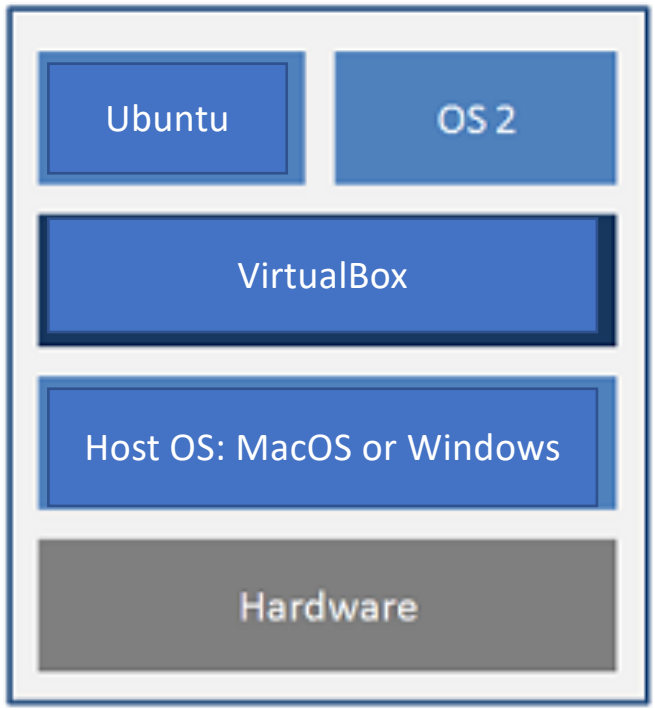
LING 408/508: Programming for Linguists

Lecture 5

Last Time

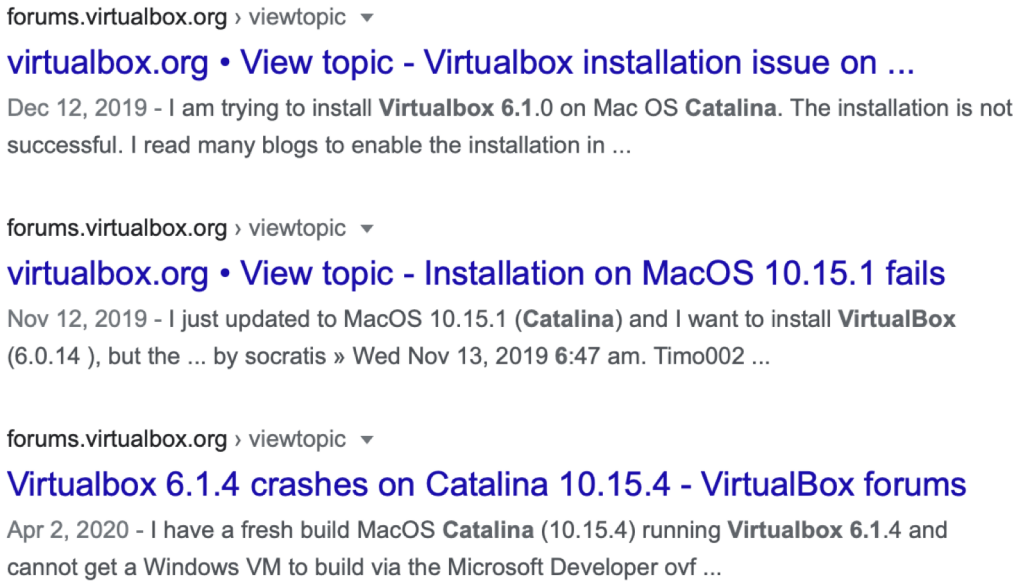
- Installing Ubuntu 18.04 LTS on top of VirtualBox
- Your Homework 2: *did everyone succeed*?
- Next slide: MacOS Catalina Problem



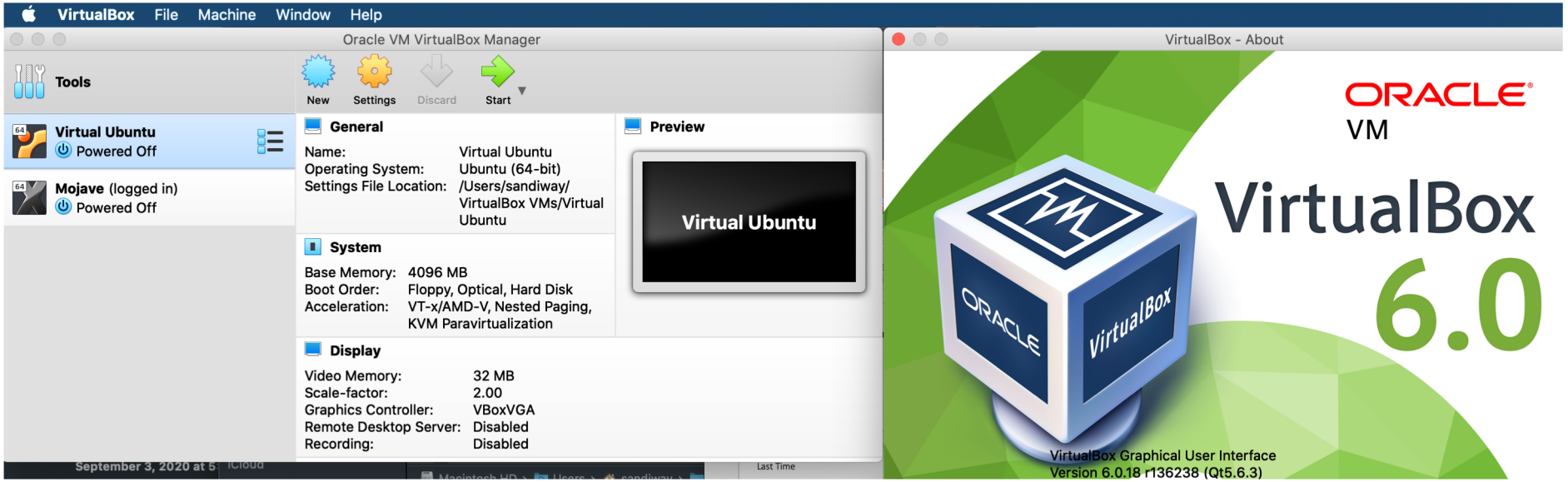
Hosted Architecture

MacOS Catalina

[Homar Aguilar] solved the issue by installing VB 6.0 instead of VB 6.1.(https://www.virtualbox.org/wiki/Download_Old_Builds_6_0)

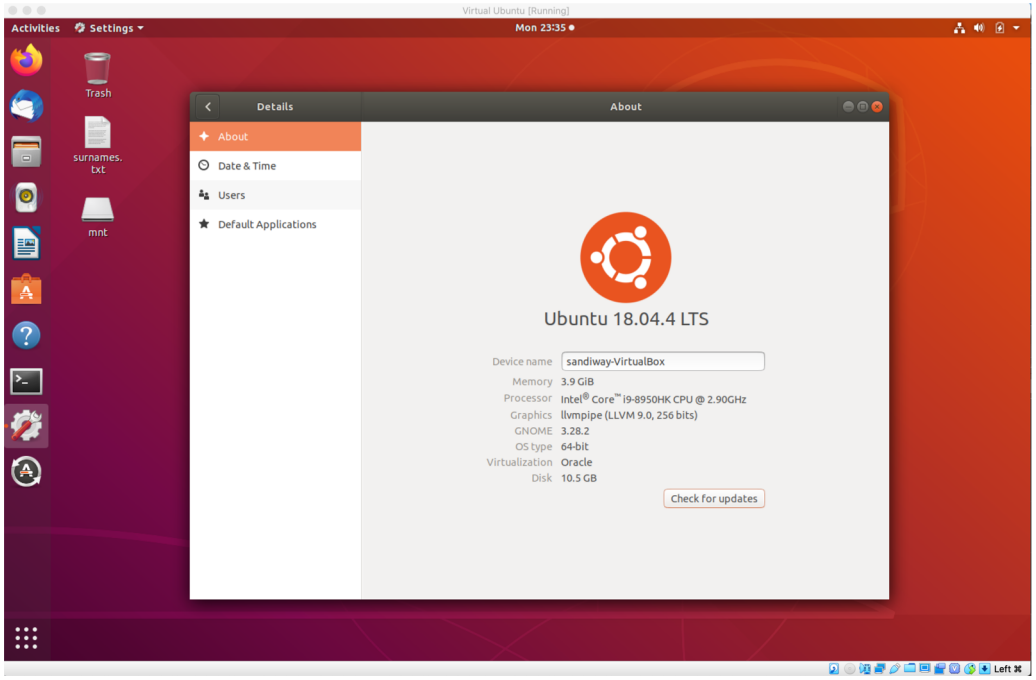


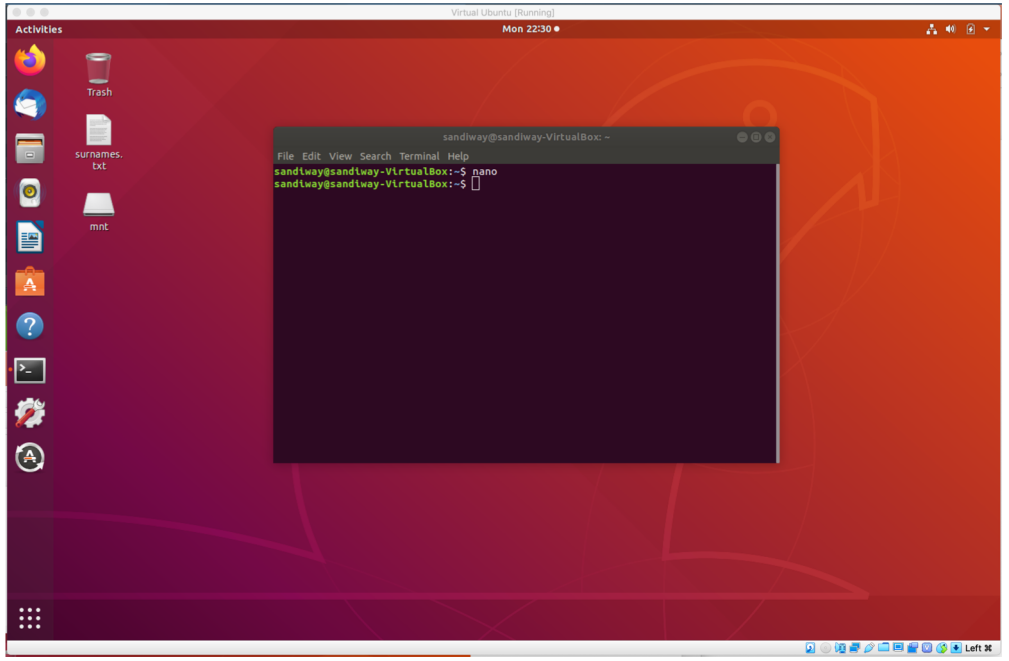
My installation (MacOS Catalina)



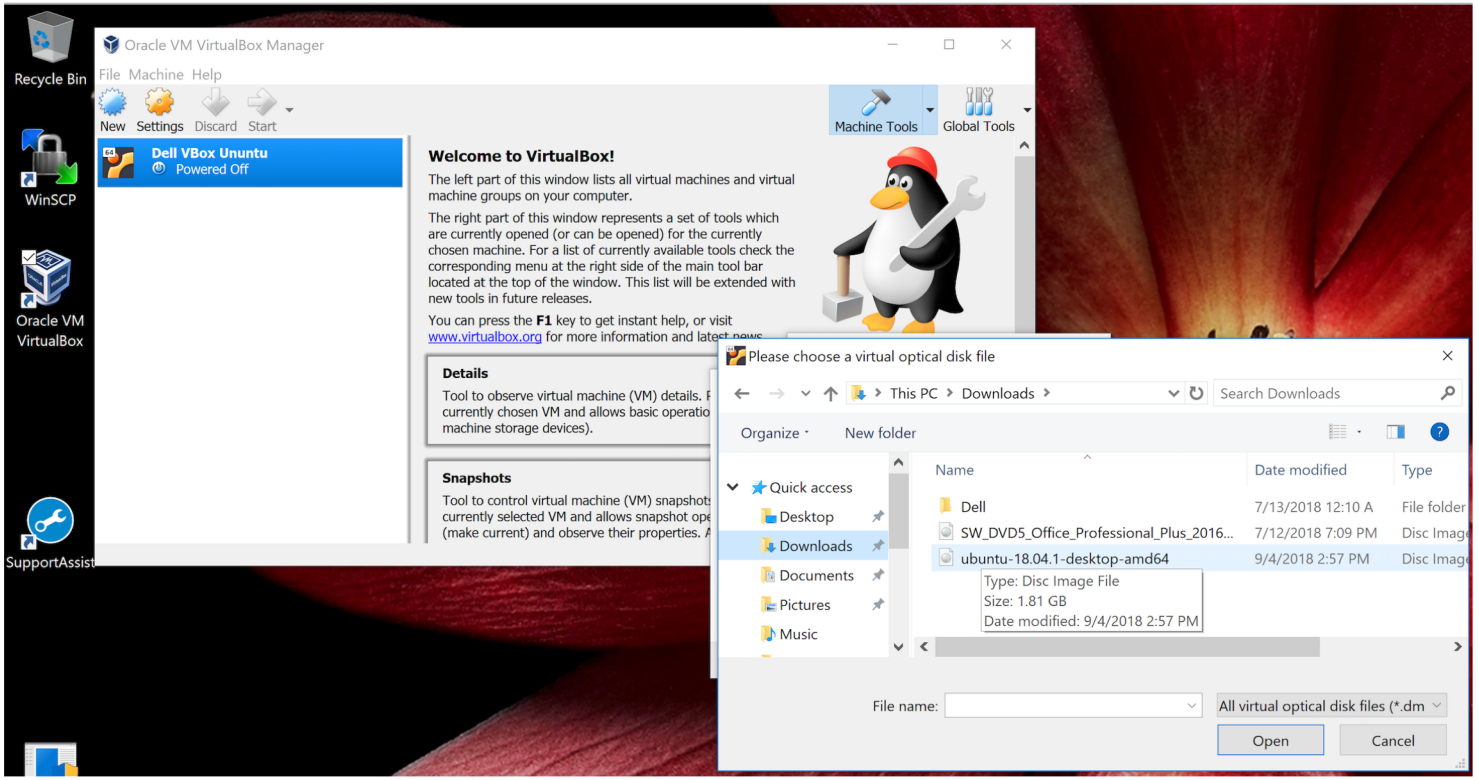
My installation (MacOS Catalina)

- Terminal then nano:

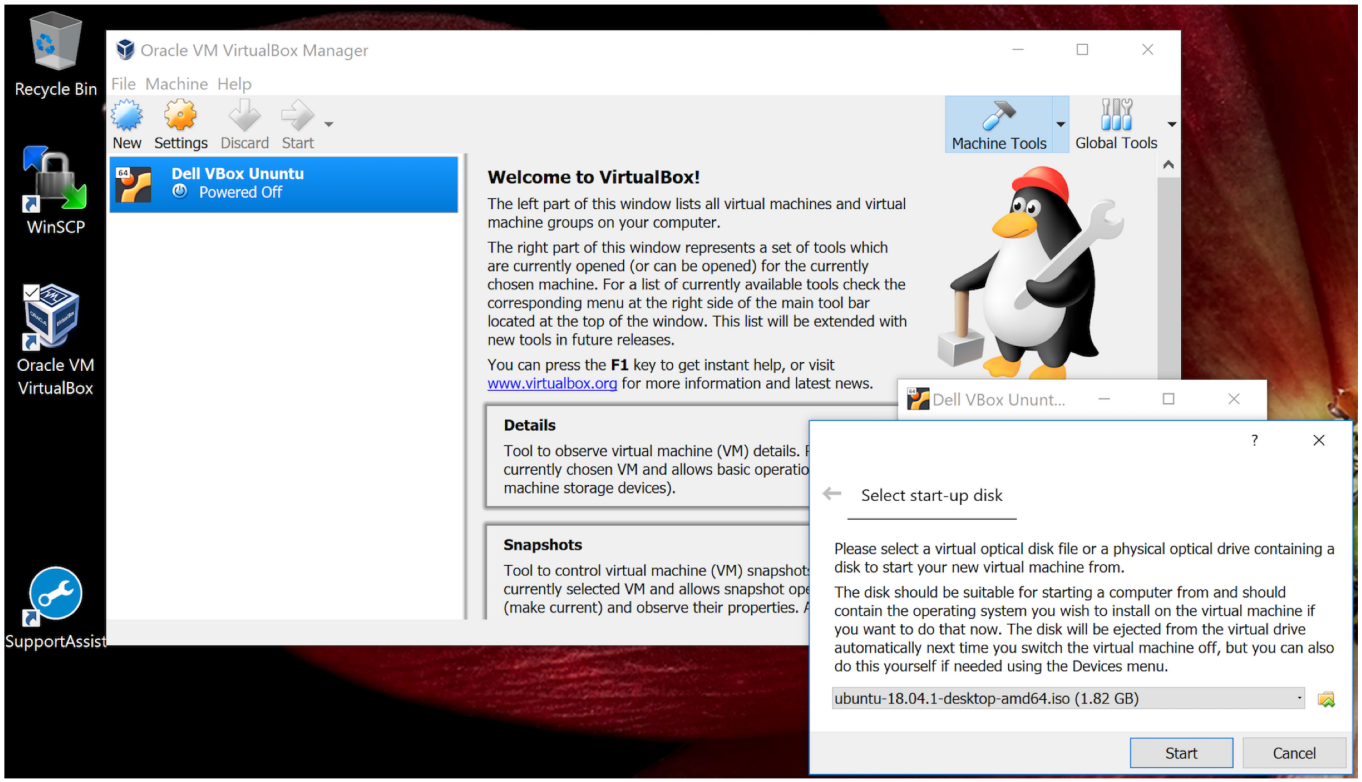




My Windows 10 install

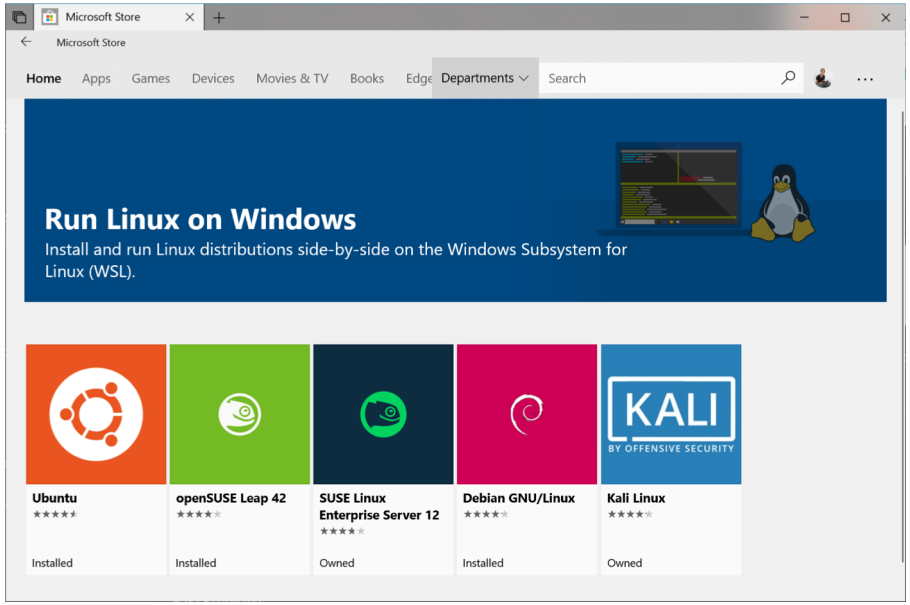


My Windows 10 install

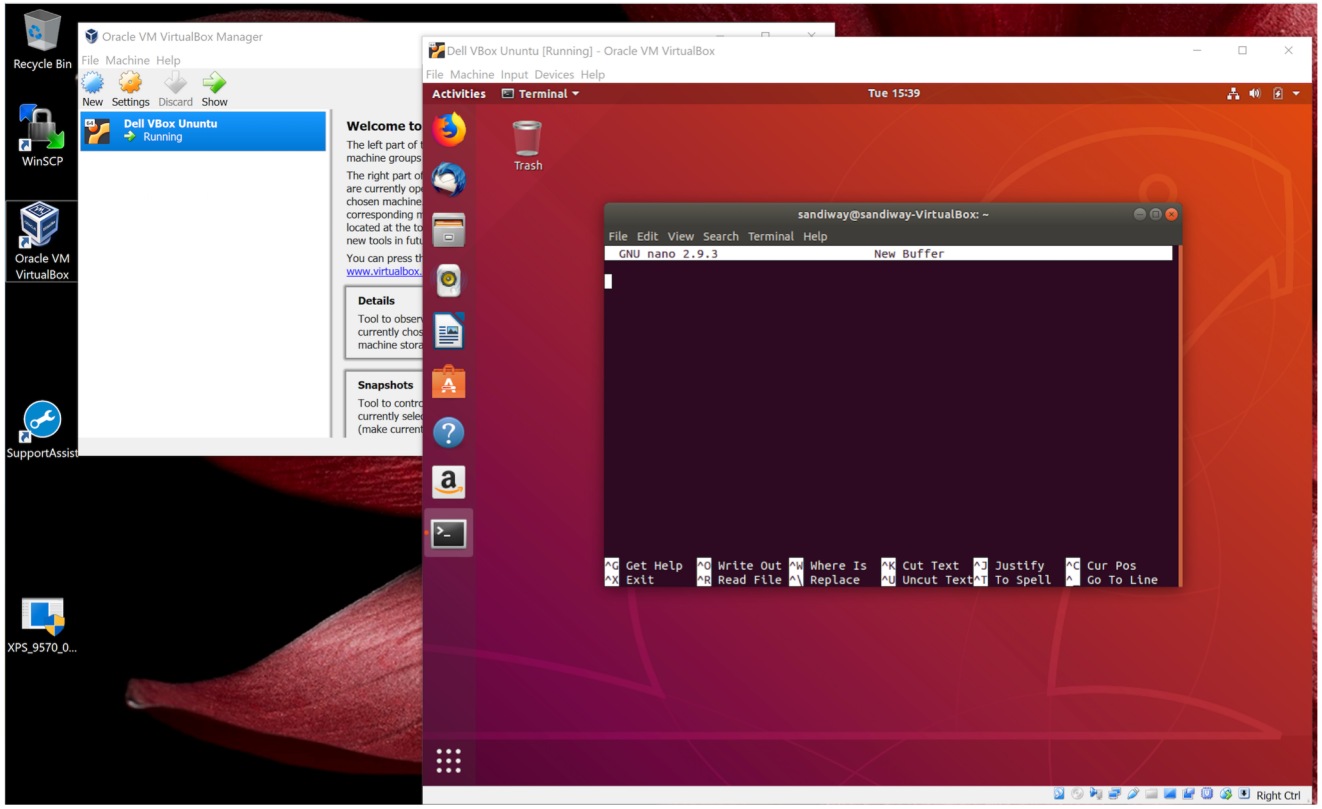


A Windows 10 alternative

- Windows Subsystem for Linux
  - https://docs.microsoft.com/en-us/windows/wsl/



Ubuntu

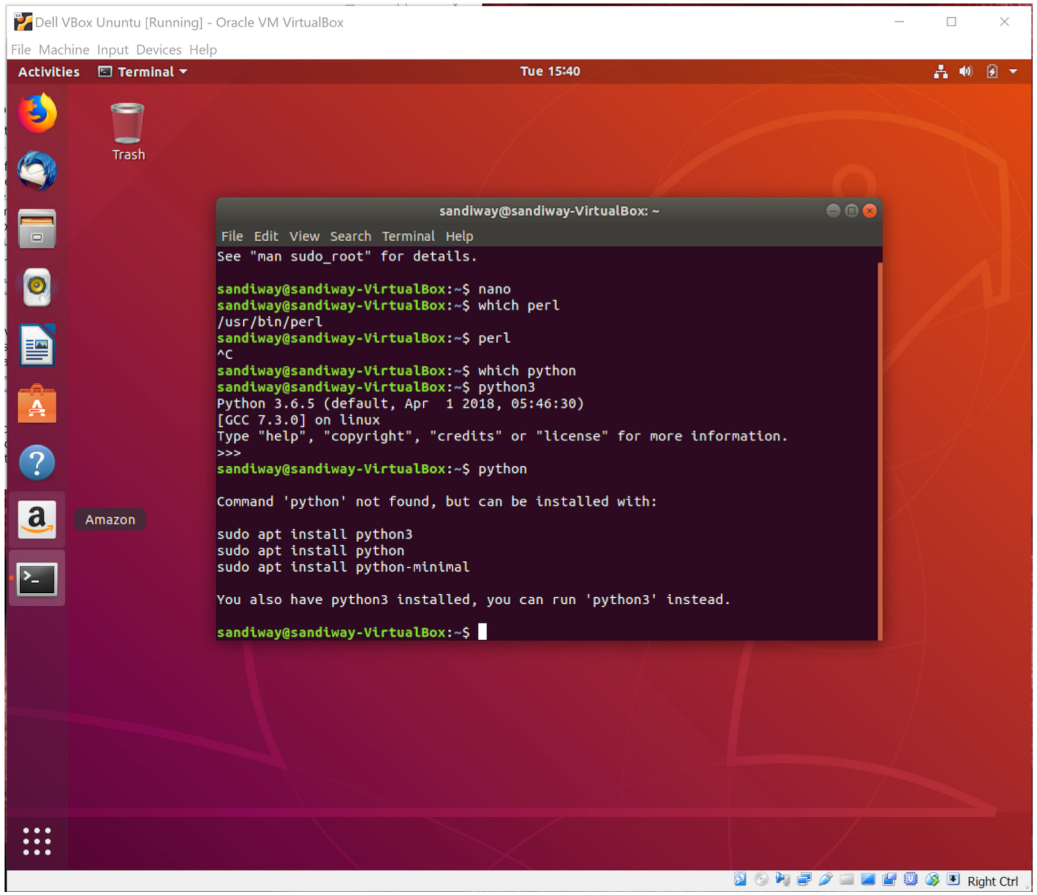


- Text editor (built in)
  - `nano` is a decent one for (use inside `Terminal`)
  - can install others via
    - `sudo apt-get install`

  or

    - `sudo apt install`

Ubuntu



- Notes:
  - `which` *command* returns path to *command* if found
  - `perl` is pre-installed.
  - `python3` is pre-installed.

Ubuntu

- Terminal:
  - runs a shell: **bash**
  - enter commands: some are built-in to the shell, others are executable files in specified directories ($PATH), still others will require apt-get install or apt install.

The shell has a programming language

```
sandiway@sandiway-VirtualBox:~$ swipl
The program 'swipl' is currently not installed. You can install
 it by typing:
sudo apt-get install swi-prolog-nox
```

  - command history is editable (up-arrow to retrieve…)
  - pre-defined environment variables: **env**
  - lots of packages are pre-loaded: **wish**, **python**, **perl**, etc.
  - **dpkg** (package manager for Debian)
  - **man** *command-name* (brings up manual page)

`dpkg -l` (list)

Shell

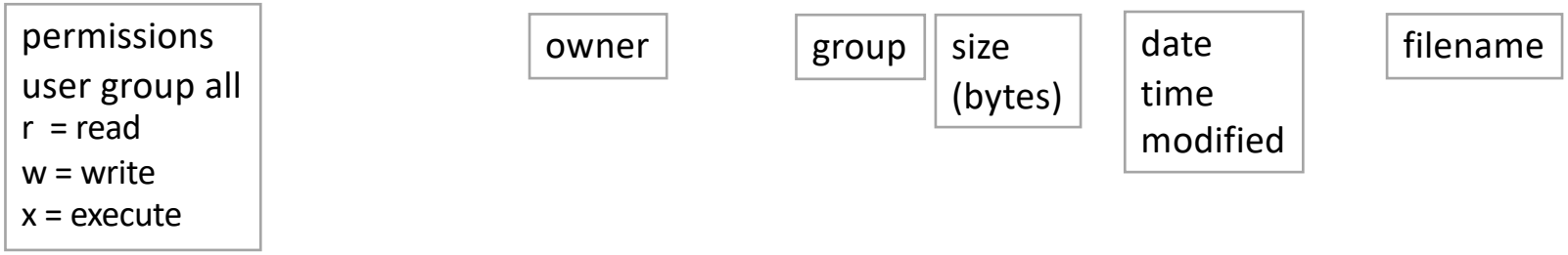
- simple commands:
  - **`pwd`** — print working directory
  - **`ls`** (**`ls –a`**) — list current directory (**-a** option: show . (dot) files too)
  - **`cd`** — change directory
  - **`mkdir`** — create a new directory
  - **`which`** *name* — prints the directory where command *name* is located, or nothing if it can't be found in the PATH
  - **`man`** *name* — display manual page for command *name*
  - **`echo $SHELL`** — prints the shell ($ prefixes a variable)
  - **`echo $PATH`** — shows the directories where the shell will look for commands

Shell

Directory shortcuts:

- your home directory: **~**
- current directory: **.**
- parent directory: **..**
- Examples:
  - **cd ..** (go to parent directory)
  - **mkdir ~/tmp** (create a new directory called tmp in your home directory)
  - **touch tmp** (create a new file tmp in the current directory if tmp doesn't already exist, or update the timestamp)
  - **ls –l tmp** (list attributes of file tmp in long format)

```
-rw-r--r--  1 sandiway  staff  0 Sep  4 09:26 tmp
```



`cat` command

- See http://www.linfo.org/cat.html

1.	cat *file1*		(print contents of file1)
2.	cat *file1* > *file2*		(‘>’ = redirect output to file2)
3.	cat *file2* | more		(‘|’ = pipe output to command more)
4.	more *file1*	– *easier*	(stops at end of screen, hit space to show more)
5.	less *file1*	– *easier*	(allows page by page display)
6.	cat > *file1*		(create file1 with input from terminal until **Control-D** EOF)
7.	cat		(input from terminal goes to terminal)
8.	cat >> *file1*		(append input from terminal to file file1)
9.	cat *file1* > *file2*		(file copy)
10.	cp *file1 file2* – *easier*		(cp = copy)
11.	cat file1 file2 file3		(prints all 3 files)
12.	cat *file1 file2 file3* > *file4*		(prints all 3 files to file4)
13.	cat *file1 file2 file3* | sort > *file4*		(3 files sorted alphabetically to file4)
14.	cat – *file5* > *file6*		(‘-’ = input from terminal)
15.	cat *file7* - > *file8*		

Shell Arithmetic

- at the shell prompt:
  - expr 1 + 3 (Need spaces cf. expr 1+3)
  - expr 2 '*' 2 (cf. expr 2 * 2)
  - echo `expr 1 + 3`
  - i=2 (NO SPACES! cf. i = 2)
  - expr $i + 1

  - let x=1+3 (cf. let x=1 + 3)
  - echo $x
  - let i=$i+1 (also ok let i=i+1)
  - echo $i

  - ((x = 1+ 3)) (spaces not significant)
  - echo $x
  - echo $((1+3))
  - ((i=i+1)) (also ok let i=$i+1)

Comparison operators

- Format:

  if [$x OP $y]; then

  …

  (else/elif…)

  fi

  - [….] is known as *test*
  - OP:
    - -eq *equals*
    - -ne *not equals*
    - -gt *greater than*
    - -ge *greater than or equals*
    - -lt *less than*
    - -le *less than or equals*

- Examples:
  - `echo $x $i`

    2 5
  - `test $x -le $i`
  - `echo $?` (*exit status*)

    0

  - `test $x -le $i -a $i -lt $x`
  - `echo $?`

    1

Input

- At a terminal:
  - `read –p “Name: ” name`
  - `read –p “Enter X and Y: ” x y`
  - `echo $x`
  - `echo $y`