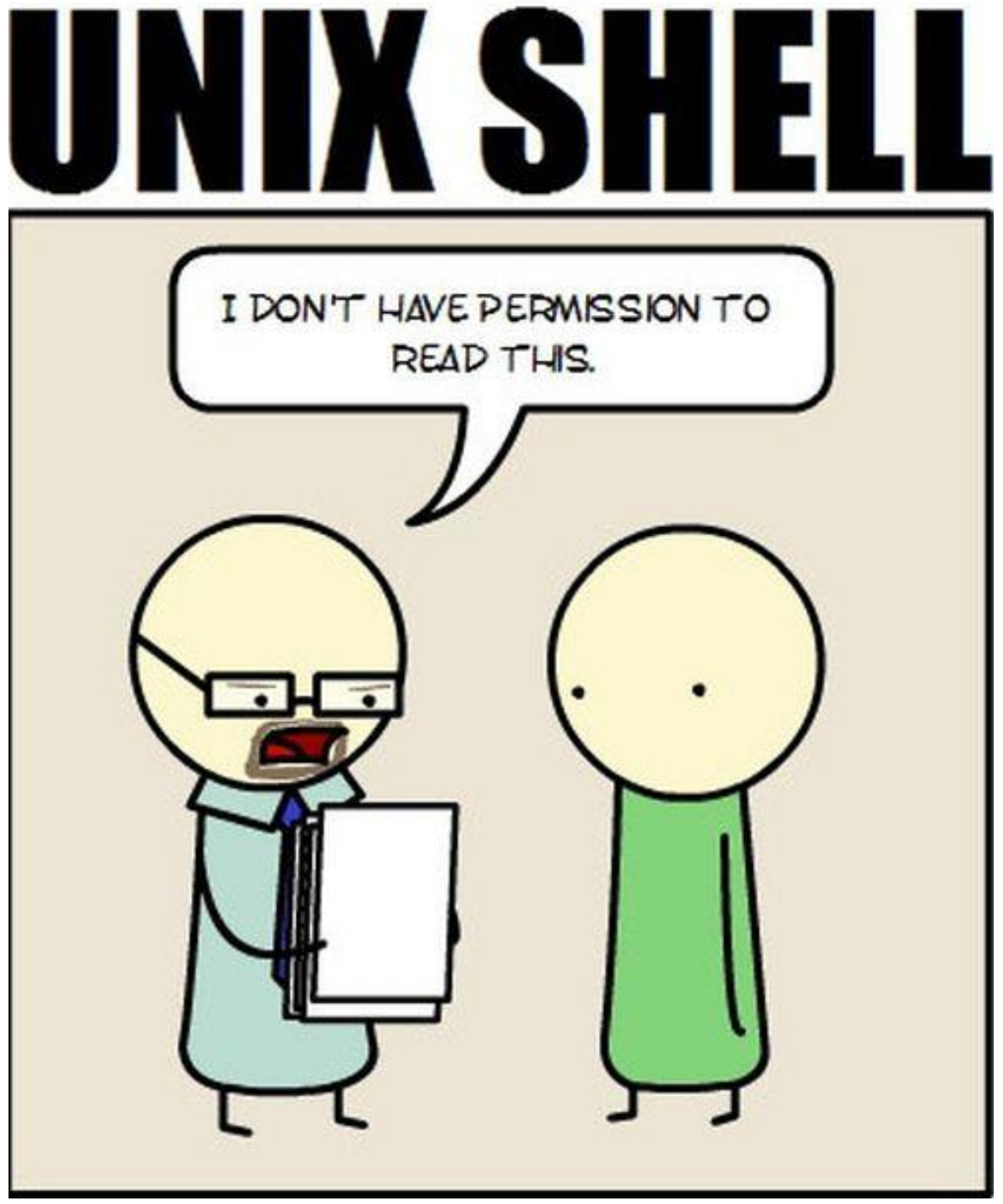
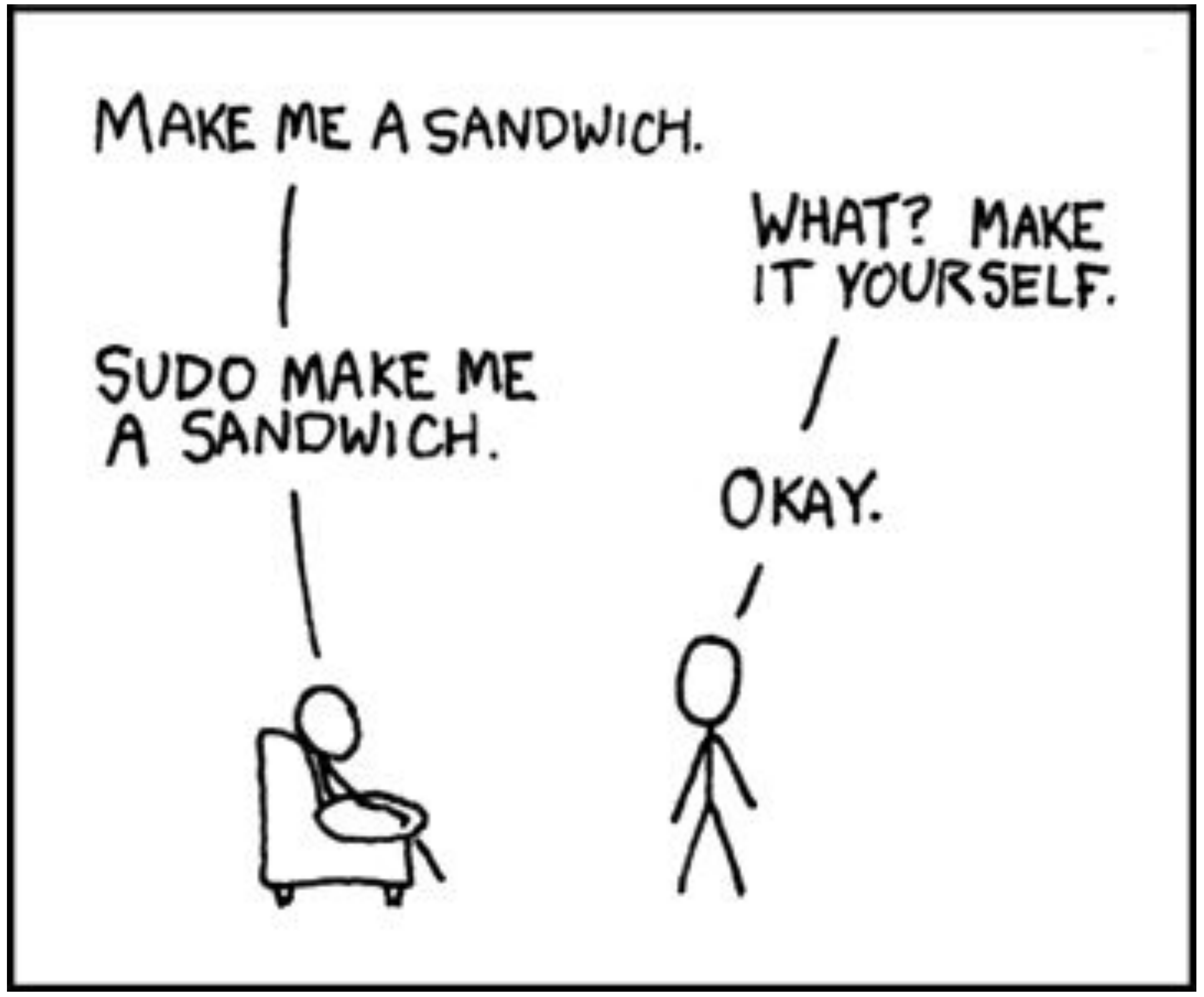
Data analysis with the shell

Spring 2025, Week 3
January 31, 2025





Check-in

Outline

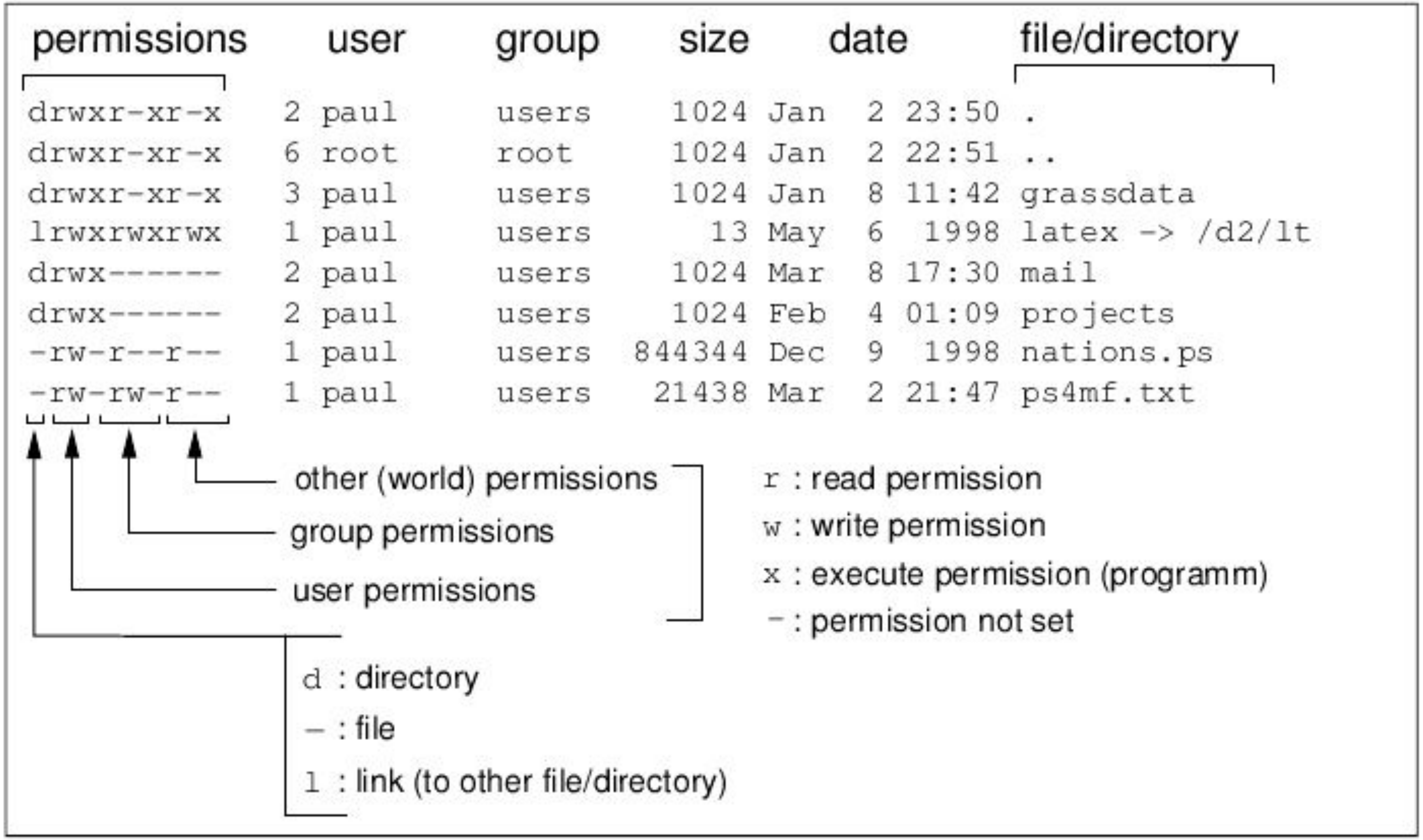
- Installing/running programs
- ‘for’ loops
- Shell scripts
- Assignment (in class)

Installing/running programs

Computers only understand their native machine language

2 types of programming languages

Permissions (‘ls -l’)



```
drwxr-xr-x   2 paul     users     1024 Jan  2 23:50 .
drwxr-xr-x   6 root     root      1024 Jan  2 22:51 ..
drwxr-xr-x   3 paul     users     1024 Jan  8 11:42 grassdata
lrwxrwxrwx   1 paul     users       13 May  6  1998 latex -> /d2/lt
drwx------   2 paul     users     1024 Mar  8 17:30 mail
drwx------   2 paul     users     1024 Feb  4 01:09 projects
-rw-r--r--   1 paul     users   844344 Dec  9  1998 nations.ps
-rw-rw-r--   1 paul     users    21438 Mar  2 21:47 ps4mf.txt
```

`r` : read permission
`w` : write permission
`x` : execute permission (programm)
`-` : permission not set

`d` : directory
`-` : file
`l` : link (to other file/directory)

Changing Permissions

`chmod` (change mode)

Add execute for User: `chmod u+x file.txt`

Add read and write for Group: `chmod g+rw file.txt`

Remove write and execute for Other: `chmod o-wx file.txt`

All three in one command: `chmod u+x,g+rw,o-wx file.txt`

$PATH

- A list of directories
  - Locations your computer looks for command-line software
- Searched in the order listed
- To view: `echo $PATH`
- To add a directory: `PATH=“$PATH:path/to/new/dir”`

Recommendation

Create three directories in your home directory:

1. *`scripts`* (your own custom scripts, PCfB p. 85-88)
2. *`programs`* (ready-to-use code downloaded from others)
3. *`source`* (source code that needs to be compiled)

Add *`scripts`* and *`programs`* to $PATH

Dependencies

Installing program demo

‘for’ loops

`for` loop

- Simple, but powerful way to repeatedly execute the same commands for different files, parameter values, etc.
- Can be included in scripts or run directly on command line

Basic syntax

```
for file in *.sh; do chmod u+x $file; done
```

Directory contents:

```
script1.sh
script2.sh
script3.py
script4.sh
```

```
for file in *.sh; do chmod u+x $file; done
```

Directory contents:

```
script1.sh
script2.sh
script3.py
script4.sh
```

Equivalent to running:

```
chmod u+x script1.sh
chmod u+x script2.sh
chmod u+x script4.sh
```

‘for’ loop examples

```
for file in *.fasta;
do print_fasta_seq_lengths.py $file;
done
```

```
for file in *txt; do cp $file copy_$file;
mkdir dir_$file; mv $file dir_$file; done
```

‘for’ loop demo

Shell scripts

Why use shell scripts?

1. Automate a series of commands
   a. particularly useful when each command takes a long time to run
2. Record of commands run
3. Easy format for rerunning commands

Two ways to specify the interpreter to use

Specify interpreter inside script

File extensions

- Recommended (but not required) to save script with specific file extension
- Allows recognition from file name
- Syntax-specific coloring in text editors
- For shell script:

Shell + Regexp method

1. Use the shell to generate a list of files/directories
2. Use regular expressions within your text editor to turn those file/directory names into a list of commands

Shell script demo