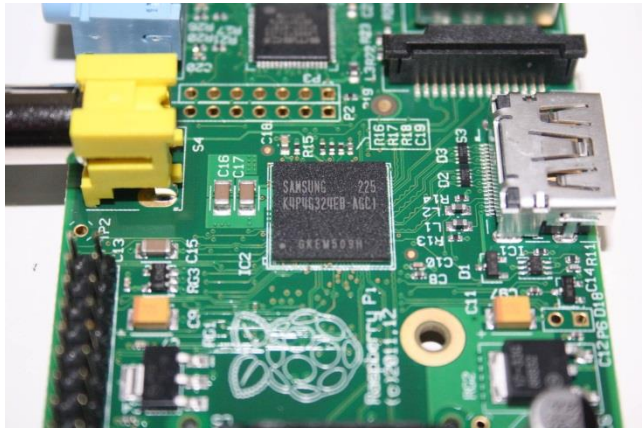
RaspberryPi

Raspberry pi interfacing

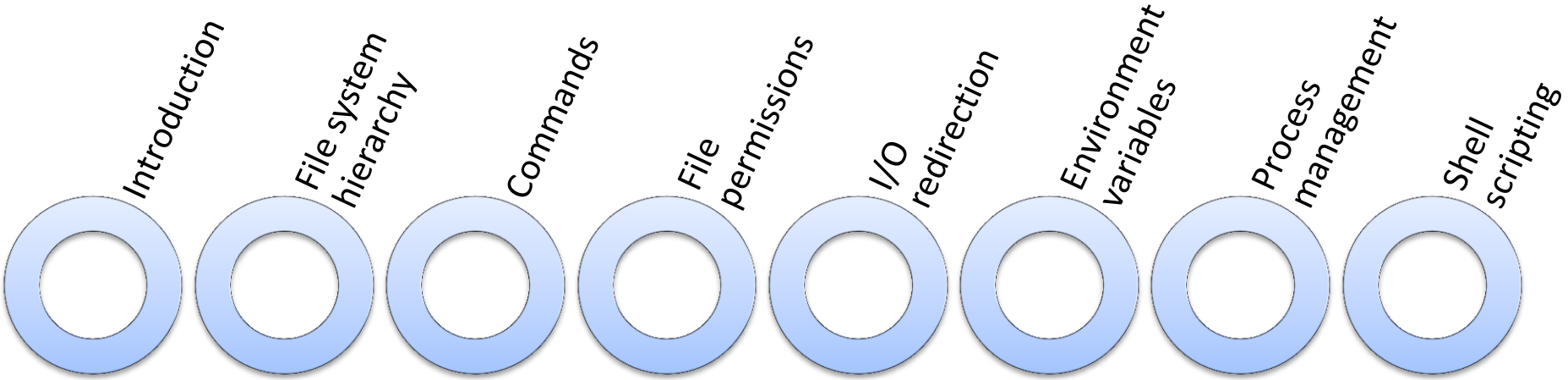
Lecture 2: Linux OS



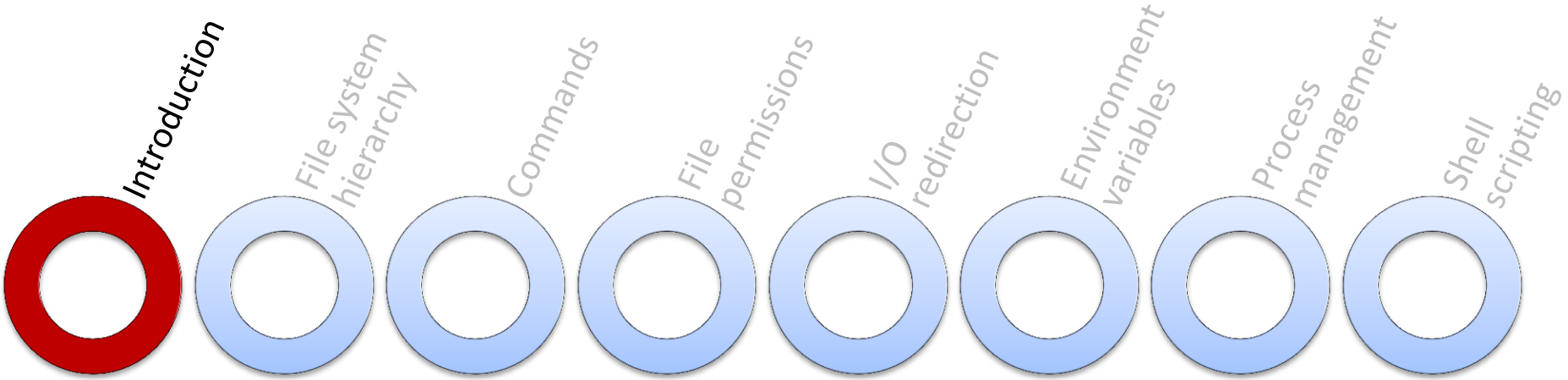
Linux OS Agenda

- Introduction to Linux OS.
- Linux file system hierarchy.
- Linux commands.
- Files permissions.
- Input-Output redirection.
- Environment variables.
- Process management.
- Linux shell scripting.

Linux OS Agenda



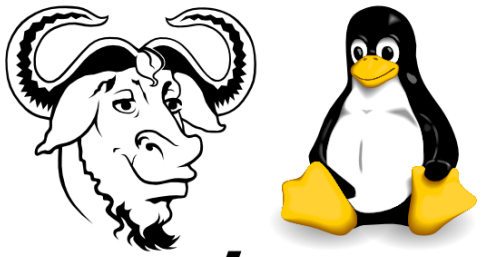
Introduction to Linux operating system



Introduction to Linux operating system

What is Linux operating system ?

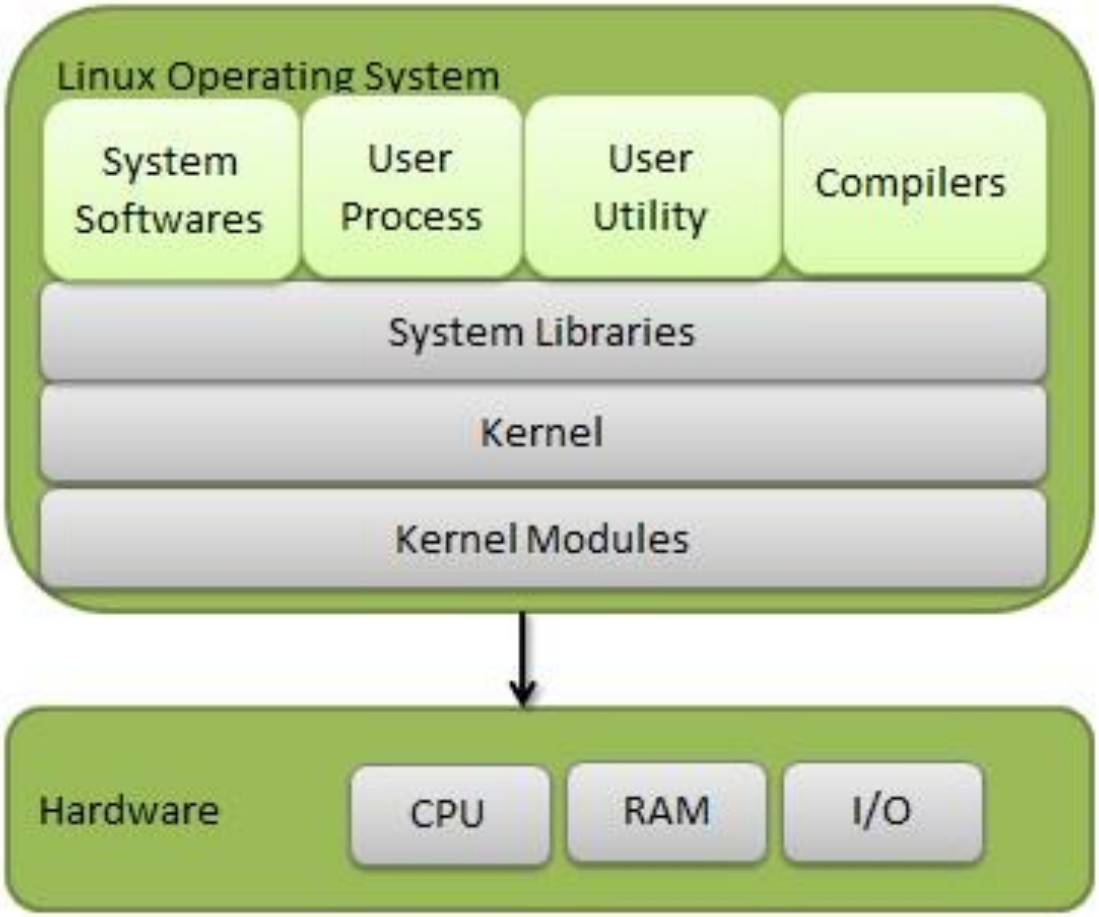
- In 1983, Richard Stallman, founder of the Free Software Foundation, set forth plans of a complete Unix-like operating system, called GNU, composed entirely of free software.
- By 1991 the lower level (kernel, device drivers, system-level utilities and daemons) was still mostly lacking.
- In 1991, Linus Torvalds released the first version of the Linux kernel. Early Linux developers ported GNU code, including the GNU C Compiler, to the kernel. The free software community adopted the use of the Linux kernel as the missing kernel for the GNU operating system.



GNU/Linux

Introduction to Linux operating system

- **Linux OS main components**



Introduction to Linux operating system

Linux features

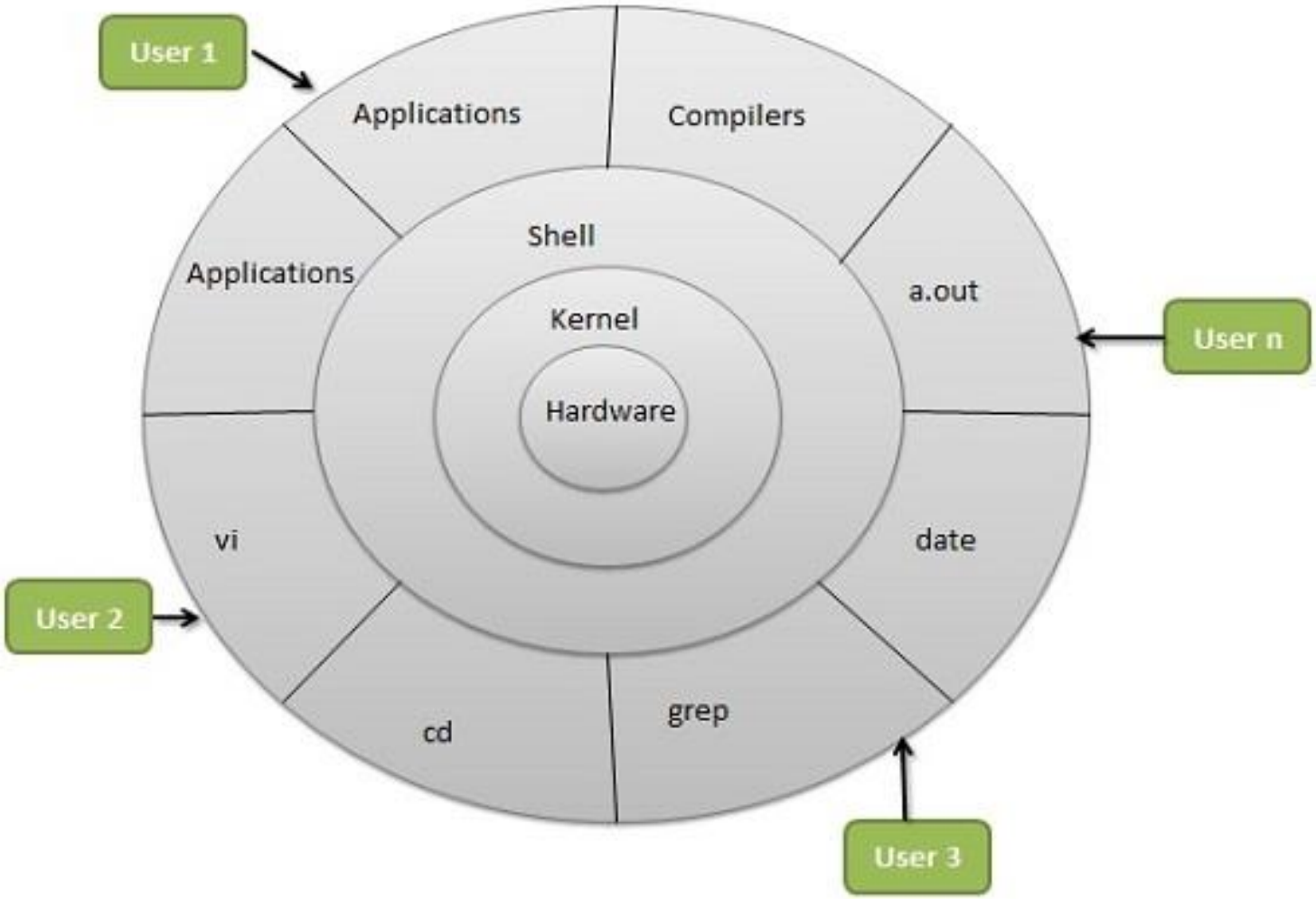
- **Open Source:** Linux source code is freely available and it is community based development project. Multiple teams works in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User:** Linux is a multiuser system means multiple users can access system resources like memory, ram, application programs at same time.
- **Multiprogramming:** Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System:** Linux provides a standard file structure in which system files, user files are arranged.

Introduction to Linux operating system

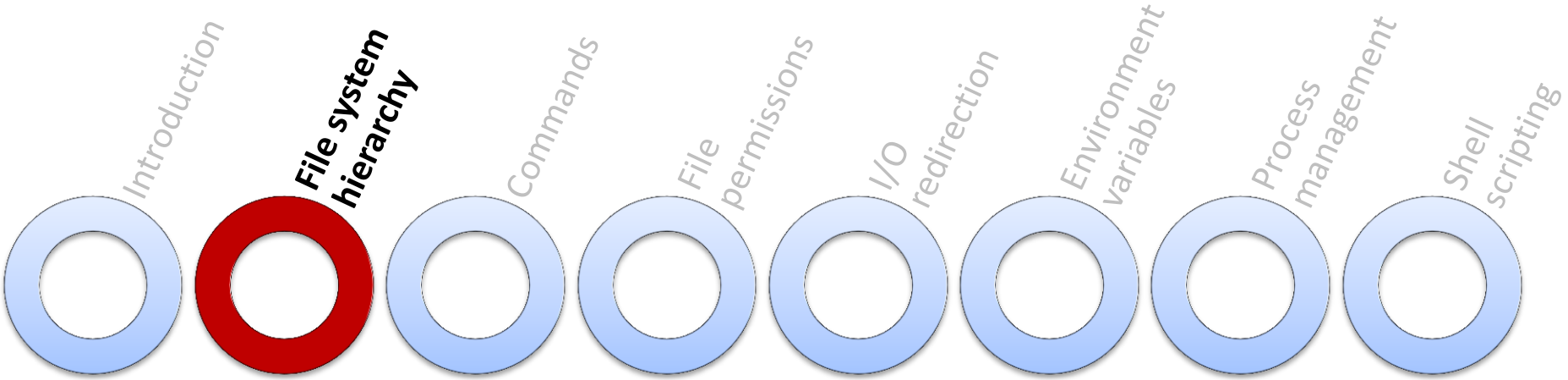
Linux features

- **Shell:** Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.
- **Security:** Linux provides user security using authentication features like password protection, controlled access to specific files, encryption of data.

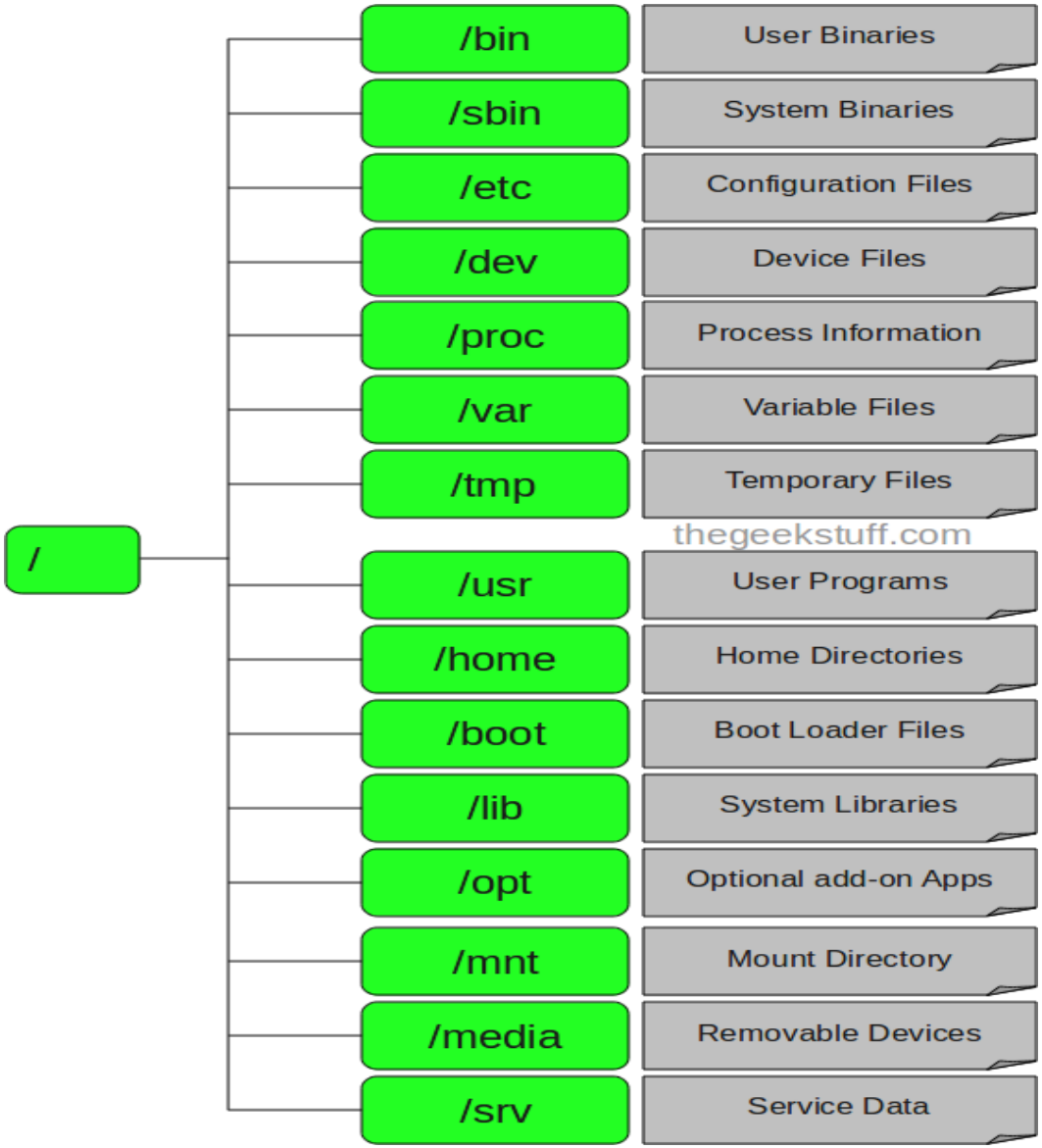
Introduction to Linux operating system



Linux file system hierarchy



Linux file system hierarchy



Linux file system hierarchy

- **/etc**
  - Contains systems configurations files.
- **/usr/bin**
  - Contains commands for all users.
- **/usr/sbin**
  - Contains commands for root user.
- **/dev**
  - Contains hardware devices tree.
- **/home/username**
  - Home directory for user of name “username”.

Linux file system hierarchy

- **/etc/passwd**
  - Contains one line for each user account, with seven fields delimited by colons (":"). These fields are:

  [login name : encrypted password : user ID : group ID : comment : home dir : default shell]

- **/etc/shadow**
  - Contains the password information for the system's accounts and optional aging information.
  - Each line contains 9 fields, separated by colons (":"), in the following order:

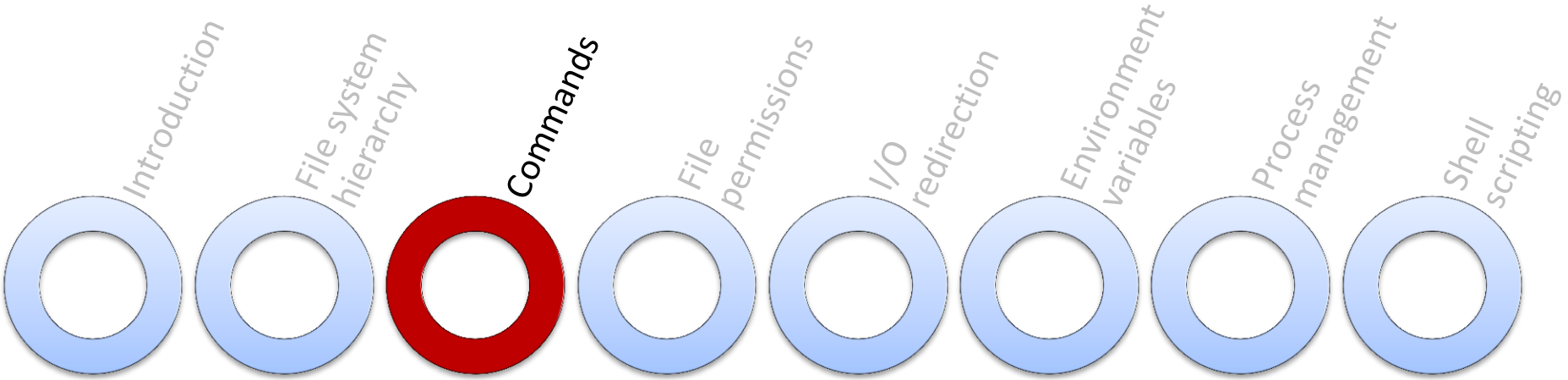
  [login name : encrypted password : last change : min pass age : max pass age : pass warning period : pass inactivity period : account expiration date : reserved field]

Linux file system hierarchy

- **/etc/group**
  - Contains the groups on the system with one line per group..
  - Each line contains 4 fields, separated by colons (“:”), in the following order:

  [group name : password : group ID : members usernames separated by commas]

Linux commands



Linux commands

- **Command format**

Command option argument

Command: required command to execute.

Option: options given to command.

Argument: which the command is executed on.

Linux commands

- **Commands examples**

  $ ls /etc : list directory content

  $ ls –l /etc : long list option

  $ ls –t /etc : sort with time

  $ ls –l –t /etc : long list and sort with time

  $ cal : calendar of current month

  $ cal 2014 : calendar of 2014

  $ cal 5 2014 : calendar of 5/2014

  $ alias display="ls –l" : create an alias

  $ unalias display : delete an alias

Linux commands

- **Commands examples**

  $ man command : get manual for any command

  $ man –k : search in manual by keyword

  $ man –s1 : search in section 1 (s1 for user commands, s4 for file formats)

  $ pwd : print working directory

  $ cd /etc : change directory to /etc

  $ cd .. : change directory to parent directory

  $ cd : change directory to home directory

  $ cd ~ : same as previous

Linux commands

- **Commands examples**

  $ mkdir dirname : make new directory with name “dirname”

  $ mkdir /dir1/dir2 : make new directory and create any parent directory

  $ rmdir : remove empty directory

  $ touch : create new empty file

  $ rm : remove file

  $ rm –r : remove folder with all its content

  $ find startPoint criteria itemToSearchFor

  $ grep pattern files : search for a pattern in files

  $ echo Text : print Text to screen

Linux commands

- **Commands examples**

  $ cat : display file content

  $ more : display file content according to screen size

  $ less : same as more, and allows backward movement

  $ head -3 : display first 3 lines of a file

  $ head : display by default 10 lines

  $ tail -2 : display 2 lines from the file end

  $ sort –k1 –t: -r /etc/passwd : sort file according to field 1, with field separator of : in descending order according to ASCII

  $ sort –nk3 : sort in numeric order

  $ sort –o output file : save result in same file

Linux commands

- **Commands examples**

  $ wc : display count of lines, words and characters of a file

  $ wc –l : display number of lines

  $ wc –w : display number of words

  $ wc –c : display number of characters

  $ cp source destination : copy file from source to destination

  $ cp –r : copy directory

  $ mv : move file or directory

  $ mv -i: ask for overwrite if exists

Linux commands

- **Commands examples**

  $ date : display current day date

  $ who : display logged in users

  $ df : display all disk partitions with size in blocks of 512 bytes

  $ df – k : size in Kbytes

  $ df –h : size in Gbytes

  $ cut –f1,7 –d: /etc/passwd : display fields 1 and 7 with separator : in file /etc/passwd

  $ cut –f3-5 –d: /etc/passwd : from field 3 to field 5

  $ cut –f3- –d: /etc/passwd : from field 3 to the end

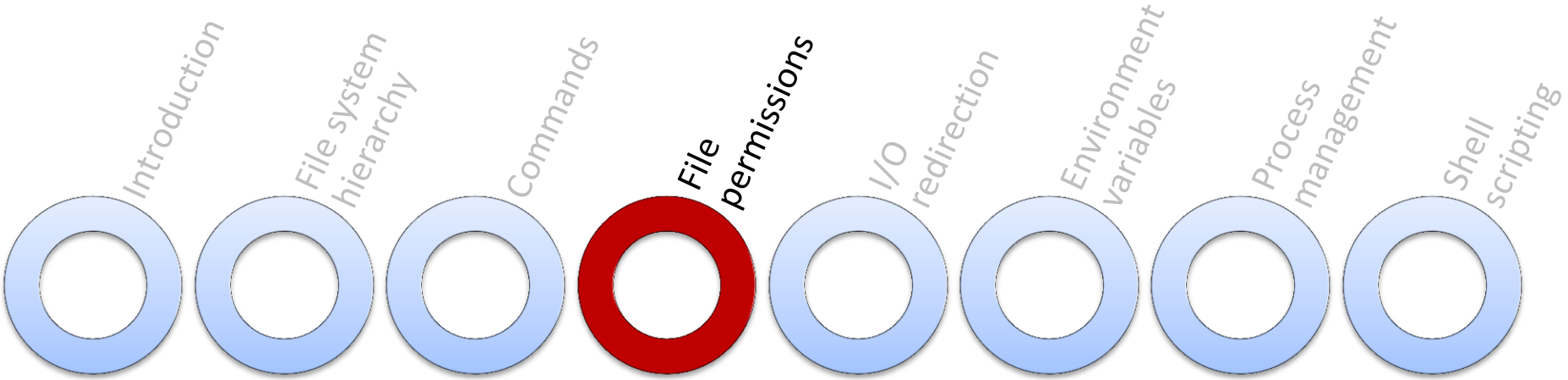
  $ cut –f-3 –d: /etc/passwd : from first field to field 3

File name expansion

- **~ :** home directory
- *** :** any combination (alpha-num)
- **? :** one character (alpha-num)
- **[ab3sc] :** one character of this set
- **[a-e0-4] :** one character in this ranges

- **$ ls *.[abc0-5] :** list directory content which names contains any set of characters followed by ‘dot .’ then followed by one character of (a, b, c, 0, 1, 2, 3, 4, 5)

Files permissions



Files permissions

$ ls –l /etc/passwd

-rw-r- -r-- 1 root root 988 ….. ….

-rw-r- -r-- : file type, owner permissions, group permissions, other permissions

1 : number of hard links

Root : owner

Roor : group

988 : size

Creation time and file name

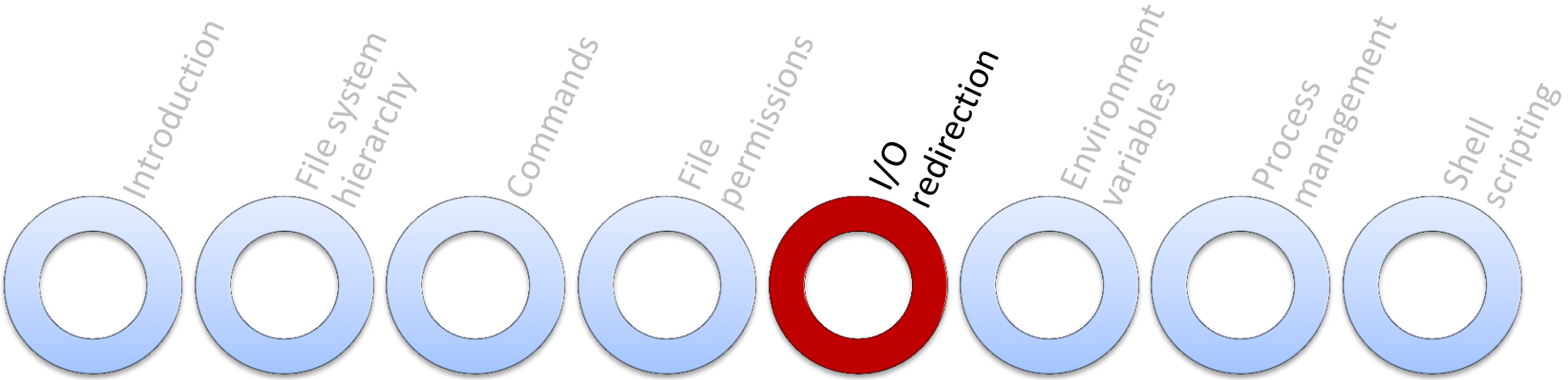
$ ls –d /etc/ : list directory permissions

Files permissions

Permission	On file	On directory
R	Read file content	List files
W	Modify file content	Modify directory from outside
X	Execute file	Change to directory or access any file inside it

$ chmod u+x : add execute permission to owner
$ chmod u=rx,g=-,o=x : make owner read execute, group none, other execute only
$ chmod 7 file1 : change permissions to be 007
$ umask 777 : set umask to be 111 to set default permissions to be complement of umask
$ umask : display current default permissions

Input-Output redirection



Input-Output redirection

$ ls –l /etc/passwd 1> file1

Redirect output list to file “file1”

1 > output redirection, overwrite

1 >> append output

2 > error redirection

$ Cat < file : input redirection

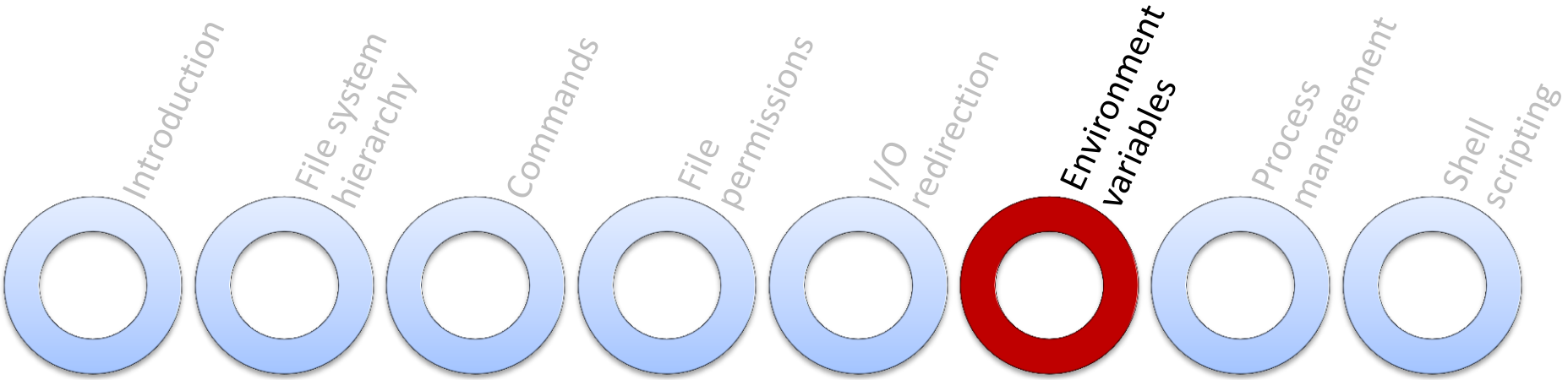
Pipelining

$ command1 | command 2

Pipeline command 1 output to be as input to command 2

$ cut –d: -f1,3 /etc/passwd | sort –t: -nk2

Environment variables



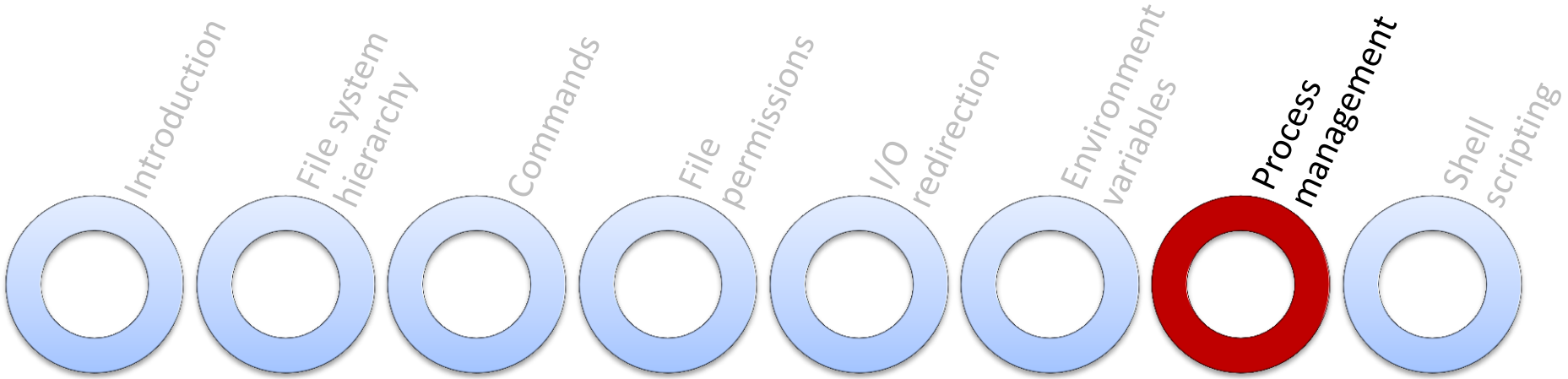
Environment variables

- **Built in variables:**
  - **$ echo $PWD :** display value of $PWD that saves current path
  - **$HOME :** save home directory
  - **$LOGNAME :** save login name
  - **$PS1 :** shell command prompt
  - **$PS2 :** appears when command didn't end yet
  - **$PS3 :** appears when user is asked to enter a choice
  - **$PATH :** contains all paths that shell uses to search for commands
  - **$$:** process ID (PID) of current shell

Environment variables

- **User defined variables:**
  - **$ x=4 :** create variable x and put 4 inside it
  - **$ y=abc :** create variable y and put “abc” inside it
  - **$ z=x :** create variable z and put ‘x’ inside it, not value of x
  - **$ z=$x :** create variable z and put value of x inside it
  - **$ set :** display all declared variables (Built-in and user defined)
  - **$ export x :** export variable x to child process

Process management



Process management

- **CTRL+Z :** pause current process in the background
- **$ bg :** continue paused process, continue will be in the background
- **$ fg :** continue paused process, continue will be in the foreground
- **$ jobs :** display all processes that exist in the background
- **$ ls –R / & :** start the process in the background

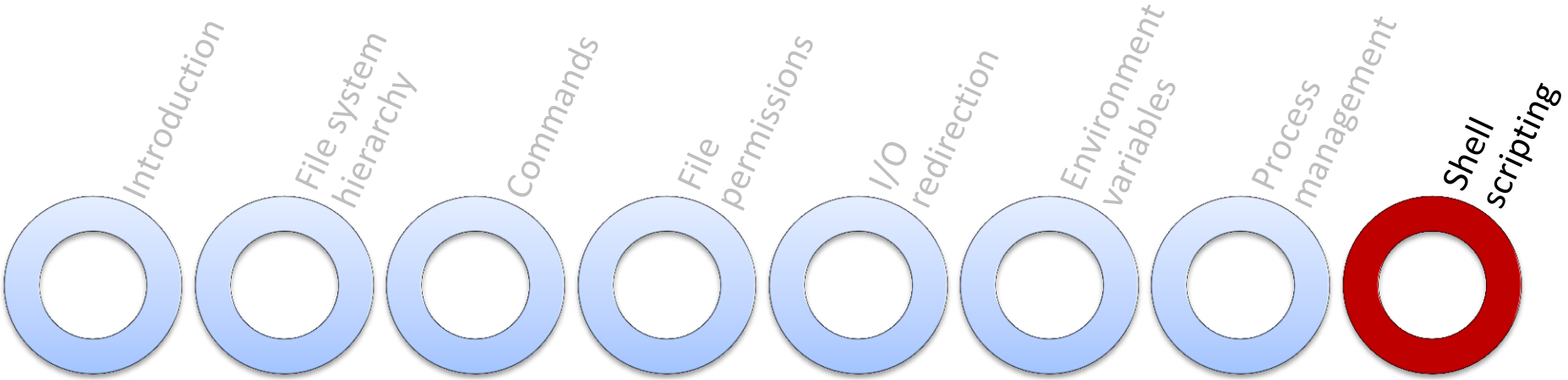
Process management

- **ps :** display processes status in current shell
- **ps –e :** display processes status in all the system
- **Ps –u name :** display processes status for specific user name
- **pstree :** display process tree
- **Pgrep text :** display PID of any process contains “text” in its name
- **Pgrep -l text :** display PID and name of any process contains “text” in its name

Process management

- **Process signals**
  - **$ kill -l :** display all available signals
  - **$ man –s3head signal :** display manual for signals
  - **$ kill -2 pid1 pid2 :** send signal 2 to processes with IDs pid1 and pid2
  - **$ kill pid :** send default signal which is signal 15 (terminate)
  - **$ pkill pname :** send signal to any process contains “pname” in its name
  - **Default action for some signals:**
    - **Signal 9:** kill immediately
    - **Signal 15:** terminate (not mandatory to end process)
    - **Signal 2:** interrupt

Linux shell scripting



Linux shell scripting

Shell scripting

- Shell scripting is an interpreted language executed line by line during runtime so it is slower than other languages like C for example.
- To execute a shell script, it should have rw (read-write) permissions.
- To add rw permissions to script, after creating the script:

```
$ chmod u+rx scriptname
```

Linux shell scripting

- **Arrays**
  - Arr[5]=hello : create array called Arr, and store at index 5 new element and make its value equals “hello”
  - Echo ${Arr[*]} : will print > hello
  - Arr[3]=23 : store at index 3 new element and make its value equals 23
  - Arr[100]=abc : store at index 100 new element and make its value equals “abc”
  - Echo ${Arr[*]} : will print > 23 hello abc
  - Echo ${#Arr[*]} : display number of elements in Arr which is 3 now

Linux shell scripting

- **Command substitution**
  - Y=`ls` : Y will be the result of executing ls command
  - Y=$(ls) : same as previous example
- **Arithmetic operations**
  - Y=5;z=10
  - Echo $(($Y+$z)) : will print 15
  - Echo 5+10 | bc : same as previous example

Linux shell scripting

- **Notes**
  - #!/bin/shellname: written at the beginning of the script to decide which shell will be used to run the script
    - Ex. #!/bin/bash
  - $# : variable contains number of arguments sent to script
  - $* : variable contains values of all arguments sent to script
  - $1 : variable contains value of first argument sent to script
  - $2 : variable contains value of second argument sent to script
  - $0 : variable contains script name

Linux shell scripting

- **sed command**
  - sed command filename: reads line from file, executes command on it, prints line after command execution, repeats for all lines.
  - sed ‘p’ /etc/passwd : will print each line twice of file /etc/passwd
  - sed ‘3p’ /etc/passwd : will print each line once, except line 3 will be printed twice.
  - sed ‘3p;10p’ /etc/passwd : will print each line once, except line 3 and line 10 will be printed twice.
  - sed ‘3,10p’ /etc/passwd : will print each line once, except from line 3 to line 10 will be printed twice.
  - sed –n ‘3,10p’ /etc/passwd : will print only line 3 to 10 one time.

Linux shell scripting

- **sed command**
  - sed –n '/bin/p' /etc/passwd : will print only lines containing bin keyword.
  - sed –n '/^bin/p' /etc/passwd : will print only lines containing bin keyword at the start of the line.

Linux shell scripting

- **Flow control for integers : returns Zero if True**
  - test $x –eq $y : == comparator
  - test $x –nq $y : != comparator
  - test $x –gt $y : > comparator
  - test $x –ge $y : >= comparator
  - test $x –lt $y : < comparator
  - test $x –le $y : <= comparator

Linux shell scripting

- **Flow control for strings: returns Zero if True**
  - test $x = $y : == comparator
  - test $x != $y : != comparator
  - Test $x –eq $y –a $x –eq $z : x==y and x==z

Linux shell scripting

- **If statment**

```
if test $x –eq $y
  then
      echo they are equal
elif test $x –gt $y
   then
      echo x greater than y
else
      echo x less than y
fi
```

Linux shell scripting

- **While loop**

```
while condition
        do
                echo newline
done
```

It will execute (echo newline) as long as the condition is true.

Linux shell scripting

- **While loop**

```
while read x
    do
        echo x
done < /etc/passwd
```

It will read each line in the file and print it.

Linux shell scripting

- **Until loop**

```
until condition
        do
                echo newline
done
```

It will execute (echo newline) as long as the condition is false.

Linux shell scripting

- **Signal handling**

  trap “command” signalnumber

  The “command” will be executed when the “signalnumber” is received.

  If command is leaved empty “ “, signal will be ignored.

  The only signal that can’t be ignored is signal 9 which kills the process.

Linux shell scripting

- **Functions**

```
Function() {
        echo line
}
```

Function take arguments like script not like functions in c language.