CS 332/532 Systems Programming

Lecture 8

-UNIX Shells, Shell Scripting-

Professor : Mahmut Unan – UAB CS

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 derivates 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

https://www.softwaretestinghelp.com/unix-vs-linux/#:~:text=Linux%20refers%20to%20the%20kernel,family%20of%20derived%20operating%20systems.

Differences	Linux	Unix
Origins	Linux was developed in the 1990s by Linus Torvalds as a free and open-source alternative to Unix.	Unix was developed in the 1970s at Bell Labs
Introduction	Linux is Open Source, and a large number of programmers work together online and contribute to its development.	Unix was developed by AT&T Labs, different commercial vendors, and non-profit organizations.
Licensing	Linux, on the other hand, is open-source software and can be used freely without any licensing fees.	Unix is a proprietary operating system, meaning that it requires a license to use.
Kernels	both have a similar design but are less complex than the Unixhold-upthat kernel.	both have a similar design but larger and more complex than the Linux kernel.
Availability	On the other hand, Linux is widely used on both enterprise and personal computers.	Unix is typically found on enterprise-level servers and workstations and is less commonly used on personal computers.

https://www.geeksforgeeks.org/linux-vs-unix/

Use Cases	It is used everywhere from servers, PCs, smartphones, tablets to mainframes.	It is used on servers, workstations, and PCs.
Shell Compatibility	The default interface is BASH (Bourne Again Shell). Anybody can use Linux whether a home client, developer or a student.	It initially used Bourne shell. But it is also compatible with other GUIs. Developed mainly for servers, workstations, and mainframes.
Source Code Availability	The source is accessible to the general public.	The source is not accessible to the general public.
Hardware Compatibility	Originally developed for Intel's x86 hardware processors. It is available for more than twenty different types of CPU which also includes an ARM.	It is available on PA-RISC and Itanium machines.
Virus Threats	It has about 60-100 viruses listed to date.	It has about 85-120 viruses listed to date (rough estimate).
Operating System Versions	Some Linux versions are Ubuntu, Debian GNU, Arch Linux, etc.	Some Unix versions are SunOS, Solaris, SCO UNIX, AIX, HP/UX, ULTRIX, etc.

https://www.geeksforgeeks.org/linux-vs-unix/

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 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

```
# 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 Subsytem 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

shellScriptLecture — -bash — 76×23

```
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

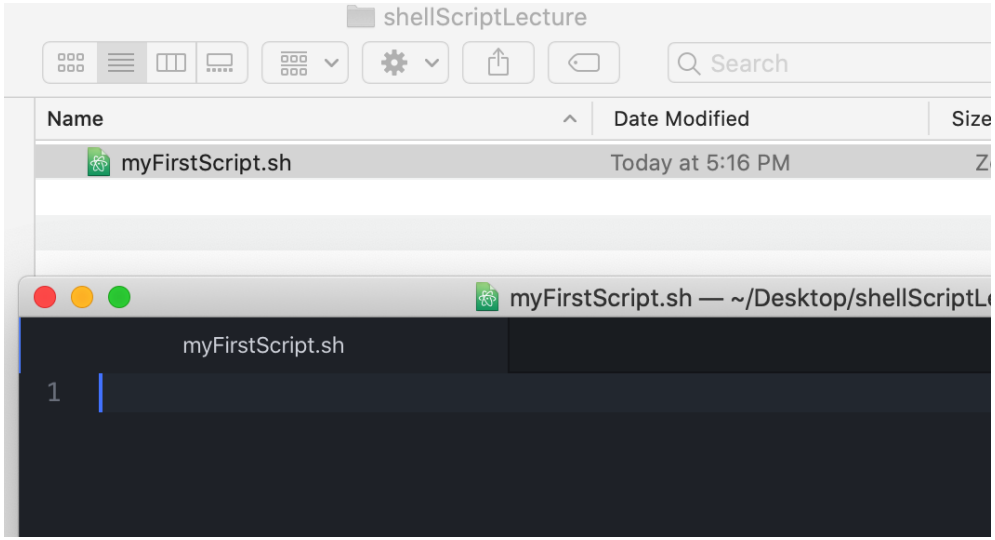
- It is a script programmed for bash
  - It contains instructions written in the Bash language
  - It can be executed by typing text commands within the shell's command-line interface.
- How to run the .sh file?
  - First, give the execute permission
    - chmod 755 somefilename.sh
  - Next, run your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



myFirstScript.sh

```
#!/bin/bash

# some comment
echo Hello CS332!!!

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

echo -n "How old are you: "
read AGE
if [[ $AGE -ge 18 ]]
then
  echo "You can vote"
else
  echo "You are not eligible to vote"
fi

```

myFirstScript.sh

```
#!/bin/bash

# some comment
echo Hello CS332!!!

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

```
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
`-e`	To check if the file exists.
`-r`	To check if the file is readable.
`-w`	To check if the file is writable.
`-x`	To check if the file is executable.
`-s`	To check if the file size is greater than 0.
`-d`	To check if the file is a directory.

https://dyclassroom.com/unix/shell-programming-file-operators

```
#!/bin/sh

FILE_NAME="someFileThatDoesntExist.txt"

# check
if [ -e $FILE_NAME ]
then
  echo "Heyyooo, the file exists!"
else
  echo "OOPPSSS, the file does not exists!"
fi
```

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

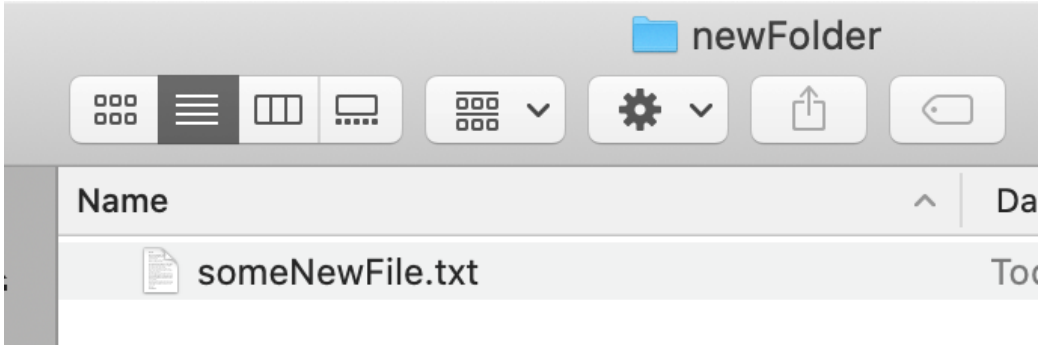
Loops & Arrays

```
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$
```

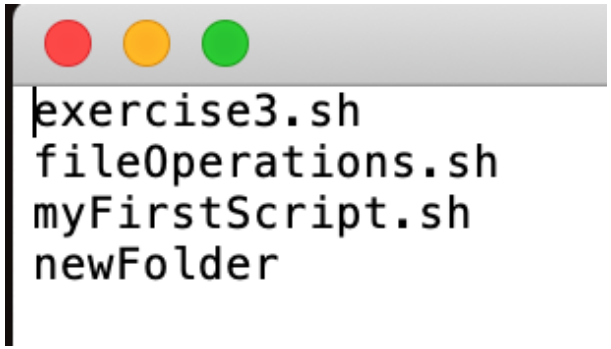


```
#!/bin/sh
clear
echo "Current Directory :"
pwd
echo "What is in this directory? :"
ls

head "myFirstScript.sh"

echo "Disk Usage :"
df -h

exit
```

```
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/l02njqgs56v17md44tfknd7c0000gn/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”`



- Input Redirection
  - < filename

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

https://www.guru99.com/linux-redirection.html

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
```