Unix Tools	Lecture 1
CS114 Fall 2002	Monday, September 30, 2002

Introduction to Unix

When you connect to a Unix machine (either through telnet, or by walking up to a machine running Unix), you will be presented with a login prompt that looks like this:

```
login:
```

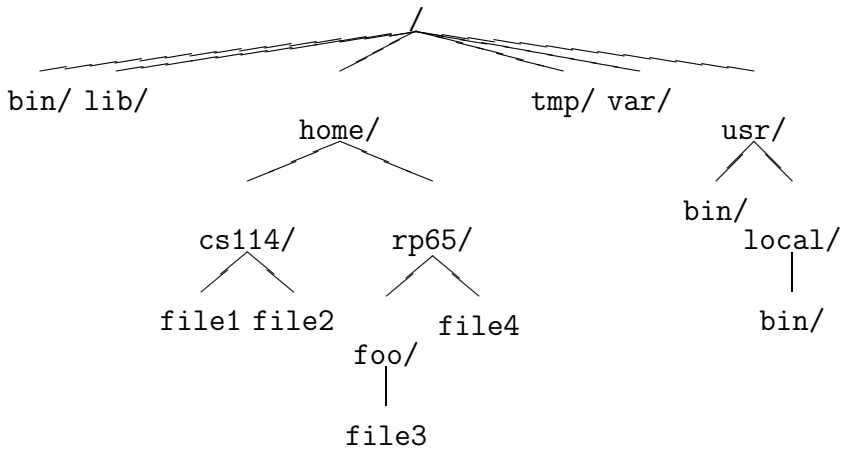
at which point the system is asking you for your *username*, the identification by which you are known to the system. By default, on `babbage`, your username is just your NetID. The system then asks you for your password. If you enter it succesfully, you end up at the Unix prompt, which on `babbage` looks like:

```
babbage%
```

The prompt is where you interact with Unix. Typically, you issue commands, possibly with arguments. Some commands start programs, some commands just manage the environment, some commands give you information.

Everything under Unix lives in *files*: programs, data, etc. These files are distributed amongst different directories. Directories contain files, and may also contain other directories. As such, directories form a tree-like hierarchy starting from a base directory call the *root directory*. This hierarchy is often called the *filesystem*. (This filesystem may be on disk, it may be accross the network, in fact we will not worry exactly where the files are physically stored.) Each user on the system gets a special directory, called their *home directory*, where they can put their files (and create subdirectories, etc).

Here's a typical filesystem (directories are indicated by names trailing with a `/`:



In words, directory / contains 6 directories, `bin/`, `lib/`, `home/`, `tmp/`, `var/`, and `usr/`. Of those, `home/` and and `usr/` have content indicated in the graph. If we look at `home/`, it contains two directories, `cs114/` and `rp65/`, where `cs114/` contains two files (`file1`) and `file2`), and `rp65/` contains a directory and a file (`foo/` and `file4`).

Paths

A *path* is used to give the location of a file or directory one is interested in. There are two types of path:

- An *absolute path* gives the path to a file or directory starting from the root path, which is indicated by `/`. Hence, a path such as `/home/rp65/foo/my-file` is an absolute path. An absolute path is a nonambiguous way of indicated where a file or directory can be found.
- A *relative path* gives the path to a file relative to some directory. For instance, relative to the directory `/home/rp65/`, the path to file `also-my-file` is `../cs114/also-my-file`.

The following sequences of characters have a special meaning in a path. The sequence `.` simply indicates the current directory the path points at. For example, `/home/./rp65/` is equivalent to `/home/rp65/`. The single `.` directory is useful in relative paths, to indicate the current directory: the path `.` relative to directory `/home/` is just `/home/` itself. The sequence `..` goes up one directory. Thus, the path `../cs114/` relative to `/home/rp65/` indicates the directory `/home/cs114/`. Similarly, the path `/home/rp65/../cs114/` indicates the directory `/home/cs114/`

Relative paths are always given relative to a directory called the *current directory*. When you login, your current directory is your home directory, namely `/home/`*`username`*`/`.

A shortcut way to refer to anyone's home directory is to use the special prefix ∼*username*. This is an absolute path. For instance, the home directory of user `cs114` can be referred to as ∼`cs114/`. To refer to your own home directory, you can use ∼, in ∼`/foo/bar`.

Some commands

The command `cd` is used to change your current directory. If you just type in `cd` by itself, it will change you back to your initial directory (usually, your home directory). If you specify a path, as in `cd /home/rp65/` or `cd ..`, it will change your current directory into the directory that you specify.

If you happen to lose track of what your current directory is, the command `pwd` will print it out for you.

Finally, the command `ls` will display the content of the current directory (i.e. it will list all the files and all the directories it contains). If you specify an argument, `ls` *directory*, then `ls` will display the content of *directory*.