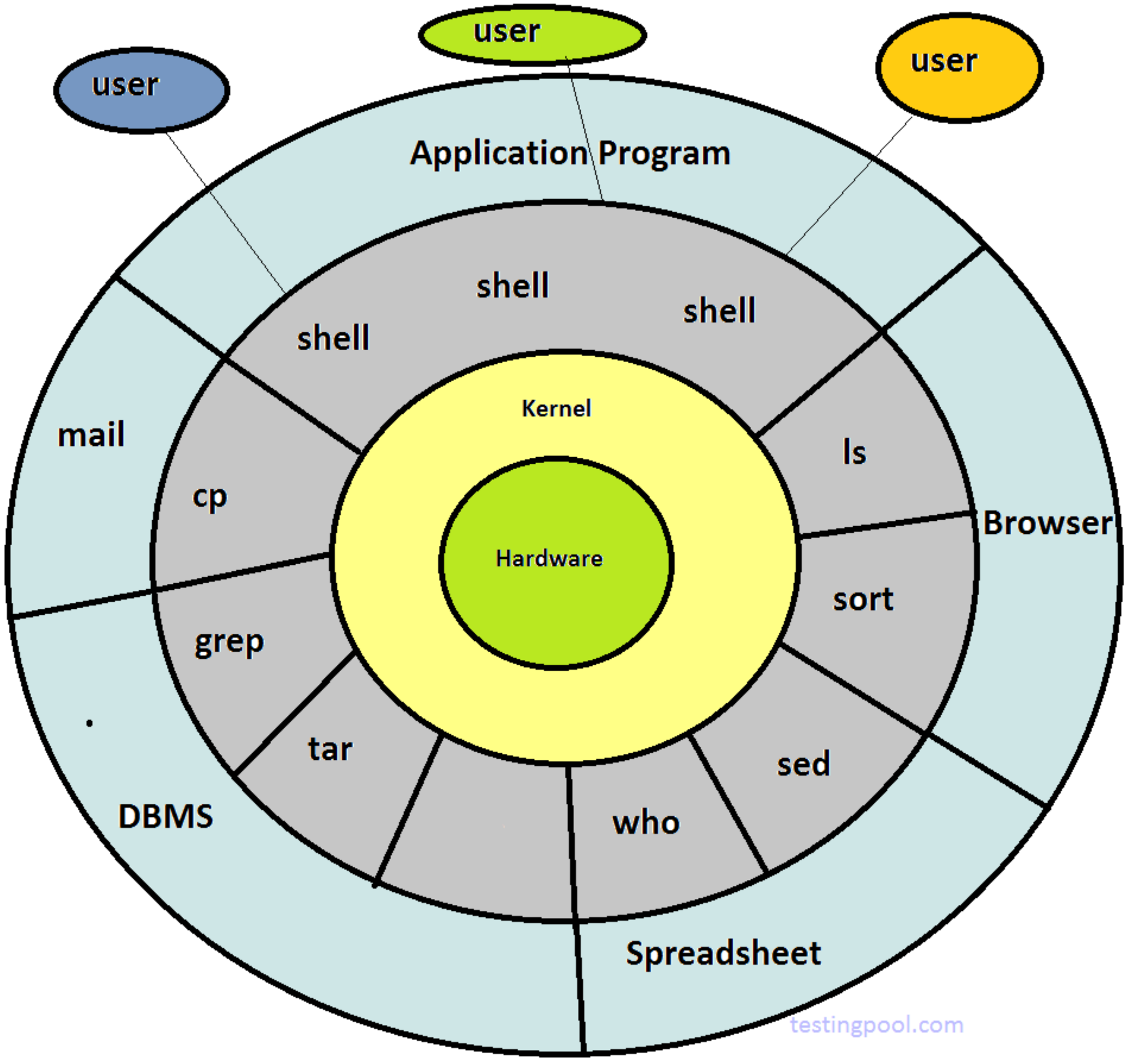
CS5354
UNIX TOOL PROGRAMMING

Shell Programming

- A. Sawarkar

Unix Architecture



Shell Programming

- Due to Shell programming many tasks that can be quickly automated

- A command is work as tool

e.g “sort” task of sorting a coomand

- A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter

- Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Shell Programming

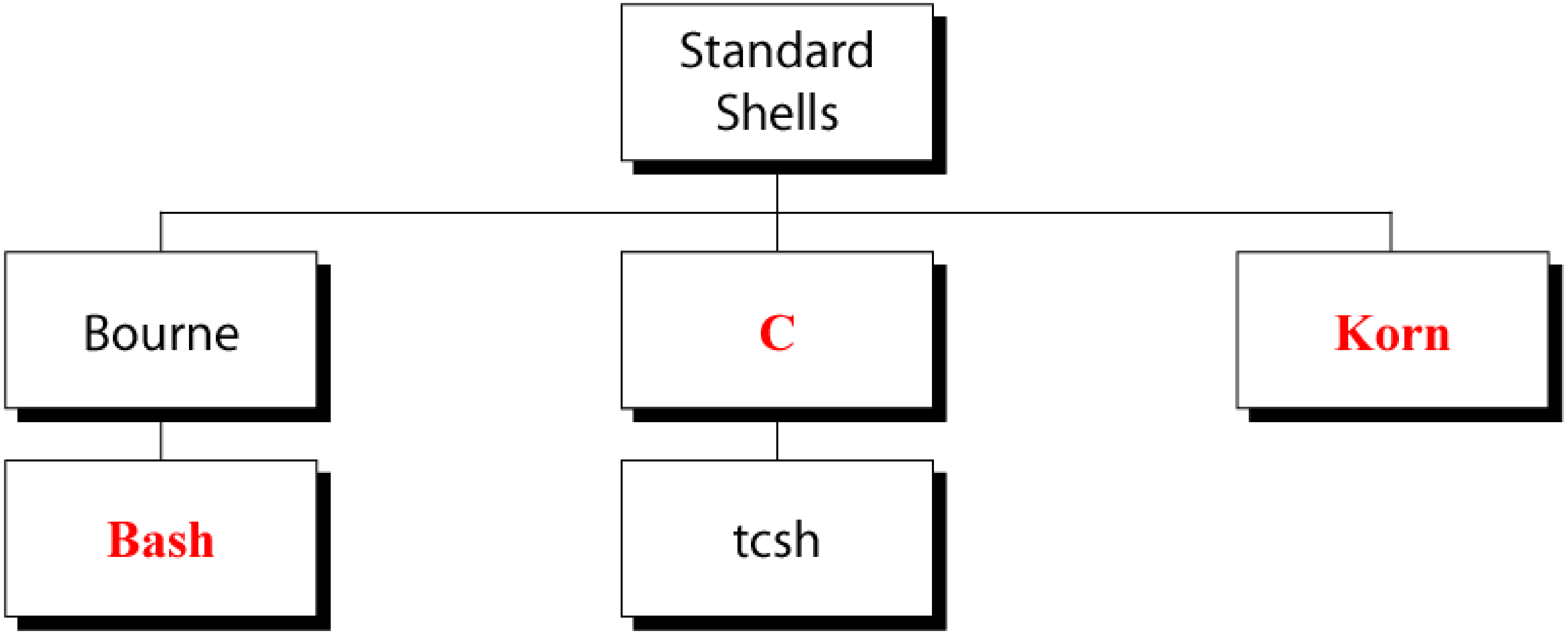
- A shell program (sometimes called a shell script) is a text file that contains standard UNIX and shell commands.

- Each line in a shell program contains a single UNIX command exactly as if you had typed them in yourself.

- The difference is that you can execute all the commands in a shell program simply by running the shell program

- Shell programs are interpreted and **not compiled** programs

UNIX COMMAND INTERPRETERS
Different types of Shell



UNIX COMMAND INTERPRETERS
Different types of Shell

- **Bourne shell (sh)**

  Original UNIX shell written by Steve Bourne of Bell Labs,which is available on all UNIX systems.

  Does not have the interactive facilities provided by modern shells. (e.g. C shell and Korn shell)

  Permits shell scripts to be written and executed.

- **C shell (csh)**

  Provides a C-like language with which to write shell scripts.

- **TC shell (tcsh)**

  Available in the public domain.

  Provides all the features of the C shell together with emacs style editing of the command line.

UNIX COMMAND INTERPRETERS
Different types of Shell

- **Korn shell (ksh)**

  Shell written by David Korn of Bell labs, which is now provided as the standard shell on UNIX systems.

  It provides all the features of the C and TC shells together with a shell programming language similar to that of the original Bourne shell.

  It is the most efficient shell.

- **Bourne Again SHell (bash)**

  This is a public domain shell written by the Free Software Foundation under their GNU initiative.

  Widely used within the academic community.

  Provides all the interactive features of the C shell (csh) and the Korn shell (ksh).

  Its programming language for shell scripts is compatible with the Bourne shell (sh).

.

INTRODUCTION TO SHELL PROGRAMMING

- Shell programming is one of the most powerful features on any UNIX system
- If you cannot find an existing utility to accomplish a task, you can build one using a shell script

SHELL PROGRAM STRUCTURE

- A shell program contains high-level programming language features:
  - Variables for storing data
  - Decision-making control (e.g. if and case statements)
  - Looping abilities (e.g. for and while loops)
  - Function calls for modularity
- A shell program can also contain:
  - UNIX commands
  - Pattern editing utilities (e.g. grep, sed, awk)

YOUR SHELL PROGRAMMING LIBRARY

- Naming of shell programs and their output
  - Give a meaningful name
  - Program name example: findfile.csh
  - Do not use: script1, script2
  - Do not use UNIX command names
- Archive for shell programs
  - If you develop numerous shell programs, place them in a directory (e.g. bin or shellprogs)
  - Update your path to include the directory name where your shell programs are located

STEPS TO CREATE SHELL PROGRAMS

- Specify shell to execute program
  - Script must begin with #! (pronounced “shebang”)
    to identify shell to be executed

Examples:

```
#! /bin/sh                          (defaults to bash)
#! /bin/bash
#! /bin/csh
#! /usr/bin/tcsh
```

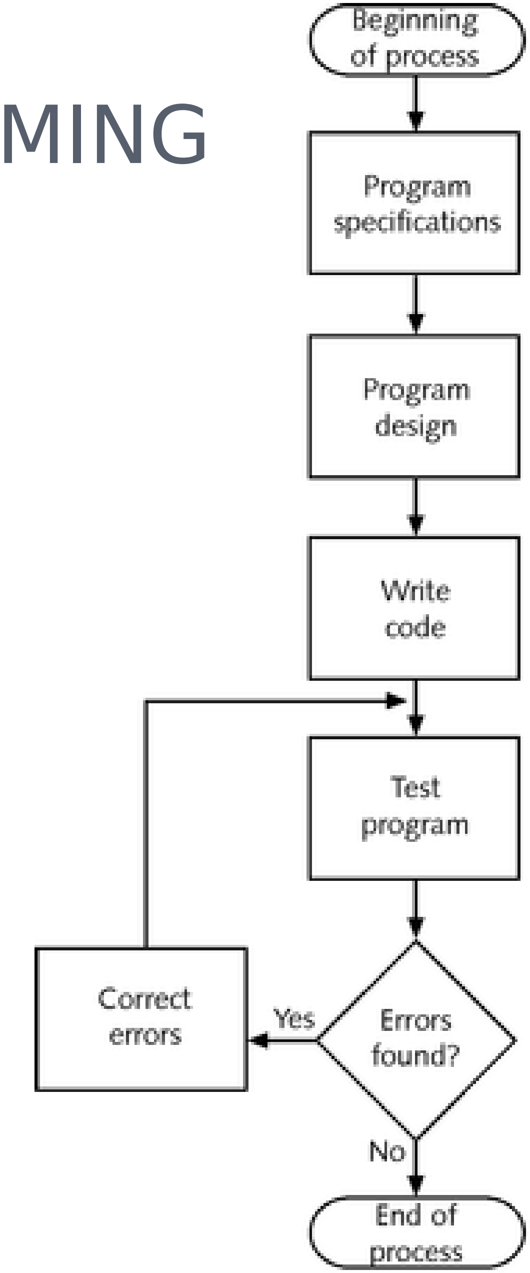
- Make the shell program executable
  - Use the “chmod” command to make the program/script file executable

FORMATTING SHELL PROGRAMS

- Formatting of shell programs
  - Indent areas (3 or 4 spaces) of programs to indicate that commands are part of a group
  - To break up long lines, place a \ at the end of one line and continue the command on the next line
- Comments
  - Start comment lines with a pound sign (#)
  - Include comments to describe sections of your program
  - Help you understand your program when you look at it later

STEPS OF PROGRAMMING

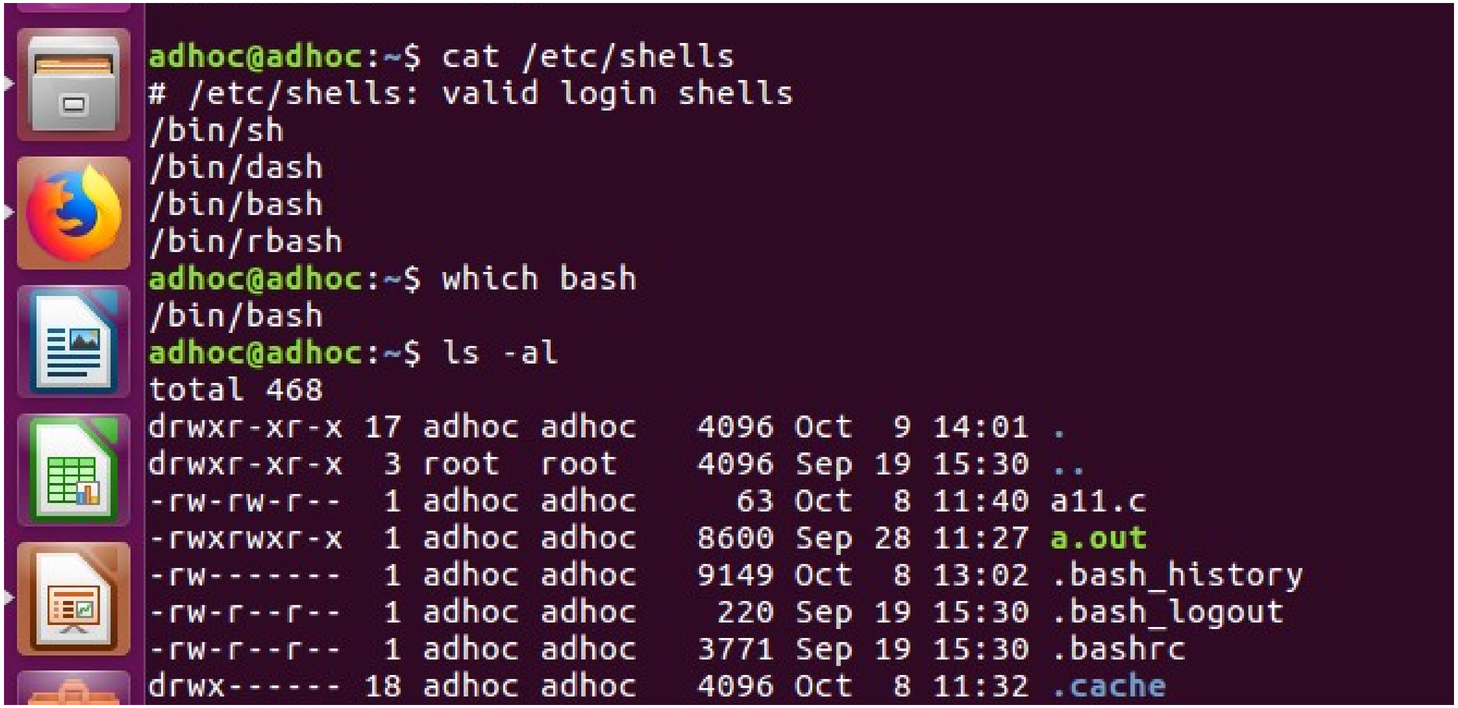
- Guidelines:
  - use good names for
    - script
    - variables
  - use comments
    - lines start with #
  - use indentation to reflect logic and nesting



Command or Shell sript

- **Cat /etc/shells:->** show all avialble shells which your system can suuport
- **Which bash:->** where bash is located
- **touch name.sh:->** creat a shell script
- **ls -al:->** show permission for you,group,user

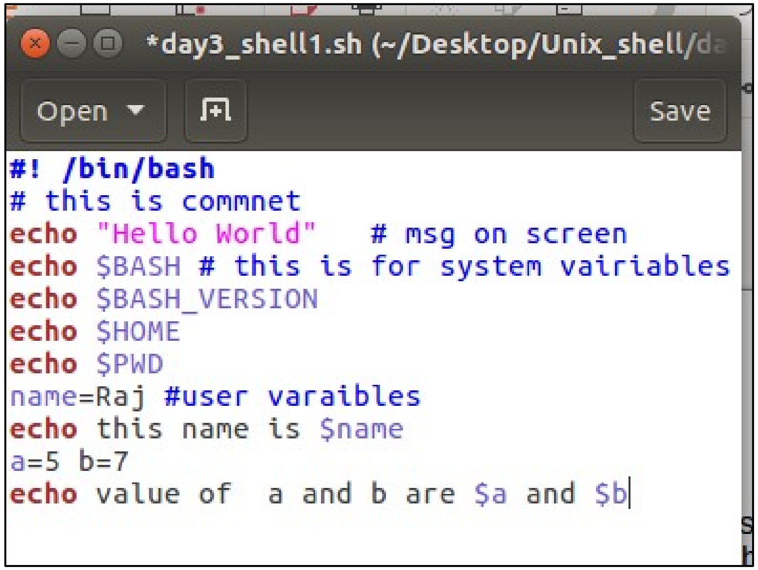
e.g rwx (read, write, execute)



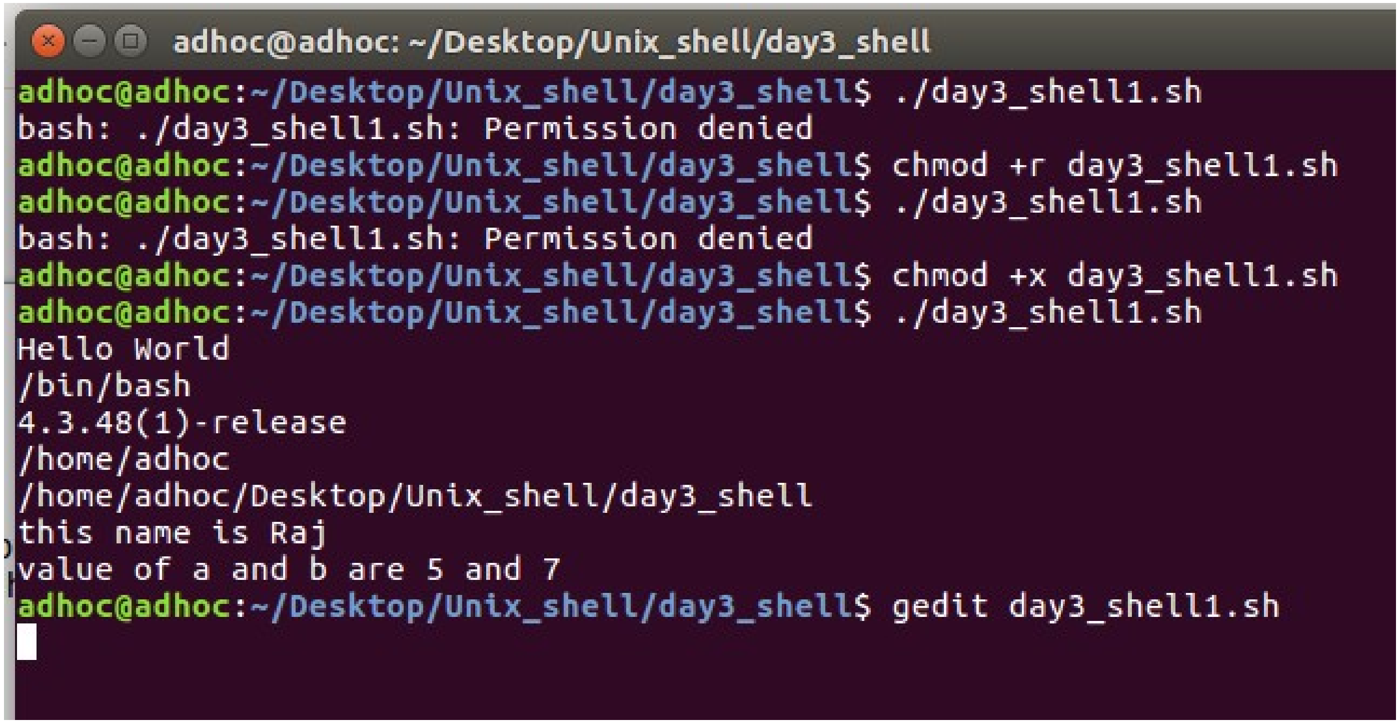
```
adhoc@adhoc:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
adhoc@adhoc:~$ which bash
/bin/bash
adhoc@adhoc:~$ ls -al
total 468
drwxr-xr-x 17 adhoc adhoc   4096 Oct  9 14:01 .
drwxr-xr-x  3 root  root    4096 Sep 19 15:30 ..
-rw-rw-r--  1 adhoc adhoc     63 Oct  8 11:40 a11.c
-rwxrwxr-x  1 adhoc adhoc   8600 Sep 28 11:27 a.out
-rw-------  1 adhoc adhoc   9149 Oct  8 13:02 .bash_history
-rw-r--r--  1 adhoc adhoc    220 Sep 19 15:30 .bash_logout
-rw-r--r--  1 adhoc adhoc   3771 Sep 19 15:30 .bashrc
drwx------ 18 adhoc adhoc   4096 Oct  8 11:32 .cache
```

Shell sript
(Variables- System/User)

- #!/bin/bash
- echo “messge as hello world”
- Permission :-> chmod +x hello.sh



```
#! /bin/bash
# this is comnnet
echo "Hello World"    # msg on screen
echo $BASH # this is for system vairiables
echo $BASH_VERSION
echo $HOME
echo $PWD
name=Raj #user varaibles
echo this name is $name
a=5 b=7
echo value of  a and b are $a and $b
```



```
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ ./day3_shell1.sh
bash: ./day3_shell1.sh: Permission denied
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ chmod +r day3_shell1.sh
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ ./day3_shell1.sh
bash: ./day3_shell1.sh: Permission denied
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ chmod +x day3_shell1.sh
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ ./day3_shell1.sh
Hello World
/bin/bash
4.3.48(1)-release
/home/adhoc
/home/adhoc/Desktop/Unix_shell/day3_shell
this name is Raj
value of a and b are 5 and 7
adhoc@adhoc:~/Desktop/Unix_shell/day3_shell$ gedit day3_shell1.sh
```

EXAMPLE: “HELLO” SCRIPT

```
#! /bin/csh
echo "Hello $USER"
echo "This machine is `uname -n`"
echo "The calendar for this month is:"
cal
echo "You are running these processes:"
ps
```

EXAMPLE SCRIPT OUTPUT

```
% chmod u+x hello
% ./hello
Hello ege!
This machine is turing
The calendar for this month is
   February 2008
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
You are running these processes:
   PID TTY      TIME CMD
 24861 pts/18   0:00 hello.csh
 24430 pts/18   0:00 csh
```