Welcome To: Module 5

Linux System Administration

Linux File Editor

- A text editor is a program which enables you to create and manipulate data (text) in a Linux file

- There are several standard text editors available on most Linux systems
  - **vi** - **Visual editor**
  - **ed** - **Standard line editor**
  - **ex** - **Extended line editor**
  - **emacs** - **A full screen editor**
  - **pico** - **Beginner's editor**
  - **vim** - **Advance version of vi**

- Our editor = vi (available in almost every Linux distribution)

Introduction to vi Editor

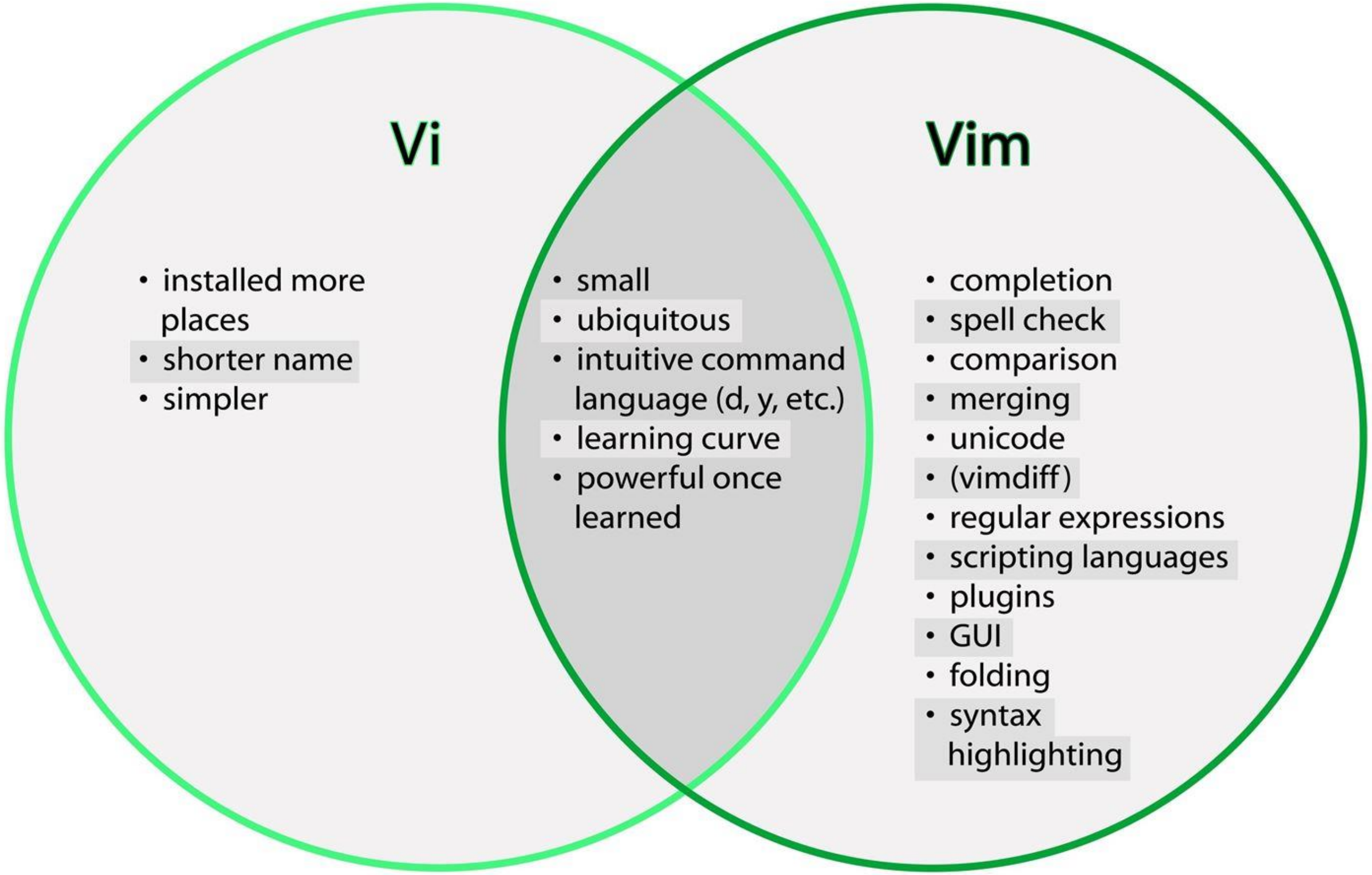
- vi supplies commands for:
  - Inserting and deleting text
  - Replacing text
  - Moving around the file
  - Finding and substituting strings
  - Cutting and pasting text

- Most common keys:
  - `i – insert`
  - `Esc – Escape out of any mode`
  - `r – replace`
  - `d – delete`
  - `:q! – quit without saving`
  - `:wq! – quit and save`

Difference Between vi and vim Editor

- As far as functionality is concerned, both editors work in the same manner. Which editor you choose is a matter of personal choice. Some people recommend learning the vim editor instead of the vi editor. Due to added features, learning and using vim editor is much easier than the vi editor.
- Since vim is based on the vi, when you will learn how to use the vim editor, you will automatically learn how to use the vi editor.
- vim has all the features as vi with some excellent addition
- There's also a comprehensive help system and lots of customization options available.

Difference Between vi and vim Editor



“vim” Interactive Learning Tools

- There are many websites that offer free vim interactive training:
  - https://www.openvim.com/
  - http://www.vimgenius.com
  - https://vim-adventures.com/ (Games)

“sed” Command

- Replace a string in a file with a newstring
- Find and delete a line
- Remove empty lines
- Remove the first or n lines in a file
- To replace tabs with spaces
- Show defined lines from a file
- Substitute within vi editor
- And much more…

User Account Management

Commands

- useradd
- groupadd
- userdel
- groupdel
- usermod

Files

- /etc/passwd
- /etc/group
- /etc/shadow

Example:

```
useradd -g superheros -s /bin/bash -c “user description” -m -d
/home/spiderman spiderman
```

The /etc/login.def File

- The chage command – per user
  - **Example**

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E
expiredate] [-W warndays] user
```

- File = /etc/login.def
  - PASS_MAX_DAYS   99999
  - PASS_MIN_DAYS   0
  - PASS_MIN_LEN   5
  - PASS_WARN_AGE   7

The chage Command

- The chage command – per user
  - **Example**

```
chage [-d lastday] [-m mindays] [-M maxdays] [-W warndays] [-I
inactive] [-E expiredate] user
```

-d = 3. Last password change **(lastchanged)** : Days since Jan 1, 1970 that password was last changed
-m = 4. Minimum : The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
-M = 5. Maximum : The maximum number of days the password is valid (after that user is forced to change his/her password)
-W = 6. Warn : The number of days before password is to expire that user is warned that his/her password must be changed
-I = 7. Inactive : The number of days after password expires that account is disabled
-E = 8. Expire : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used.

Switch Users and sudo Access

Commands

- `su - username`
- `sudo command`
- `visudo`

File

- `/etc/sudoers`

Monitor Users

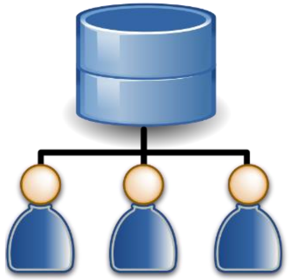
- `who`
- `last`
- `w`
- `finger`
- `id`

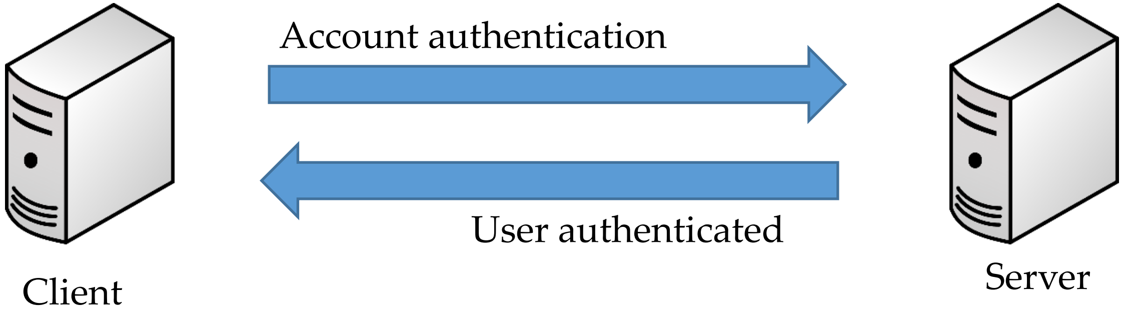
Talking to Users

- `users`
- `wall`
- `write`

Linux Account Authentication

- Types of Accounts
  - Local accounts
  - Domain/Directory accounts





Account authentication

User authenticated

Client

Server

- Windows = Active Directory
- Linux = LDAP?

Difference between Active Directory, LDAP, IDM, WinBIND, OpenLDAP etc.

- Active Directory = Microsoft
- IDM = Identity Manager
- WinBIND = Used in Linux to communicate with Windows (Samba)
- OpenLDAP (open source)
- IBM Directory Server
- JumpCloud
- LDAP = Lightweight Directory Access Protocol

System Utility Commands

- `date`
- `uptime`
- `hostname`
- `uname`
- `which`
- `cal`
- `bc`

Processes and Jobs

- Application = Service
- Script
- Process
- Daemon
- Threads
- Job

Process / Services Commands

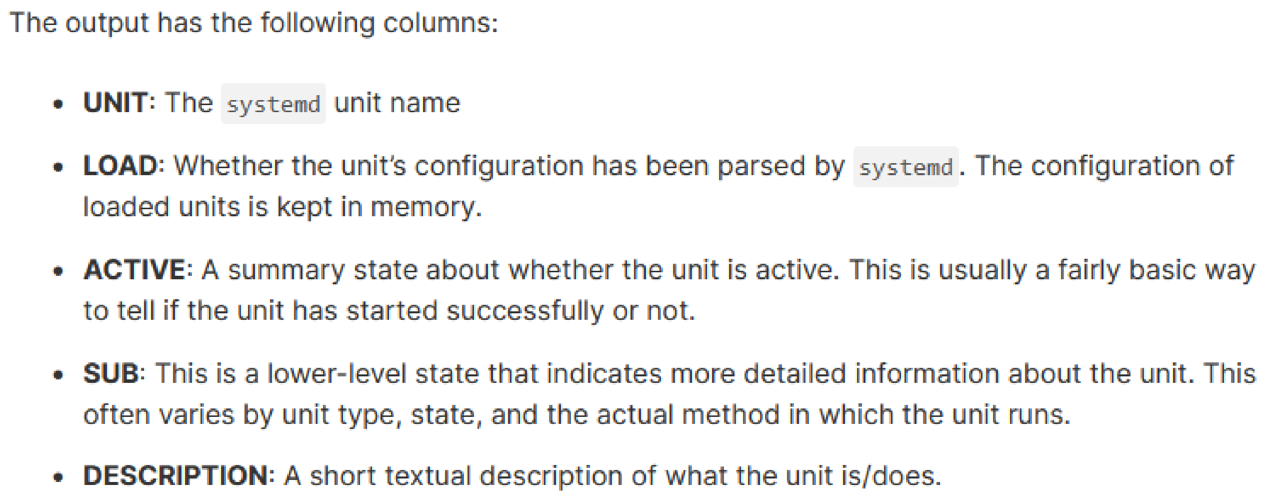
- systemctl or service
- ps
- top
- kill
- crontab
- at.

systemctl command

- systemctl command is a new tool to control system services
- It is available in version 7 and later and it replaces the **service** command

- Usage example:

```
systemctl start|stop|status servicename.service     (firewalld)
systemctl enable servicename.service
systemctl restart|reload servicename.service
systemctl list-units --all
```



The output has the following columns:

- **UNIT**: The `systemd` unit name
- **LOAD**: Whether the unit's configuration has been parsed by `systemd`. The configuration of loaded units is kept in memory.
- **ACTIVE**: A summary state about whether the unit is active. This is usually a fairly basic way to tell if the unit has started successfully or not.
- **SUB**: This is a lower-level state that indicates more detailed information about the unit. This often varies by unit type, state, and the actual method in which the unit runs.
- **DESCRIPTION**: A short textual description of what the unit is/does.

`systemctl` command

- To add a service under systemctl management:

  Create a unit file in `/etc/systemd/system/servicename.service`

- To control system with systemctl

```
systemctl poweroff
systemctl halt
systemctl reboot
```

“**ps**” command

- **ps** command stands for process status and it displays all the currently running processes in the Linux system

Usage examples:

- **ps** = Shows the processes of the current shell

PID = the unique process ID
TTY = terminal type that the user logged-in to
TIME = amount of CPU in minutes and seconds that the process has been running
CMD = name of the command

- **ps -e** = Shows all running processes
- **ps aux** = Shows all running processes in BSD format
- **ps -ef** = Shows all running processes in full format listing *(Most commonly used)*
- **ps -u username** = Shows all processes by username.

By: Imran Afzal
www.utclisolutions.com

“top” command

- top command is used to show the Linux processes and it provides a real-time view of the running system
- This command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel
- When the top command is executed then it goes into interactive mode and you can exit out by hitting **`q`**

- **`Usage: top`**

PID: Shows task’s unique process id
USER: Username of owner of task
PR: The “PR” field shows the scheduling priority of the process from the perspective of the kernel
NI: Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
VIRT: Total virtual memory used by the task
RES: Memory consumed by the process in RAM
SHR: Represents the amount of shared memory used by a task
S: This field shows the process state in the single-letter form
%CPU: Represents the CPU usage
%MEM: Shows the Memory usage of task
TIME+: CPU Time, the same as ‘TIME’, but reflecting more granularity through hundredths of a second.

“top” command

- `top –u iafzal` = shows tasks/processes by user owned
- `top then press c` = shows commands absolute path
- `top then press k` = kill a process by PID within top session
- `top then M and P` = To sort all Linux running processes by Memory usage

Please note:
Top command refreshes the information every 3 seconds

“`kill`” command

- `kill` command is used to terminate processes manually
- It sends a signal which ultimately terminates or kills a particular process or group of processes

Usage:

```
kill [OPTION] [PID]
```

OPTION = Signal name or signal number/ID
PID = Process ID

`kill -l`	=	to get a list of all signal names or signal number

Most used signals are:

`kill PID`	=	Kill a process with default signal
`kill -1`	=	Restart
`kill -2`	=	Interrupt from the keyboard just like Ctrl C
`kill -9`	=	Forcefully kill the process
`kill -15`	=	Kill a process gracefully

“kill” command

- Other similar kill commands are:

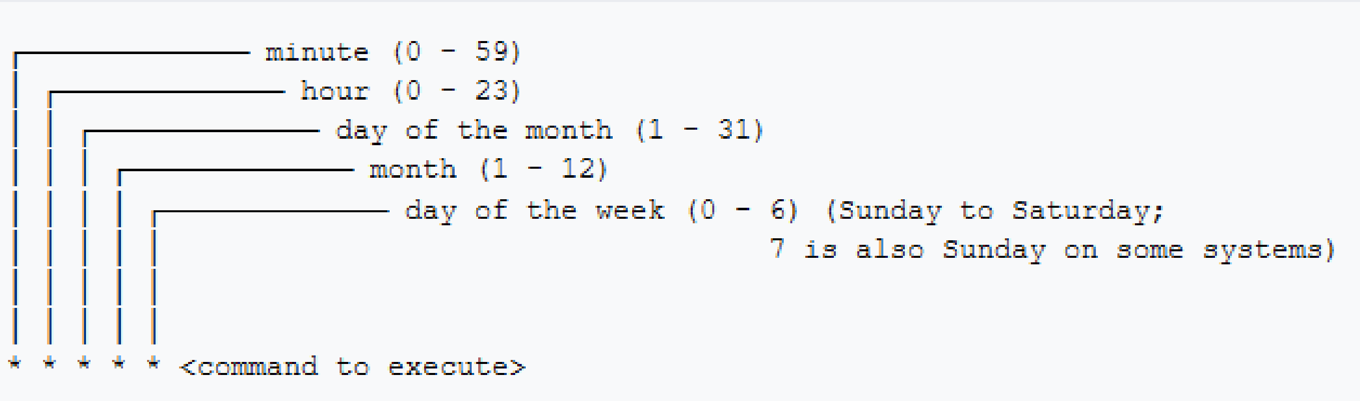
```
killall
pkill
```

“crontab” command

- Crontab command is used to schedule tasks

Usage:

- ○ **crontab -e** = Edit the crontab
- ○ **crontab -l** = List the crontab entries
- ○ **crontab -r** = Remove the crontab
- ○ **crond** = crontab daemon/service that manages scheduling
- ○ **systemctl status crond** = To manage the crond service



```
┌───────────── minute (0 - 59)
│ ┌───────────── hour (0 - 23)
│ │ ┌───────────── day of the month (1 - 31)
│ │ │ ┌───────────── month (1 - 12)
│ │ │ │ ┌───────────── day of the week (0 - 6) (Sunday to Saturday;
│ │ │ │ │                                   7 is also Sunday on some systems)
│ │ │ │ │
│ │ │ │ │
* * * * * <command to execute>
```

- ○ Create crontab entry by scheduling a task:

  crontab -e

  schedule, echo “This is my first crontab entry” > crontab-entry

“at” command

- at command is like crontab which allows you to schedule jobs but only once
- When the command is run it will enter interactive mode and you can get out by pressing **Ctrl D**

Usage:

○ **at HH:MM PM**	=	Schedule a job
○ **atq**	=	List the at entries
○ **atrm #**	=	Remove at entry
○ **atd**	=	at daemon/service that manages scheduling
○ **systemctl status atd**		= To manage the atd service

- Create at entry by scheduling a task:

```
at 4:45PM → enter
echo “This is my first at entry” > at-entry
Crtl D
```

“at” command

Other future scheduling format:

- **`at 2:45 AM 101621`** = Schedule a job to run on Oct 16th, 2021 at 2:45am
- **`at 4PM + 4 days`** = Schedule a job at 4pm four days from now
- **`at now +5 hours`** = Schedule a job to run five hours from now
- **`at 8:00 AM Sun`** = Schedule a job to 8am on coming Sunday
- **`at 10:00 AM next month`** = Schedule a job to 10am next month

By: Imran Afzal
www.utclisolutions.com

Additional Cron Jobs

- By default, there are 4 different types of cronjobs
  - Hourly
  - Daily
  - Weekly
  - Monthly
- All the above crons are setup in
  - **/etc/cron.___** (directory)
- The timing for each are set in
  - **/etc/anacrontab** **--** except hourly
- For hourly
  - **/etc/cron.d/0hourly**

By: Imran Afzal

Process Management

- Background = Ctrl-z, jobs and bg
- Foreground = fg
- Run process even after exit = nohup process &

  OR = nohup process > /dev/null 2>&1 &
- Kill a process by name = pkill
- Process priority = nice (e.g. nice –n 5 process)

  The niceness scale goes from -20 to 19. The lower the number more priority that task gets
- Process monitoring = top
- List process = ps.

By: Imran Afzal

System Monitoring

- `top`
- `df`
- `dmesg`
- `iostat 1`
- `netstat`
- `free`
- `cat /proc/cpuinfo`
- `cat /proc/meminfo`

By: Imran Afzal

Log Monitoring

Another and most important way of system administration is log monitor

Log Directory = **`/var/log`**

- **`boot`**
- **`chronyd = NTP`**
- **`cron`**
- **`maillog`**
- **`secure`**
- **`messages`**
- **`httpd`**

System Maintenance Commands

- `shutdown`
- `init 0-7`
- `reboot`
- `halt`

By: Imran Afzal

Changing System Hostname

- hostnamectl – set-hostname newhostname

- Version 7 = Edit /etc/hostname
- Version 6 = Edit /etc/sysconfig/network

Finding System Information

- `cat /etc/redhat-release`
- `uname -a`
- `dmidecode`

System Architecture

- **Differences between a 32-bit and 64-bit CPU**

A big difference between 32-bit processors and 64-bit processors is the number of calculations per second they can perform, which affects the speed at which they can complete tasks. 64-bit processors can come in dual core, quad core, six core, and eight core versions for home computing. Multiple cores allow for an increased number of calculations per second that can be performed, which can increase the processing power and help make a computer run faster. Software programs that require many calculations to function smoothly can operate faster and more efficiently on the multi-core 64-bit processors

- **Linux = arch**
- **Windows = My computer → Properties**

Terminal Control Keys

Several key combinations on your keyboard usually have a special effect on the terminal.

These "control" (CTRL) keys are accomplished by holding the CTRL key while typing the second key.
For example, CTRL-c means to hold the CTRL key while you type the letter "c".

The most common control keys are listed below:

- **`CTRL-u   - erase everything you've typed on the command line`**
- **`CTRL-c   - stop/kill a command`**
- **`CTRL-z   - suspend a command`**
- **`CTRL-d   - exit from an interactive program (signals end of data).`**

Terminal Commands

- **`clear`**

Clears your screen

- **`exit`**

Exit out of the shell, terminal or a user session

- **`script`**

The script command stores terminal activities in a log file that can be named by a user, when a name is not provided by a user, the default file name, typescript is used

Recover Root Password

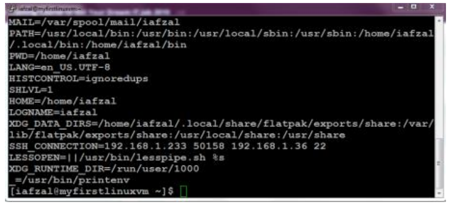
- Restart your computer
- Edit grub
- Change password
- reboot

SOS Report

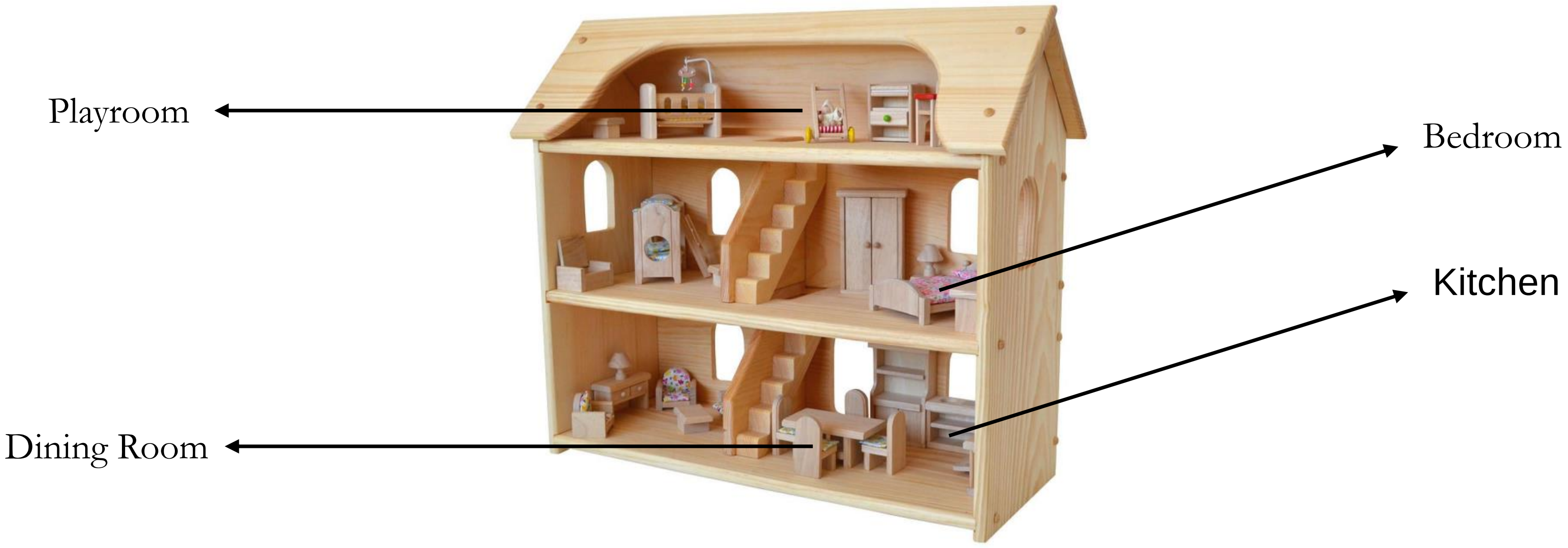
- What is SOS Report?
  - Collect and package diagnostic and support data
- Package name
  - sos-version
- Command
  - sosreport

Environment Variables

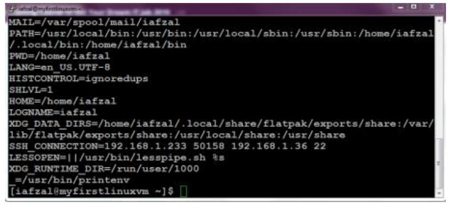
- What are environment variables?
  - An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer. They are part of the environment in which a process runs.
  - In simple words: set of defined rules and values to build an environment
  - E.g.



```
MAIL=/var/spool/mail/iafzal
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/iafzal
/.local/bin:/home/iafzal/bin
PWD=/home/iafzal
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/iafzal
LOGNAME=iafzal
XDG_DATA_DIRS=/home/iafzal/.local/share/flatpak/exports/share:/var/
lib/flatpak/exports/share:/usr/local/share:/usr/share
SSH_CONNECTION=192.168.1.233 50158 192.168.1.36 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/1000
_=/usr/bin/printenv
[iafzal@myfirstlinuxvm ~]$
```



Environment Variables

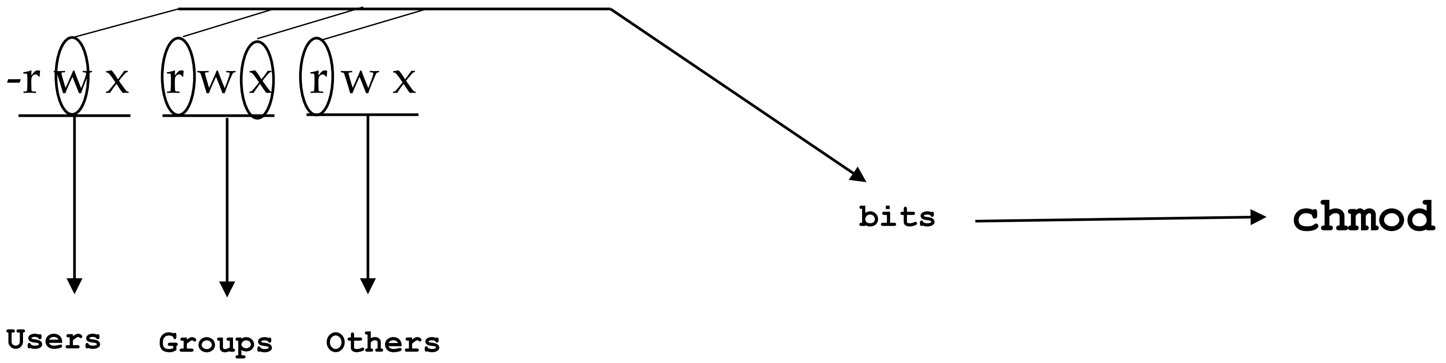


```
MAIL=/var/spool/mail/iafzal
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/iafzal
/.local/bin:/home/iafzal/bin
PWD=/home/iafzal
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/iafzal
LOGNAME=iafzal
XDG_DATA_DIRS=/home/iafzal/.local/share/flatpak/exports/share:/var/
lib/flatpak/exports/share:/usr/local/share:/usr/share
SSH_CONNECTION=192.168.1.233 50158 192.168.1.36 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/1000
_=/usr/bin/printenv
[iafzal@myfirstlinuxvm ~]$
```

- To view all environment variables
    - **printevn OR env**
- To view ONE environment variable
    - **echo $SHELL**
- To set the environment variables
    - **export TEST=1**
    - **echo $TEST**
- To set environment variable permanently
    - **vi .bashrc**
    - **TEST='123'**
    - **export TEST**
- To set global environment variable permanently
    - **vi /etc/profile or /etc/bashrc**
    - **Test='123'**
    - **export TEST**

Special Permissions with `setuid, setgid and sticky bit`

- All permissions on a file or directory are referred as bits



Not actual commands

- There are 3 additional permissions in Linux
  - **`setuid`**: bit tells Linux to run a program with the effective user id of the owner instead of the executor: *(e.g.* **`passwd`** *command)* → **`/etc/shadow`**
  - **`setgid`**: bit tells Linux to run a program with the effective group id of the owner instead of the executor: *(e.g.* **`locate`** *or* **`wall`** *command)*

    Please note: This bit is present for only files which have executable permissions

  - **`sticky bit`**: a bit set on files/directories that allows only the owner or root to delete those files

Special Permissions with `setuid, setgid and sticky bit`

- To assign special permissions at the user level
  - `chmod u+s xyz.sh`
- To assign special permissions at the group level
  - `chmod g+s xyz.sh`
- To remove special permissions at the user or group level
  - `chmod u-s xyz.sh`
  - `chmod g-s xyz.sh`
- To find all executables in Linux with setuid and setgid permissions
  - `find / -perm /6000 -type f`

Please note:
These bits work on c programming executables not on bash shell scripts

Sticky bit

- It is assigned to the last bit of permissions

-r w x r w x r w t

- Why?

Example of **`/tmp`** directory

Special Permissions with `setuid, setgid and sticky bit`

Lab exercise:

- Become root and create a directory allinone in / = **`mkdir /allinone`**
- Assign all rwx permissions to that directory = **`chmod 777 /allinone`**
- Become iafzal and create directory inside of /allinone = **`mkdir imrandir`**
- Give all rwx permissions to that directory = **`chmod 777 imrandir`**
- Create 3 files in that directory = **`touch a b c`**
- Open another terminal and login as spiderman
- Go to /allinone directory and delete imrandir directory = **`rm -rf imrandir`**
  - *You will see the directory is deleted*
- Now become root again and assign sticky bit permission to /allinone = **`chmod +t /allinone`**
- Become iafzal and create directory again inside of /allinone = **`mkdir imrandir`**
- Give all rwx permissions to that directory = **`chmod 777 imrandir`**
- Create 3 files in that directory = **`touch a b c`**
- Become spiderman user again
- Go to /allinone directory and try to delete imrandir directory = **`rm -rf imrandir`**
  - *Now as spiderman you cannot delete the directory*