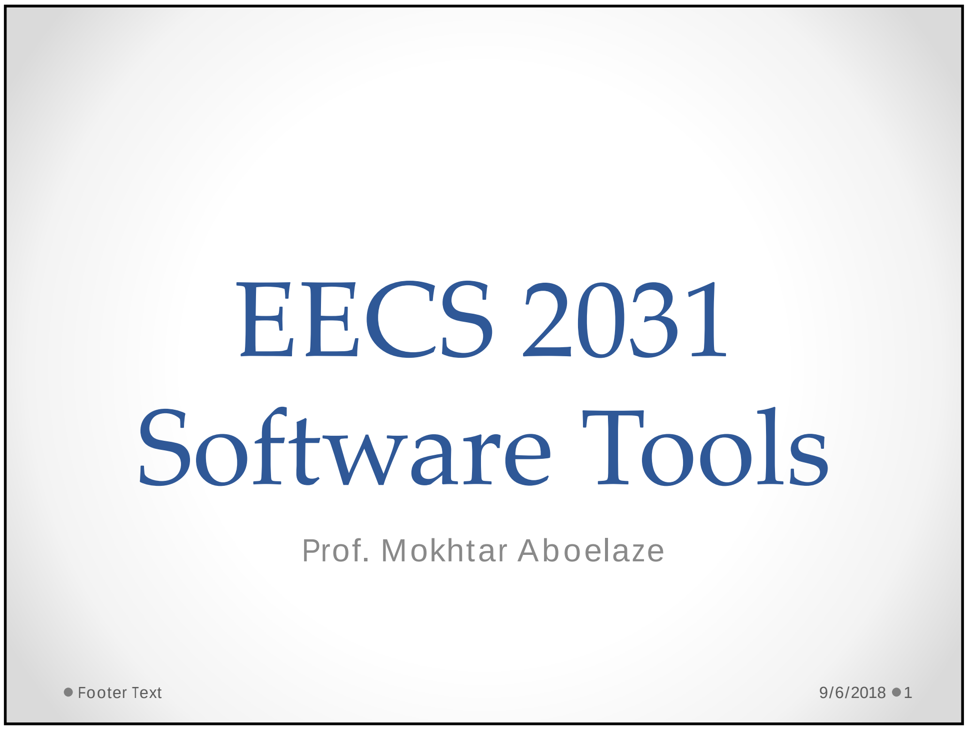
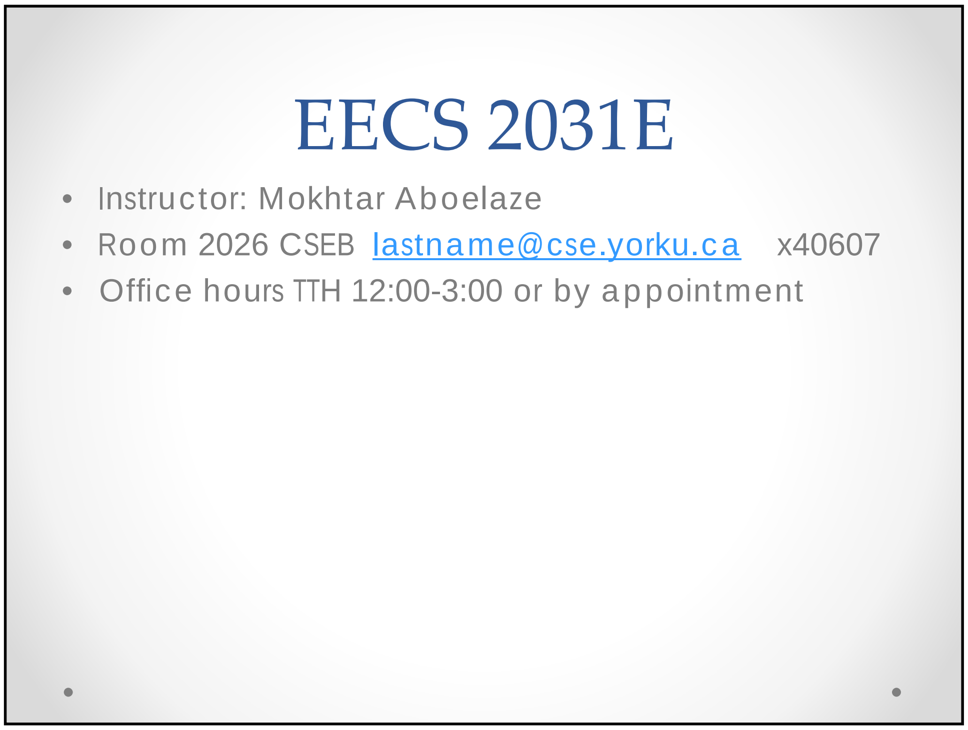


EECS 2031 Software Tools

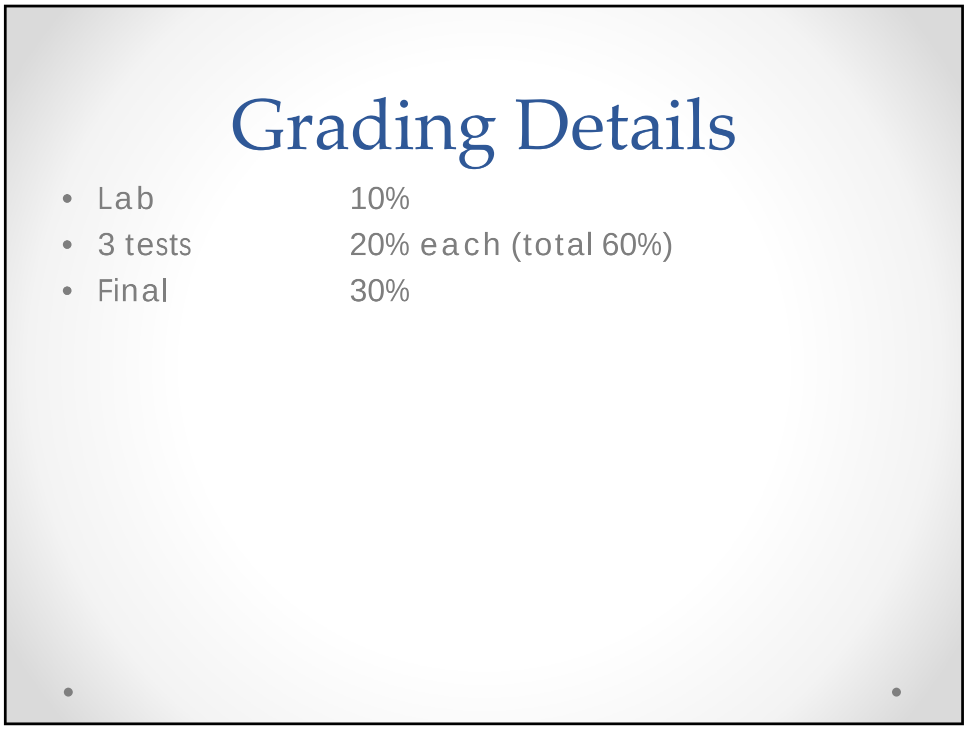
Prof. Mokhtar Aboelaze

Footer Text 9/6/2018 1



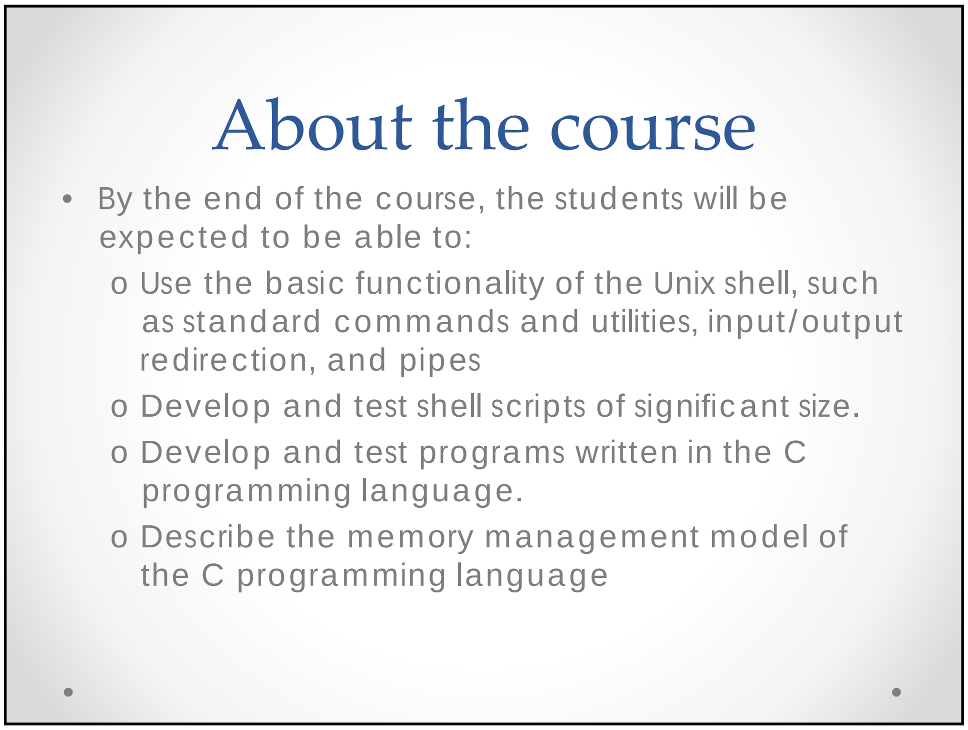
EECS 2031E

- Instructor: Mokhtar Aboelaze
- Room 2026 CSEB lastname@cse.yorku.ca x40607
- Office hours TTH 12:00-3:00 or by appointment



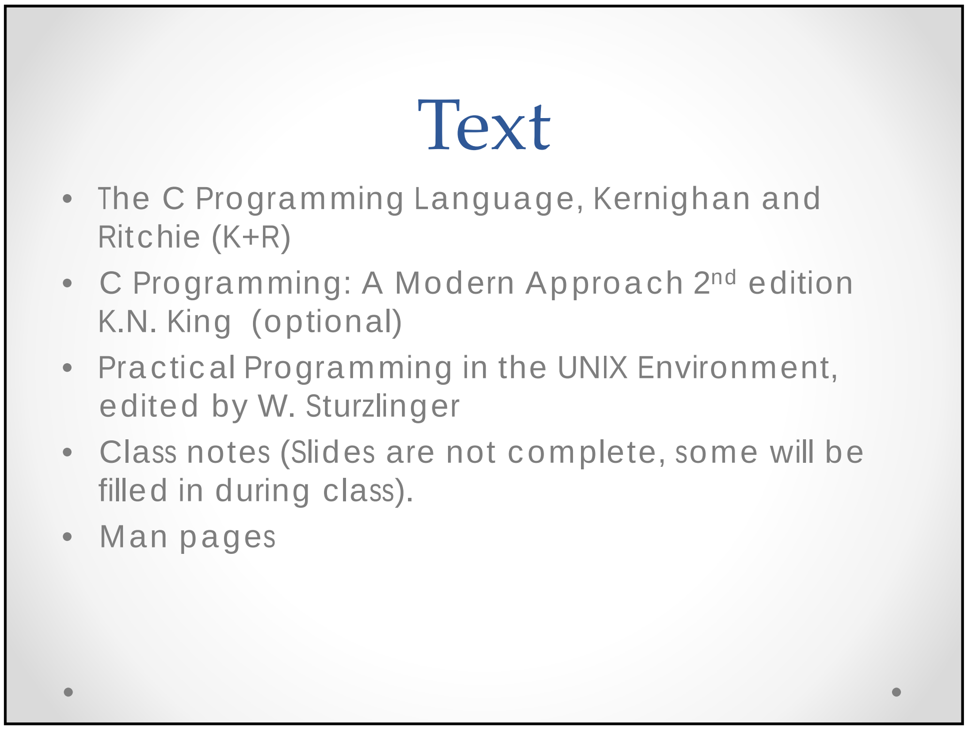
Grading Details

• Lab	10%
• 3 tests	20% each (total 60%)
• Final	30%



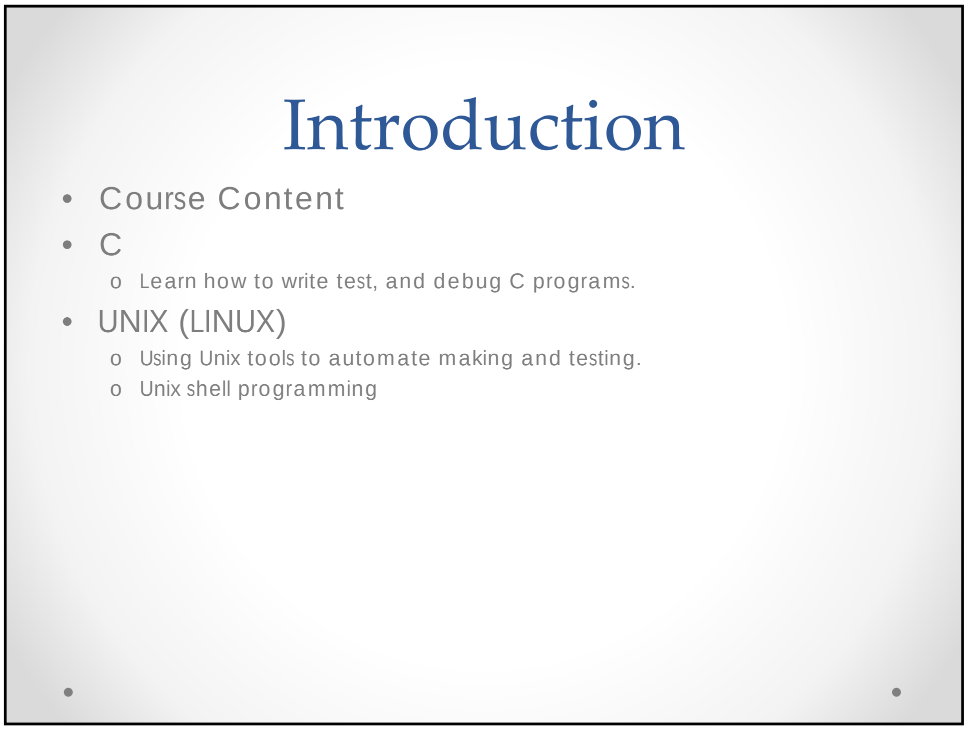
About the course

- By the end of the course, the students will be expected to be able to:
  - o Use the basic functionality of the Unix shell, such as standard commands and utilities, input/output redirection, and pipes
  - o Develop and test shell scripts of significant size.
  - o Develop and test programs written in the C programming language.
  - o Describe the memory management model of the C programming language



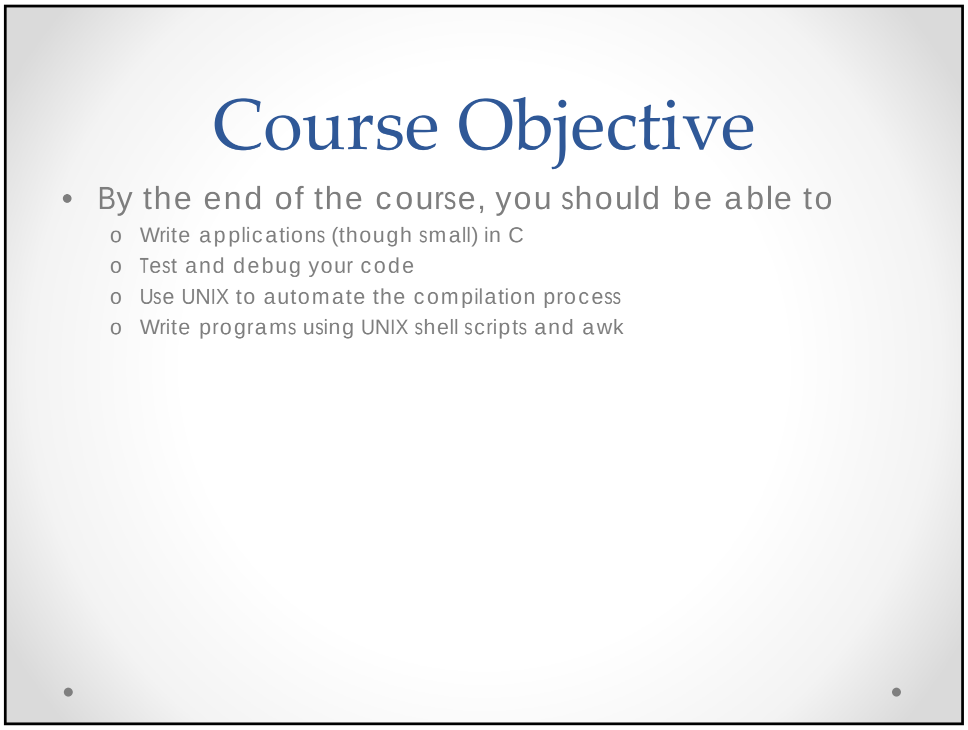
Text

- The C Programming Language, Kernighan and Ritchie (K+R)
- C Programming: A Modern Approach 2nd edition K.N. King (optional)
- Practical Programming in the UNIX Environment, edited by W. Sturzlinger
- Class notes (Slides are not complete, some will be filled in during class).
- Man pages



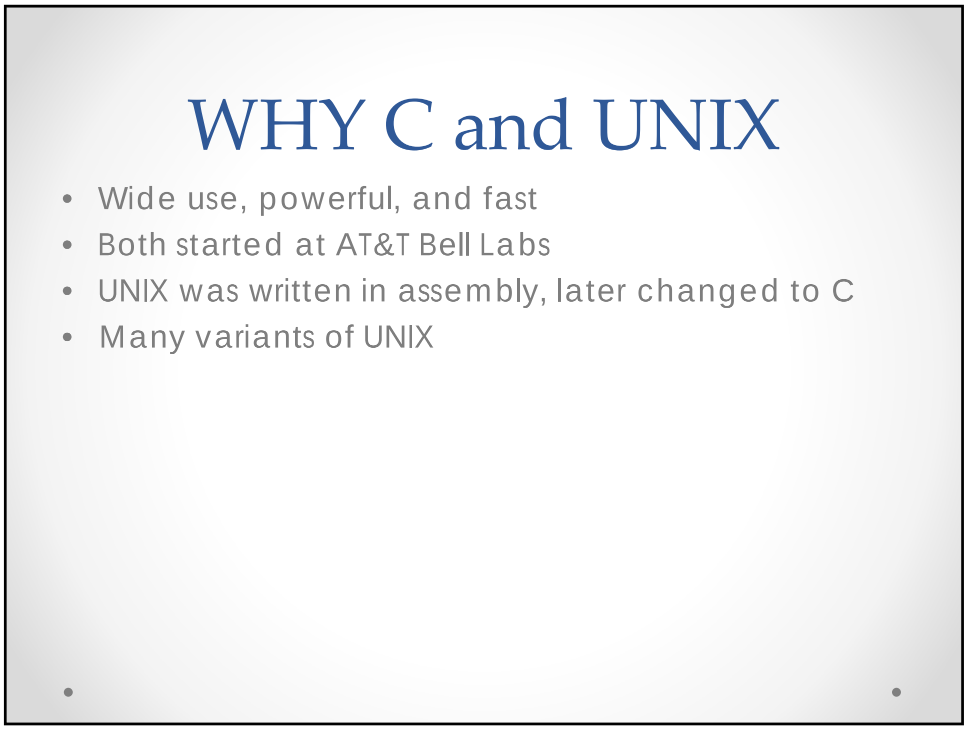
Introduction

- Course Content
- C
  - Learn how to write test, and debug C programs.
- UNIX (LINUX)
  - Using Unix tools to automate making and testing.
  - Unix shell programming



Course Objective

- By the end of the course, you should be able to
  - Write applications (though small) in C
  - Test and debug your code
  - Use UNIX to automate the compilation process
  - Write programs using UNIX shell scripts and awk



WHY C and UNIX

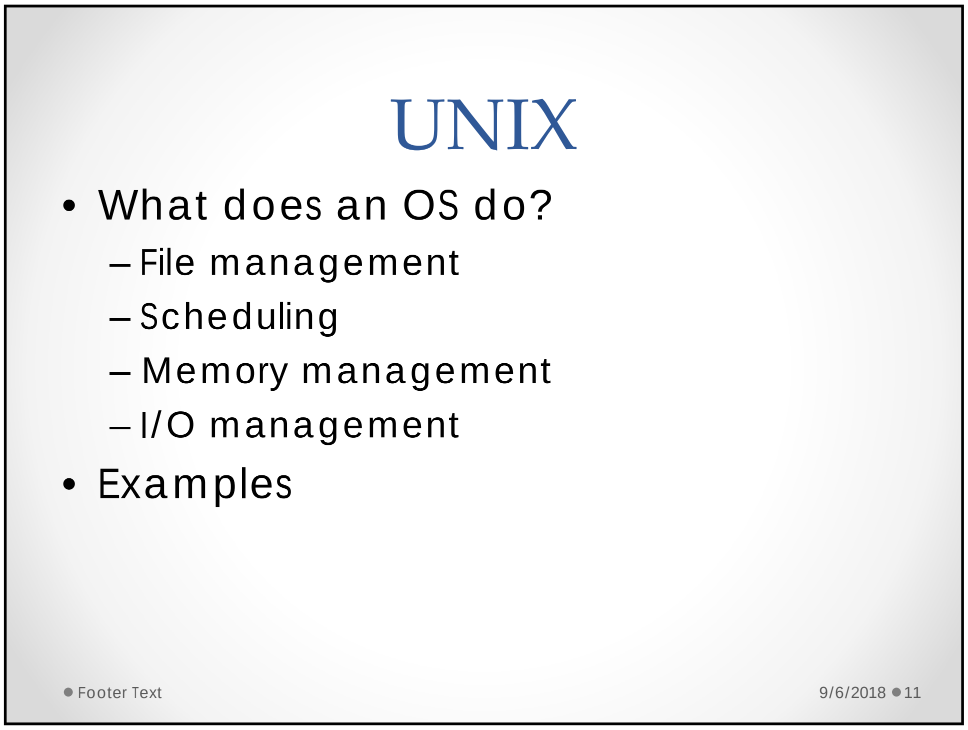
- Wide use, powerful, and fast
- Both started at AT&T Bell Labs
- UNIX was written in assembly, later changed to C
- Many variants of UNIX

WHY C and UNIX

- The first part of the course is Linux and bash
- The second part is C
- We will start with a quick introduction to Unix to be able to start the labs.
- Lab 1 is this week (introduction to Unix)
- Lab policy

Introduction to Unix

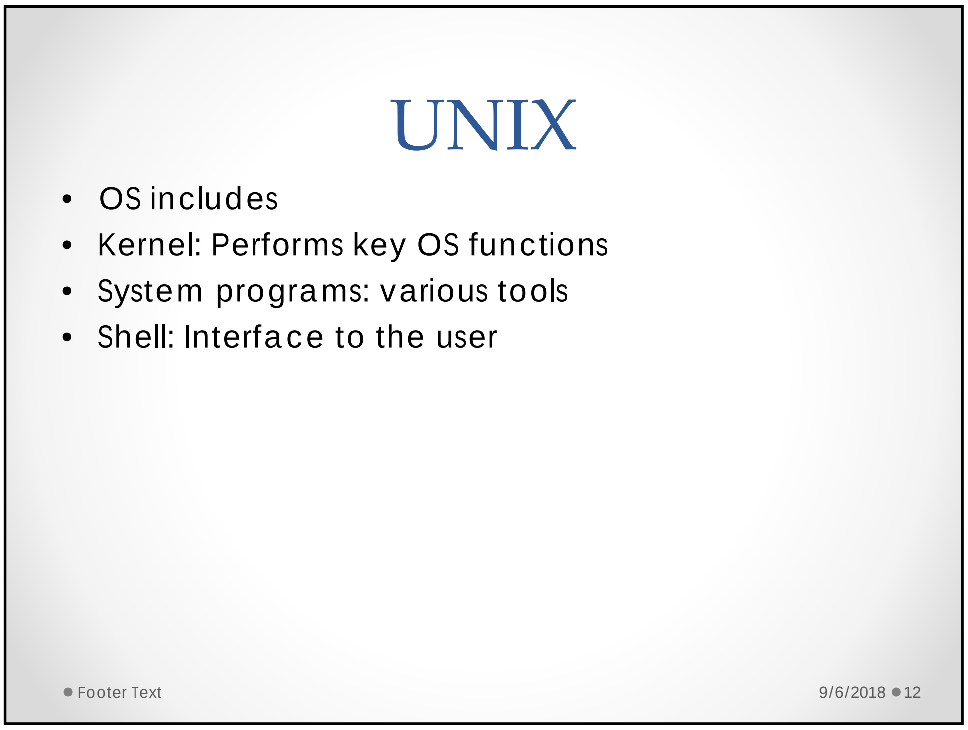
- Please check the tutorial at http://www.cs.sfu.ca/~ggbaker/reference/unix/
- The first 4 tutorials
- Blackboard



UNIX

- What does an OS do?
  - File management
  - Scheduling
  - Memory management
  - I/O management
- Examples

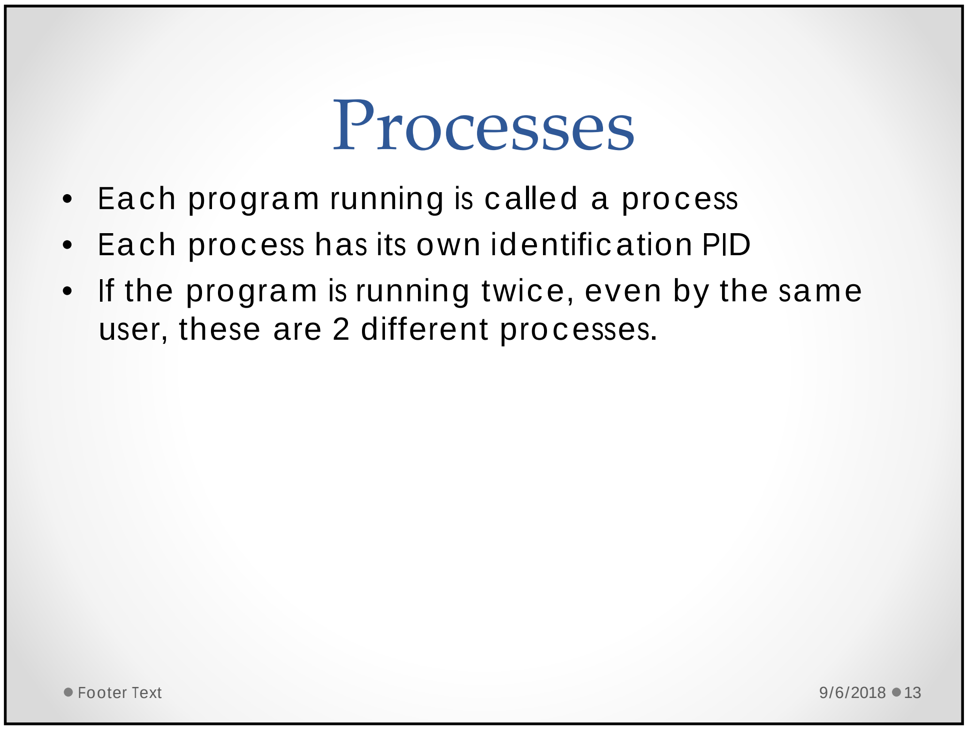
Footer Text 9/6/2018 11



UNIX

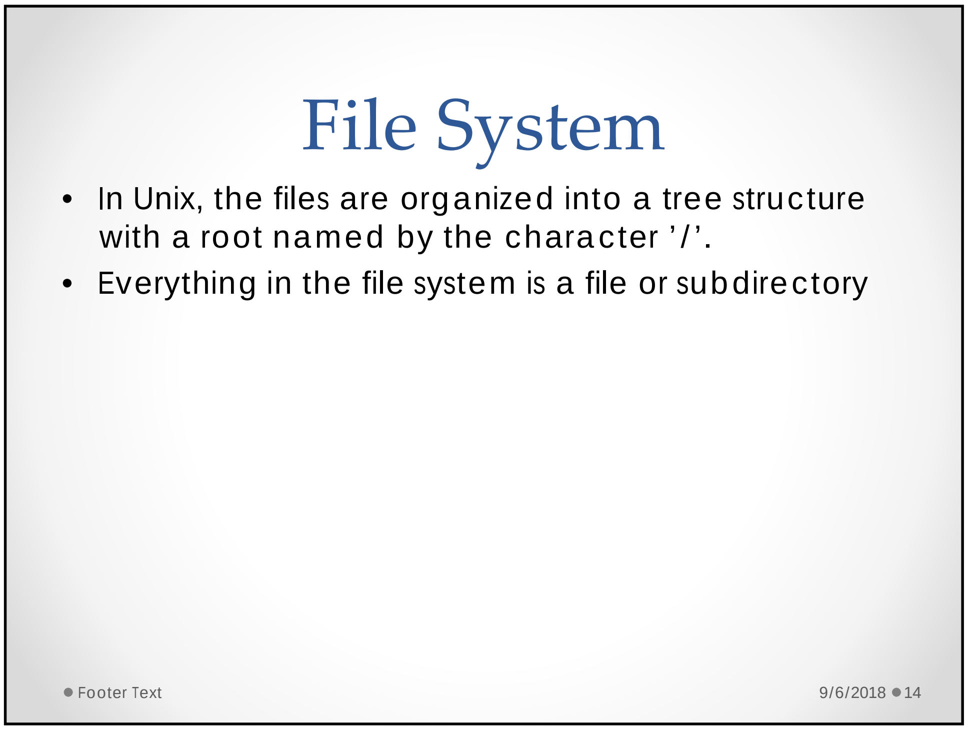
- OS includes
- Kernel: Performs key OS functions
- System programs: various tools
- Shell: Interface to the user

Footer Text 9/6/2018 12



Processes

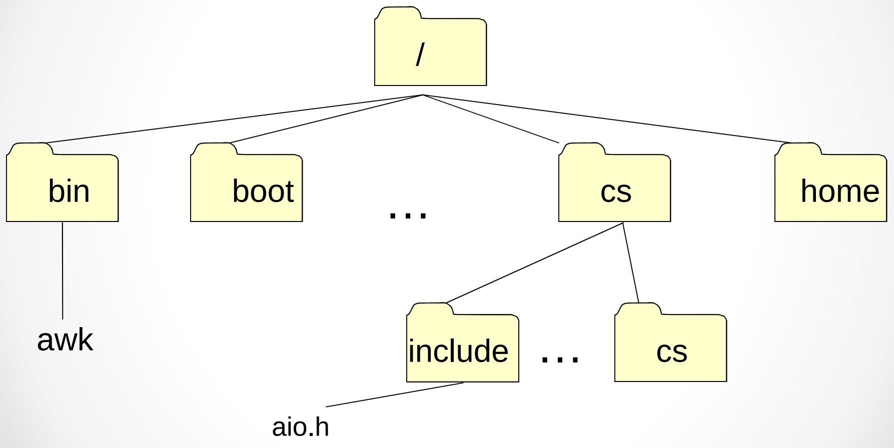
- Each program running is called a process
- Each process has its own identification PID
- If the program is running twice, even by the same user, these are 2 different processes.

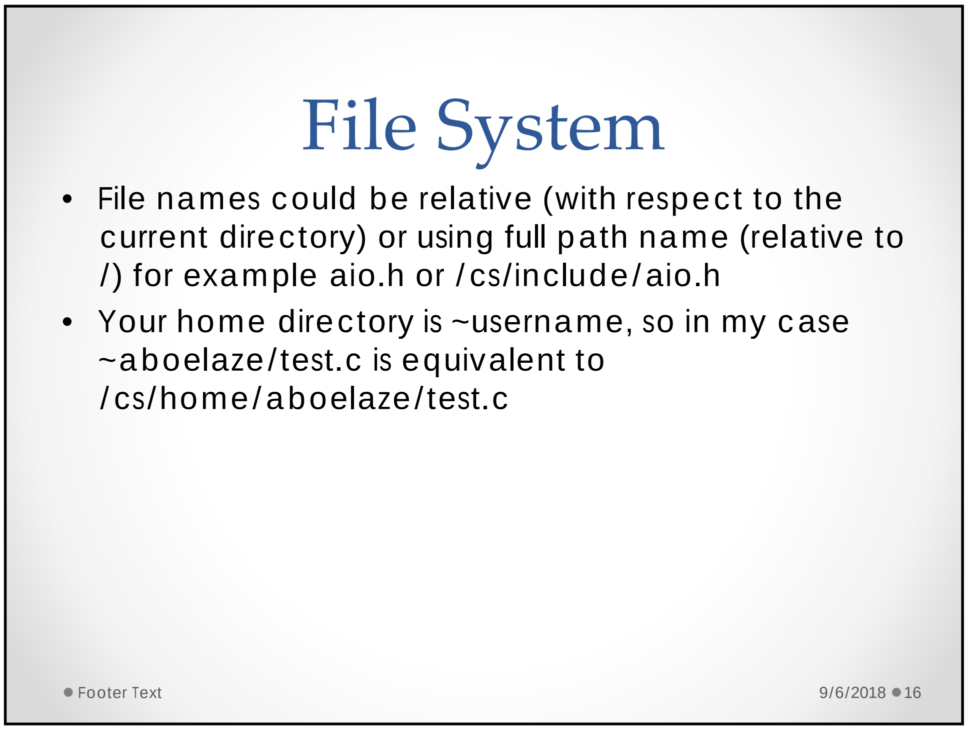


File System

- In Unix, the files are organized into a tree structure with a root named by the character '/'.
- Everything in the file system is a file or subdirectory

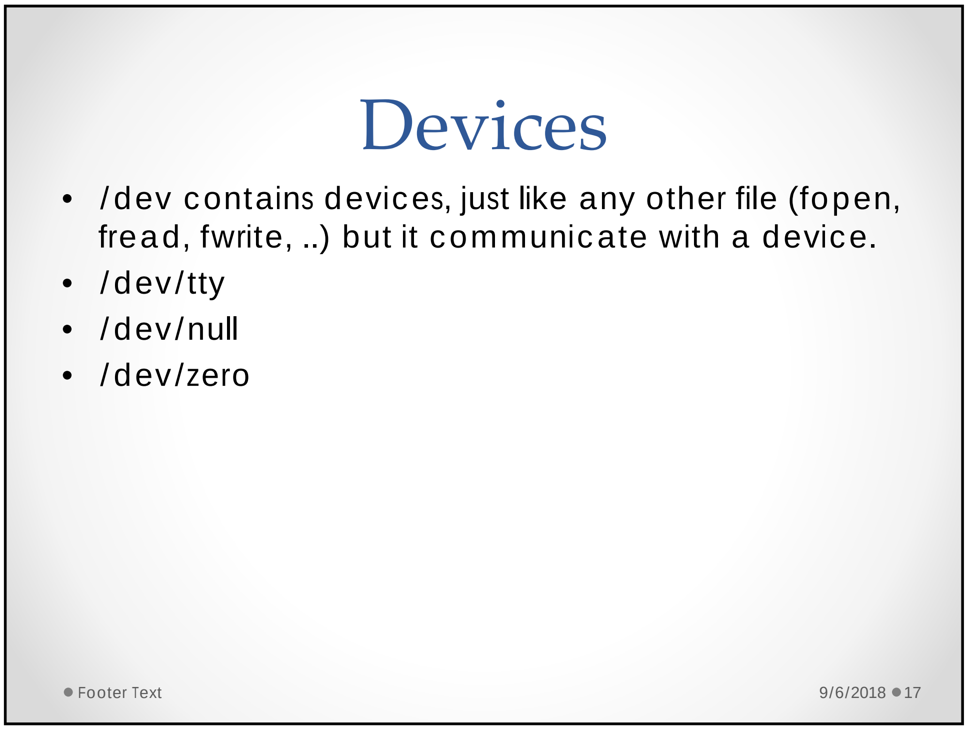
File System





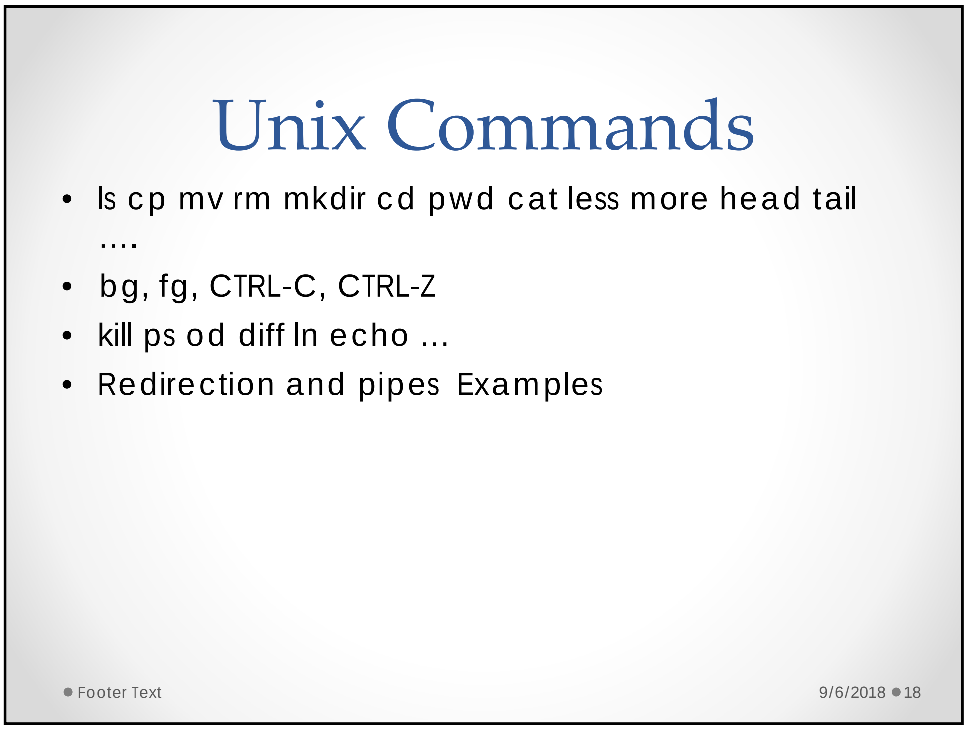
File System

- File names could be relative (with respect to the current directory) or using full path name (relative to /) for example aio.h or /cs/include/aio.h
- Your home directory is ~username, so in my case ~aboelaze/test.c is equivalent to /cs/home/aboelaze/test.c



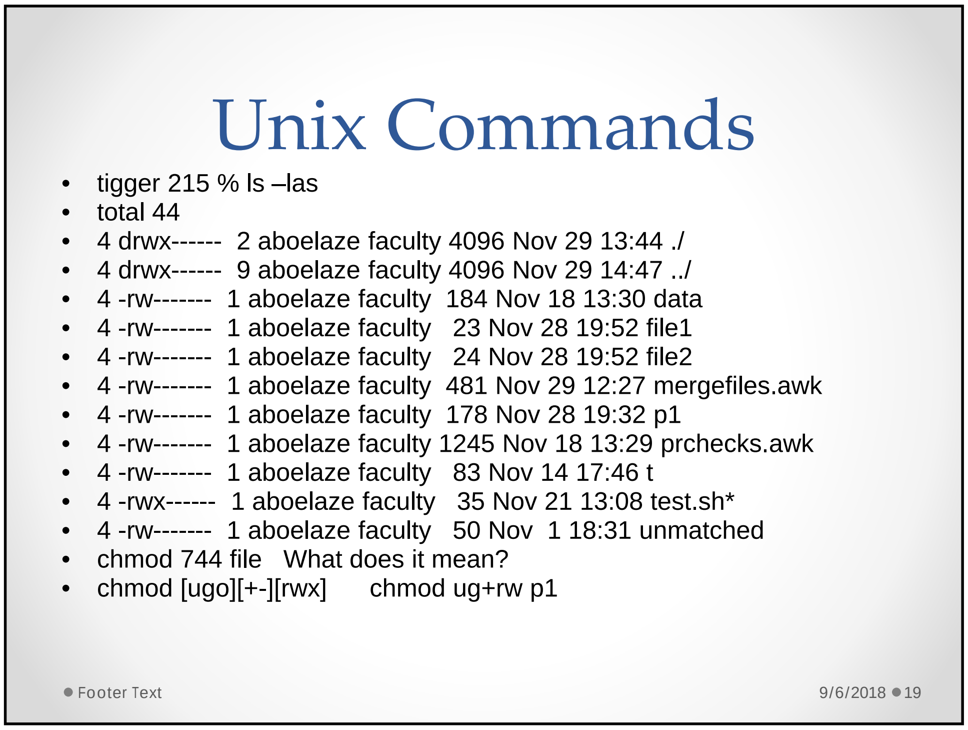
Devices

- /dev contains devices, just like any other file (fopen, fread, fwrite, ..) but it communicate with a device.
- /dev/tty
- /dev/null
- /dev/zero



Unix Commands

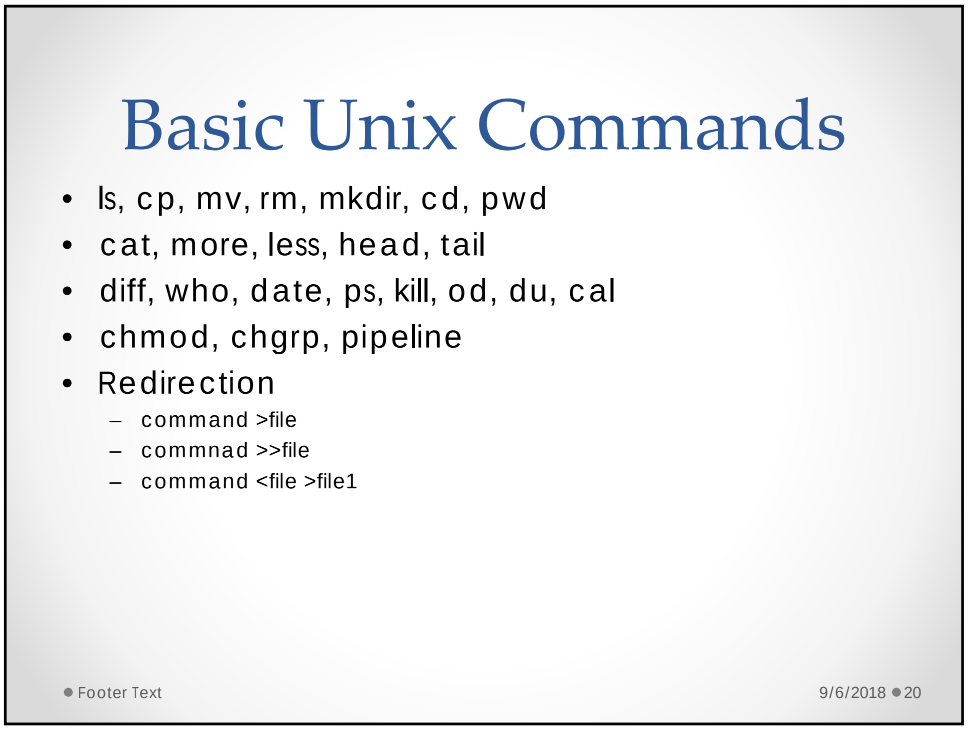
- ls cp mv rm mkdir cd pwd cat less more head tail ….
- bg, fg, CTRL-C, CTRL-Z
- kill ps od diff ln echo …
- Redirection and pipes Examples



Unix Commands

- tigger 215 % ls –las
- total 44
- 4 drwx------ 2 aboelaze faculty 4096 Nov 29 13:44 ./
- 4 drwx------ 9 aboelaze faculty 4096 Nov 29 14:47 ../
- 4 -rw------- 1 aboelaze faculty 184 Nov 18 13:30 data
- 4 -rw------- 1 aboelaze faculty   23 Nov 28 19:52 file1
- 4 -rw------- 1 aboelaze faculty   24 Nov 28 19:52 file2
- 4 -rw------- 1 aboelaze faculty 481 Nov 29 12:27 mergefiles.awk
- 4 -rw------- 1 aboelaze faculty 178 Nov 28 19:32 p1
- 4 -rw------- 1 aboelaze faculty 1245 Nov 18 13:29 prchecks.awk
- 4 -rw------- 1 aboelaze faculty   83 Nov 14 17:46 t
- 4 -rwx------ 1 aboelaze faculty   35 Nov 21 13:08 test.sh*
- 4 -rw------- 1 aboelaze faculty   50 Nov 1 18:31 unmatched
- chmod 744 file   What does it mean?
- chmod [ugo][+-][rwx]   chmod ug+rw p1

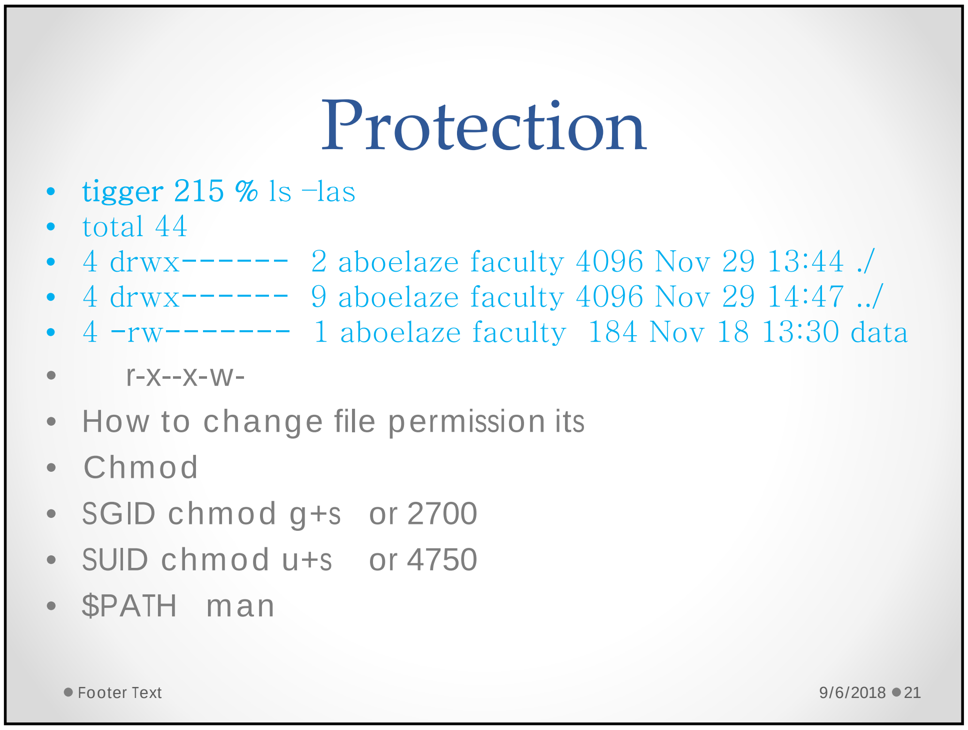
Footer Text   9/6/2018 19



Basic Unix Commands

- ls, cp, mv, rm, mkdir, cd, pwd
- cat, more, less, head, tail
- diff, who, date, ps, kill, od, du, cal
- chmod, chgrp, pipeline
- Redirection
  - command >file
  - commnad >>file
  - command <file >file1

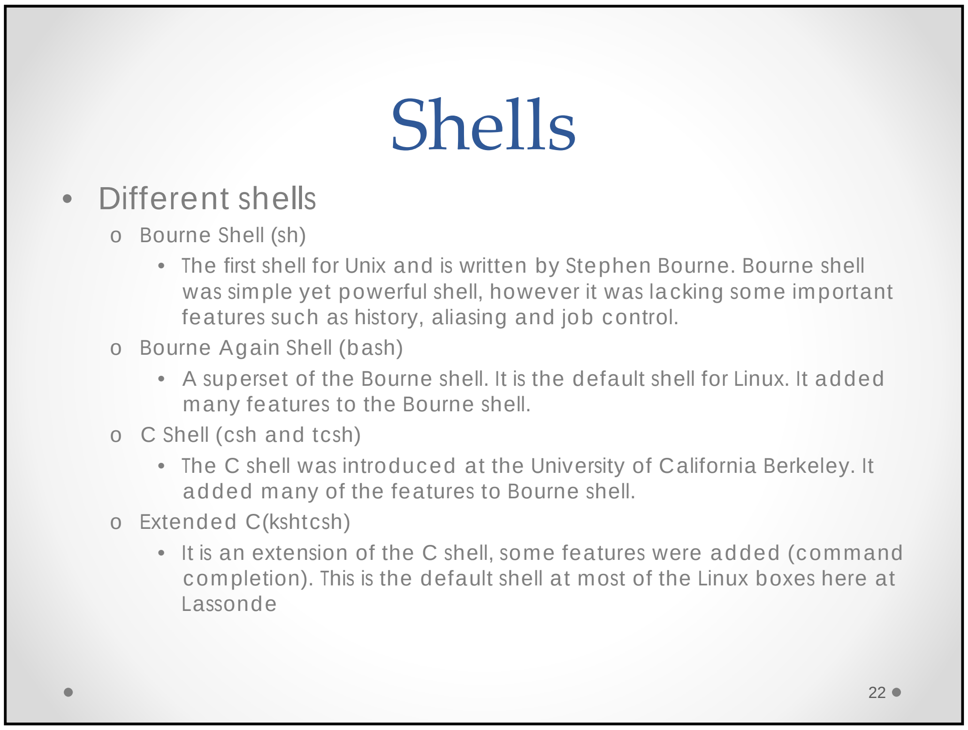
Footer Text   9/6/2018 20



Protection

- tigger 215 % ls –las
- total 44
- 4 drwx------ 2 aboelaze faculty 4096 Nov 29 13:44 ./
- 4 drwx------ 9 aboelaze faculty 4096 Nov 29 14:47 ../
- 4 -rw------- 1 aboelaze faculty 184 Nov 18 13:30 data
- r-x--x-w-
- How to change file permission its
- Chmod
- SGID chmod g+s or 2700
- SUID chmod u+s or 4750
- $PATH man

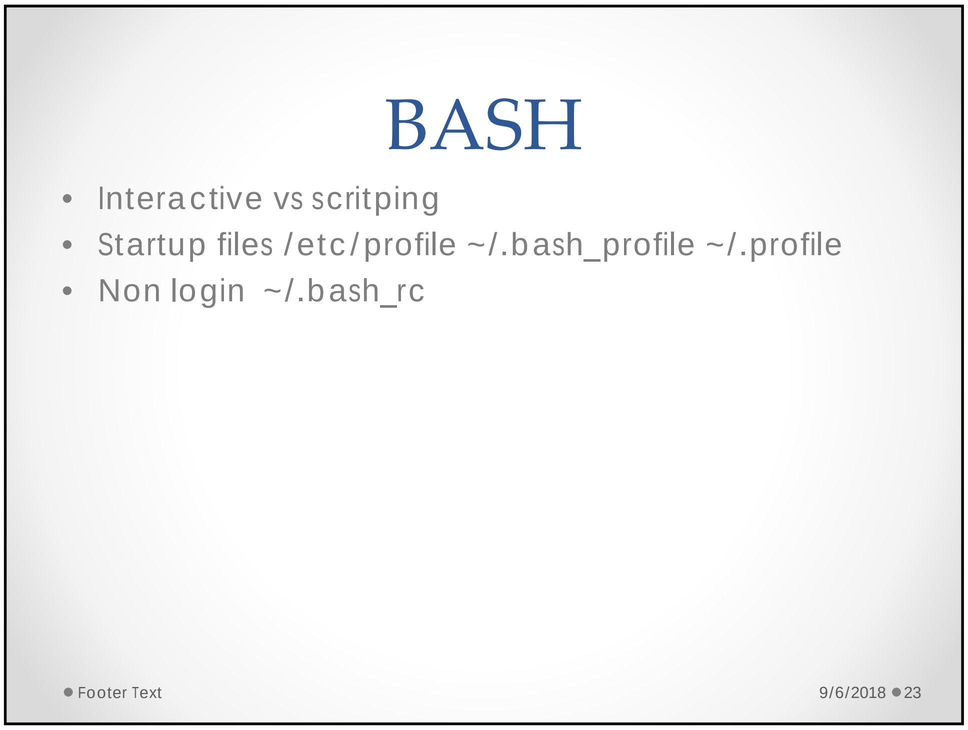
Footer Text 9/6/2018 21



Shells

- Different shells
  - Bourne Shell (sh)
    - The first shell for Unix and is written by Stephen Bourne. Bourne shell was simple yet powerful shell, however it was lacking some important features such as history, aliasing and job control.
  - Bourne Again Shell (bash)
    - A superset of the Bourne shell. It is the default shell for Linux. It added many features to the Bourne shell.
  - C Shell (csh and tcsh)
    - The C shell was introduced at the University of California Berkeley. It added many of the features to Bourne shell.
  - Extended C(kshtcsh)
    - It is an extension of the C shell, some features were added (command completion). This is the default shell at most of the Linux boxes here at Lassonde

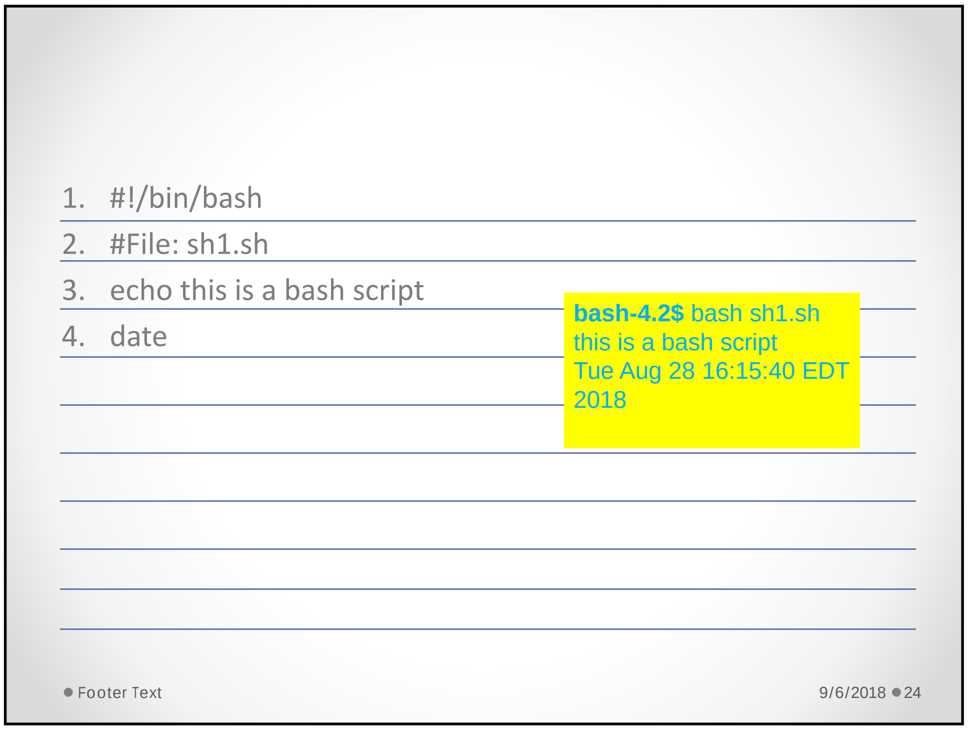
22



BASH

- Interactive vs scritping
- Startup files /etc/profile ~/.bash_profile ~/.profile
- Non login ~/.bash_rc

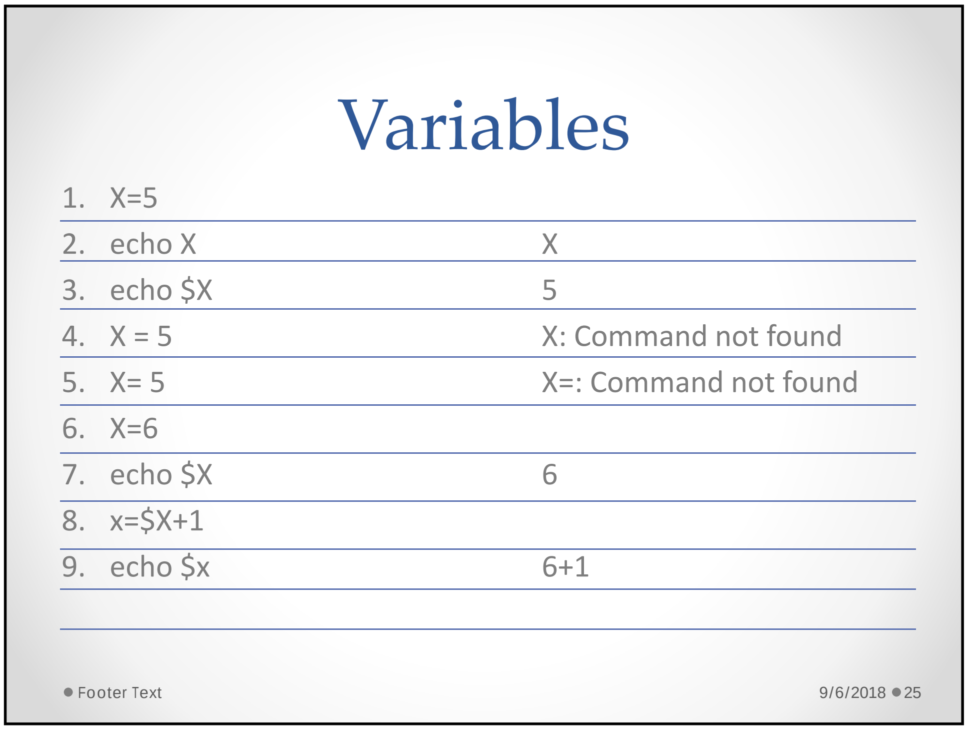
Footer Text 9/6/2018 23



1. #!/bin/bash
2. #File: sh1.sh
3. echo this is a bash script
4. date

```
bash-4.2$ bash sh1.sh
this is a bash script
Tue Aug 28 16:15:40 EDT
2018
```

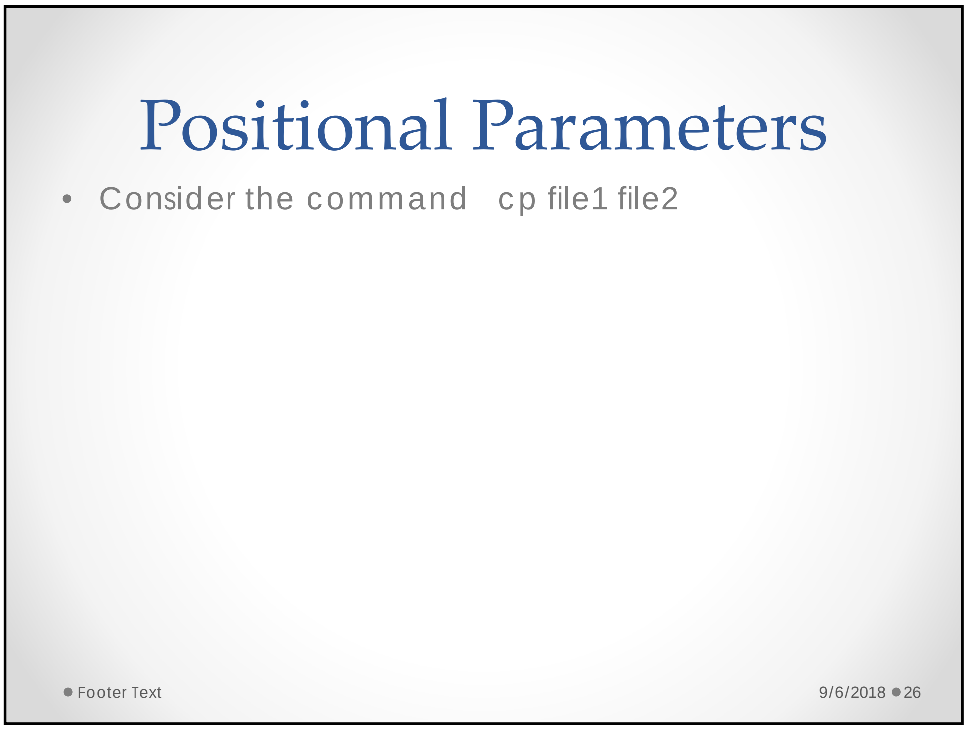
Footer Text 9/6/2018 24



Variables

1. X=5	
2. echo X	X
3. echo $X	5
4. X = 5	X: Command not found
5. X= 5	X=: Command not found
6. X=6	
7. echo $X	6
8. x=$X+1	
9. echo $x	6+1

Footer Text 9/6/2018 25



Positional Parameters

- Consider the command cp file1 file2

Footer Text 9/6/2018 26

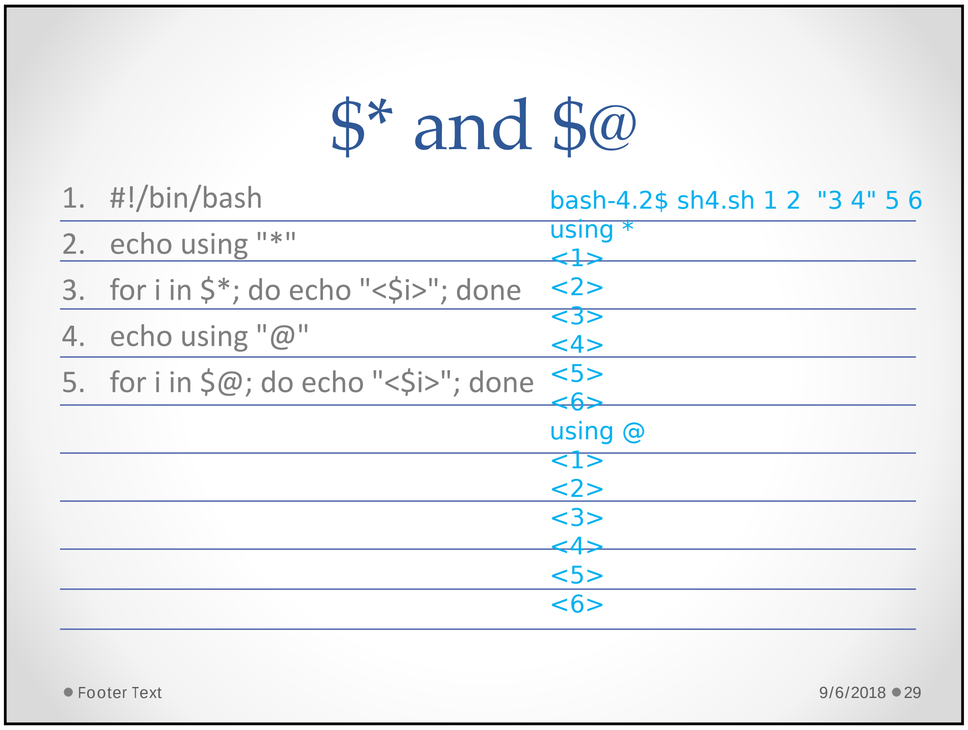
1. #!/bin/bash
2. #File: sh2.sh
3. echo The first parameter is $1
4. echo The second is $2

```
bash-4.2$ sh2.sh a 7856
The first parameter is a
The second is 7856
```

5. shift 2 ?????

Positional Variables

Command	Result and explanation
$0	The name of the script
$1 $2 …. $9 ${10} ….	The different positional parameters
$*	list of all positional parameters
$#	The number of parameters
$@	list of all arguments differs from $* only when in double quotes
$?	The exit status of the process
$$	The pid of the current shell (not subshell)
$IFS	The input field separator

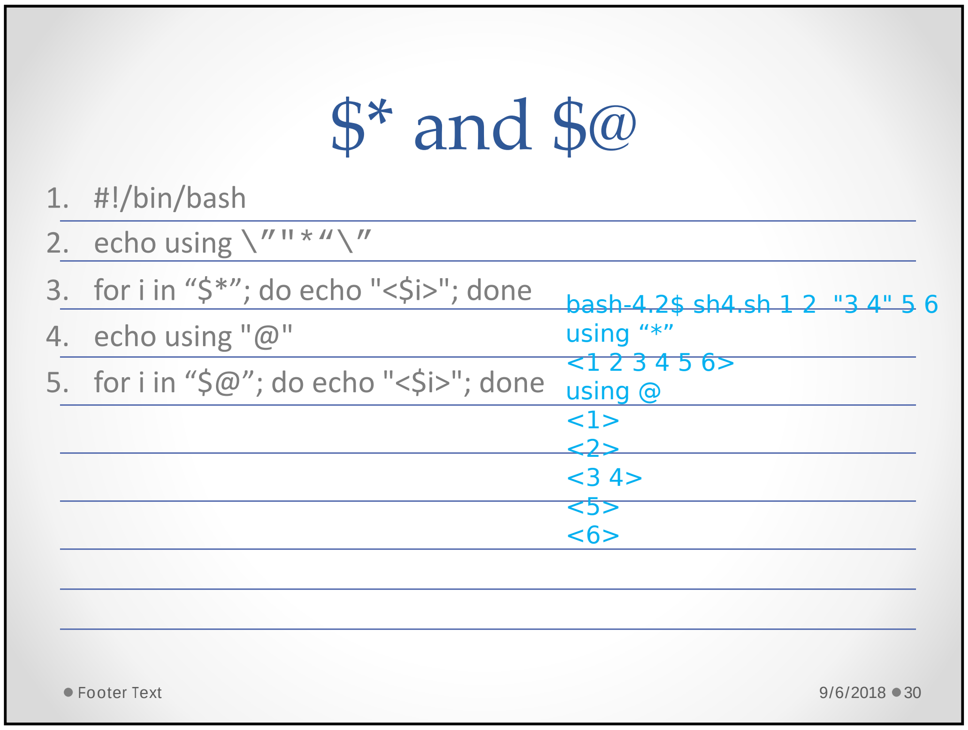


$* and $@

```
#!/bin/bash
echo using "*"
for i in $*; do echo "<$i>"; done
echo using "@"
for i in $@; do echo "<$i>"; done
```

```
bash-4.2$ sh4.sh 1 2  "3 4" 5 6
using *
<1>
<2>
<3>
<4>
<5>
<6>
using @
<1>
<2>
<3>
<4>
<5>
<6>
```

Footer Text 9/6/2018 29

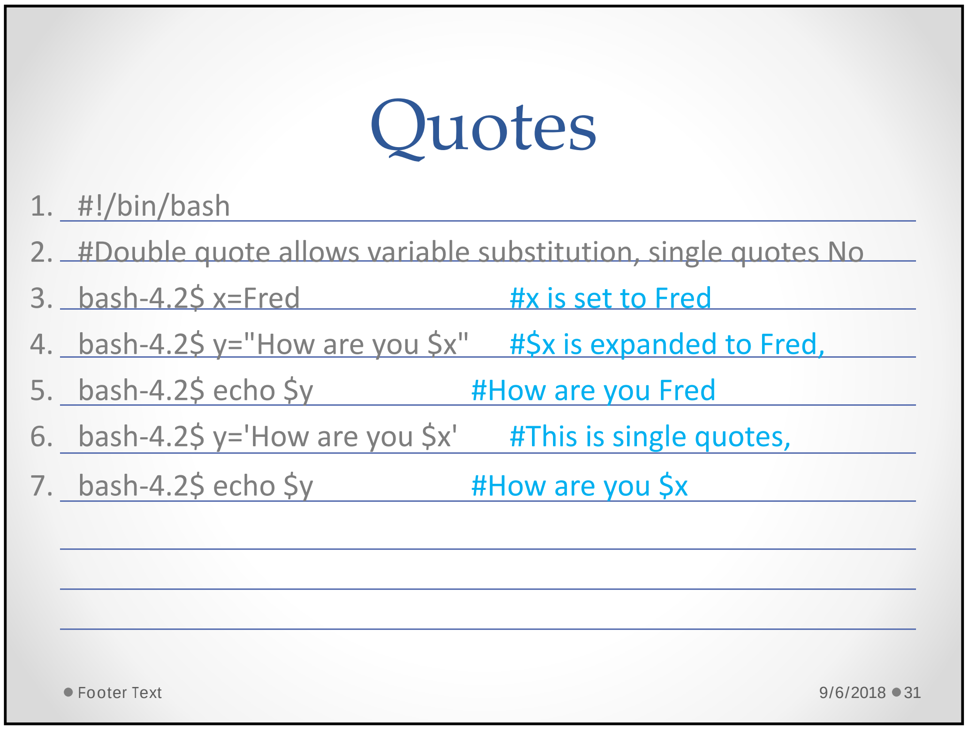


$* and $@

```
#!/bin/bash
echo using \”"*“\”
for i in “$*”; do echo "<$i>"; done
echo using "@"
for i in “$@”; do echo "<$i>"; done
```

```
bash-4.2$ sh4.sh 1 2  "3 4" 5 6
using “*”
<1 2 3 4 5 6>
using @
<1>
<2>
<3 4>
<5>
<6>
```

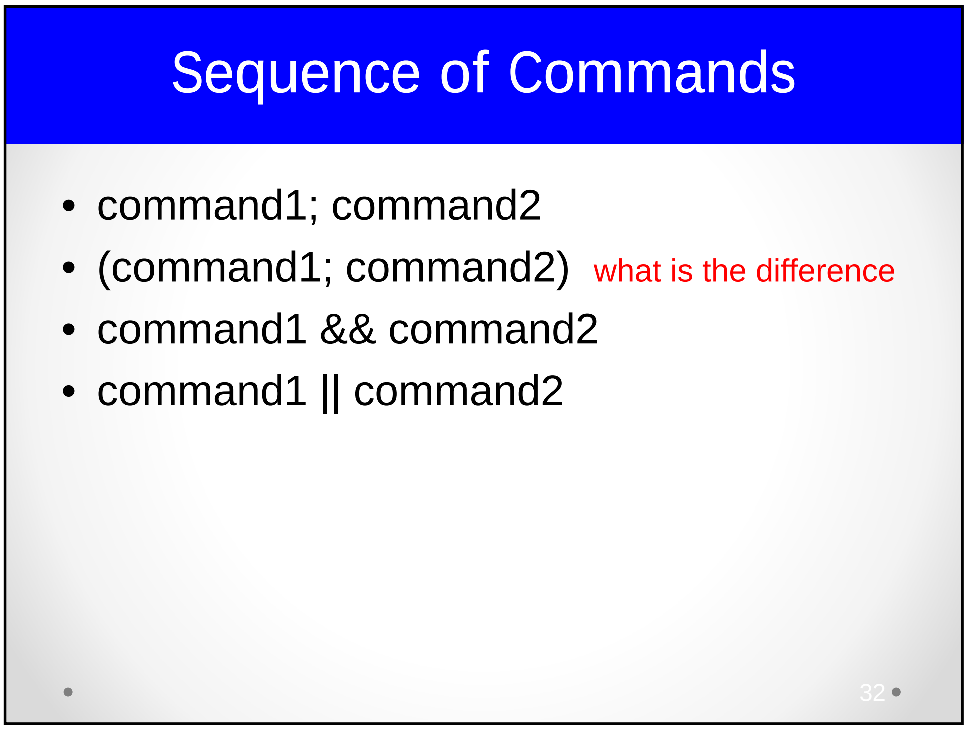
Footer Text 9/6/2018 30



Quotes

```
#!/bin/bash
#Double quote allows variable substitution, single quotes No
bash-4.2$ x=Fred                        #x is set to Fred
bash-4.2$ y="How are you $x"            #$x is expanded to Fred,
bash-4.2$ echo $y                       #How are you Fred
bash-4.2$ y='How are you $x'            #This is single quotes,
bash-4.2$ echo $y                       #How are you $x
```

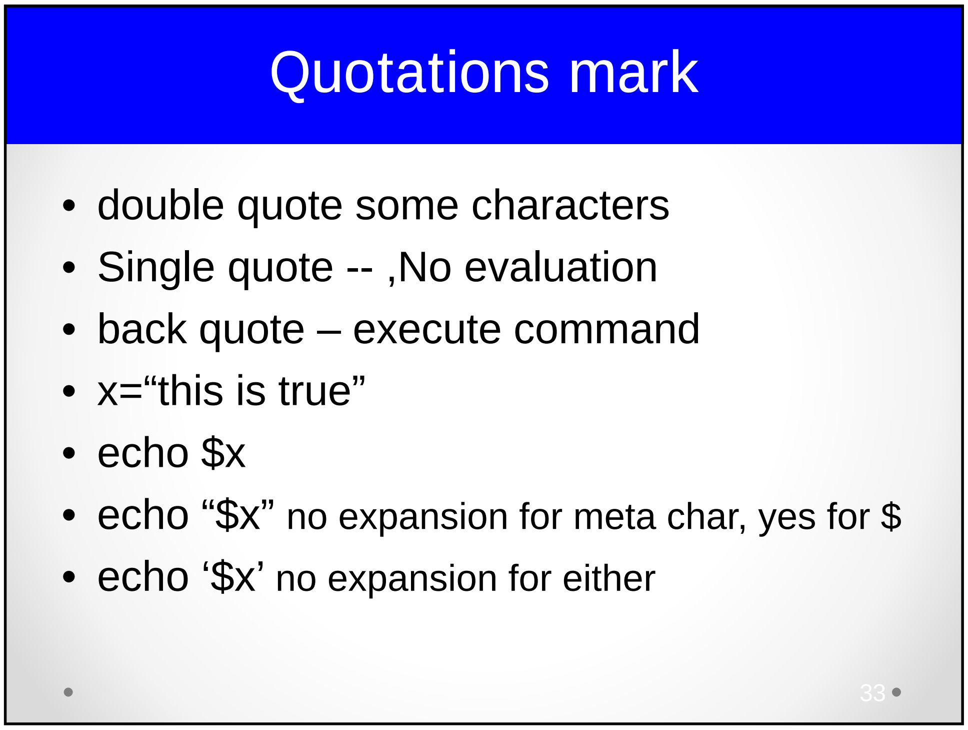
Footer Text 9/6/2018 31



Sequence of Commands

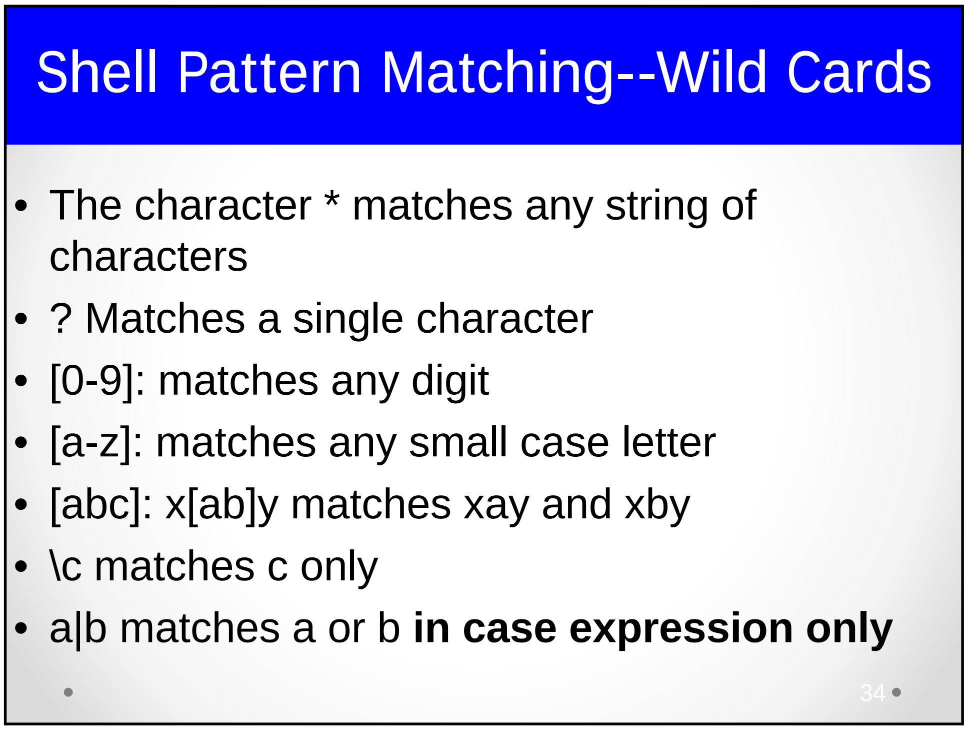
- command1; command2
- (command1; command2) what is the difference
- command1 && command2
- command1 || command2

32



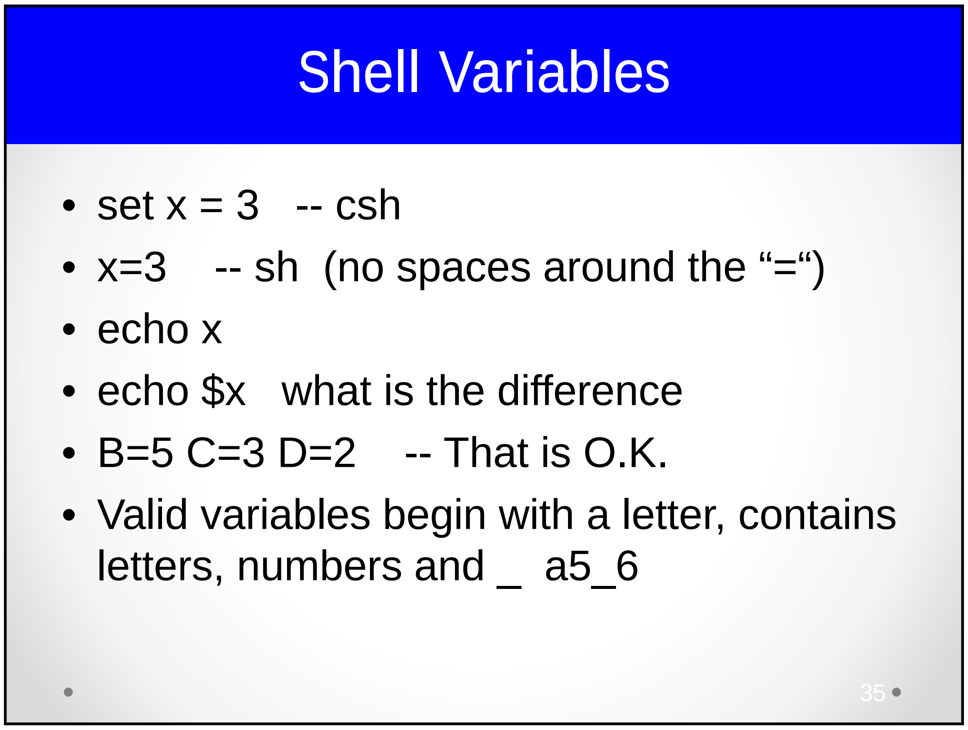
Quotations mark

- double quote some characters
- Single quote -- ,No evaluation
- back quote – execute command
- x=“this is true”
- echo $x
- echo “$x” no expansion for meta char, yes for $
- echo ‘$x’ no expansion for either



Shell Pattern Matching--Wild Cards

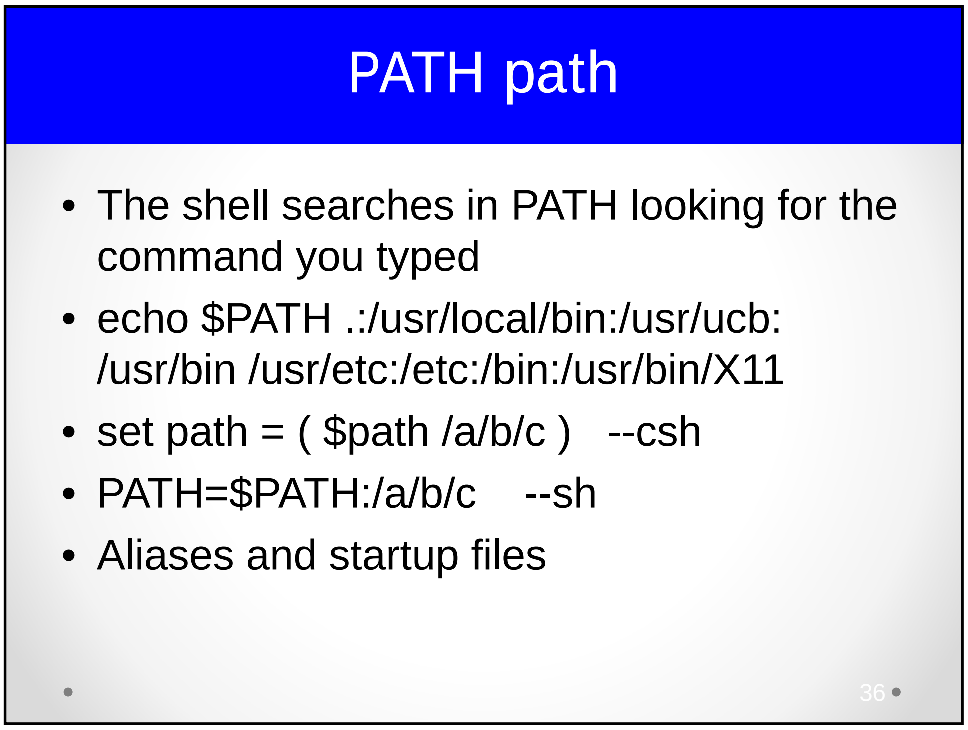
- The character * matches any string of characters
- ? Matches a single character
- [0-9]: matches any digit
- [a-z]: matches any small case letter
- [abc]: x[ab]y matches xay and xby
- \c matches c only
- a|b matches a or b **in case expression only**



Shell Variables

- set x = 3   -- csh
- x=3   -- sh (no spaces around the “=“)
- echo x
- echo $x   what is the difference
- B=5 C=3 D=2   -- That is O.K.
- Valid variables begin with a letter, contains letters, numbers and _ a5_6

35



PATH path

- The shell searches in PATH looking for the command you typed
- echo $PATH .:/usr/local/bin:/usr/ucb: /usr/bin /usr/etc:/etc:/bin:/usr/bin/X11
- set path = ($path /a/b/c)   --csh
- PATH=$PATH:/a/b/c   --sh
- Aliases and startup files

36

Shell scripting

```
#!/cs/local/bin/sh
echo “Hello World”
```

```
tigger 397 % script1
Hello World
tigger 398 %
```

```
echo  -n “Hello
World”
```

```
tigger 393 % script1
Hello Worldtigger 394 %
```

```
#!/cs/local/bin/sh
echo "Now I will guess your OS"
echo  -n "Your OS is : "
uname
```

```
tigger 399 % script1
Now I will guess your OS
Your OS is : Linux
tigger 400 %
```

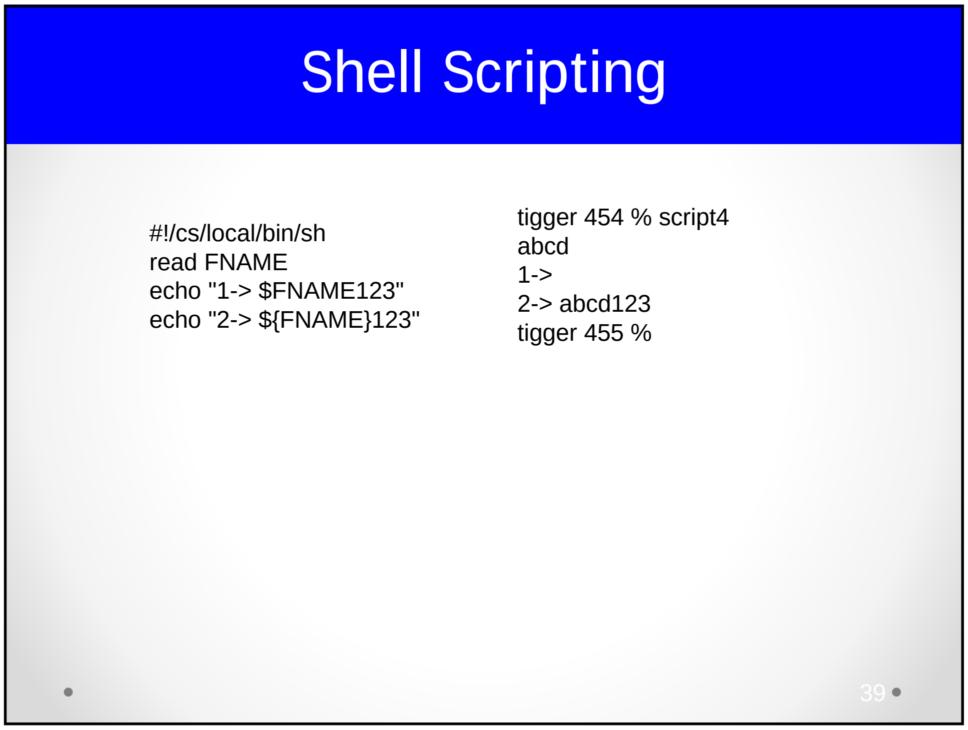
37

Shell Scripting

```
#!/cs/local/bin/sh
echo -n "Please enter your first name : "
read FNAME
echo -n "Last name pelase : "
read LNAME
MESSAGE=" Your name is : $LNAME , $FNAME"
echo "$MESSAGE"
```

```
tigger 439 % script3
Please enter your first name : Mokhtar
Last name pelase : Aboelaze
 Your name is : Aboelaze , Mokhtar
```

38

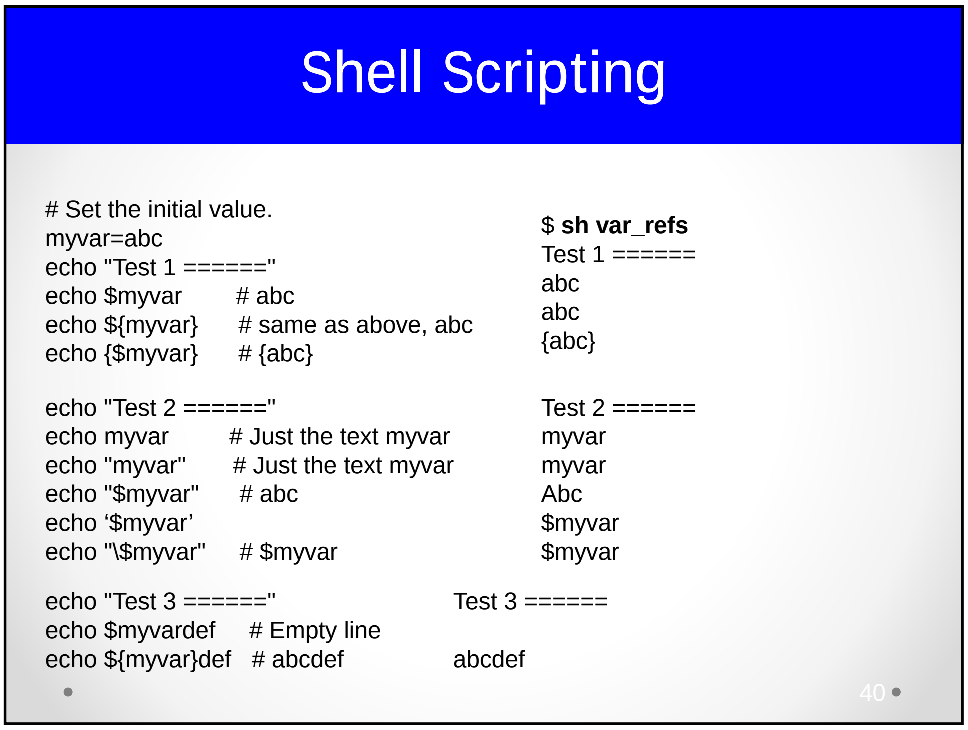


Shell Scripting

```
#!/cs/local/bin/sh
read FNAME
echo "1-> $FNAME123"
echo "2-> ${FNAME}123"
```

```
tigger 454 % script4
abcd
1->
2-> abcd123
tigger 455 %
```

39



Shell Scripting

```
# Set the initial value.
myvar=abc
echo "Test 1 ======"
echo $myvar        # abc
echo ${myvar}      # same as above, abc
echo {$myvar}      # {abc}

echo "Test 2 ======"
echo myvar         # Just the text myvar
echo "myvar"       # Just the text myvar
echo "$myvar"      # abc
echo '$myvar'
echo "\$myvar"     # $myvar

echo "Test 3 ======"
echo $myvardef     # Empty line
echo ${myvar}def   # abcdef
```

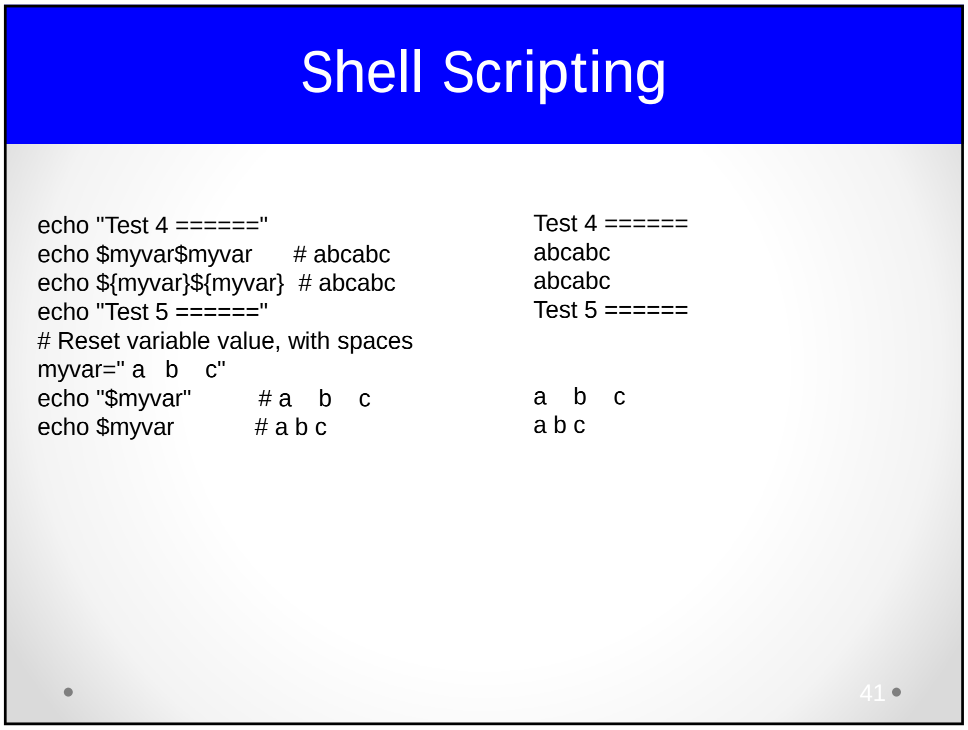
```
$ sh var_refs
Test 1 ======
abc
abc
{abc}

Test 2 ======
myvar
myvar
Abc
$myvar
$myvar

Test 3 ======

abcdef
```

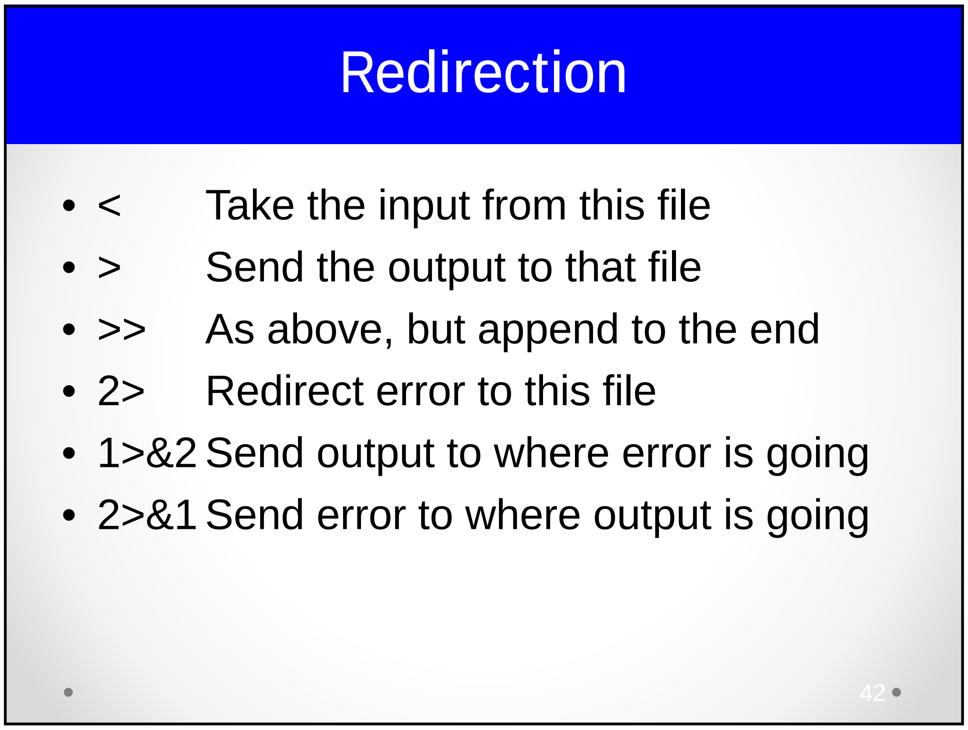
40



Shell Scripting

```
echo "Test 4 ======"                                    Test 4 ======
echo $myvar$myvar       # abcabc                       abcabc
echo ${myvar}${myvar}  # abcabc                        abcabc
echo "Test 5 ======"                                    Test 5 ======
# Reset variable value, with spaces
myvar=" a   b    c"
echo "$myvar"          # a    b    c                   a    b    c
echo $myvar            # a b c                          a b c
```

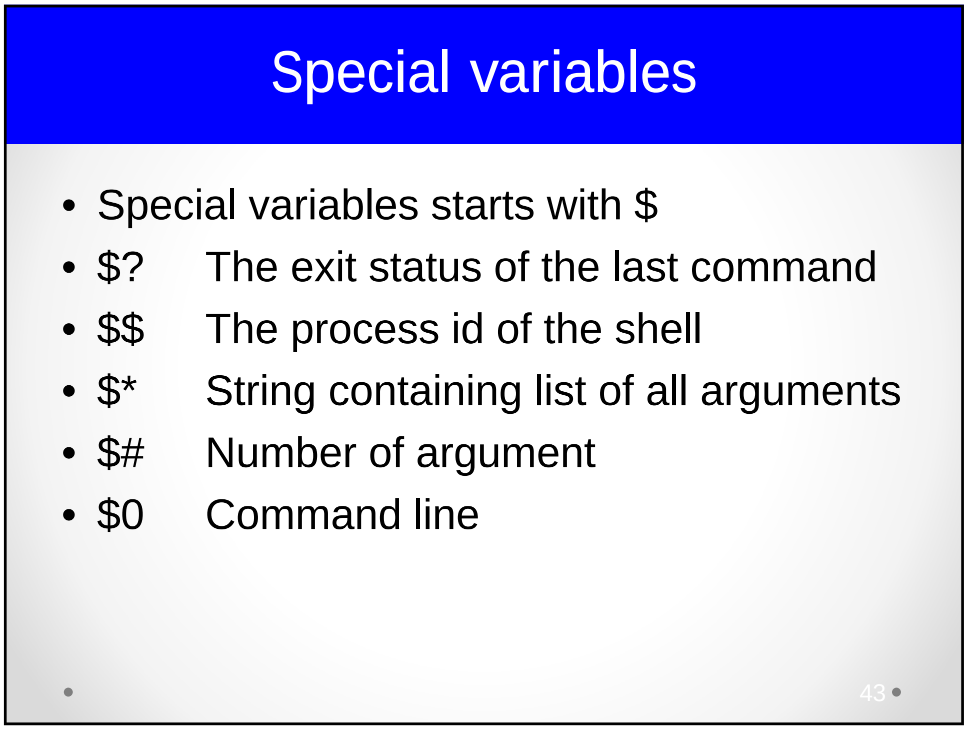
41



Redirection

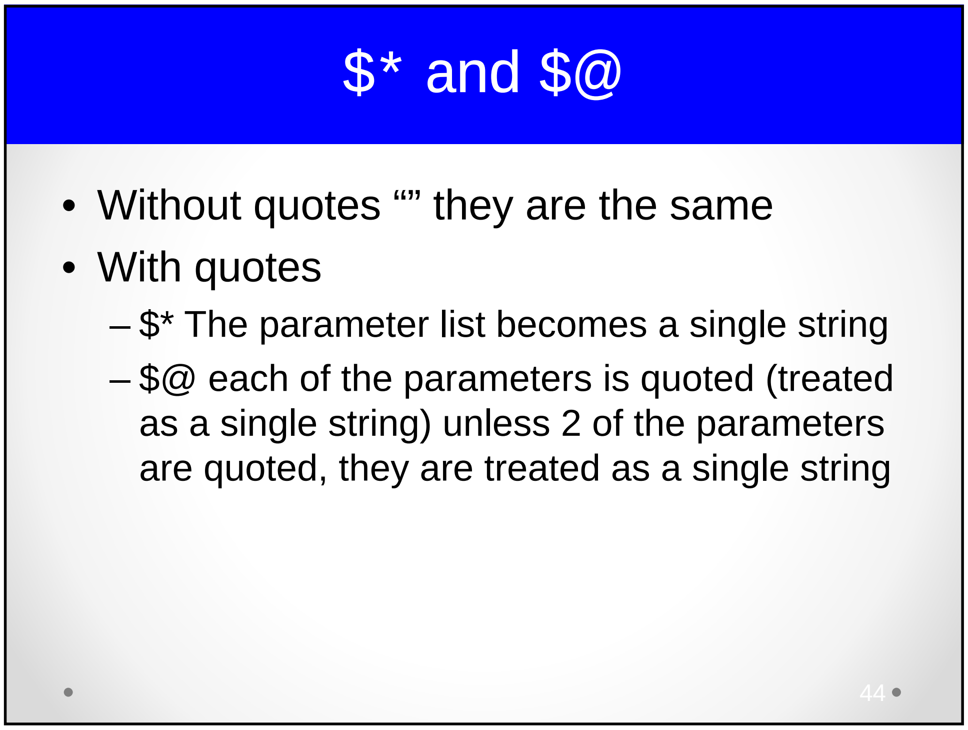
- <   Take the input from this file
- >   Send the output to that file
- >>   As above, but append to the end
- 2>   Redirect error to this file
- 1>&2 Send output to where error is going
- 2>&1 Send error to where output is going

42



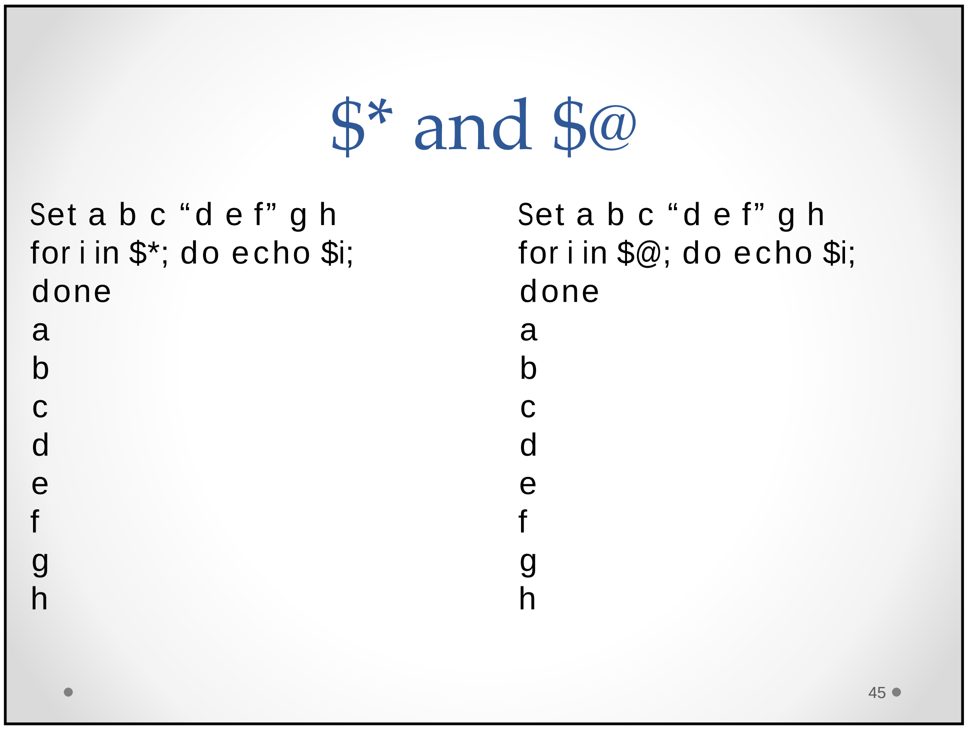
Special variables

- Special variables starts with $
- $? The exit status of the last command
- $$ The process id of the shell
- $* String containing list of all arguments
- $# Number of argument
- $0 Command line



$* and $@

- Without quotes “” they are the same
- With quotes
  - $* The parameter list becomes a single string
  - $@ each of the parameters is quoted (treated as a single string) unless 2 of the parameters are quoted, they are treated as a single string

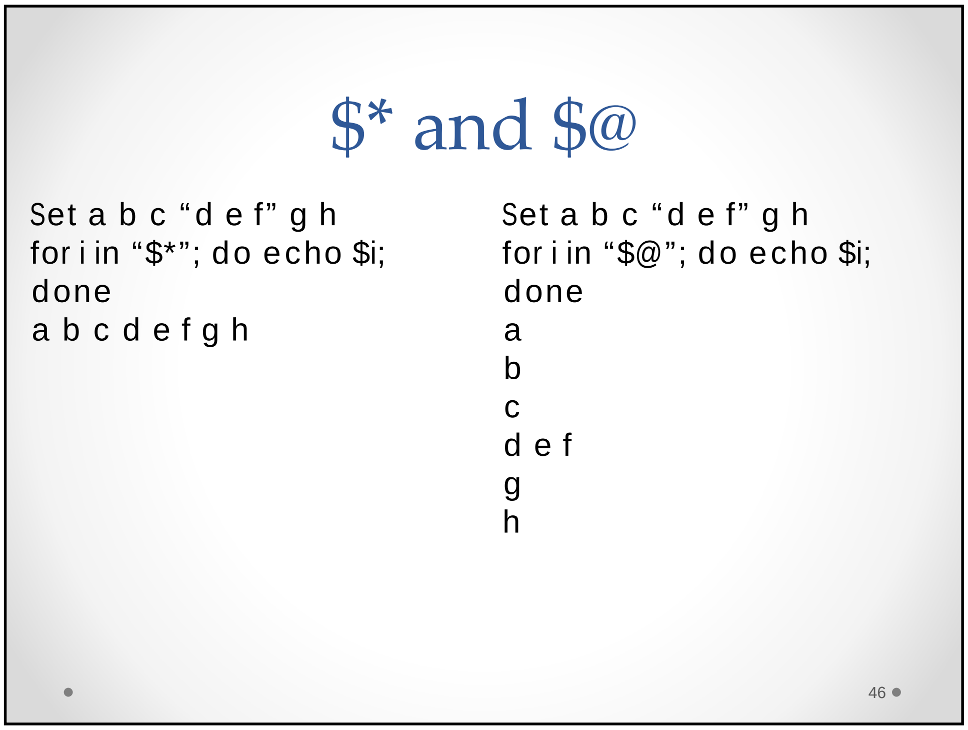


$* and $@

```
Set a b c “d e f” g h
for i in $*; do echo $i;
done
a
b
c
d
e
f
g
h
```

```
Set a b c “d e f” g h
for i in $@; do echo $i;
done
a
b
c
d
e
f
g
h
```

45

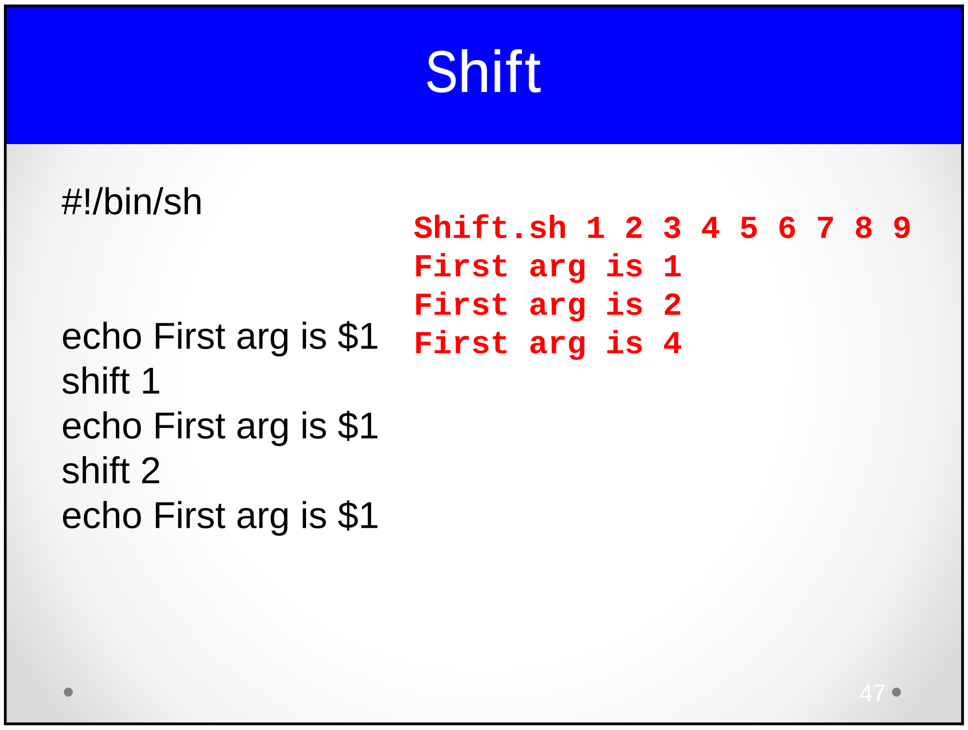


$* and $@

```
Set a b c “d e f” g h
for i in “$*”; do echo $i;
done
a b c d e f g h
```

```
Set a b c “d e f” g h
for i in “$@”; do echo $i;
done
a
b
c
d e f
g
h
```

46



Shift

```
#!/bin/sh


echo First arg is $1
shift 1
echo First arg is $1
shift 2
echo First arg is $1
```

```
Shift.sh 1 2 3 4 5 6 7 8 9
First arg is 1
First arg is 2
First arg is 4
```

47