

CS 332/532 Systems Programming

Lecture 8

-UNIX Shells, Shell Scripting-

Professor : Mahmut Unan – UAB CS

Announcement

- Monday
 - Git & GitHub Workshop

Agenda

- UNIX Shells
- Shell Scripting
- Man Page

Linux vs UNIX

- Linux refers to the kernel of the GNU/Linux operating system. More generally, it refers to the family of derived distributions.
- Unix refers to the original operating system developed by AT&T. More generally, it refers to family of derived operating systems.
- GNU/Linux and derivatives like Debian and Fedora.
System-V Unix and derivatives like IBM-AIX and HP-UX; Berkeley Unix and derivatives like FreeBSD and macOS
- Linux is broadly available as configurable software download and installer. UNIX is typically shipped along with hardware e.g. MacBook

Working in the UNIX Environment

- UNIX like OS
 - Solaris
 - FreeBSD
 - macOS
 - NetBSD
 -
- Logging In
 - login name - password
 - password file
 - /etc/passwd

Shells

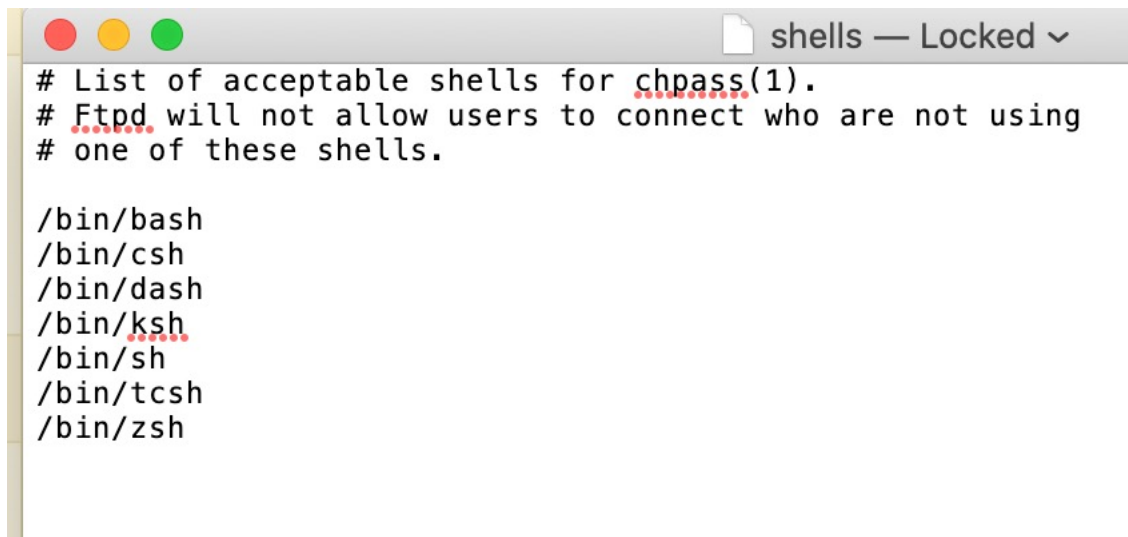
- A shell is the interface between the user and the kernel.
- Users can interact with the shell using shell commands in terminal or from a file (shell script).
- The common shells are;

Name	Path	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10
Bourne shell	/bin/sh	•	•	copy of bash	•
Bourne-again shell	/bin/bash	optional	•	•	•
C shell	/bin/csh	link to tcsh	optional	link to tcsh	•
Korn shell	/bin/ksh	optional	optional	•	•
TENEX C shell	/bin/tcsh	•	optional	•	•

Figure 1.2 Common shells used on UNIX systems

MacOS users

- Start the Terminal app on your Mac
- Terminal > Preferences, then click General.
- Under “Shells open with,” select “Command (complete path),” then enter the path to the shell you want to use.
- If you want to check the available shells in your mac;
 - go to `/etc` folder and check the `shells` file



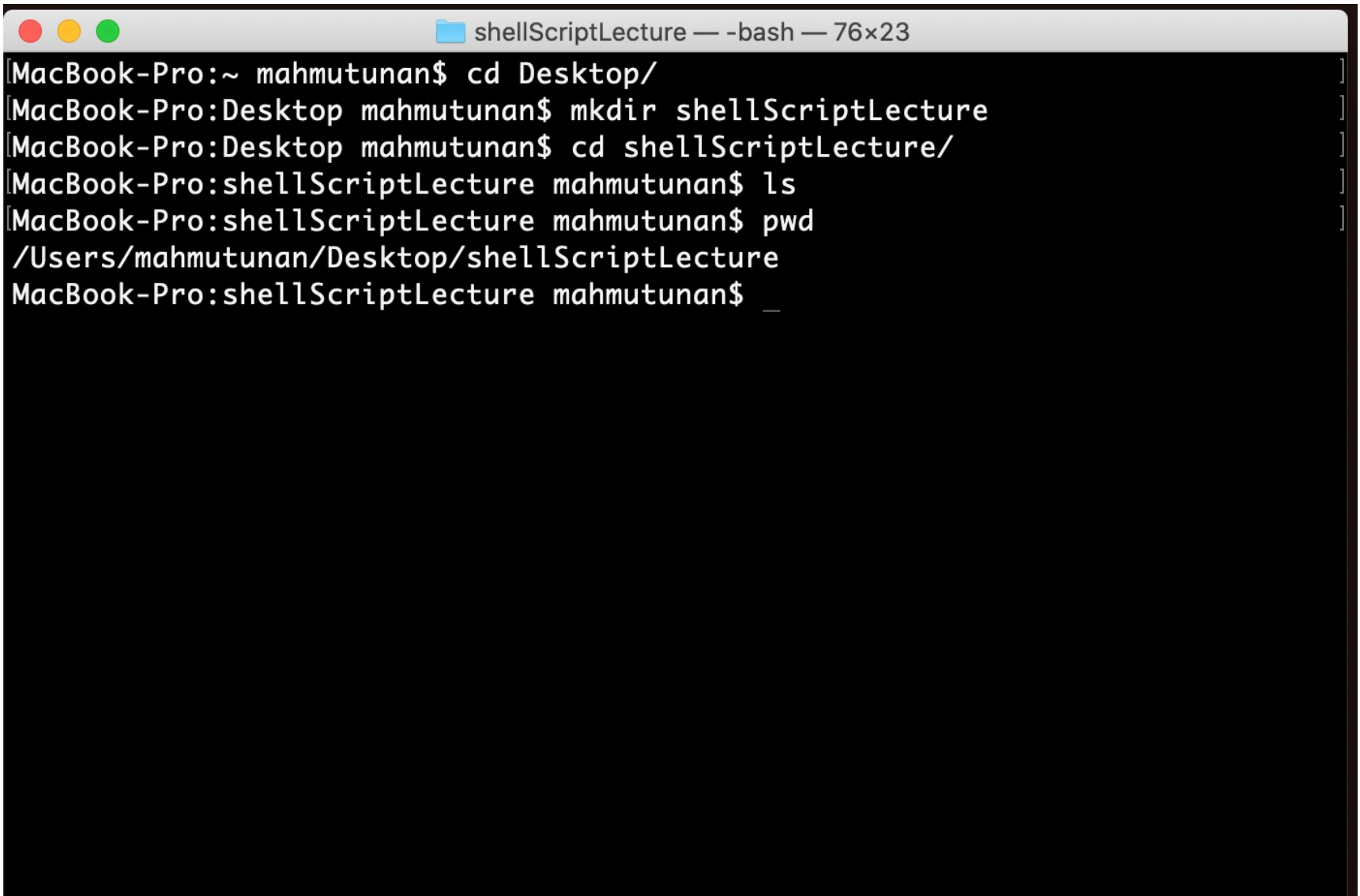
```
shells — Locked v
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/bash
/bin/csh
/bin/dash
/bin/ksh
/bin/sh
/bin/tcsh
/bin/zsh
```

Windows Users

- Windows Subsystem for Linux
 - Bash Shell
- Git Bash
- <https://www.geeksforgeeks.org/use-bash-shell-natively-windows-10/>
- <https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/>

Exercise 1 - first script file

A terminal window titled "shellScriptLecture — -bash — 76x23" with standard macOS window controls (red, yellow, green buttons). The terminal shows a series of commands and their outputs: 1. "cd Desktop/" changes the directory to Desktop. 2. "mkdir shellScriptLecture" creates a new directory named shellScriptLecture. 3. "cd shellScriptLecture/" changes the directory to the newly created one. 4. "ls" lists the contents of the current directory, which is empty. 5. "pwd" prints the current working directory as "/Users/mahmutunan/Desktop/shellScriptLecture". 6. The prompt returns to "MacBook-Pro:shellScriptLecture mahmutunan\$ _".

```
MacBook-Pro:~ mahmutunan$ cd Desktop/
MacBook-Pro:Desktop mahmutunan$ mkdir shellScriptLecture
MacBook-Pro:Desktop mahmutunan$ cd shellScriptLecture/
MacBook-Pro:shellScriptLecture mahmutunan$ ls
MacBook-Pro:shellScriptLecture mahmutunan$ pwd
/Users/mahmutunan/Desktop/shellScriptLecture
MacBook-Pro:shellScriptLecture mahmutunan$ _
```

.sh file

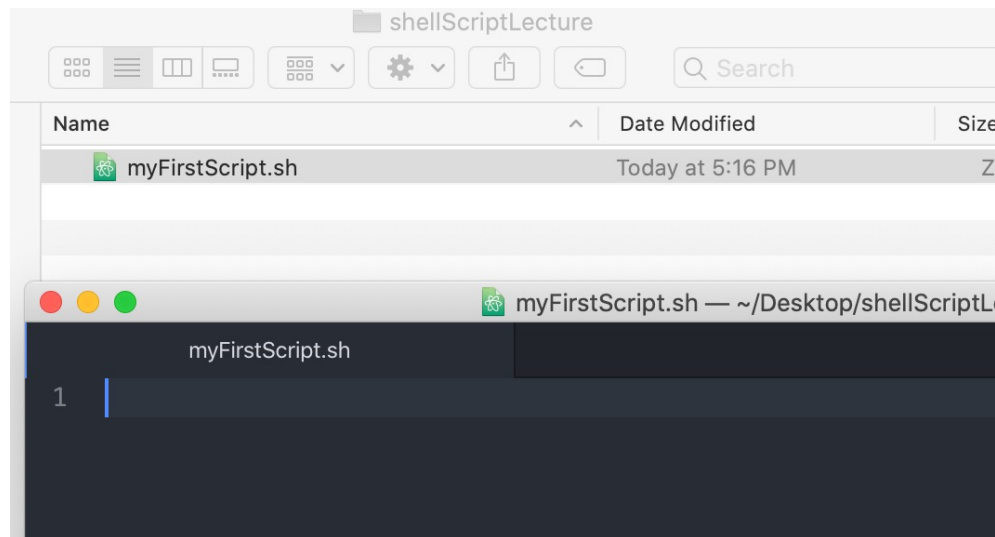
- It is a script programmed for bash
 - It contains instructions written in the Bash language
 - It can be executed by typing text commands within the shell's command-line interface.
- How to run the .sh file?
 - First, give the execute permission
 - `chmod 755 somefilename.sh`
 - Next, run your script file
 - `sh somefilename.sh`
 - `bash somefilename.sh`
 - `./somefilename.sh`
 - if you want to run it as a root user
 - `sudo bash somefilename.sh`

myFirstScript.sh

- You can use your terminal to create the file and use nano to edit the file

```
MacBook-Pro:shellScriptLecture mahmutunan$ touch myFirstScript.sh
MacBook-Pro:shellScriptLecture mahmutunan$ nano myFirstScript.sh
```

- OR, you can use any editor to create and edit the file



```
1  #!/bin/bash
2
3  # some comment
4  echo Hello CS332!!!
5
6  LECTURE="Lecture 8"
7  echo "This is $LECTURE"
8
9  echo -n "How old are you: "
10 read AGE
11 if [[ $AGE -ge 18 ]]
12 then
13     echo "You can vote"
14 else
15     echo "You are not eligible to vote"
16 fi
17
```

myFirstScript.sh

```
1  #!/bin/bash
2
3  # some comment
4  echo Hello CS332!!!
5
6  LECTURE="Lecture 8"
7  echo "This is $LECTURE"
8
```

```
MacBook-Pro:shellScriptLecture mahmutunan$ bash myFirstScript.sh
Hello CS332!!!
This is Lecture 8
How old are you: 21
You can vote
MacBook-Pro:shellScriptLecture mahmutunan$ bash myFirstScript.sh
Hello CS332!!!
This is Lecture 8
How old are you: 11
You are not eligible to vote
MacBook-Pro:shellScriptLecture mahmutunan$
```

FILE Conditions

File operators

Operator	Note
<code>-e</code>	To check if the file exists.
<code>-r</code>	To check if the file is readable.
<code>-w</code>	To check if the file is writable.
<code>-x</code>	To check if the file is executable.
<code>-s</code>	To check if the file size is greater than 0.
<code>-d</code>	To check if the file is a directory.

```
1  #!/bin/sh
2
3  FILE_NAME="someFileThatDoesntExist.txt"
4
5  # check
6  if [ -e $FILE_NAME ]
7  then
8      echo "Heyyooo, the file exists!"
9  else
10     echo "00PPSSS, the file does not exists!"
11  fi
```

```
MacBook-Pro:shellScriptLecture mahmutunan$ bash fileOperations.sh
00PPSSS, the file does not exists!
```

Loops & Arrays

```
MY_COURSES="CS203 CS330 CS332"  
for COURSE in $MY_COURSES  
do  
    echo $COURSE  
done
```

```
CS203  
CS330  
CS332
```



```
mkdir newFolder  
touch "newFolder/someNewFile.txt"  
echo "This message goes to the file" >> "newFolder/someNewFile.txt"  
echo "This message appears on the terminal"
```

This message appears on the terminal
MacBook-Pro:shellScriptLecture mahmutunan\$ _



```
1  #!/bin/sh
2  clear
3  echo "Current Directory :"
4  pwd
5  echo "What is in this directory? :"
6  ls
7
8  head "myFirstScript.sh"
9
10 echo "Disk Usage :"
11 df -h
12
13 exit
14
```

```

Current Directory :
/Users/mahmutunan/Desktop/shellScriptLecture
What is in this directory? :
exercise3.sh          fileOperations.sh      myFirstScript.sh      newFolder
#!/bin/bash

# some comment
echo Hello CS332!!!

LECTURE="Lecture 8"
echo "This is $LECTURE"

echo -n "How old are you: "
read AGE

Disk Usage :

```

Filesystem	Size	Used	Avail	Capacity	iused	ifree	%iused
/dev/disk1s6	466Gi	10Gi	221Gi	5%	488316	4881964564	0%
devfs	231Ki	231Ki	0Bi	100%	800	0	100%
/dev/disk1s1	466Gi	208Gi	221Gi	49%	1063805	4881389075	0%
/dev/disk1s4	466Gi	15Gi	221Gi	7%	15	4882452865	0%
/dev/disk1s5	466Gi	10Gi	221Gi	5%	487048	4881965832	0%
map auto_home	0Bi	0Bi	0Bi	100%	0	0	100%
ome							
Box	466Gi	208Gi	221Gi	49%	1063805	586421779	0%
/Users/mahmutunan/Documents/Atom.app	466Gi	199Gi	243Gi	45%	971867	4881481013	0%
k/l02njags56v17md44tfknd7c0000gn/T/AppTranslocation/71F900AC-7A89-4E16-BC6F-66BBF16E283E							
/dev/disk1s3	466Gi	1.0Gi	221Gi	1%	94	4882452786	0%
/dev/disk3s1	309Mi	229Mi	80Mi	75%	1433	4294965846	0%

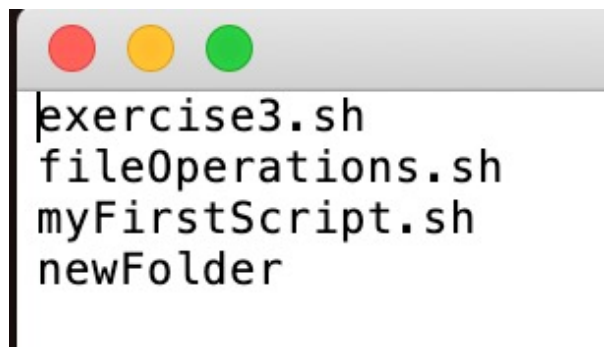
```

MacBook-Pro:shellScriptLecture mahmutunan$ _

```

I/O Redirection

- Regular UNIX system commands;
 - take input from terminal (stdin)
 - writes output to terminal (stdout)
- Output redirection
 - Output to a file
 - `> filename` notation will be used
 - `ls >> "newFolder/someNewFile.txt"`



```
exercise3.sh
fileOperations.sh
myFirstScript.sh
newFolder
```

A terminal window with a grey title bar containing three colored circles (red, yellow, green). The terminal content shows a directory listing with four items: exercise3.sh, fileOperations.sh, myFirstScript.sh, and newFolder.

- Input Redirection
 - < filename

```
Mail -s "Subject" to-address < Filename
```

The diagram shows a terminal command: `guru99@VirtualBox:~$ mail -s "News Today" abc@ymail.com < NewsFlash`. Handwritten labels with red arrows point to specific parts of the command: "Attachment File" points to `< NewsFlash`, "E-mail Subject" points to `"News Today"`, and "E-mail Address" points to `abc@ymail.com`.

```
guru99@VirtualBox:~$ mail -s "News Today" abc@ymail.com < NewsFlash
```

Attachment File

E-mail Subject

E-mail Address

Man Page

\$man command

\$man cat

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

NAME
    cat -- concatenate and print files

SYNOPSIS
    cat [-benstuv] [file ...]

DESCRIPTION
    The cat utility reads files sequentially, writing them to the standard output. The file operands are processed in command-line order. If file is a single dash ('-') or absent, cat reads from the standard input. If file is a UNIX domain socket, cat connects to it and then reads it until EOF. This complements the UNIX domain binding capability available in inetd(8).

    The options are as follows:

    -b      Number the non-blank output lines, starting at 1.

    -e      Display non-printing characters (see the -v option), and display a dollar sign ('$') at the end of each line.

    -n      Number the output lines, starting at 1.

    -s      Squeeze multiple adjacent empty lines, causing the output to be single spaced.

    -t      Display non-printing characters (see the -v option), and display tab characters as '^I'.

    -u      Disable output buffering.

    -v      Display non-printing characters so they are visible. Control characters print as '^X' for control-X; the delete character (octal 0177) prints as '^?'. Non-ASCII characters
```