Bash basics

Sorting your downloads with Bash

About me

- Programming in some way for 7 years
- Experience with Java, Visual Basic, C, C++, and Bash
- Designed game engines, Minecraft mods, operating systems

What is BASH?

- Shell scripting language
- Default shell for Unix and Linux systems
- Bourne Again SHell
- Used mainly for automation and lower level OS interaction

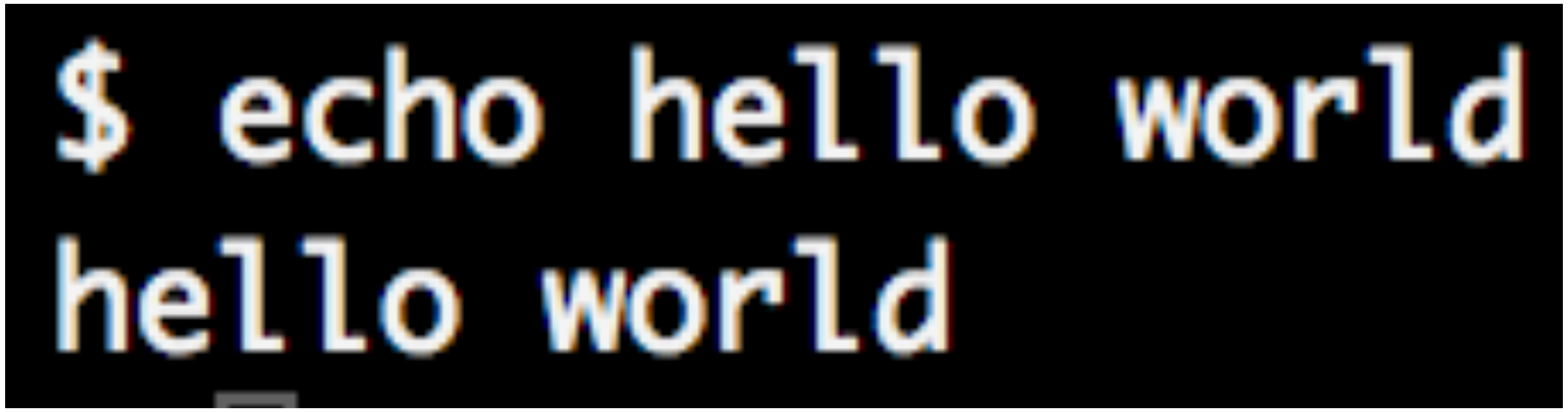
Where is BASH?

- Computers
- Smartphones
- Cars
- mp3 players
- Home appliances

Hello world!

echo - prints out argument passed to it

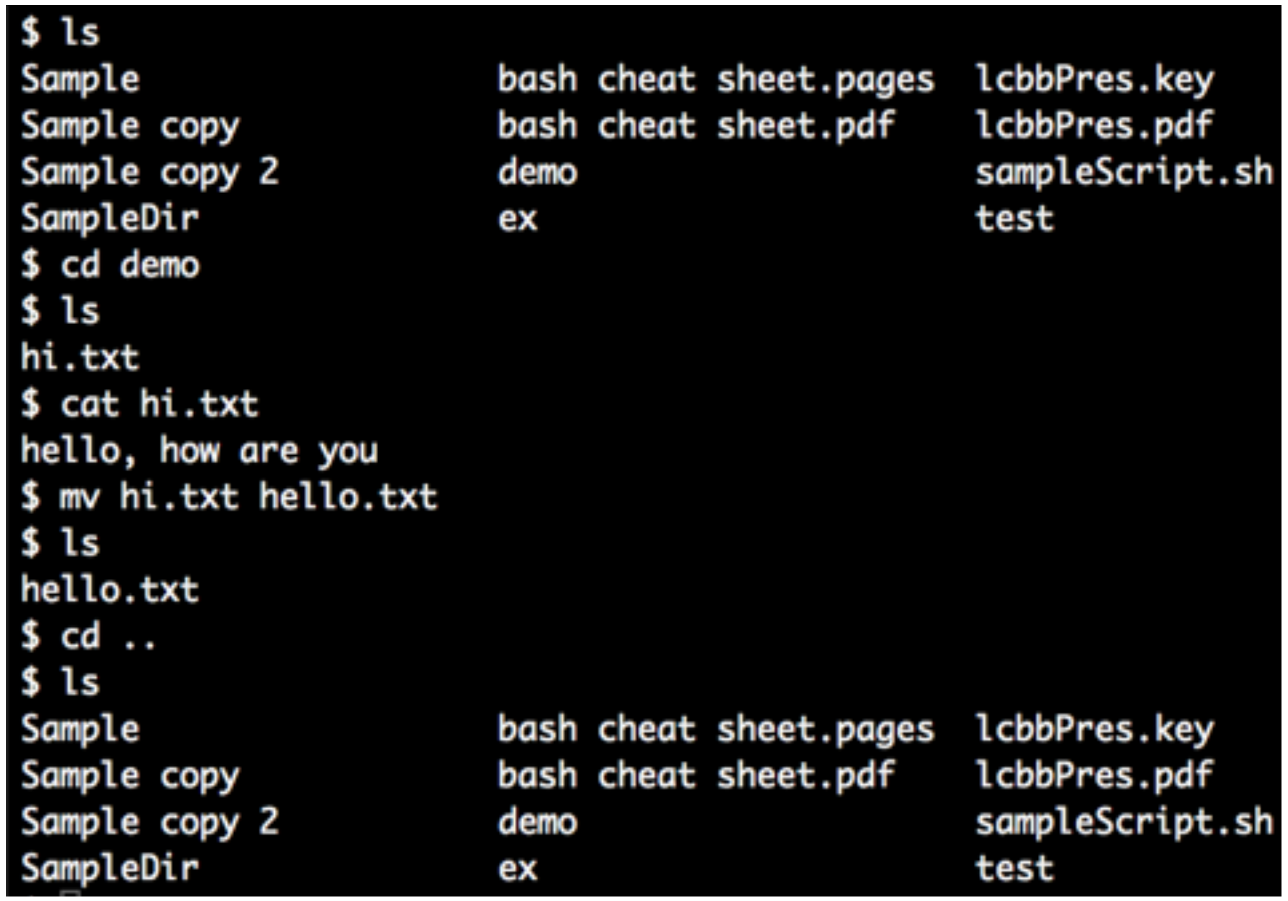
To print “Hello world” enter the command **`echo Hello world`**



```
$ echo hello world
hello world
```

Navigating your file system with BASH

- `cd` to change current directory
- `ls` to list files in current directory
- `cat` to read file
- `mv` moves files and directories
- `rm` deletes files



```
$ ls
Sample                   bash cheat sheet.pages   lcbbPres.key
Sample copy              bash cheat sheet.pdf     lcbbPres.pdf
Sample copy 2            demo                     sampleScript.sh
SampleDir                ex                       test
$ cd demo
$ ls
hi.txt
$ cat hi.txt
hello, how are you
$ mv hi.txt hello.txt
$ ls
hello.txt
$ cd ..
$ ls
Sample                   bash cheat sheet.pages   lcbbPres.key
Sample copy              bash cheat sheet.pdf     lcbbPres.pdf
Sample copy 2            demo                     sampleScript.sh
SampleDir                ex                       test
```

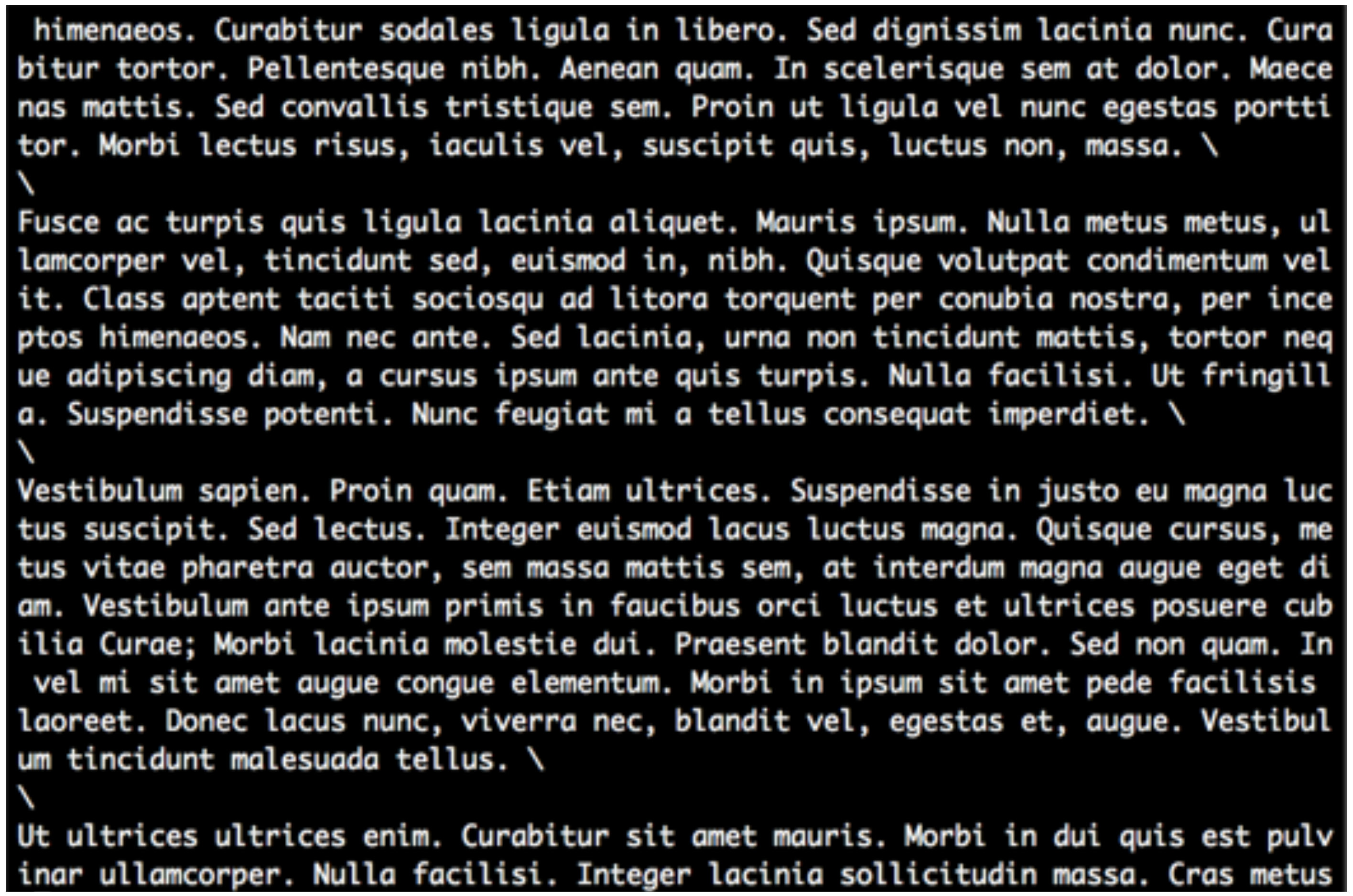
- All part of the POSIX standard commands that exists across most operating systems
- . is the current directory, .. is the directory above the current one

Bash shortcuts

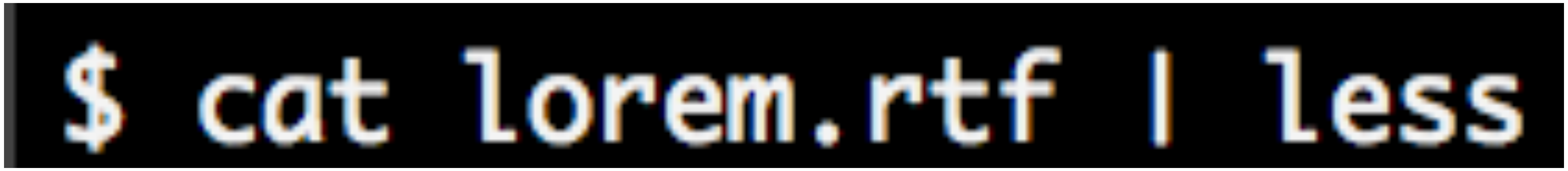
- Hit tab at any point to have bash attempt to autocomplete a command or file name
- Hold control + c during a long process to terminate it
- Use up and down arrow keys to find previously entered commands

Linking it all together

- **`less`** - pagifies input
- pipes - `|`



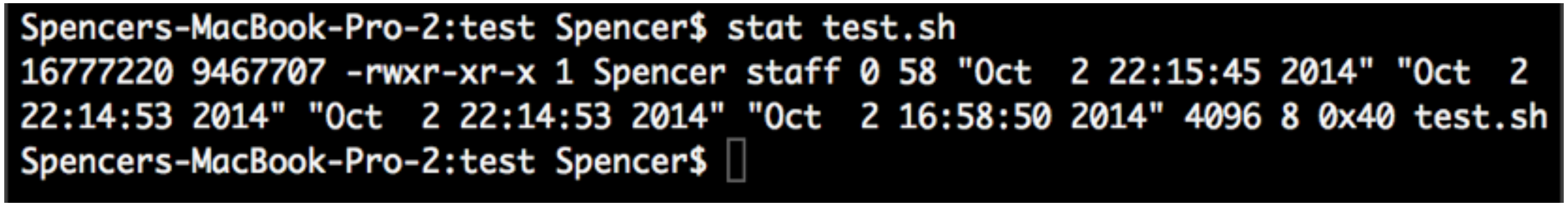
```
 himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Cura
bitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maece
nas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas portti
tor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. \
\
Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ul
lamcorper vel, tincidunt sed, euismod in, nibh. Quisque volutpat condimentum vel
it. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per ince
ptos himenaeos. Nam nec ante. Sed lacinia, urna non tincidunt mattis, tortor neq
ue adipiscing diam, a cursus ipsum ante quis turpis. Nulla facilisi. Ut fringill
a. Suspendisse potenti. Nunc feugiat mi a tellus consequat imperdiet. \
\
Vestibulum sapien. Proin quam. Etiam ultrices. Suspendisse in justo eu magna luc
tus suscipit. Sed lectus. Integer euismod lacus luctus magna. Quisque cursus, me
tus vitae pharetra auctor, sem massa mattis sem, at interdum magna augue eget di
am. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cub
ilia Curae; Morbi lacinia molestie dui. Praesent blandit dolor. Sed non quam. In
 vel mi sit amet augue congue elementum. Morbi in ipsum sit amet pede facilisis
laoreet. Donec lacus nunc, viverra nec, blandit vel, egestas et, augue. Vestibul
um tincidunt malesuada tellus. \
\
Ut ultrices ultrices enim. Curabitur sit amet mauris. Morbi in dui quis est pulv
inar ullamcorper. Nulla facilisi. Integer lacinia sollicitudin massa. Cras metus
```



```
$ cat lorem.rtf | less
```

Getting file access times with stat

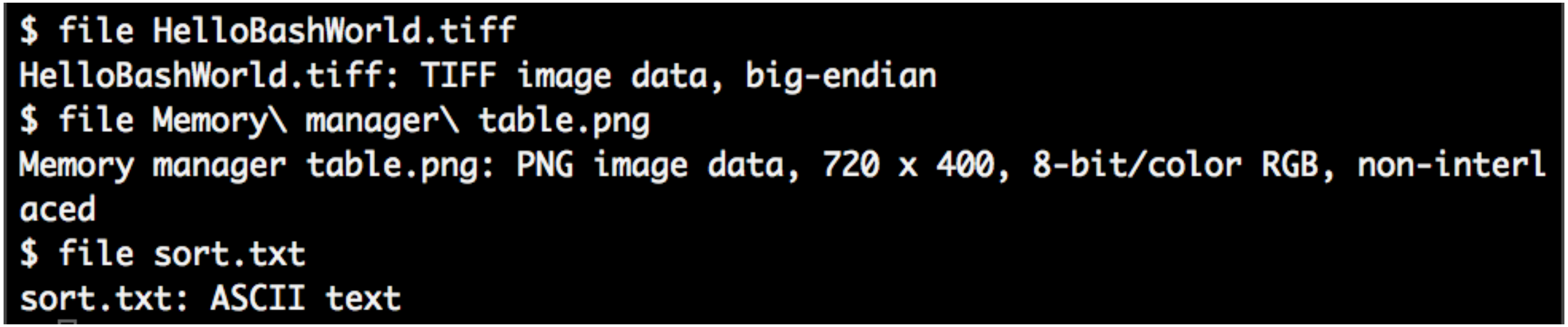
- Gives information about file
- Can output access time since epoch, access date and time in english, permissions, the creator



```
Spencers-MacBook-Pro-2:test Spencer$ stat test.sh
16777220 9467707 -rwxr-xr-x 1 Spencer staff 0 58 "Oct  2 22:15:45 2014" "Oct  2
22:14:53 2014" "Oct  2 22:14:53 2014" "Oct  2 16:58:50 2014" 4096 8 0x40 test.sh
Spencers-MacBook-Pro-2:test Spencer$ 
```

Analyzing files with “file”

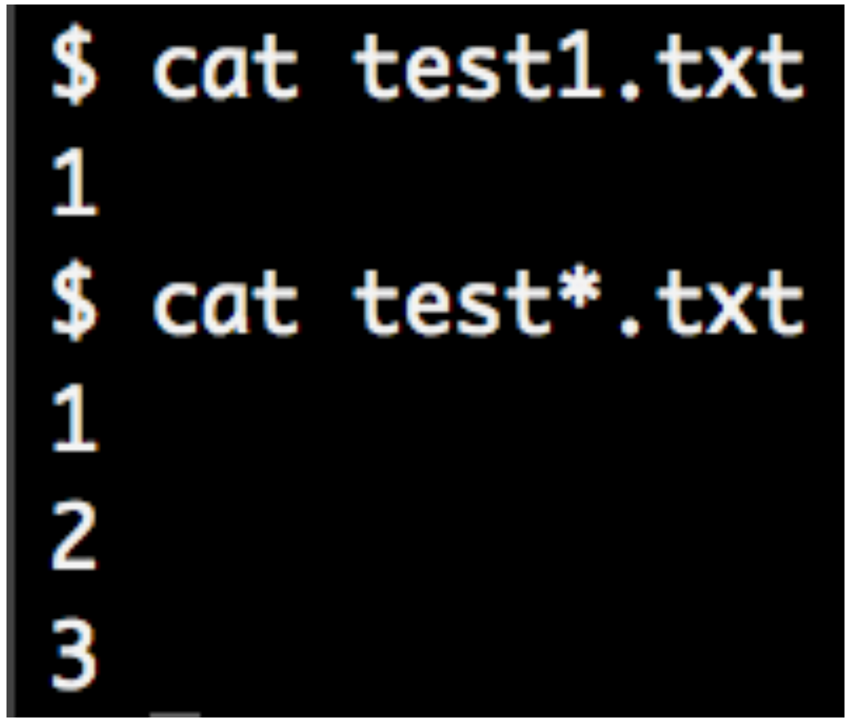
- Outputs information about a given file
- Capable of determining the resolution and color depth of images
- Identifies audio files and their corresponding metadata
- Able to identify archives
- Determines type by looking at the contents of the file
- Use file on “Space” to find
  out what type of file it is.



```
$ file HelloBashWorld.tiff
HelloBashWorld.tiff: TIFF image data, big-endian
$ file Memory\ manager\ table.png
Memory manager table.png: PNG image data, 720 x 400, 8-bit/color RGB, non-interl
aced
$ file sort.txt
sort.txt: ASCII text
```

Wildcards

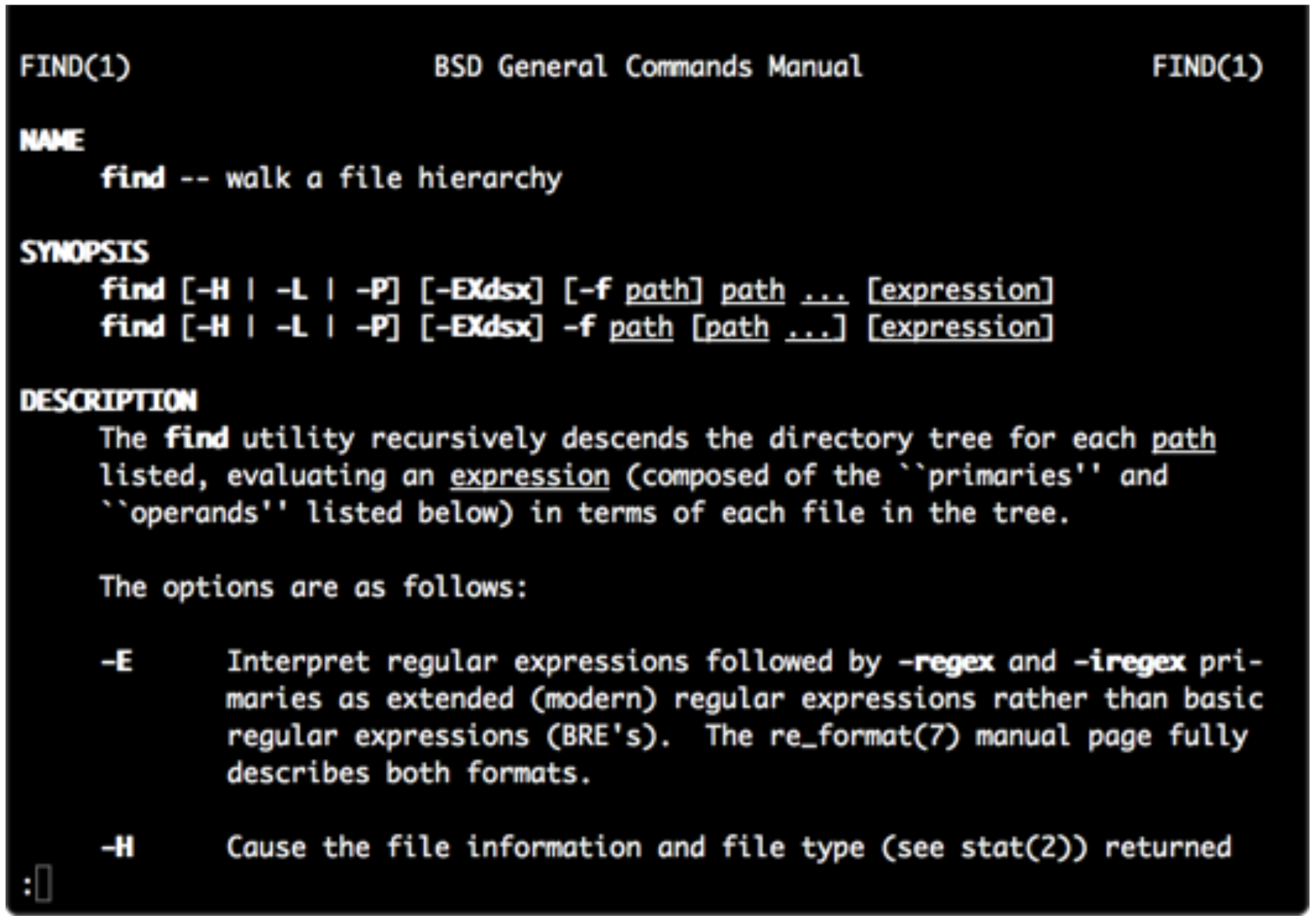
- Usually “*”
- A wildcard signifies that anything can go there
- * can represent anything, and file* represents “file” with any suffix



```
$ cat test1.txt
1
$ cat test*.txt
1
2
3
```

Man pages

- Man is a manual built into many unix and linux systems
- Can be used to find syntax and usage of Bash / POSIX commands and functions, as well as other programs that add man pages
- Type in **`man file`** to get the manual entry for the **`file`** command
- Down and up arrows to scroll, Q key exits the man page



```
FIND(1)                 BSD General Commands Manual                FIND(1)

NAME
     find -- walk a file hierarchy

SYNOPSIS
     find [-H | -L | -P] [-EXdsx] [-f path] path ... [expression]
     find [-H | -L | -P] [-EXdsx] -f path [path ...] [expression]

DESCRIPTION
     The find utility recursively descends the directory tree for each path
     listed, evaluating an expression (composed of the ``primaries'' and
     ``operands'' listed below) in terms of each file in the tree.

     The options are as follows:

     -E      Interpret regular expressions followed by -regex and -iregex pri-
             maries as extended (modern) regular expressions rather than basic
             regular expressions (BRE's).  The re_format(7) manual page fully
             describes both formats.

     -H      Cause the file information and file type (see stat(2)) returned
:
```

Writing a script

- Enter all commands in order in a text document
- Shell script files usually end with .sh
- Start shell script with **`bash myScript.sh`** or **`./myScript.sh`**

Variables

- set variable with **`someVar=something`**
- get variable with **`$someVar`**
- set variable to user input with **`read someVar`**
- **`echo $someVar`** prints the value of someVar

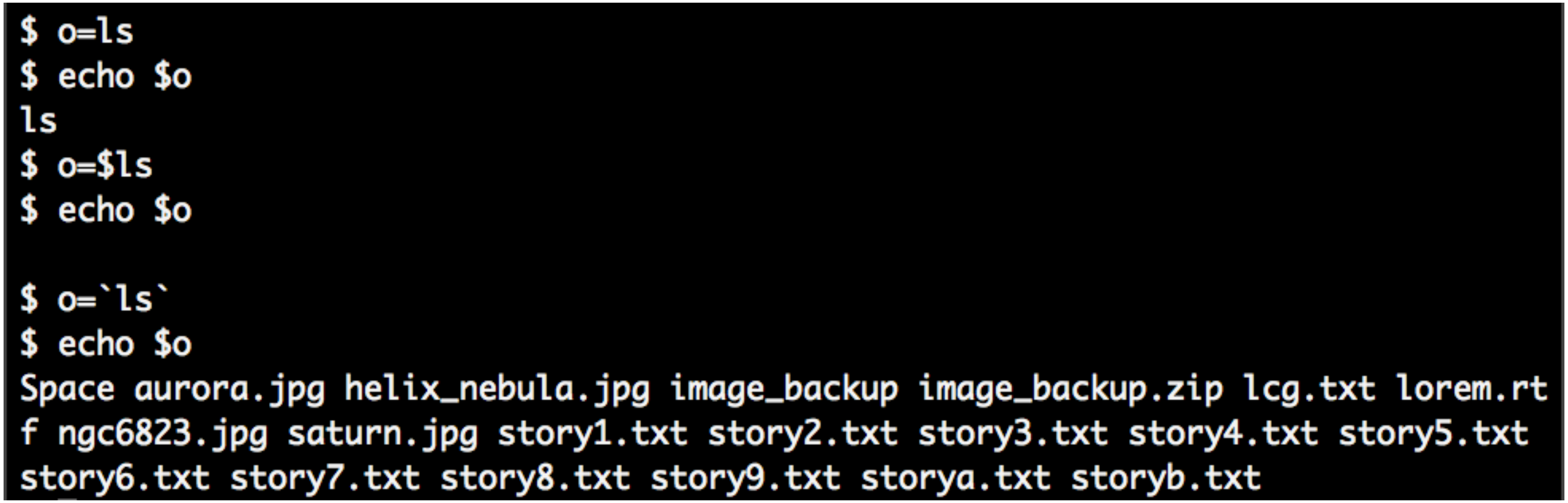
Experiment

Set a variable to the output of ls, and print out the value of that variable

What’s wrong?

When setting a variable to the output of a program, you must wrap it in ``

The proper syntax is **`` v=`ls` ``**



```
$ o=ls
$ echo $o
ls
$ o=$ls
$ echo $o

$ o=`ls`
$ echo $o
Space aurora.jpg helix_nebula.jpg image_backup image_backup.zip lcg.txt lorem.rt
f ngc6823.jpg saturn.jpg story1.txt story2.txt story3.txt story4.txt story5.txt
story6.txt story7.txt story8.txt story9.txt storya.txt storyb.txt
```

Find

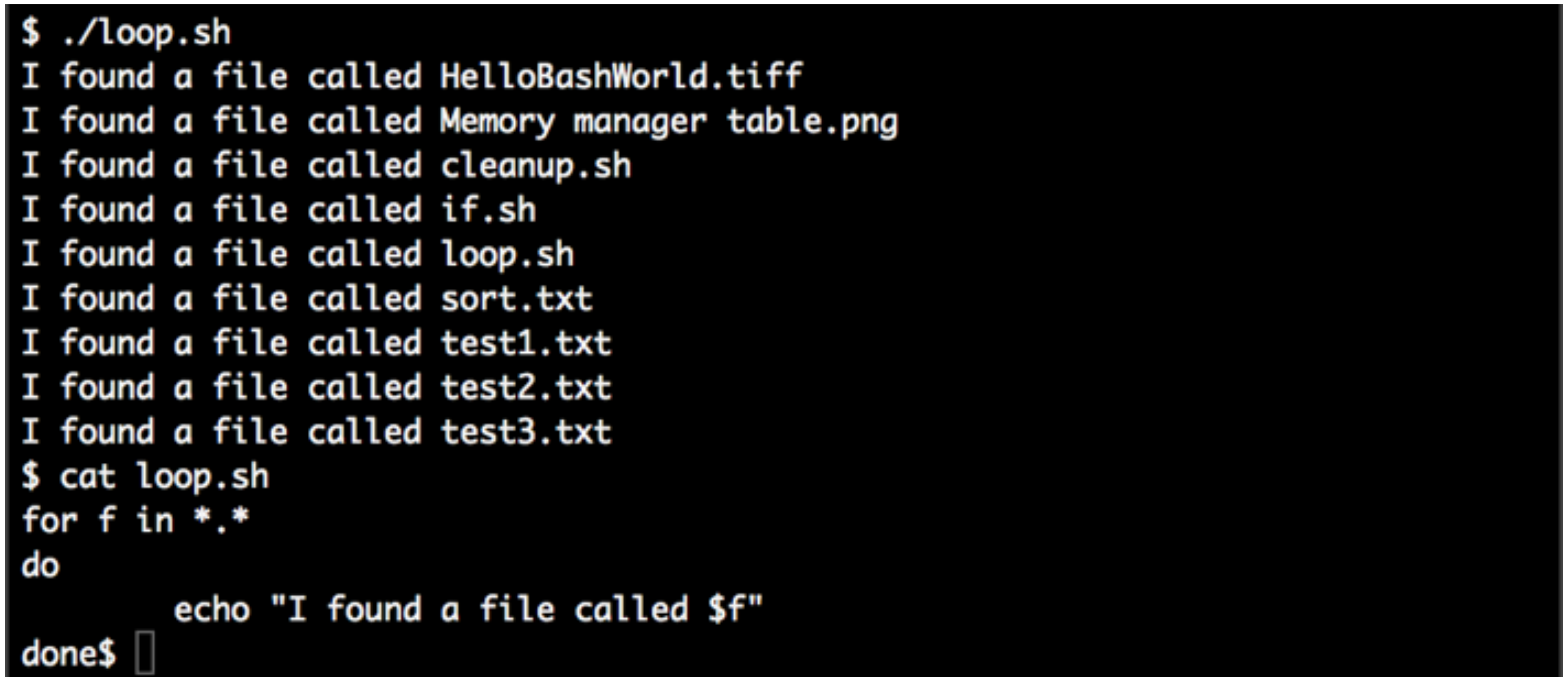
- Used to search for files
- Can find files modified or accessed before or after a time
- Can apply an operation to said files with **`-exec`**
- **`-maxdepth`** and **`-mindepth`** will specify how many folders find will look in
- **`-type`** can specify whether to find files, folders or other file system objects
- **`-name`** to specify the name to look for
- **`-atime`** to specify the amount of time since the file was accessed
- **`-ctime`** to specify the amount of time since the file was changed
- **`-mtime`** and **`-ctime`** both use a format of (+/-)n(s/m/h/d/w), where n is a number

Exercise

- Use man pages to identify how to remove the file name in the output of **`file`**
- Do not use other commands to remove parts of the output of **`file`**

Looping

- Iterate over sets with **`for`**
- Can be used to iterate over files in a directory, or even just count



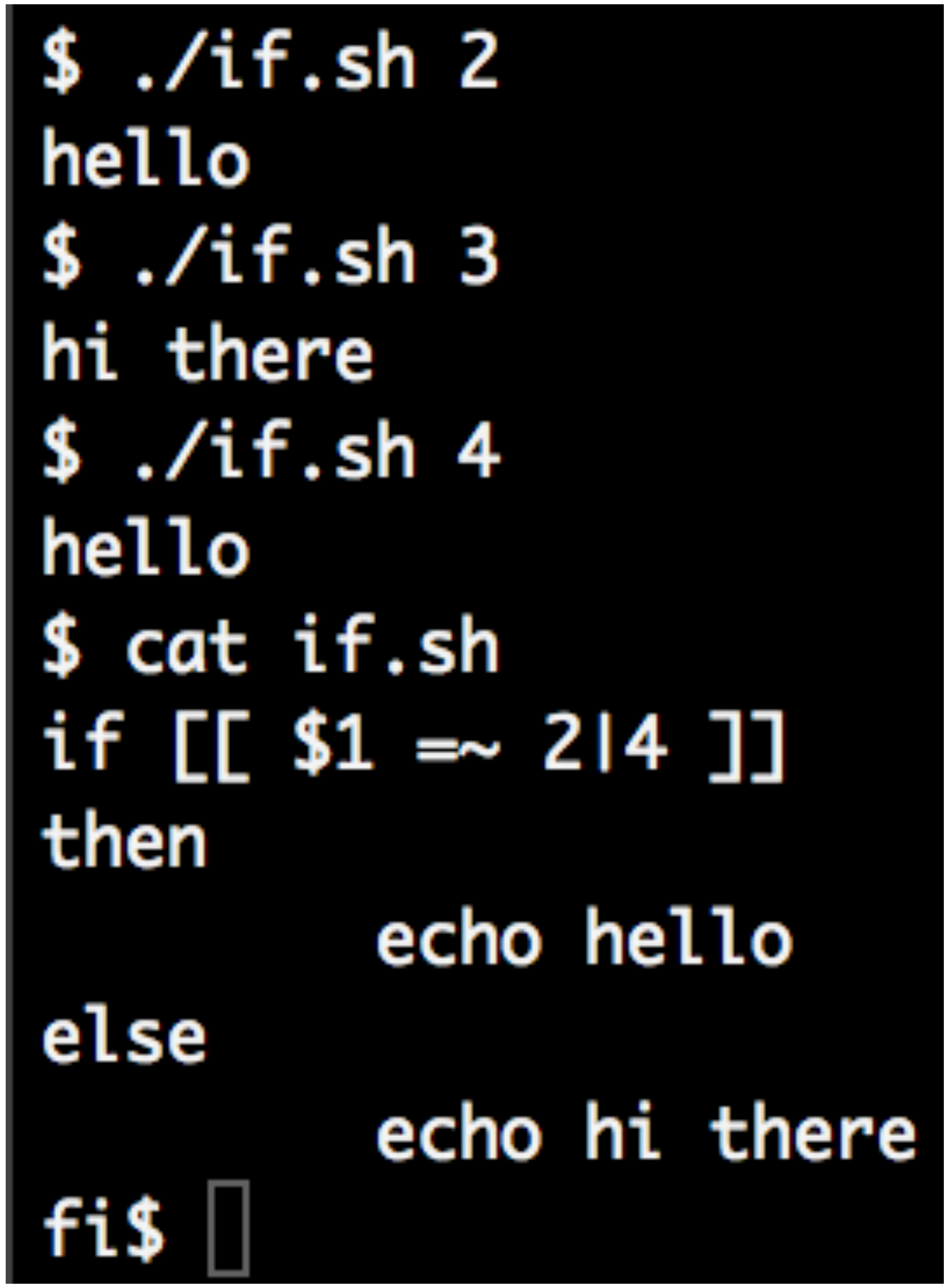
```
$ ./loop.sh
I found a file called HelloBashWorld.tiff
I found a file called Memory manager table.png
I found a file called cleanup.sh
I found a file called if.sh
I found a file called loop.sh
I found a file called sort.txt
I found a file called test1.txt
I found a file called test2.txt
I found a file called test3.txt
$ cat loop.sh
for f in *.*
do
        echo "I found a file called $f"
done$ 
```

Comparing

- **`==`** is true if the left and right sides are equal (**`$a == $b`**)
- **`!=`** is true if the left and right sides are not equal (**`$n != 4`**)
- **`=~`** will compare the left side to items on the right side separated by |'s (**`$a =~ A*|B|$n`**)

ifs

- Syntax is:
  ```
  if [[ $a == $b ]]
  then
    echo hello
  fi
  ```
- will echo hello if a equals b
- Else syntax is:
  ```
  if [[ $a == $b ]]
  then
    echo true
  else
    echo false
  fi
  ```
- Can use any comparator inside of brackets



```
$ ./if.sh 2
hello
$ ./if.sh 3
hi there
$ ./if.sh 4
hello
$ cat if.sh
if [[ $1 =~ 2|4 ]]
then
        echo hello
else
        echo hi there
fi$ 
```

What have we learned?

- File system navigation
- Control flow
- File analysis
- Searching the file system
- Variables
- Man pages

Let’s write some code!

```
#! env bash

mkdir images
mkdir archives
mkdir documents
mkdir unidentified

for f in *.*
do
     t=$(file -b $f)
     if [[ $t =~ PNG*|TIFF*|JPEG* ]]
     then
          mv $f images/$f
     fi

     if [[ $t =~ Zip*|TAR* ]]
     then
          if [[ $f =~ *.docx|*.doc|*.pptx|*.ppt ]]
          then
               mv $f documents/$f
          else
               mv $f archives/$f
          fi
     fi
     if [ $t == UTF-8 ]
     then
          mv $f documents/$f
     fi
done

find ./ -atime +1w -type f -maxdepth 1 -name "*" -exec mv {} unidentified/{}
\;
```