Basic Command Line Actions

Here are a few example uses of Linux command-line commands. There are thousands of possible commands, one for each kind of computer program that might be useful. Each command also can have many options, most of which are be rarely used. The most useful commands are described in this note.

This note assumes that you have set up your Raspberry Pi by following the instructions in

http://yosemitefoothills.com/ACM_Workshop_Notes/SettingUpRaspberryPi-June-26-2016.pdf

When you boot up a Raspberry Pi, it can be set to boot up with a console command-line window that needs a user name an password to get started, or it can be set to boot up with a graphical desktop that does not require a password. In either case, commands that require superuser privileges can be used by the "pi" user by preceding the command by "sudo".

There are numerous sources for learning Linux commands, but here are a couple
http://linuxcommand.org/index.php
and
https://www.raspberrypi.org/documentation/
within which you can find general Linux information at
https://www.raspberrypi.org/documentation/linux/README.md
and specifically information about a few common commands at
https://www.raspberrypi.org/documentation/linux/usage/commands.md

The following is a re-edited version of notes written for grandchildren. Try not to be offended when the discussion is less technical than you want.

Some example commands

Click on the menu option named **Menu/Accessories/Terminal** to open up a **command-line window** (or log into a console). At the left side of the top line, a command-line processing program called **bash** will write **`pi@raspberrypi ~ $`,** called the **command prompt**, and then wait for you to give it a command.

Try typing `date` and press the **enter key**. You will be running a **program** named `date` which will print out the current date and time on the next line and then a new prompt will be printed on the following line.

Typing `hostname` (followed by the enter key) causes it to reply with **`raspberrypi`**, or whatever has been set as the network name of your computer.

Typing `whoami` returns your **user name**, **`pi`**.

Most commands have **options**. If you type `date -u`, the option 'u' preceded by a '`-`' causes the `date` command to return the time in universal time, UTC, defined as the local time in Greenwich, England. `date --utc` also does this by using a longer form of the option.

All options for date can be seen by typing `date --help` or `man date`. Here, `man` is short for "manual" and usually causes the computer to display a detailed explanation of the command that is given after it (`date` in this example).

Safely halting and rebooting

You should not just turn off the power to shutdown the computer, but rather ask it to halt by typing the two-word command `sudo halt` . That allows it to tidy up various things before shutting down. About 10 seconds later when the green LED stops blinking, you can turn off its power. Here, `sudo` is necessary in front of the `halt` because you are doing something that is risky and must be done with superuser privileges. Using `sudo` temporarily gives you the power of the `root` user. There is also a command `sudo reboot` that is sometimes needed to restart the

computer after changing the operating system.

Failure to follow this procedure will usually not lead to problems. As with all computers, unexpected power failures happen and only cause problems when they happen at a very inopportune time like when it is writing to the disk.

The SD memory card

Nearly all of what the Raspberry Pi "knows" is stored on the tiny **SD card** plugged into a slot underneath it. Think of the SD card as being like a giant library with two buildings. One building is kind of secret and rarely entered by visitors, the other is huge and has one huge room with some books but that room also has other rooms within it. Those have still other books and more rooms, etc.. Each book and room has a name and some access permissions. Each book is made up of many individual entities called **bytes**. There are 256 kinds of bytes, some can represent alphabet letters, others digits and punctuation marks. One is simply a blank space like we use between words.

In computer terminology, the buildings of the library are called "**drives**." The rooms are called "**directories**." The books are called "**files**." The first directory (room) that contains all other rooms is called the **root directory** and is given the single-character name "/". It is also called the **top directory** if one thinks of the library is being like an upside-down tree with a single root at the top and branches extending downward.

So the library is basically a large collection of bytes. Your SD card might hold 32 billion bytes. Some of the directories hold files that tell the Raspberry Pi how to do things, these are called programs. Others contain bytes that form documents full of words. Still others have bytes that can be made into pictures or even sounds.

A special user called the **`root user`** has full access to everything, but **ordinary users** like `pi` have more restricted access. `pi` might not be able to read some books, change some, or even see what is in some rooms. `pi`, however, might be allowed special permission to act like the `root user` for a while, but must ask to do so only when really necessary.

Exploring the SD memory card Directories

Typing `pwd`, which stands for "print working directory, lets you know where your are in the entire library building. If you just started a command-line session, `pwd` gives `/home/pi` which means that you are in the `pi` directory which itself is in the `home` directory which in turn is in the top most directory `/`. This particular directory is called the **home directory of the user `pi`** and is often represented by a **tilda** "`~`".

Now try typing the command `ls` which is short for "list". It will print out something like this:

```
Desktop  Documents  Downloads  Music  Pictures  Programming  python_games  Templates Videos
```

These are the names of files and directories in the your current working directory, `/home/pi`. The `ls` command result showed that in your home directory are some other directories (**subdirectories**) colored blue with the names `Desktop`, `Documents`, `etc.` There are actually no individual files shown above, but if they were present, they would be shown in white.

When file and directory names are made from multiple words, they are either run together like `GeigerCounter` or combined with an underscore "_" character like `python_games`. Putting a space between words making up a file or directory name is allowed, but makes certain operations more difficult. I try to avoid names with spaces.

Capitalization matters! Trying to run the `date` program by typing `Date` will not work.

The `ls` command can be followed by a directory name. For example, typing `ls Documents` will produce the contents of the `Documents` directory:

```
BlueJ Projects  Greenfoot Projects  Scratch Projects
```

Here, my preferred naming convention of avoiding spaces in file and directory names has not been followed by the person assembling the default Raspbian system; there are just three files here, each name has two words and quotes or a backslash (`\`) character must be used to handle the spaces. For example, this is clearly seen using the `ls`

command with the option `-l` (letter `l`, not number `1`). Typing `ls -l Documents/Scratch\ Projects` or `ls -l Documents/"Scratch Projects"` gives

```
total 2276
-rw-r--r-- 1 pi pi   42396 Apr 28  2013 Asteroid Blaster.sb
-rw-r--r-- 1 pi pi  488285 Feb  7  2013 Pacman for Scratch.sb
-rw-r--r-- 1 pi pi 1793692 May  9  2013 Scratch Invaders.sb
```

Here the "`-`" at the start means that the entry is a file; a "d" in that location would signify that the name is a directory. Other letters signify other types of files.

The next 9 characters and the two "`pi`" entries have to do with **permissions** to access and use the file.
The number 1 is the number of "**hard links**" to the file. We can ignore hard links and permissions for now. The remainder of the line contains the file **size in bytes**, the **date it was last changed**, and its **name**.

If you do that ls -l command without the quotes or backslash, you will get an error because it will be looking for two items Scratch and Projects instead just one "Scratch Projects". Typing `ls -l Documents/Scratch Projects` gives

```
ls: cannot access Documents/Scratch: No such file or directory
ls: cannot access Projects: No such file or directory
```

When you started `Terminal` or are in a console and received the prompt `pi@raspberrypi ~ $`, you were automatically placed in your home directory `/home/pi`. The prompt lets you know that your user name is `pi`, you are on the `raspberrypi` computer, and in your home directory `~`. It may seem silly to be informed who you are and which computer you are using, but in the networked world, you might be on another computer somewhere else in the world and acting as a different user.

The disk is organized like the branches of an upside-down tree with a single root at the top. To see the first set of files and directories under the root, we simply type `ls /` and find that it contains 19 directories, one of which is named `home`. To see what is in the `home` directory, we type `ls /home` and find that it contains 1 directory named `pi`. To see what is inside of the `pi` directory, we can type `ls /home/pi`. Next, let's go deeper into the upside-down directory tree by typing `ls /home/pi/Music` and find two more directories `Mp3` and `Midi`. (These were installed as part of our Setting Up instructions.) Going still deeper with `ls /home/pi/Music/Mp3`, we find a directory named `AFA_Music`, and a file named `PlayingMp3`. Finally, upon using the command `ls /home/pi/Music/Mp3/AFA_Music`, we see a list of songs in files with names that end in `.mp3` .

The file `PlayingMp3` tells us how to play these files. To read what is in that file, we can type the command `cat /home/pi/Music/Mp3/PlayingMp3`, and we will see its contents which are

> The command to play an `.mp3` file that is in your current working directory is `omxplayer` followed by a space and the name of the song.

To follow these instructions, we need to understand how to change our working directory. Remember that the command `pwd` tells you your present working directory. The `cd` command is used to change the working directory. To start from `/home/pi` and change to `/home/pi/Music/Mp3/AFA_Music`, we could do

```
cd /home/pi/Music/Mp3/AFA_Music
```

Here, you *must* have a `/` before `home`. But there is a shortcut if you are already in `/home/pi`. (Doing cd alone changes your working directory to your home directory no matter where your are currently.) From there, you could have taken the shortcut:

```
cd
cd Music/Mp3/AFA_Music
```

In this case, be sure you do *not* have a `/` before `Music`. **A common mistake is to put an initial '`/`' where you don't need it or to fail to put one where you do need it.**

After doing this working directory change, the `pwd` command will now show `/home/pi/Music/Mp3/AFA_Music` and `ls` alone will show the list of music pieces from the Air Force Academy (no copyright problem with these).

Following the directions in PlayingMp3 and making the directory change just described, we play the piece afsong.mp3 by typing

```
omxplayer afsong.mp3 .
```

The cd command alone, without any file, puts you in your home directory again.

```
cd
```

The omxplayer is also used to play video files with endings of .mp4, .h264, and some others.

To play Midi files (which end in .mid), we follow a similar procedure except that we use timidity instead of omxplayer:

```
pi@raspberrypi ~ $ cd /home/pi/Music/Midi
pi@raspberrypi ~/Music/Midi $ timidity 2001SprachZarathustra-RichardStrauss.mid
```

The prompts are shown here to illustrate that the tilda ~ before the $ in the prompt actually represents your home directory /home/pi and when you change to /home/pi/Music/Midi, this part of the prompt changes to ~/Music/Midi. Also, the cd command could have been written as cd ~/Music/Midi .

Finally, there are some short sound files in /home/pi/Programming/Java/originals/sounds which end in .wav (installed during our Setting Up procedure). These are played by the command aplay:

```
cd /home/pi/Programming/Java/originals/sounds
aplay train.wav
aplay startup.wav
```

Pictures and other images have data in files with endings like .jpg, .jpeg, .png, … To see them on the graphics screen you use the command feh. For example,

```
feh /home/pi/Pictures/ValleyAboveBennettville.jpg
```

List of some commands used so far:

```
aplay                   feh                     omxplayer
cat                     hostname                pwd
cd                      ls                      sudo halt
date                    ls /home/pi             sudo reboot
date -u                 ls -l                   timidity
date –utc               man                     whoami
```

Reading and writing documents

Earlier, we used the cat program to show what was in a simple text file /home/pi/Music/Mp3/playingMp3. To create that file and many others, I used an editor program called vim. It is a fast editor that can be run without any graphical desktop and is easily used over a network connection. For that reason, it is a favorite for many programmers and system managers.

Other editors which are available require the graphical desktop and are activated from the menu button of the desktop. The default install contains a simple graphical editor that produces text files. It is called leafpad opened by doing Menu/Accessories/Text Editor. When writing computer programs, vim is much better because it uses colors to reveal important aspects of the program contents. This can be seen by doing

```
vim /home/pi/Programming/Python/NoGraphics/powersOf2.py
```

and comparing with

```
leafpad /home/pi/Programming/Python/NoGraphics/powersOf2.py
```

leafpad is very easy to learn to use. Just start it up and play around.

A more advanced way to create documents uses libreoffice. libreoffice is somewhat tricky to use and has some advanced features that can annoy the beginner, but it can be used to create professional documents for

publication.

Documents created by `libreoffice` have an filename ending in `.odt` . They are not simple text documents. If you attempt to display them using the `cat` command, they will be unreadable. You should read them using `libreoffice`. To create a new .odt file named `MyNotes.odt` in the directory `~/WorkshopNotes/`, start libreoffice by doing `libreoffice`, create your text and then save it to the `WorkshopNotes` directory with the name `MyNotes`. The suffix `.odt` will be added automatically.

You must, of course, have the graphical desktop operating to use `libreoffice`; it is a word processing program, not just an editor.

PDF files are viewed by the program `evince`. For example,

```
evince ~/WorkshopNotes/BasicCommandLineActionsForWorkshop.pdf
```

How to use the `vim` editor

`vim` is a bit harder to use because it is entirely run from the keyboard and therefore has different modes of operation where the same key might do different things in the different modes.

A beginner using `vim` to create or edit a file named `/home/pi/WorkshopNotes/myFirstFile` should type:

```
cd /home/pi/WorkshopNotes
vim myFirstFile
```

You are started in **COMMAND mode**, but you should immediately type the letter "**i**" to change to **INSERT mode**. The word INSERT will then appear at the lower-left corner of the window. **If you forget to go into INSERT mode, your typing will do weird things since each key stroke is likely to execute some special editing command.**

Next, type a few lines of text. The editing keys like backspace, cursor up/down/left/right, shift, and enter should function as expected.

When finished, use the **escape** key (Esc) to leave INSERT mode and go back to COMMAND mode.

Then, type the colon key "**:**" remembering to use shift and see a colon appear in the lower left corner of the screen.

Next, type a "**w**" key to tell it to write what you have.

Finally, type a "**q**" key followed by the enter key to tell it to quit.

If you then do

```
cat myFirstFile
```

you should see your creation.

To summarize: **`vim myFirstFile, Enter`** (to start it), **i**, do your stuff, **`escape`**, **`:wq`**, **`Enter`**

When in COMMAND mode (INSERT **not** shown on bottom line), you can do the following:

i	Switch to INSERT mode and start inserting text before current character.
a	Switch to INSERT mode and start inserting text after current character.
dd	Delete current line and place its contents into a copy buffer.
7dd	Delete current line and the 6 lines following it and put them into the copy buffer.
yy	Yank (copy) the current line and put it into the copy buffer.
3yy	Yank (copy) the current line and the following 2 lines and put them into the copy buffer.
p	Place the copy buffer contents (the last deleted or yanked lines) just after the current line.
u	Undo the last change.
:wq	Save changes and quit.

:q!	Quit without saving changes.
:r `afile`	Insert the contents of the text file named `afile` between the current line and the next line.
:set number	Show line numbers. They will not be part of the text, but are useful during editing.
:set nonumber	Stop showing line numbers.
:w aNewName	Write the contents with a different name than used to open the file, e.g. for a backup copy.

There is much more capability in the vim editor, but this list gives you a start.

Other useful commands

To keep the system up-to-date, you need to occasionally (like weekly) run the following commands:

`sudo apt-get update`	This checks what updated programs are available.
`sudo apt-get dist-upgrade`	This upgrades the programs on your system.
`sudo reboot`	This is only necessary if the `rpi-update` program made any changes.

When you want to add a new program, for example the `shotwell` photo organizer, you do:

```
sudo apt-get install shotwell
```

This will give you a list of software packages it needs to install to support the `shotwell` program and how much SD card space will be required. You can then say “Y” or “n” for yes or no, with “Y” being the default.

If you need to stop a running program, you can usually do so by typing <ctrl>-c, a “c” with the control key pressed.

To copy a file from one place to another, use the `cp` command:

```
cp WorkshopNotes/MyNotes.odt WorkshopNotes/MyNotes.backup.odt
```

To copy an entire directory tree, do:

```
cp -a WorkshopNotes WorkshopNotesBackup
```

To move a file from one place to another or to just rename a file, use the `mv` command:

```
mv /home/pi/WorkshopNotes/MyFirstFile /home/pi/MyOldFirstFile
```

There are two common abbreviations used in referring to directories, “.” and “..”. The first refers to the current directory and the second refers to the parent directory above the current one. If these are used with file or directory names after them, a “/” must be used (e.g. “`./afile`” or “`../afile`”).

To execute a program that has not been made part of the system, you must use a “./” in front of the program name. For example, even when you are in the `~/Programming/Python/NoGraphics` directory and want to execute the `powersOf2.py` program, you must do

```
cd ~/Programming/Python/NoGraphics
./powersOf2.py
```

(This assumes that the first line of `powersOf2.py` states `#! /usr/bin/python3` which tells the system that it is a python3 program. Also, it must have been given execute permission by doing `chmod +x powersOf2.py`.)

Return to your home directory by doing

```
cd
```

If the `mv` command above were being executed from `/home/pi`, it could have been written as:

```
mv WorkshopNotes/MyFirstFile MyOldFirstFile
```

If you do not want to change the name, just move the file, a simple `.` will suffice.

```
mv WorkshopNotes/MyFirstFile .
```

Forgetting the space and dot at the end of the `cp` or `mv` commands is a common error.

Make a new directory named `MyDocuments` in the `/home/pi/WorkshopNotes` directory by doing

```
mkdir WorkshopNotes/MyDocuments
```

To move `WorkshopNotes/MyNotes.odt` to `WorkshopNotes/MyDocuments` directory.

```
mv WorkshopNotes/MyNotes.odt WorkshopNotes/MyDocuments/
```

To remove a file, use the `rm` command. There is no mercy. When gone, it cannot be recovered. So be careful.

```
rm junkFile
```

To remove a directory, it must first be emptied using the `rm` command, then do

```
rmdir uselessDirectory
```

(The `rm` command has a `-i` option which can be used to ask you if you are sure. Also there is a very dangerous way to use the `rm` command which can delete everything in a directory including all files and subdirectories.)

There are files that are normally hidden from view. They are not secret files, just ones that users generally do not want to pay attention to. Such files have names that begin with a period, like `/home/pi/.profile`, and hold configuration information specific to the user `pi` (in this case). You will rarely need to deal with these files so it is best for them to be "hidden." To see them, you can use the 'a' option to the `ls` command:

```
 ls -a /home/pi
.            .bash_logout  .config  Documents       .local       Pictures  python_games .themes     .xsession-errors
..           .bashrc       .dbus    Downloads       Music        .profile  .ssh         Videos
.bash_history .cache       Desktop  .gstreamer-0.10  Public    Templates    .Xauthority
```

Additional New Commands

`sudo`	`cp`	`rmdir`
`apt-get update`	`rm`	`ls -a`
`apt-get dist-upgrade`	`mv`	
`apt-get install`	`mkdir`	

Remember that manuals for the commands are available by using the `man` program, e.g. `man feh` .
Also, the command with the option `--help` is often also helpful, e.g. `timidity --help` .

Backing up all home directory files to a thumb drive

Backing up is a bit more complicated because the system on the Raspberry Pi uses permission data associated with its files, and that data must also be backed up with the files. The `tar` command is used to do this correctly.

Check to see what drives are currently attached when the thumb drive is NOT plugged in:

```
df -T
Filesystem     Type     1K-blocks    Used Available Use% Mounted on
/dev/root      ext4      15392432 3664628  11048056  25% /
devtmpfs       devtmpfs    469540       0    469540   0% /dev
tmpfs          tmpfs       473872       0    473872   0% /dev/shm
tmpfs          tmpfs       473872    6432    467440   2% /run
tmpfs          tmpfs         5120       4      5116   1% /run/lock
tmpfs          tmpfs       473872       0    473872   0% /sys/fs/cgroup
/dev/mmcblk0p1 vfat         64456   20968     43488  33% /boot
tmpfs          tmpfs        94776       0     94776   0% /run/user/1000
```

Plug the thumb drive into a USB port and check again

```
df -T
Filesystem     Type     1K-blocks    Used Available Use% Mounted on
/dev/root      ext4      15392432 3664628  11048056  25% /
devtmpfs       devtmpfs    469540       0    469540   0% /dev
tmpfs          tmpfs       473872       0    473872   0% /dev/shm
```

```
tmpfs           tmpfs       473872    6432    467440   2% /run
tmpfs           tmpfs         5120       4      5116   1% /run/lock
tmpfs           tmpfs       473872       0    473872   0% /sys/fs/cgroup
/dev/mmcblk0p1 vfat          64456   20968     43488  33% /boot
tmpfs           tmpfs        94776       0     94776   0% /run/user/1000
/dev/sda1       ext2       3857220   43952   3617328   2% /media/pi/b1205542-46e9-4826-9438-5be3883f5261
```

In this example, an entry for `/dev/sda1` has now appeared. This is your thumb drive and its contents are in the directory `/media/pi/b1205542-46e9-4826-9438-5be3883f5261`. It has 3617328 kilobytes available.

The tab key will very helpful when doing the following commands – it usually does automatic completion of file names so that you do not need to type the long series of digits after `/media/pi`. You just type the first few characters and hit the tab key. If there is no ambiguity, it will be automatically completed.

Check the size of what is in your home directory in kilobytes:

```
du -sk /home/pi
39356   /home/pi
```

We see that this is less than the `3617328` kilobytes available in the thumb drive so there is ample room for our home directory contents.

The following commands make sure you are in your home directory and copy all of your home directory to the file `June-19-2016-home-pi.tar` in the directory `/media/pi/b1205542-46e9-4826-9438-5be3883f5261/` on the thumb drive. Using a date in the file name is quite helpful, but eventually will fill your drive with old backups and you will need to prune some of them.

```
cd /home/pi
sudo tar cf /media/pi/b1205542-46e9-4826-9438-5be3883f5261/June-19-2016-home-pi.tar .
```

The space and period at the end are essential; the period signifies that the current directory is the starting point for the copy.

The above step may take quite a while. After it completes, check that the size of `home.tar` is about right (It will be close, but not exactly the same.):

```
du -sk /media/pi/b1205542-46e9-4826-9438-5be3883f5261/June-19-2016-home-pi.tar
36252   /media/pi/b1205542-46e9-4826-9438-5be3883f5261/June-19-2016-home-pi.tar
```

You can also see a list of all the files in `pi.tar` by using the `tar -tf` command:

```
tar -tf /media/pi/b1205542-46e9-4826-9438-5be3883f5261/June-19-2016-home-pi.tar
```

For practice, delete the file /home/pi/Music/Midi/PlayingMidi by doing

```
cd /home/pi
rm Music/Midi/PlayingMidi
```

```
Confirm that it is gone
```

```
ls Music/Midi/PlayingMidi
ls: cannot access Music/Midi/PlayingMidi: No such file or directory
```

and then recover it from the thumb drive

```
tar -xf /media/pi/b1205542-46e9-4826-9438-5be3883f5261/June-19-2016-home-pi.tar ./Music/Midi/PlayingMidi
```

The "./" in front of Music is essential.

Beware, files brought in from the thumb drive will replace any with the same name in the specified directory.

You can confirm that it has reappeared by doing

```
ls Music/Midi/PlayingMidi
```

```
Music/Midi/PlayingMidi
```

`sudo` is *not* required here because you are writing the file into your own directory.

When done, you can unmount the thumb drive by using the `umount` command:

```
sudo umount /dev/sda1
```

or you can type a `sync` command, wait a few seconds, and remove the drive.

Still More New Commands

```
du -sk /home/pi                sudo mount                     tar xf
df -T                          sudo umount                    tar tf
rm                             sudo tar cf
```