# Installation of Ubuntu

# Setting wifi through DWA-131

I used some URL for RTL… Installed git, git clone…, sudo make, sudo make install,… And everything was fine. Did not use any of DWA-131 linux driver.

**Direct in a computer**

Create an Ubunto ISO file in removable disk. Boot from removable disk.

**VM**

Download Oracle VM: Create hard disk: In optical drive show path of removable disk created in step 1.

Start the VM. It will automatically install Ubuntu

**Started on 21-07-2018**

Good understanding of file system

<http://www.thegeekstuff.com/2010/09/linux-file-system-structure/>

Intermediate level ubuntu commands

<http://www.vogella.com/tutorials/Ubuntu/article.html#ubuntu>

Linux shell is programming interface through which we run commands against the system. There is BASH, CSH, TCSH etc.

Every Linux system has a user with name “root”. This is like any other user but can do everything. This is activated when you use sudo. ‘/root’ is home directory of user with name ‘root’.

By Canonical. Every year two releases. April release, October release. April release stable. Release names like 18.04, 18.10 for 2018 April and 2018 Oct release.

Email provided ThunderBird.

Editor: search by editor

Create user: Search box 🡪 Users 🡪Unlock 🡪 Add user

Manage user permission and groups: need another tool to install.

sudo apt-get install gnome-system-tools

This will add facility for groups and permissions in the Users window.

# Some commands

|  |  |  |
| --- | --- | --- |
| Command | Example | Details |
| ls –option directoryname | ls -l | Give more details of files |
| Find | find sample.\* |  |
|  | whoami | Current login user |
|  | pwd | Current working directory |
|  |  |  |
|  |  |  |
|  |  |  |

“~” denotes home and “..” denotes parent, “./” is current

cd ~ changes to home directory

### Installing Samba in Ubutu

Sudo apt-get install samba samba-common. After installing Samba it is possible to communicate and transfer files from windows system.

# Update and upgrade

sudo apt-get update just updates the **list** of packages for your system. It does not download the latest versions of packages. Sudo apt-get upgrade downloads the latest versions of all packages in your system. This should be done after sudo update.

# Understand file system

Linux has single tree file system irrespective of how many drives are there. There is only one base directory and then files and sub directories. The first directory or base directory is called root directory.

**pwd** is print working directory. At login the directory is user’s home directory wherein he is allowed to write. It is /home/<user name>.

**ls** lists all files and directories. **cd** change directory. cd /usr/bin changes to directory /usr/bin. This is absolute path. cd ./bin changes to bin directory which is same as cd bin.

cd ~<username> changes to home directory of <username>

## File names

* Starting with a ‘.’ is hidden file
* File names are case sensitive
* Extensions like .pdf or .docx etc. does not mean anything in linux. There is no concept of file extension.
* Long file names are supported with any characters in between.

**ls** dir1 dir2 will list two directories. ls -l means output in long format.

Most commands in linux look like

**command -options arguments**

* ls -lt lists the files in long format with sorted by time (-lt option means long and time)
* To determine the file type use **file** command. file <filename>. Extension of a file does not mean anything.
* less <filename> shows contents of text file. If more text then scroll available. ‘q’ means quit from less.
* Double click a file name to copy and middle click to paste.
* There are symbolic links which point to a file.

cp (copy), mv (move), mkdir(make directory), rem(remove), ln(link)

cp file1 file2. Complex wild card format exists. mv is used to move / rename.

ln command creates symbolic and hard links for a file. Symbolic link is like file shortcut. Hard link is like another name for the same file.

‘/’ is root directory, means base, means highest in hierarchy. Only user with name root has access to this directory. User with name root cannot be logged in. There are directory level permissions to users and not file level. All folders with (s) like /sbin, /usr/sbin, /usr/local/sbin are for system user only.

1. /boot : Contains the boot loader
2. /home : Contains the home directories of users.
3. /bin : All the executable binaries and commands used by all the users on the system are located here.
4. /sbin : This contains the system executable binaries typically used by system administrators.
5. /lib : Contains the system libraries that support the binaries in /bin and /sbin.
6. /etc : Contains the configuration files for network, boot-time, etc.
7. /dev : This has the device files i.e. usb, terminal device or any other device attached to the system are shown here.
8. /proc : Contains information about the process running.
9. /tmp : This is the temporary directory where many processes create the temporary files required. This is purged each time the machine is booted.

Linux follows FHS (Filesystem Hierarchy Standard). All directories are in “/” irrespective of physical drives. There are no “A:”,”B:”,”C:” etc.

There is a ‘home’ directory in ‘/’ folder. You will find a folder corresponding to each user in this folder. In each user’s folder you will find desktop, documents, downloads and several other folders. User can also create new folders in his subfolders.

It appears that a user has full control over the folder in his name inside the ‘/home’ folder sat ‘/Home/sushant’. Other folders outside of home folder of user, the user can work based upon his permission. When you open a command prompt the default directory is /Home/<user name>.

Most of installations in linux writes in different folders apart from user’s folder in /Home directory. Since a user has no default permissions over these folder hence a sudo prefix is required to enable root power. If a user wants to write anything in those folders outside his own Home then he needs permission.

# Commands

Commands can be:

* C compiled binaries, python, ruby, perl, shell etc.
* A shell built in command like cd
* Shell function
* Alias to other command which we can define

Type command is built in and tell what type of command it it

type <command>

multiple commands in single line separated by ;

**cd /usr; ls; cd –**

change directory to usr, list and come back to original directory.

## alias, unalias

alias foo = ‘cd /usr;ls;cd-’

unalias foo

above creates a new command and then removes it.

# Redirection and pipe

Redirect input and output of a command to and from file. Also connect multiple commands and create pipeline.

cat - Concatenate files

sort - Sort lines of text

uniq - Report or omit repeated lines

grep - Print lines matching a pattern

wc - Print newline, word, and byte counts for each file

head - Output the first part of a file

tail - Output the last part of a file

tee - Read from standard input and write to standard output and files

In linux everything is file. If a ls command outputs something that means it writes to a file stdout which is connected to screen.

ls -l /usr/bin > ls-output.txt will write results to the file. To append existing file use >> operator.

There are ways to redirect error and output to files using > and >> operators.

Pipe operator ‘|’

Output of a command is passed to input of another command.

ls -l /usr/bin | less

Output will be in form of paginated display.

**ls /bin /usr/bin | sort | uniq -d | less**

grep *pattern* [*file...*]

grep is used to find pattern in file.

**ls /bin /usr/bin | sort | uniq | grep zip** prints all lines where zip word occurs.

**head** and **tail** commands provide first few and last few lines of a file.

# SSH

Secure Shell. You can remotely connect to command prompt of a linux server remotely in secured manner. This is as good as your are having the command prompt of server at your desktop. It uses cryptographic keys for secured transmission. In windows Putty is used for SSH.

# Tips

Ctrl+alt+t Opens a terminal

Vim editor

* i - Enter interactive mode to edit the file
* Escape - Leave interactive mode
* / - Search in file
* :wq - Saves the file and exists vim
* :q! - Exists vim without saving

# Terminal commands

* **cd**change directory
* **ls**List directory contents
* **pwd**show working directory
* **cat**see the contents of a file
* **mkdir**create a folder
* **cp**copy a file
* **cp –a**copy a directory
* **mv**move a file
* **rm**remove a file
* **rm –rf**remove a directory ( no warning given)
* **df –h**see hard disk usage
* **locate**locate a file
* **du –h**show a the size of a specific directory
* **shutdown –h**shutdown the system
* **reboot**restarts the system
* **history**display all  previous commands typed in terminal
* **uptime**see how long the system has been running
* **ifconfig**see the TCP/IP settings of the system
* **chown**change the ownership of a file or directory
* **chmod**change permissions of a file
* **tar**compress/decompress Linux tool

crusader

# Shell

The terminal or CLI or command. Prompt is like username@machinename followed by current working directory (CWD). If last character is ‘#’ then admin privilege.

# Permissions

id – Display user identity

chmod – Change a file's mode

umask – Set the default file permissions

su – Run a shell as another user

sudo – Execute a command as another user

chown – Change a file's owner

chgrp – Change a file's group ownership

passwd – Change a user's password

Access rights may be read / write / execute. In linux everybody of ‘windows’ is ‘world’.

# Environment

There are *environment variables* and *shell variables.* There are also *shell functions*.

printenv | less shows all environment variables

set | less shows env variables, shell variables and shell functions.

**printenv USER**  displays user

**env $USER** also displays user

There is $PATH environment variable which like windows is search path for your programs. If you have an executable in bin folder of your home then you can include that in $PATH variable and you are ready to go with your program.

After modifying PATh variable, you can do export PATH.

# Permissions

id – Display user identity

chmod – Change a file's mode

umask – Set the default file permissions

su – Run a shell as another user

sudo – Execute a command as another user

chown – Change a file's owner

chgrp – Change a file's group ownership

passwd – Change a user's password

world means everyone in linux.

User accounts are defined in /etc/passwd and groups are defined in /etc/group file.

Access rights to files

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-r-- 1 me me 0 2008-03-06 14:52 foo.txt

First 10 chars are file attributes. 1st is File type

**Attribute File Type**

- A regular file.

d A directory.

l A symbolic link. Notice that with symbolic links

c A *character special file*. This file type refers to a device that

handles data as a stream of bytes, such as a terminal or modem.

b A *block special file*. This file type refers to a device that handles

data in blocks, such as a hard drive or CD-ROM drive

The remaining nine characters of the file attributes, called the *file mode,* represent the

read, write, and execute permissions for the file's owner, the file's group owner, and

everybody else:

**Owner Group World**

rwx rwx rwx

r: Read, w: Write, x: Execute.

chmod: change file mode

File owner and superuser can change file mode. Two ways. Octal, Symbolic. Octal: 3 digits: so fits well with file mode.

0 000 ---

1 001 --x

2 010 -w-

3 011 -wx

4 100 r--

5 101 r-x

6 110 rw-

7 111 rwx

[me@linuxbox ~]$ **> foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-r-- 1 me me 0 2008-03-06 14:52 foo.txt

[me@linuxbox ~]$ **chmod 600 foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw------- 1 me me 0 2008-03-06 14:52 foo.txt

Symbolic

u: user, g: group, o: world, a: all.

u+x: add execute for user, u-x: remove execute from user, +x: add execute to user, group and world, go=rw: group and world read and write permission, all other permissions removed.

chown can change file owner or group.

# Processes

ps – Report a snapshot of current processes

top – Display tasks

jobs – List active jobs

bg – Place a job in the background

fg – Place a job in the foreground

kill – Send a signal to a process

killall – Kill processes by name

shutdown – Shutdown or reboot the system

## System boot up

Kernal start : starts init process, a series of shell scripts in /etc : some are daemons which run in background without user interface.

Kernal keeps track of processes. Process has PID, owner name, memory used etc.

**Ps x** | **less** shows all processes. **ps** shows processes in the terminal.

In a terminal Ctrl-c interrupts a program.

Put & after a command and the program will run in background.

fg %1 brings back the job in foreground.

Signals can be sent to Processes with help of kill command.

# Environment

Commands: printenv, set: sets an option, export, alias: create alias for a command

Two types of data are stored in environment. Environment Variables and Shell variables.

**printenv | less** shows environment variables and **set | less** shows both variables.

**echo $HOME** also shows the variable value.

This command updates the path variable

PATH=$PATH:$HOME/bin

This way when you enter a command this is searched in $HOME/bin folder also. This is useful for installing private programs.

**export PATH** command provides the path variable to child processes also.

nano, gedit and vi are good text editors. They are already installed.

# Packaging systems

Two packaging systems. 1) Debian camp .deb, 2) Red hat camp .rpm

Debian Style (.deb) Debian, Ubuntu, Xandros, Linspire

Red Hat Style (.rpm) Fedora, CentOS, Red Hat Enterprise Linux, OpenSUSE,

Mandriva, PCLinuxOS

To update packages from repository

apt-get update; apt-get upgrade; apt-get install package-name

remove: apt-get remove package-name

low level installation of package from non repository

dpkg –install downloaded-package-file

list installed packages: dpkg –list

# Compilation

./configure

make

sudo make install

# Shell scripts

Create script, give permissions, put in directory where shell can find it.

* Using nano I created a file hello with this content: The first line is called shebang and it tells which interpreter to use.

#!/bin/bash

echo ‘hello world’

* Changed permission

chmod 755 hello

./hello

Remember ./ for execution in current directory.

Path environment variable: is ’ :’ separated list of directories to search.

In home directory of a user there is a hidden file .bashrc. You can edit this file as nano .bashrc and include following line:

Export PATH=~/bin:”$PATH”

Close the terminal and restart. If you have kept hello file in the newly created bin folder then just by giving command ‘hello’ the script will be executed. The above export command has added /home/user-name/bin in the beginning of PATH variable separated by a ‘:’. That works.

Without closing and opening the terminal if you give command “. .bashrc” then it will read .bashrc as if it were inputted through keyboard. “.” Command is called the source command and it is builtin bash command. It reads a file as if given from input.

## Good location for scripts

For personal use: ~/bin

For everybody to use: /usr/local/bin

For system admin: /usr/local/sbin

Use /usr/local hierarchy. The usr/bin is allocated for package distributors. When writing scripts long options for command are suggested for readability. **ls --all –directory** is better than **ls -ad**

## Installing Postgresql and PGAdmin in Ubuntu

* Sudo apt-get update
* Sudo apt-get upgrade
* Sudo apt-get install postgresql postgresql-contrib
* You may get this error

# [**You need to install postgresql-server-dev-X.Y for building a server-side extension or libpq-dev for building a client-side application**](https://stackoverflow.com/questions/28253681/you-need-to-install-postgresql-server-dev-x-y-for-building-a-server-side-extensi)

Do this:

sudo apt-get install libpq-dev python-dev

sudo apt install python-pip

pip install psycopg2

For restarting postgresql

**Sudo service postgresql restart**

* For setting password of postgresql:

**Sudo -u postgres psql postgres**

\password postgres

\q

* Above creates a user postgres. But login gives an error as ‘Peer authentication failed’

In (/etc/postgresql/10/main/pg\_hba.conf

Change

local all postgres peer

To

local all postgres md5

restart as:

sudo service postgresql restart

Now you will be able to use psql as

psql -U postgres

\q exits the postgresql.

* For PgAdmin installation follow the site <https://linuxhint.com/install-pgadmin4-ubuntu/>

I installed latest version of pgadmin. For that I had to replace the version number in several commands given in the site. Pgadmin worked in Ubuntu machine.

**24th Nov 2018**

I successfully installed pgAdmin 4 in Ubuntu machine using above site. There was problem connecting pgadmin with Docker Postgresql. Thereafter I successfully connected docker postgres image with pgadmin as follows:

* Sudo docker run -p 5432:5432 postgres:11.1
* Start pgadmin from this command after going to folder pgAdmin4/pgAdmin4. Before that follow to install pgAdmin as per above site. Python lib/python2.7/site-packages/pgAdmin4/pgAdmin4.py
* In pgAdmin4 give host as 127.0.0.1. If it does not work create a sudo user ‘postgres’ in Ubuntu. Then try. It worked for me.

**Installing plv8**

* Install using sudo apt-get install
  + git
  + G++, clang++
  + Python
  + Pkg-config
* Sudo apt-get install make

$ wget https://github.com/plv8/plv8/archive/v2.3.3.tar.gz

$ tar -xvzf v2.3.3.tar.gz

$ cd plv8-2.3.3

* Make

## Install plv8 from build

Use site <https://plv8.github.io>

I tried to build plv8 from source. In brand new installations of Ubuntu the running of make was successful but otherwise it did not build. Also after successful build the make install did not work. All the time it reported that file or directory not found.

But I was successful in creating the extension plv8 version 2.3.2 by using xtuple precompiled binaries at <https://github.com/xtuple/xtuple/wiki/Installing-PLv8>

Process was as follows:

* Download precompiled binary from <http://updates.xtuple.com/updates/plv8/linux64/xtuple_plv8.tgz>
* Sudo apt install libc++1
* Pg\_config –pkglibdir
* Cd tar-file-directory
* tar xf xtuple\_plv8.tgz
* cd xtuple\_plv8
* sudo ./install\_plv8.sh
* Enter the full path to the PostgreSQL installation directory when install\_plv8.sh asks as noted above. e.g. /usr/lib/postgresql/10
* Psql -U postgres
* Create extension plv8
* \q to come out of psql

The above commands successfully installed plv8 in Ubuntu.