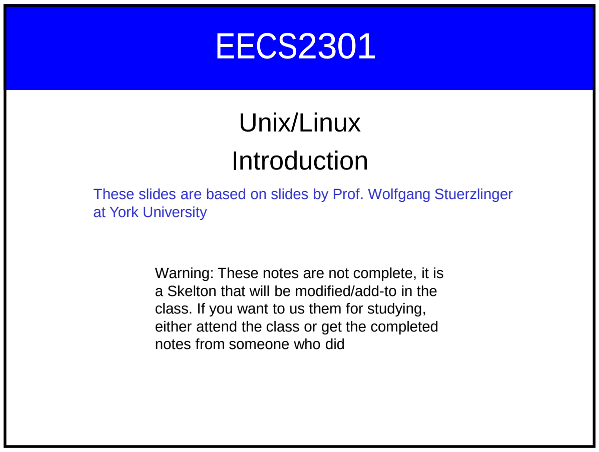


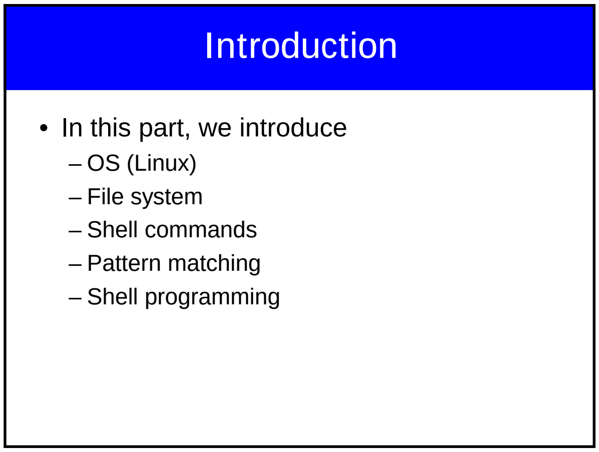
EECS2301

Unix/Linux

Introduction

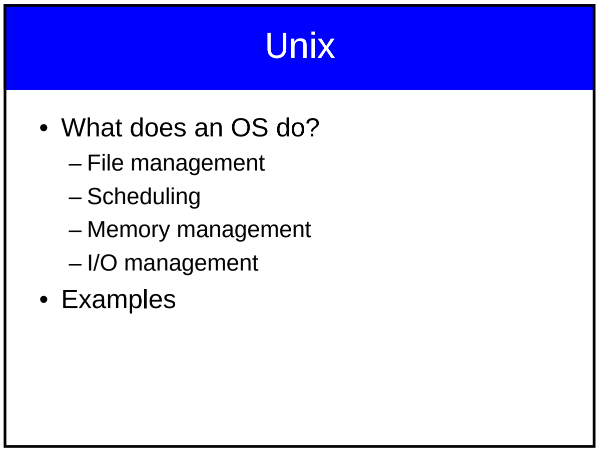
These slides are based on slides by Prof. Wolfgang Stuerzlinger at York University

Warning: These notes are not complete, it is a Skelton that will be modified/add-to in the class. If you want to us them for studying, either attend the class or get the completed notes from someone who did



Introduction

- In this part, we introduce
  - OS (Linux)
  - File system
  - Shell commands
  - Pattern matching
  - Shell programming



Unix

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

Unix

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

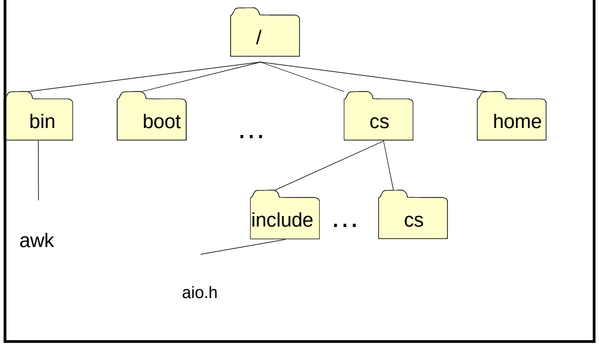
Processes

- 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

Our File System

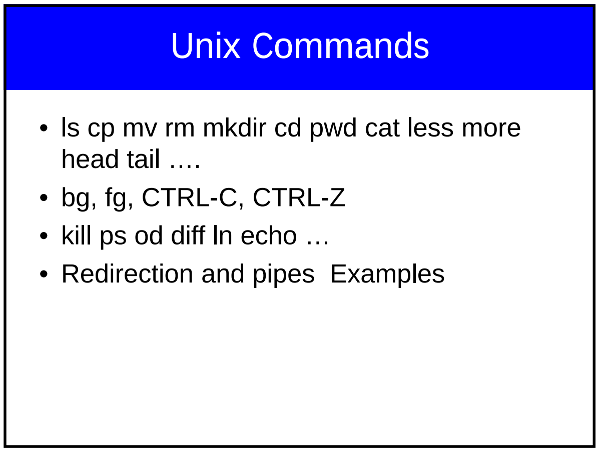


File System

- 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

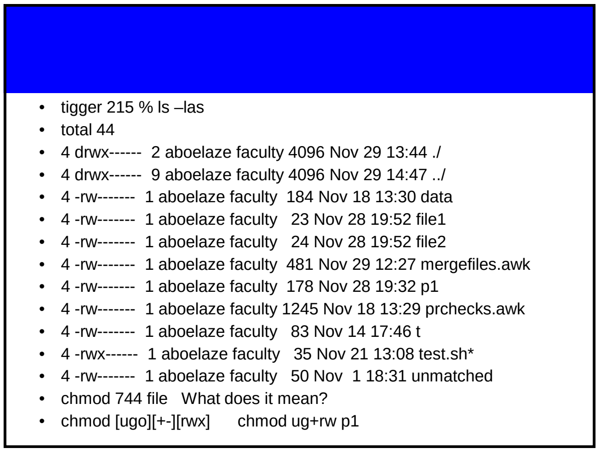
Devices

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



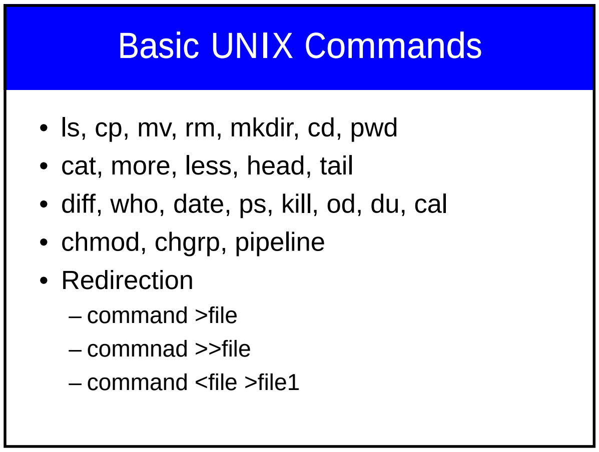
Unix Commands

- 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



```
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



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

Shells

Different shells

- Bourne Shell (sh)
- Bourne Again Shell (bash)
- C Shell (csh and tcsh)
- Korn Shell (ksh)

Sequence of Commands

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

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

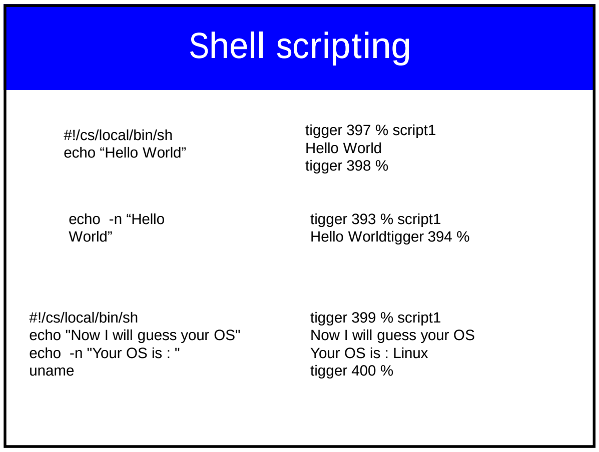
- The character * matches any string of characters
- ? Matches a single character
- [0-9]: matches any digit
- [a-z]: matches any small case letter
- [abc]: x[ab]y matches xay and xby
- \c matches c only
- a|b matches a or b **in case expression only**

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

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



Shell scripting

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

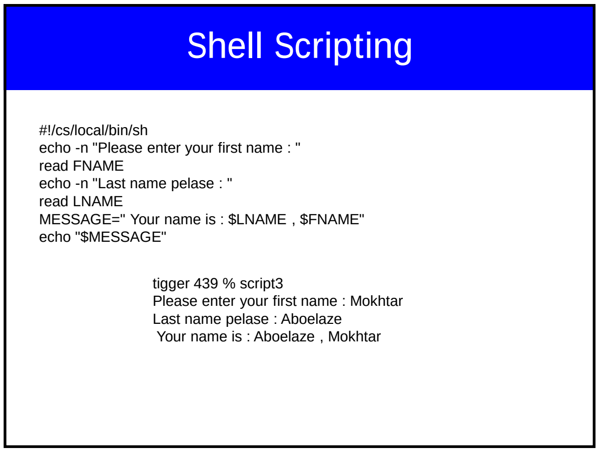
```
tigger 397 % script1
Hello World
tigger 398 %
```

```
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
```

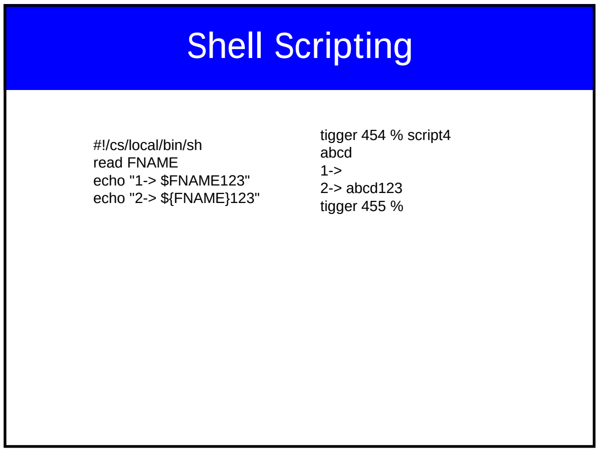
```
tigger 399 % script1
Now I will guess your OS
Your OS is : Linux
tigger 400 %
```



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"
```

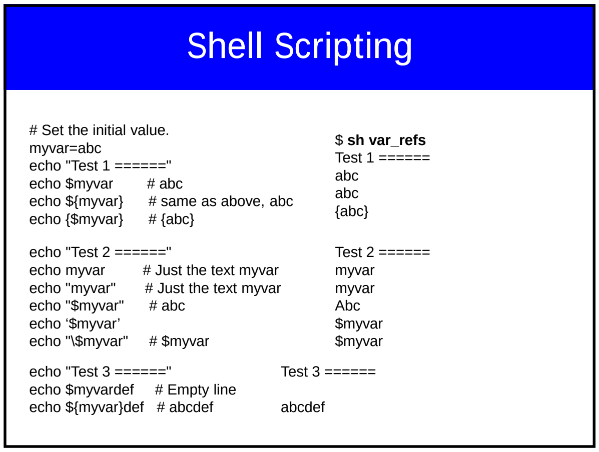
```
tigger 439 % script3
Please enter your first name : Mokhtar
Last name pelase : Aboelaze
 Your name is : Aboelaze , Mokhtar
```



Shell Scripting

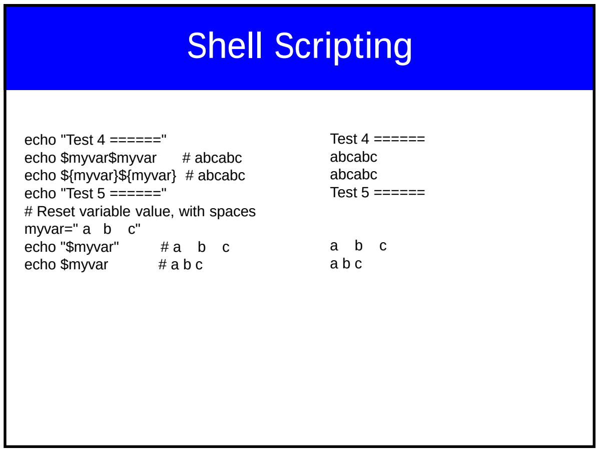
```
#!/cs/local/bin/sh
read FNAME
echo "1-> $FNAME123"
echo "2-> ${FNAME}123"
```

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



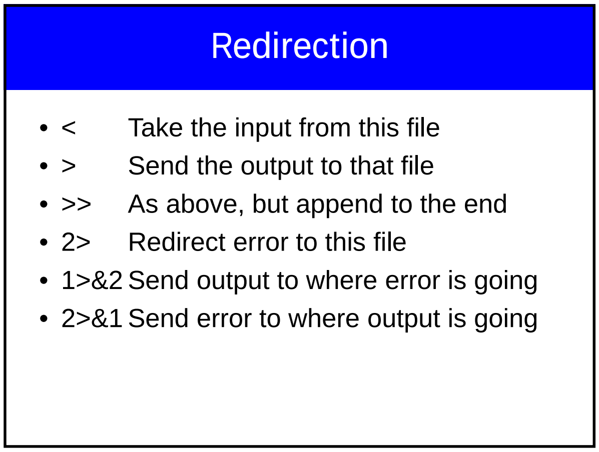
Shell Scripting

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

echo "Test 2 ======"                            Test 2 ======
echo myvar         # Just the text myvar        myvar
echo "myvar"       # Just the text myvar        myvar
echo "$myvar"      # abc                        Abc
echo ‘$myvar’                                   $myvar
echo "\$myvar"     # $myvar                     $myvar

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



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
```



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

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
  - S@ 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

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
```