CS 50:
Software Design and Implementation

Shell Programming

A shell script is a series of commands in a text file that can be executed sequentially

Three things need to happen to make a shell script executable

1. **Write a script**
   - Shell scripts are ordinary text files
   - Use a text editor (vi or emacs) to write them

2. **Make the script executable**
   - By default, scripts will not be executable (runnable)
   - Must set file permissions to allow script execution

3. **Script must be in a directory where the system can find it**
   - Linux searches current directory
   - Also searches directories in $PATH

Agenda

1. Writing script step by step
2. Loops
3. Activity

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022
```

Tell Linux to run bash (located in bin directory)

 # is normally a comment

Give a description of what this script does

List inputs and outputs (could be none)

List author and date

Say how to run the script

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

echo "parameter: $1"
echo "got: $# parameters"

exit 0
```

First make sure we got the right number of parameters and that we can access them

I like to write the exit code right away (so I don't forget to return 0 if successful)

```
tjp@plank:~/cs50/workspace$ ./myscript.sh
-bash: ./myscript1.sh: Permission denied
```

Why doesn't it run?

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

echo "parameter: $1"
echo "got: $# parameters"

exit 0
```

First make sure we got the right number of parameters and that we can access them

I like to write the exit code right away (so I don't forget to return 0 if successful)

```
tjp@plank:~/cs50/workspace$ ./myscript.sh
-bash: ./myscript1.sh: Permission denied

tjp@plank:~/cs50/workspace$ ls -l myscript.sh
-rw-r--r-- 1 d84607y thayerusers  11 Jun 27 18:09 myscript.sh
```

Why doesn't it run?

Execute permission is not set!

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

echo "parameter: $1"
echo "got: $# parameters"

exit 0
```

First make sure we got the right number of parameters and that we can access them

I like to write the exit code right away (so I don't forget to return 0 if successful)

Change file mode
Allow user (u) to execute (x)

```
tjp@plank:~/cs50/workspace$ chmod u+x myscript.sh
tjp@plank:~/cs50/workspace$ ls -l myscript.sh
-rwxr--r-- 1 d84607y thayerusers 328 Jun 27 18:09 myscript.sh*
```

User has rights to execute (but not group or others)

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

echo "parameter: $1"
echo "got: $# parameters"

exit 0
```

So far, so good!

```
tjp@plank:~/cs50/workspace$ ./myscript.sh test test2
parameter: test
got: 2 parameters

tjp@plank:~/cs50/workspace$ echo $?
0
```

Echo first parameter and count

Pass 2 parameters, test and test2

Check exit code

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

debug=0
if [ "$debug" -eq 1 ]; then
      echo "parameter: $1"
      echo "got: $# parameters"
fi

exit 0
```

Technique: set debug flag and output text only if flag is set

Square brackets invokes the test command

Use eq, ne, lt, le, gt, ge for numeric values

Use = or == for strings

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

debug=0
if [ "$debug" -eq 1 ]; then
        echo "parameter: $1"
        echo "got: $# parameters"
fi

#check for correct number of parameters
if [ $# -ne 1 ]; then
        echo >&2 "Incorrect number of parameters"
        echo >&2 "Usage: ./myscript.sh <directory name>"
        exit 1
fi

exit 0
```

Defensive programming: first check that we got the right number of parameters

Output error message on stderr and give non-zero exit code if not

```
tjp@plank:~/cs50/workspace$ ./myscript.sh
Incorrect number of parameters
Usage: ./myscript.sh <directory name>
tjp@plank:~/cs50/workspace$ ./myscript.sh test test2
Incorrect number of parameters
Usage: ./myscript.sh <directory name>
tjp@plank:~/cs50/workspace$ ./myscript.sh test
```

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

debug=0
if [ "$debug" -eq 1 ]; then
        echo "parameter: $1"
        echo "got: $# parameters"
fi

#check for correct number of parameters
if [ $# -ne 1 ]; then
        echo >&2 "Incorrect number of parameters"
        echo >&2 "Usage: ./myscript.sh <directory name>"
        exit 1
fi

exit 0
```

Defensive programming: first check that we got the right number of parameters

Output error message on stderr and give non-zero exit code if not

No parameters

Too many parameters

Right number of parameters

```
tjp@plank:~/cs50/workspace$ ./myscript.sh
Incorrect number of parameters
Usage: ./myscript.sh <directory name>
tjp@plank:~/cs50/workspace$ ./myscript.sh test test2
Incorrect number of parameters
Usage: ./myscript.sh <directory name>
tjp@plank:~/cs50/workspace$ ./myscript.sh test
```

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
#!/bin/bash
#
# myscript.sh - lists files in a directory sorted with most recently edited first
#
# input: directory
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

debug=0
if [ "$debug" -eq 1 ]; then
        echo "parameter: $1"
        echo "got: $# parameters"
fi

#check for correct number of parameters
if [ $# -ne 1 ]; then
        echo >&2 "Incorrect number of parameters"
        echo >&2 "Usage: ./myscript.sh <directory name>"
        exit 1
fi

#check if directory exits
if [ ! -d "$1" ]; then
        echo >&2 "Directory does not exist"
        exit 2
fi

exit 0
```

Check directory exists
Output to stderr and return non-zero if not

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ vi myscript.sh
```

```
# output: list of files in a directory sorted with  most recently edited first
#
# usage: ./myscript.sh <directory name>
#
# Tim Pierson, CS 50, Fall 2022

debug=0
if [ "$debug" -eq 1 ]; then
        echo "parameter: $1"
        echo "got: $# parameters"
fi

#check for correct number of parameters
if [ $# -ne 1 ]; then
        echo >&2 "Incorrect number of parameters"
        echo >&2 "Usage: ./myscript.sh <directory name>"
        exit 1
fi

#check if directory exits
if [ ! -d "$1" ]; then
        echo >&2 "Directory does not exist"
        exit 2
fi

ls -ltF --color=auto "$1"
if [ $? -ne 0 ]; then
        echo >&2 "Something went wrong listing files"
        exit 3
fi

exit 0
```

Add command and exit with 0 if successful

Use a text editor to write scripts, make sure to comment them!

```
tjp@plank:~/cs50/workspace$ ls -l
total 367807
-rwxr-xr-x 1 d84607y thayerusers       288 Jun 27 10:36 backup.sh*
-rwxr--r-- 1 d84607y thayerusers       744 Jun 27 18:49 myscript.sh*
-rw-r--r-- 1 d84607y thayerusers        25 Jun 27 17:36 out.txt
-rw-r--r-- 1 d84607y thayerusers     67938 Jun 26 19:43 passwd
drwxr-sr-x 2 d84607y thayerusers        93 Jun 27 17:38 students/
-rw-r--r-- 1 d84607y thayerusers 294299243 Jun 25 11:16 vaccine.csv

tjp@plank:~/cs50/workspace$ ./myscript.sh test
Directory does not exist

tjp@plank:~/cs50/workspace$ ./myscript.sh student
Directory does not exist

tjp@plank:~/cs50/workspace$ ./myscript.sh students
total 70
-rw-r--r-- 1 d84607y thayerusers 210 Jun 26 16:56 learning_fellows.txt
-rw-r--r-- 1 d84607y thayerusers   0 Jun 26 15:21 error.txt
-rw-r--r-- 1 d84607y thayerusers 210 Jun 26 12:33 output.txt
```

Agenda

1. Writing script step by step
2. Loops
3. Activity

For loops can be entered from the command line or used in a script

Prompt changes indicating bash is still waiting for the rest of the command

```
tjp@plank:~/cs50$ for i in Allen "North Park" Test; do
> echo $i
> done
Allen
North Park
Test
```

While works similarly while [test]; do… done

Reference loop variable with $<varname>

Any spaces or newlines in the for command will cause the shell to delineate *words* that become arguments to for

For loops can loop over file contents

File name

```
tjp@plank:~/cs50/workspace$ for i in $(<./students/learning_fellows.txt); do echo $i; done
Homer
A.
Simpson
Marge
B.
Simpson
Bart
C.
Simpson
Lisa
D.
Simpson
<snip>
```

Output breaks on spaces instead of new lines

For loops can loop over output from commands

Sed command operating on file contents

```
tjp@plank:~/cs50/workspace$ for i in $(sed -n 's/.\..*//p' < students/learning_fellows.txt);
do echo $i; done
Homer
Marge
Bart
Lisa
Maggie
Montomery
Sideshow
Millhouse
Ned
Edna
Barney
Kustry
```

Extract only first names (assumes middle initial)
Search for any single character (.) followed by dot (must escape with \.) followed by any number of characters (.*)
Replace with nothing (//)

Alternatively, we could have just searched for a space!
sed -n 's/ .*//p'
Find a space followed by any number of characters and replace with nothing

Can loop over files in a directory

```
#!/bin/bash
#
# backup.sh - make a backup copy of all the .c files in current
directory
#
# usage: backup.sh
# (no arguments)
#
# input: none
# output: a line of confirmation for each file backed up
#
# CS50, Fall 2022


for i in *.c
do
  echo "back up $i"
  cp "$i" "$i.bak"
done


exit 0
```

While works similarly
while [test]; do… done

For [test]; do… done
Loop over all files in the current directory

Agenda

1. Writing script step by step
2. Loops
3. Activity