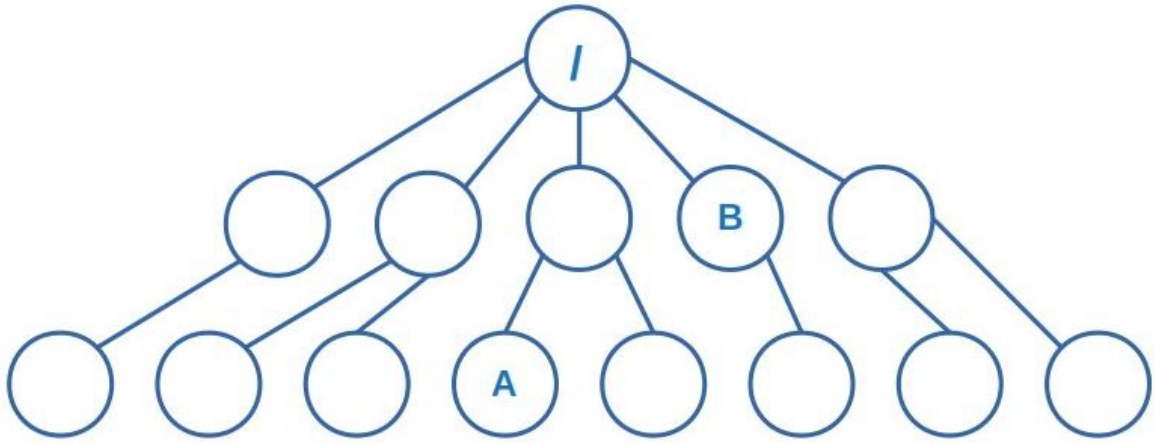


lateral movement

Start From Root (/)

- If we were to move up the tree from **A**, the command would be `cd ../../B` and it would take three turns.
- Instead, we should start from the top, which will take only one turn.
- The top is called **root** (**/**).
- To indicate that we are starting from the top instead of starting from **A**, we start our command with **/**.
- The correct command is: `cd /B` .

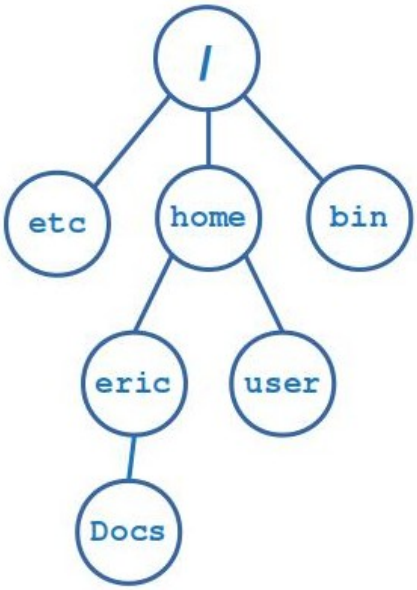
Note: There usually isn't a /B in Linux systems, so try this with `cd /bin` instead.



absolute

Navigating the Linux Filesystem II

- Now let's replace labels like **A** and **B** with some proper directory names.
- Refer back to the navigation commands you used in Assignment 1. Do they match up?



proper names

The Tree Command

Compare the tree diagrams above with the output from `tree` :

```
/
├── sample_dir
│   ├── faculty
│   ├── markham
│   ├── oxford
│   │   ├── programming
│   │   ├── report
│   │   └── security
│   ├── stenton
│   │   ├── gen_ed
│   │   └── lib_arts
│   └── todays-example
└── users
    ├── ebrau
    └── sam
```

- **Challenge: re-draw this in the 'tree' format**
- `/` has two subdirectories (two dots)
- `sample_dir` has **five** subdirectories.

Please feel free to practice navigation by entering the following command:

```
cd ~eric.brauer/uli101
```

Summary

Terminology

- *parent*
- *subdirectory*
- *home*
- *root*
- *user*

Commands

- `pwd`
- `cd`
- `ls`
- `tree`