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. Here is a list of a few of the most useful commands.

Some example commands

Click on the **desktop icon** named **LXTerminal** to open up a **command-line window**. At the left side of the top line, the LXTerminal 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`, 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).

The SD memory card

Nearly all of what the Raspberry Pi "knows" is stored on the tiny **micro 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 building is huge and has one huge room which might have some books, but which 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 of the bytes is used to represent a blank space between words in text documents.

In computer terminology, the buildings of the library are called "**drives**," the rooms are called "**directories**," and 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 holds about 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. Permission is requested by preceding a command by sudo. For example,

```
sudo vi /boot/config.txt
```

Only special users, like `pi` in our setup, are allowed this privilege.

Exploring the SD memory card

Typing the command `pwd`, which stands for "print working directory, lets you know where your are in the entire library building. When you first turn on the computer, `pwd` gives `/home/pi` which means that you are in the `pi` directory which in turn 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 of list. It will print out something like this:

```
bin      Documents  KF-BTJ    ocr_pi.png    PythonExamples  Sounds      Videos
Desktop  Music      Pictures  python_games  TypingTest      Wacom
```

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**) with the names `bin`, `Desktop`, `Documents`, `etc` are all shown in blue . The purple name `ocr_pi.png` is not a directory; it is actually a small picture file called an **icon**. The names of ordinary files are shown in white letters.

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

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: `BasicCommandLineActions.odt,` InstallationNotes, ...

The `ls` command can also be given the option `-l` to give more details. `ls -l Documents` produces

```
total 48
-rw-r--r-- 1 pi pi 92289 Nov 15 13:05 BasicCommandLineActions.odt
-rw-r--r-- 1 pi pi 13116 Nov 14 13:44 InstallationNotes
...
```

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**.

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 15 seconds later, you can turn off its power. Here, `sudo` is necessary in front of the `halt` because you are doing something that is risky and should not be done casually. 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 key operating system software.

When you started `LXTerminal` 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 21 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`. Going still deeper with `ls /home/pi/Music/Mp3`, we find directories named `AFA_Music` and `AFA_Music`, and a file named `playingMp3`. Finally, upon using the command `ls /home/pi/Music/Mp3/AFA_Music`, we see a list of music 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 the 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 can to do `cd /home/pi/Music/Mp3/AFA_Music`. Here, you *must* have a `/` before `home`. But there is a shortcut. Since you are already in `/home/pi`, you can just do `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 from the US Air Force Academy.

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. Try

```
omxplayer Videos/Badger.mp4
```

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
```

Notice here 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`. The `cd` command could have been written as `cd ~/Music/Midi`, or just `cd Music/Midi` since you are already in your home directory.

Finally, there are some short sound files in /home/pi/Sounds which end in `.au` or `.wav`. These are played by the command `asound`:

```
pi@raspberrypi ~ $: cd /home/pi/Sounds
pi@raspberrypi ~/Sounds $ asound train.au
pi@raspberrypi ~/Sounds $ asound 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` .

More commands:

```
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 in the lower left of the desktop. A simple graphical editor that produces text files is `leafpad`. 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/PythonExamples/powersOf2.py
```

and comparing with

```
leafpad /home/pi/PythonExamples/powersOf2.py
```

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

More advanced ways to create documents use `focuswriter` and `libreoffice`. `focuswriter` is fairly easy to learn and allows some font manipulation and document formatting. `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. Both `focuswriter` and `libreoffice` have spell checking capability.

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`.

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/Documents/myFirstFile` should type:

```
cd /home/pi/Documents
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.

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 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 done to the SD card.

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`**, **i**, your stuff, **Esc**, **: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 the copy buffer.

7dd	Delete current line and the 6 lines following it and put them all 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.

The typing practice program

If you type the command `TypingTest`, a screen will appear where you can type your initials. If you are new to the program, it also asks for your full name.

You can then select a test level from 1 to 15, and test duration in seconds which must be 30 or greater.

A set of words will then appear that must be typed correctly including spaces and capitalization. Use the backspace key to fix mistakes as soon as you notice them. If a mistake is not corrected by the end of a word, it turns red and you must fix it before continuing to the next word. When you reach the end of a line, another line will appear when you type the space after the last word.

When the time has expired, you will be presented with your speed, number of fixes, and number of errors. Fixes are when you correct a mistake, errors are when the program needed to alert you by turning the word red as you tried to move on to the next word without correcting a mistake. You are still required to fix the mistake.

Finally, you will be given an opportunity to see your top 20 results and a graph of all your results. This graph should provide encouragement since it shows your speed increase vs. total number of minutes of testing. The graph is closed by typing “q” and you will be asked if you wish to start over.

When it asks you to answer a yes/no question, it will show `(Y or n)` which means that typing a “`Y`”, “`y`”, or just hitting the enter key will select “`yes`”, but you must type an “`n`” to indicate “no”. (“`N`”, “`no`”, “`No`”, “`NO`”, or “`nope`” also work for “no”.) The capitalized choice, “`Y`” in this case, is taken as the default.

The words on level 1 contain only the 8 most common letters in the English language, each additional level adds new letters and finally punctuation and numbers.

Your results are saved in a file named with your initials followed by `.score` in the directory `/home/pi/TypingTest`.

A link to the `TypingTest.py` program is placed in `/home/pi/bin` which causes it to be usable simply by typing `TypingTest` anywhere in the directory system.

Kanji-Flash/BTJ – A flashcard program for learning technical words in Japanese

Typing `Kanjiflash` will start a DOS program I wrote in 1991 as a software flash card companion to the textbook *Basic Technical Japanese* written by Edward E. Daub, R. Byron Bird, Nobuo Inoue. It has some help information available in its top menu, but you should be somewhat familiar with Japanese to efficiently use the program.

The program files are in the directory `/home/pi/KF-BTJ` and a link to the program is in `/home/pi/bin` so that it can be easily run from anywhere in the directory. As you complete a section, a list of the words that gave you trouble is saved. The next time you try that section, those will be the words tested until you can complete the section with no errors. There are detailed instruction filed accessible from within the program.

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 upgrade`	This updates the programs on your system.
`sudo rpi-update`	This updates the basic processor firmware if necessary.
`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”, 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 or to just rename a file, use the `cp` command:

Set current directory to `home` directory.

```
cd
```

Make a copy.

```
cp Documents/myOriginal Documents/myOriginal.backup
```

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

Set current directory to home directory.

```
cd
```

Make a new directory in the `home/Documents` directory.

```
mkdir Documents/FinalDocuments
```

Move `Documents/myGreatNovel` to `Documents/FinalDocuments` directory.

```
mv Documents/myGreatNovel Documents/FinalDocuments/
```

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 use the `rmdir` command:

```
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.
```

More new commands

```
sudo                    cp                      rmdir
apt-get update          rm                      ls -a
apt-get 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` .

Useful command-line tricks

Alternatives: For historical reasons, when `vim` is installed, just typing `vi` gives the same editor program as when typing `vim`. `vim` is an alternative to an older editor named `vi` which is not installed.

Command completion: When typing in commands, use of the <tab> key often results in an automatic completion of the word. Try typing just the three letters `omx` at the command line prompt and then press the tab key. The line will automatically be completed to read `omxplayer`. This is true for all program names, but sometimes the system still doesn't know what you want. Try typing `vi` and then press the tab key. Nothing happens at first, but if you press the tab key again, you will be given a list of possible commands that start with `vi`. You then need to type more letters to make clear which one you need.

Command completion can often be used for names of files in you current working directory. Try doing the following:

```
cd ~/Music/Midi
timi <tab key> 2 <tab key>
```

and the second line will be appear as

```
timidity 2001SprachZarathustra-RichardStrauss.mid
```

Accessing and reusing previous commands: When at a new command prompt, you can use the up arrow key to see commands you have previously entered. When you find one you want to reuse, just hit the enter key.

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 all files which must be backed up with the files. The `tar` command is used to do this correctly.

Check to see what drives are currently attached:

```
ls /dev/sd*
```

Plug the thumb drive into a USB port and check again

```
ls /dev/sd*
```

If, for example `sda`, `sda1`, and `sg0` have appeared, you will be interested in `sda1`.

In the following, you need to be very careful about where leading `/` symbols are located or missing.

Make the current working directory be your home directory:

```
cd /home/pi
```

Check the size of what is in you home directory:

```
du -s /home/pi
```

Make a directory on which to hang the directory of the thumb drive:

```
sudo mkdir /media/sda1
```

`sudo` is required here because only the root user can make write to `/media`.

Hang the thumb drive directory on `/media/sda1`:

```
sudo mount /dev/sda1 /media/sda1
```

`sudo` is required here because only the root user can mount thumb drives to `/media/sda1`.

Confirm the the `sda1` drive is properly mounted and has enough room on the thumb drive:

```
df -T
```

The last line shown by this command should correspond to your thumb drive, `/dev/sda1`. The space available number should be greater than that shown by the `du` command above for the space required to hold `/home/pi`.

Make a directory in the thumb drive with the current date:

```
sudo mkdir /media/sda1/Nov-18-2014
```

Copy all of your home directory to that directory as a `.tar` file. The space followed by a period at the end here represents your current working directory which you have earlier made to be your home directory:

```
sudo tar cf /media/sda1/Nov-18-2014/pi.tar .
```

Check that the size of `home.tar` is about right:

```
ls -l /media/sda1/Nov-18-2014/pi.tar
```

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

```
tar -tf /media/sda1/Nov-18-2014/pi.tar
```

If, for example, you need to recover a file `/home/pi/Documents/MyGreatNovel` from `pi.tar`, you must change the current working directory to be your home directory:

```
cd /home/pi
```

and then extract the file from `pi.tar` and put it back in place. Be sure to use `./` in front of Documents here.

```
tar -xf /media/sda1/Nov-18-2014/pi.tar ./Documents/MyGreatNovel
```

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

The file should have appeared where it was originally. **The one from the thumb drive will replace any file of the same name in the specified directory.**

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

```
sudo umount /media/sda1
```

Because this backup procedure keeps adding new files without removing any, the thumb drive will eventually become full. You will then need to learn how to safely remove entire directories to clean out the older backups.

Still More New Commands

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

Computer programming

What makes computers so useful is that their behavior can be controlled by a series of instructions. Our body is controlled by the instructions in our DNA and computers are controlled by numbers in their memory. It is possible, to edit those numbers directly, but that process is very slow and extremely error-prone. Instead, we write English language instruction files that the computer could convert to the necessary numerical instructions. These files are what we normally refer to as computer programs.

When writing them, we must obey specific rules just as proper English writing requires that certain rules of grammar and spelling are obeyed. The rules for writing a computer program are defined by a computer programming language. One difference, however, is that a computer language must be extremely precise. We often say that a computer is a fast, obedient moron that must told exactly what to do. If you make an error in writing a computer program, it will either not work at all or may work, but not do what you intended.

The Python programming language

There are many different programming languages, but the best computer programming language for a beginner is the Python language. You can start out doing simple things, but eventually do very serious scientific calculations and robot control with Python.

Here is a simple Python program that prints out the greeting “Hello, world!” to the screen:

```
#! /usr/bin/env python3
print('Hello, World!')
```

To use this program, we must write those lines into a file with a suitable name, tell the computer that the file is a program, and then tell the computer to run the program.

First, create a directory for your computer programs:

```
pi@raspberrypi ~ $ mkdir ~/MyPrograms
pi@raspberrypi ~ $ cd ~/MyPrograms
pi@raspberrypi ~/MyPrograms $ vim HelloWorld.py
```

In the `vim` editor, you start by typing the letter “i” to get into INSERT mode.
The `vim` editor is set to provide line numbers, so you just need to type the following two lines:

```
#! /usr/bin/env python3
print('Hello, World!')
```

You then leave INSERT mode by hitting the <Esc> key.
And finally save and quit by typing `:wq` followed by the <Enter> key.
Next you let the computer know that this is a program file by doing:

```
pi@raspberrypi ~/MyPrograms $ chmod +x HelloWorld.py
```

Before you did this the `ls` command would show the file HelloWorld.py in white, afterwards the `ls` command will show it as green. This lets you easily know which files are programs since their names are green when shown by the `ls` program.

Next, you run your program by typing `./` and the program name:

```
pi@raspberrypi ~/MyPrograms $ ./HelloWorld.py
Hello, world!
```

If you are not in the ~/MyPrograms directory, you can run the program by typing:

```
pi@raspberrypi ~/TypingTest $ ~/MyPrograms/HelloWorld.py
```

You can display your program using the cat command:

```
pi@raspberrypi ~/TypingTest $ cat ~/MyPrograms/HelloWorld.py
#! /usr/bin/env python3
```

```
print('Hello, world!')
```

The second line in the program is pretty obvious, it does the printing. The stuff to be printed must be put inside of a pair of quotes, single quotes `'Hello, world!'` or double quotes `"Hello, world!"` and that must be enclosed in parentheses after the word `print`.

The first line tells the computer how it should run your program, that it should use version 3 (currently version 3.2 on the Raspberry Pi) of the python interpreter to figure out what you want it to do. That line should be the same for nearly all your programs although sometimes it might need to say `#! /usr/bin/env python2.7 .`

An excellent way to learn computer programming is to start messing with a working program.

For example, you could edit it to say `Hello, Dad!` , or perhaps `Dad, look at what my program does now!`

Or you could replace the word `print` with `write` and see what happens. (Hint. It doesn't know about the word `write` and will give an error message.

Or you could add an extra line that says `Goodbye for now.` so the program would look like

```
#! /usr/bin/env python3
print('Hello, world!')
print('Goodbye for now.')
```

and would print the following when run:

```
Hello, world!
Goodbye for now.
```

If you were to write a program and forget the initial line that says `#! /usr/bin/env python3` , it would say the following when you tried to run it:

```
./HelloWorld.py: line 1: syntax error near unexpected token `'Hello, world!''
./HelloWorld.py: line 1: `print('Hello, world!')'
```

Or if you forgot to do `chmod +x HelloWorld.py` to tell the computer that it is a program, it will print the following error message when you try to run it:

```
-bash: ./HelloWorld.py: Permission denied
```

If you leave off the quote marks around `HelloWorld` in the line that says `print('Hello, world!')` , you will get the following error when you try to run the program:

```
  File "./HelloWorld.py", line 2
    print(Hello, world!)
                       ^
SyntaxError: invalid syntax
```

When doing computer programming, you must be very careful to follow the rules. For Python, the rules are at the following web site:

https://docs.python.org/3.3/

The tutorial at that site is very useful.

The typing practice program `~/TypingTest/TypingTest.py` is written in Python and can be modified, but before doing so, you should copy the entire `TypingTest` directory into `~/MyPrograms` using the following command:

```
cp -a ~/TypingTest ~/MyPrograms
```

The `-a` option of the `cp` command copies the entire directory and its contents, not just one file. You can now play with that copy of `TypingTest` by doing

```
cd ~/MyPrograms/TypingTest
vim TypingTest.py
```
(change something and quit the `vim` editor)

While you are still in the directory `~/MyPrograms/TypingTest`, you can run your changed copy by doing `./TypingTest.py`

If you just type `TypingTest`, without the preceding `./`, you will run the unchanged version because the link in `~/bin` points to the original in the `~/TypingTest` directory.

The Raspberry Pi has a set of electrical contacts called GPIO pins (General Purpose Input/Output) that let it sense and control hardware. It is best to learn some basic electricity, know how to use a multimeter, avoid static electrical charges, and to be able to read and understand specifications for electronic components before experimenting with those pins since it is easy to make an unrecoverable mistake that can damage your Raspberry Pi. Fortunately, the worst case problem is that you must replace it at a cost of about $35 whereas doing that sort of experimentation with a normal computer is likely to cost hundreds of dollars.

More new commands

```
chmod +x
cp -a
```