



The Shell: The Most Powerful GUI

Anoop M. Namboodiri
Biometrics and Secure ID Lab
IIIT Hyderabad



About the Course

- Introduction of Software Systems: CS6.201
 - Lectures/Tutorials on Tuesdays and Fridays: 8:30AM – 10AM
 - Labs on Fridays: 2pm onwards
 - Lectures will end soon after Mid-Term
- Course webpage: <https://courses.iiit.ac.in/>
 - All details, resources, assignments are posted there
- Pre-requisites: C-Programming, Basics of Linux
- Textbooks: None
 - Lots of resources online.
 - Python documentation



Grading

- Tentative Weightage:
- Attendance
- Tutorials
- Project
- Academic Honesty

| Head | % of Grade |
|----------------------------|------------|
| Quiz 1 | 10 |
| Lab Exam | 10 |
| Mid-Term/Final | 15 |
| Hackathon (Final Lab Exam) | 15 |
| Project | 25 |
| Assignments | 10 |
| Labs | 15 |



Class Etiquettes

- Before the Lecture
 - This is a short course and we have lots to learn
 - Watch any videos / read any material given for preparation
 - We will start sharp at 8:30pm; Do not be late
- During the Class
 - Please pay attention in the class
 - Keep your cell phones muted
 - If you have a doubt, ask. Others are likely to have the same doubt.
 - Take notes: Not every topic discussed in the class comes from a textbook; Not everything is on the slides.



Questions?



The Shell

- Your interface to the OS
 - Command Line Interface
- Different Shells:
 - sh, ksh, bash, zsh
- Far more powerful than GUI to control the computer
 - Doing a set of commands repeatedly; and more.
- Commands, Permission and Path
- Complex commands:
 - Pipes, Redirection
- Commands and Processes

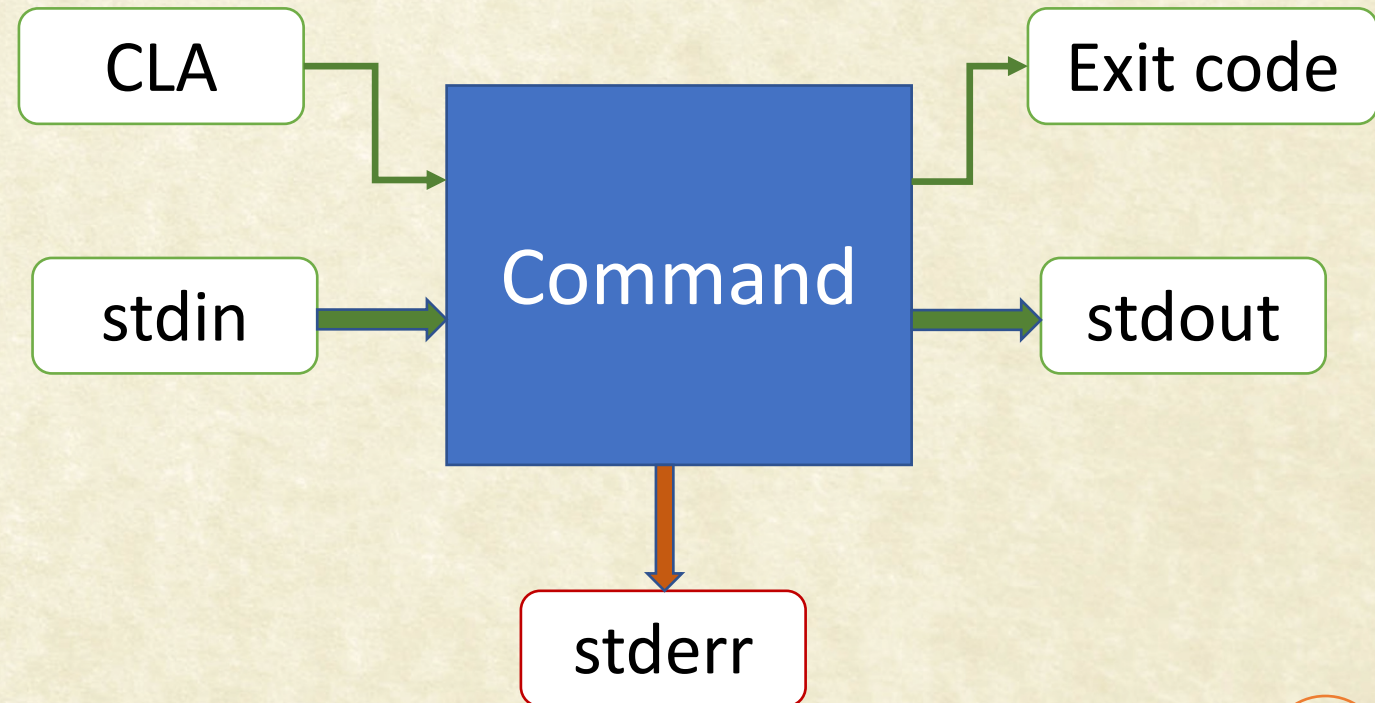
```
anoop — -zsh — 36x9
anoop@MBA ~ % ls
Applications      Library
CLionProjects     Movies
Desktop           Music
Documents         Pictures
Downloads         Public
Dropbox
anoop@MBA ~ %
```

```
anoop — -zsh — 102x6
anoop@MBA ~ % cat /etc/services | grep nntp | grep tcp | cut -w -f2 | cut -d "/" -f1 > portNumber
```




Extending Your Command Tools

- Any C-program can be a command
- Unix Philosophy: [[Eric Raymond's 17 Unix Rules](#)]
 - Do one thing and do it well
 - Work well with others
 - Handle text streams
- Stick to conventions
 - Read only from stdin
 - Output only to stdout
 - Errors only to stderr
 - Use CLA to alter behavior
 - Use exit status properly





A Shell Script

- A sequence of commands in a file
- The 'shell' will read and execute them one-by-one
 - Compiler vs. Interpreter
- The script:
 - The shell specification
 - The script, comments
 - File permission
 - File name
- The scripting language
 - Programming constructs

```
01_Shell-Scripting — vi hello.sh — 45x12
#!/bin/bash

echo "Hello World!"    # Just print Hello

echo "The current directory is: "
pwd                    #print the present working directory
~
~
~
~
"hello.sh" 7L, 131B
```




Learning a Language

- Input-Output
- Variables, Data Structures
- Operators
- Conditional Statements
- Loops
- Functions
- Classes

Bash Scripting

- Simple, strict syntax
- Large number of functions

Additional Topics

- Command-line arguments



Printing

```
#!/bin/bash
```

```
echo "Hello World"
```

```
#!/bin/bash
```

```
echo "Hello World!"    # Just print Hello
```

```
echo "The current directory is: "
```

```
pwd    #print the present working directory
```




Variables, Input, Operators

```
#!/bin/bash
```

```
STRING="Hello World"  
echo $STRING
```

```
#!/bin/bash
```

```
bkf=Backup_$(date +%Y%m%d).tar.gz  
tar -czf $bkf ~/Documents
```

- declare
- let
- factor

```
#!/bin/bash
```

```
echo "Hi, What is your name? !"  
read name  
echo "Hello $name, Good to hear from you!!"
```

```
#!/bin/bash
```

```
read -p "Input two numbers: " num1 num2  
res=$((num1+num2)) # arithmetic expansion  
echo "The sum is: $res"
```

Operators [let, declare, \$(())]

- ++X, X++, --X, X--
- +, -, *, /
- %, **, ^
- &&, ||, !
- &, |, ^, ~
- <=, <, >=, >, ==, !=
- =



Conditional Statements

```
#!/bin/bash
num=30
if [ $num -eq 30 ]
then
    echo "num is correct"
else
    echo "num is incorrect"
fi
```

- Conditional Operators
 - eq, -ne, -gt, -lt, -ge, le
 - f myFile: if myFile exists
 - d myDir: if myDir exists
 - see [man test](#) for more.



Loops, Arrays

```
#!/bin/bash
#Declare an array with 5 elements
ARRAY=( Red Green Blue Yellow Orange )

ELEMENTS=${#ARRAY[@]} #number of elements in array

# loop over each element in array
for (( i=0;i<$ELEMENTS;i++)); do
    echo ${ARRAY[$i]}
done
```

Other Loops

```
for var in list; do
    commands
done
```

```
while [ ]; do
    commands
done
```

```
Until [ ]; do
    commands
done
```




Command Line Arguments

```
#!/bin/bash
```

```
# passing values as command line arguments
```

```
echo '$1 $2 $3': $1 $2 $3
```

```
args=("$@")
```

```
echo 'Printing as Array: ' ${args[0]} ${args[1]} ${args[2]}
```

```
#Use $@ to print out all arguments at once
```

```
echo 'Printing $@: ' $@
```

```
# $# gives the number of arguments
```

```
echo Number of arguments passed: $#
```




Extending your Scripting Toolset

- Text/table processing
 - **cut**: split a string
 - **sed**: stream editor
 - **awk**: table processing
 - **grep/egrep**: pattern matching
 - **sort**: what it says
 - **wc**: word count
 - **tee**: fork the output
 - **xargs**: feed input as arguments
- **time**: time to execute a command
- **date**: current time, date, day, year
- File operations
 - **touch** : create/update a file
 - **diff**: changes between files
 - **head/tail**: lines from a file
 - **cat/tac**: concatenate files
 - **split**: split a file
 - **which/locate/find**: find a file
- **du/df**: Disk operations
- **bc**: command line calculator
- **trap**: capture interrupts
- **select, case, printf**



Questions?