
Exploring the Raspberry Pi Documentation

Release 0.1

Amit Saha

Nov 13, 2017

Contents

1	Installing Fedora Linux	3
1.1	Configuration	4
1.2	Setting up for remote access	7
1.3	Web-based Access	8
1.4	Updating and Adding RPM fusion	10
1.5	Time Synchronization	10
2	Know your Raspberry Pi	11
2.1	Exploring /proc	11
2.2	USB Devices	12
2.3	Mount points and mounting disks	12
3	Programming	15
3.1	Python	15
3.2	Web application using Flask	17
3.3	Ruby	19
3.4	Web application using Sinatra	20
3.5	CouchDB	20
3.6	Daemonizing applications	20
3.7	GPIO Pins	20
4	Half-baked Projects with Full Potential	23
4.1	Fedora Infrastructure Bus Scraper	23
4.2	Network Sniffer	24
4.3	Zer0mq Broadcaster	24
5	Next Things to do	25
6	Miscellaneous	27
6.1	Source for the book	27
6.2	Contact	27

You have been using Linux on your big computers so far and want a light reading about exploring the Raspberry Pi? Or, you are just learning Linux with the Raspberry Pi? Hop on. This guide is for you.

However, this guide is aimed at wannabe power users of the Raspberry Pi. You won't even see the window manager or the GUI if you follow this guide. All work is performed on the Pi in a headless manner - you ssh into your Pi. If that puts you off, you are better off fetching any of the other guides.

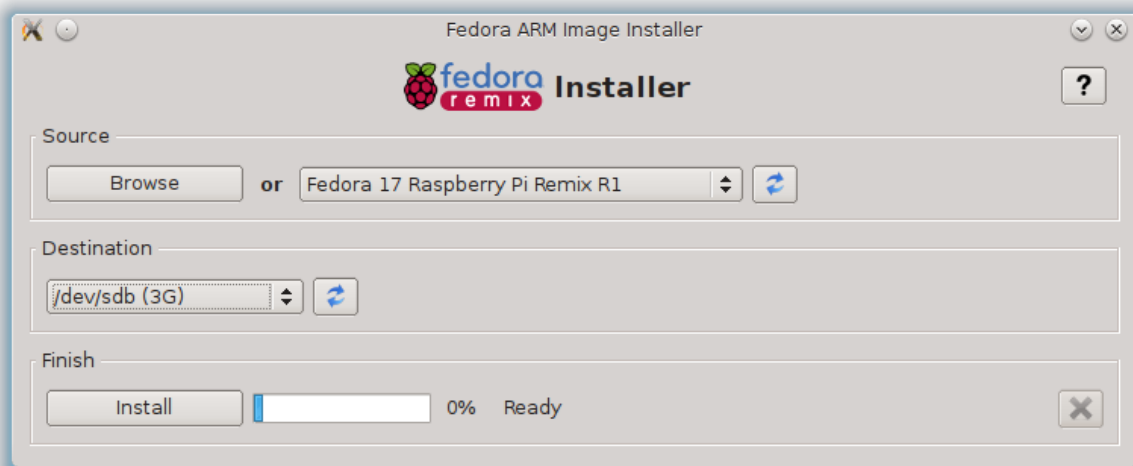
Let's get started.

CHAPTER 1

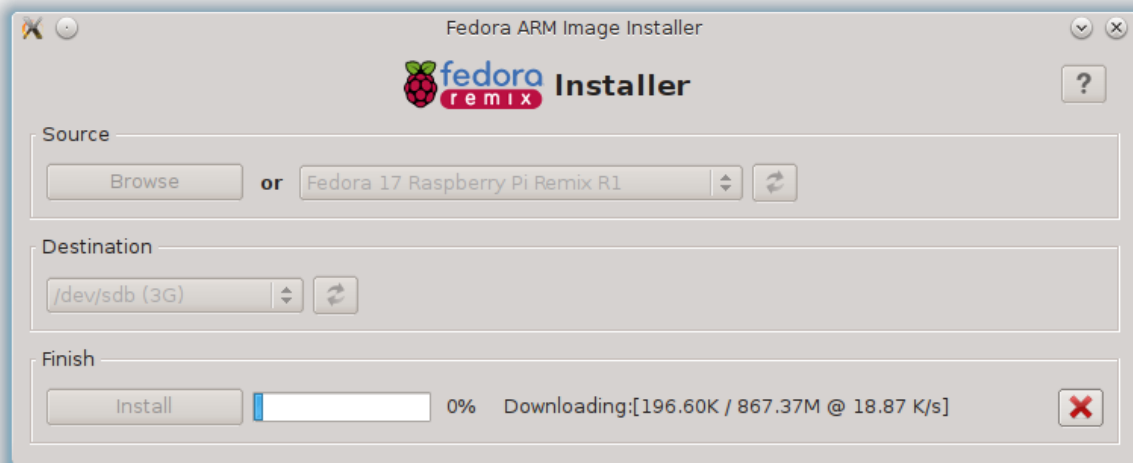
Installing Fedora Linux

Start from [here](#).

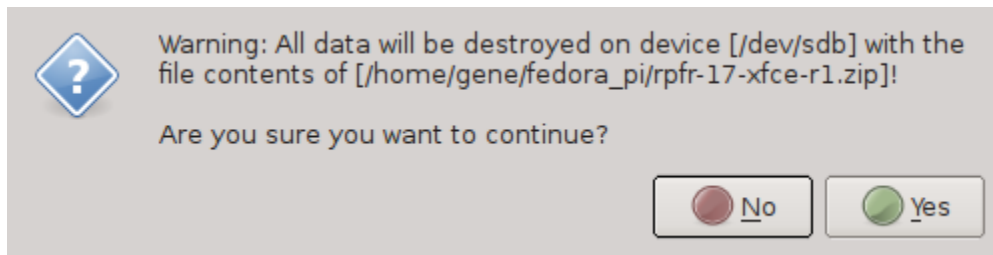
Installer:



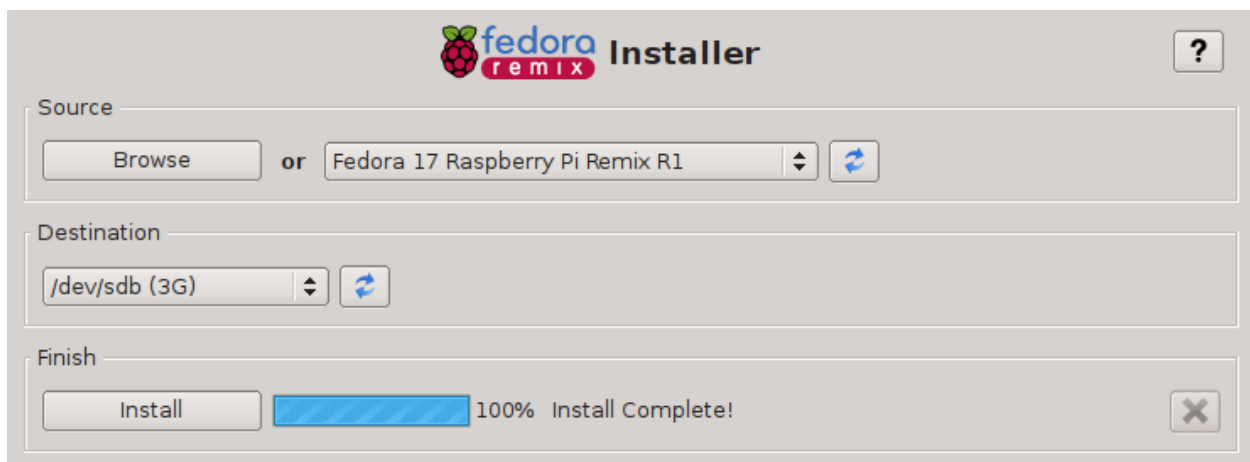
Downloading image:



Write to the disk:

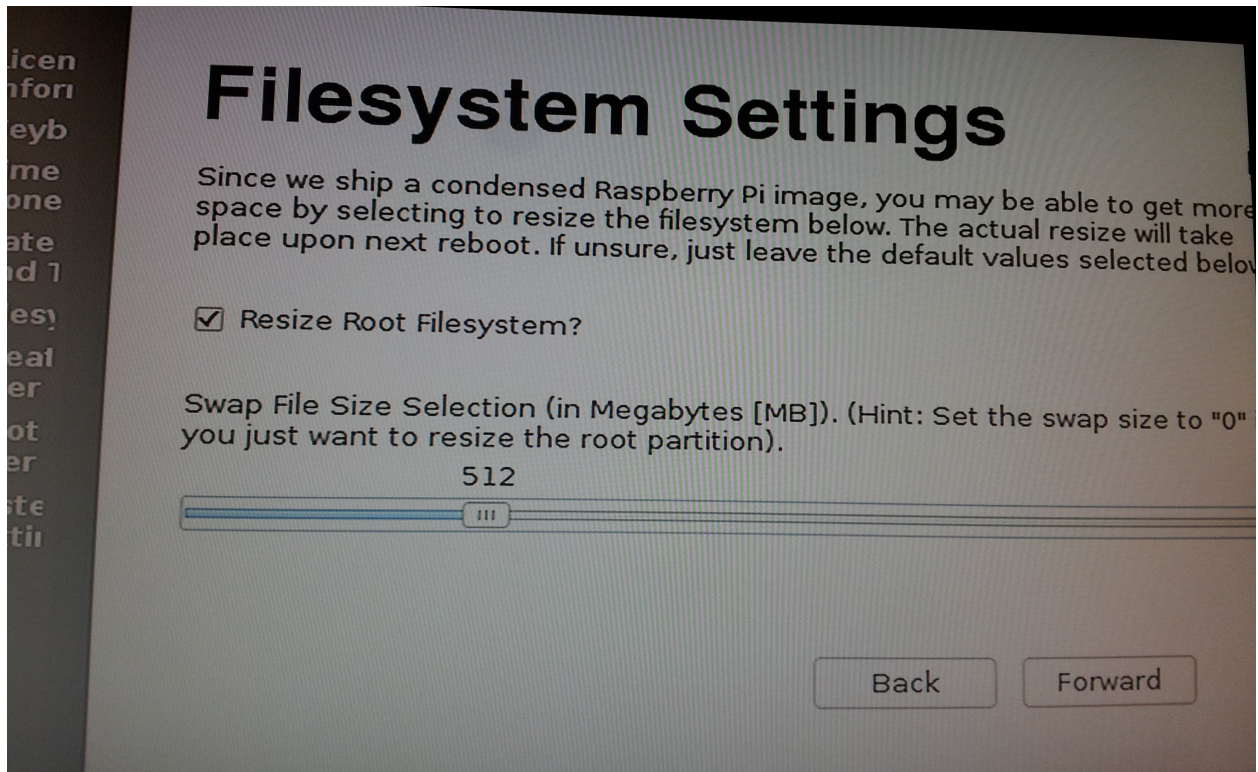


Install complete:

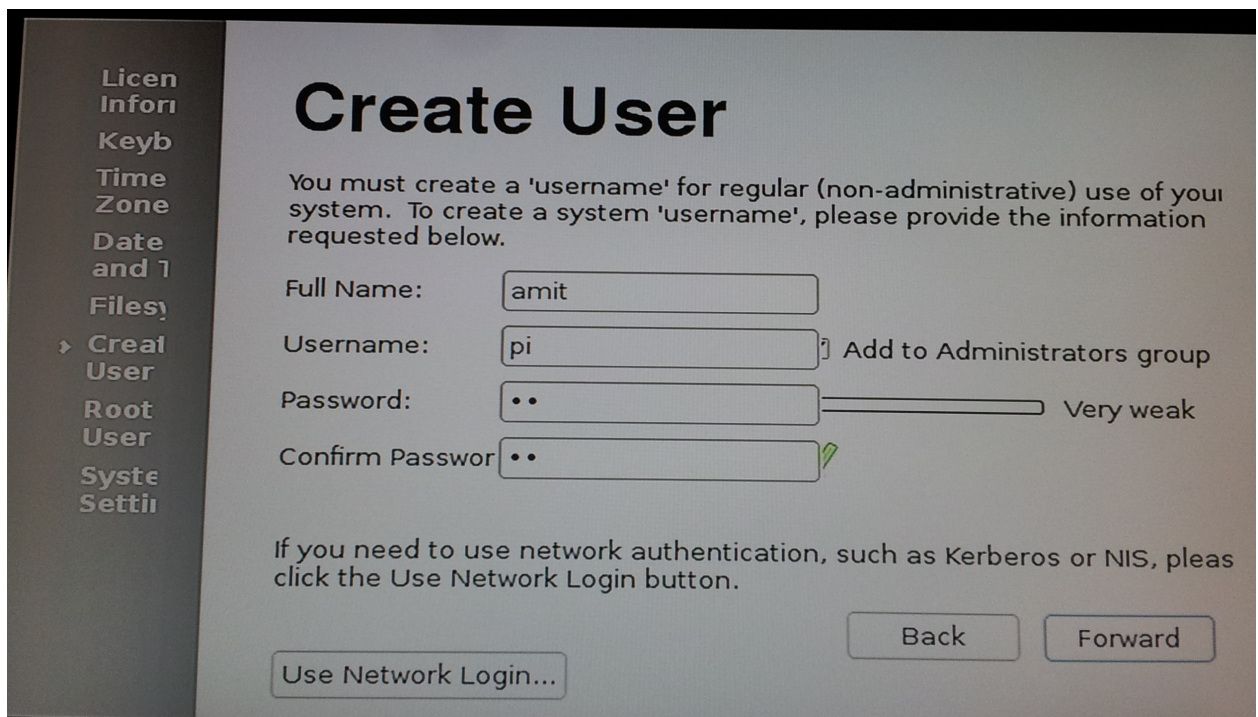


1.1 Configuration

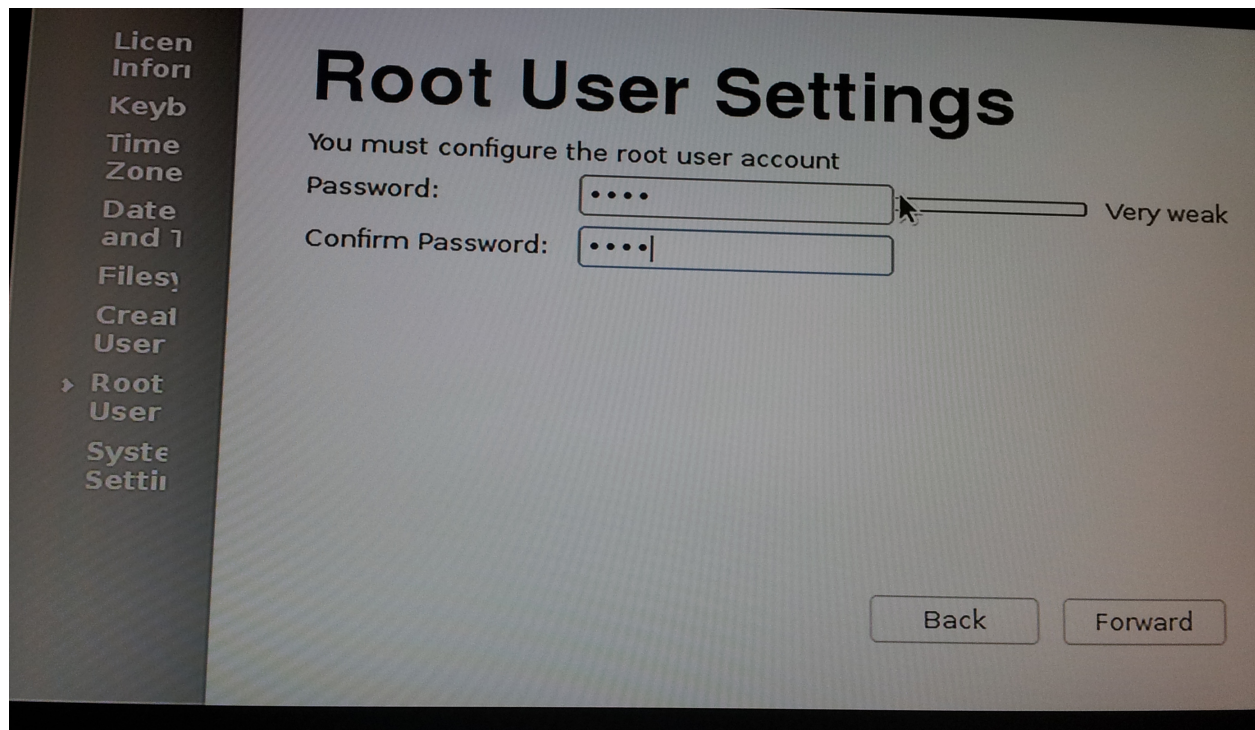
Root file system resize:



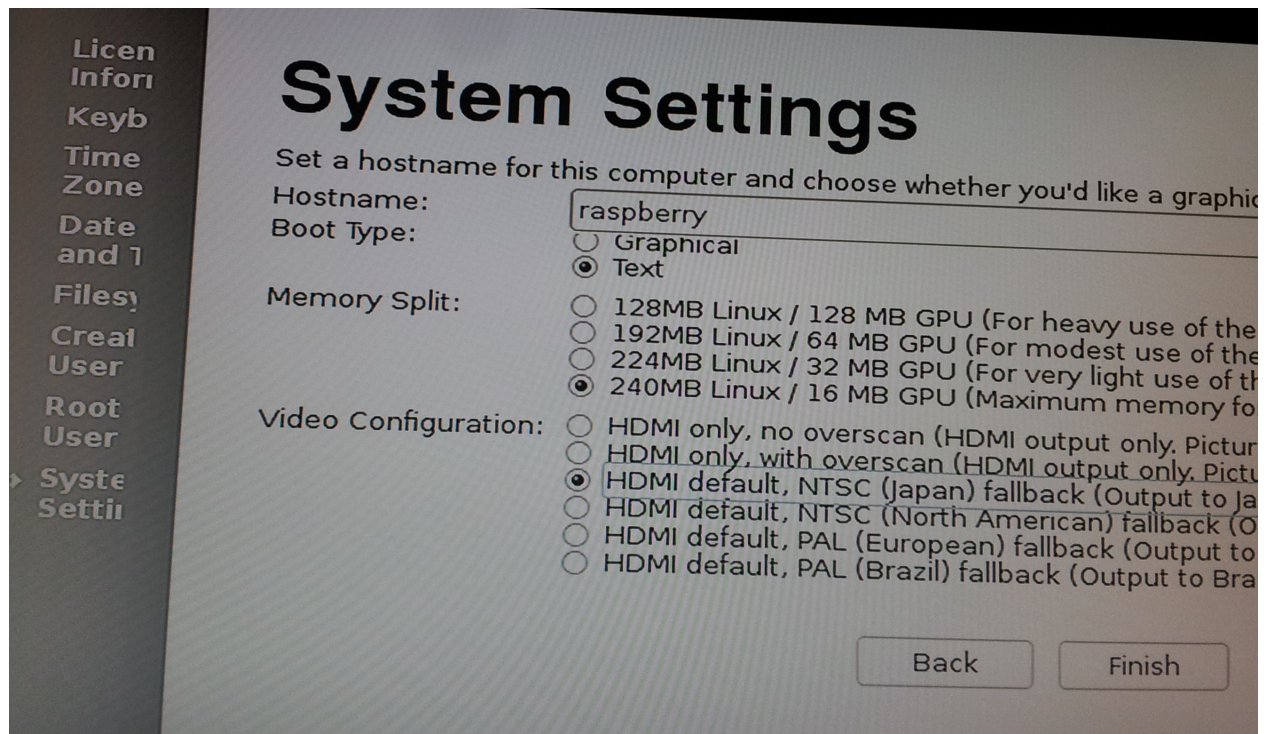
Create user:



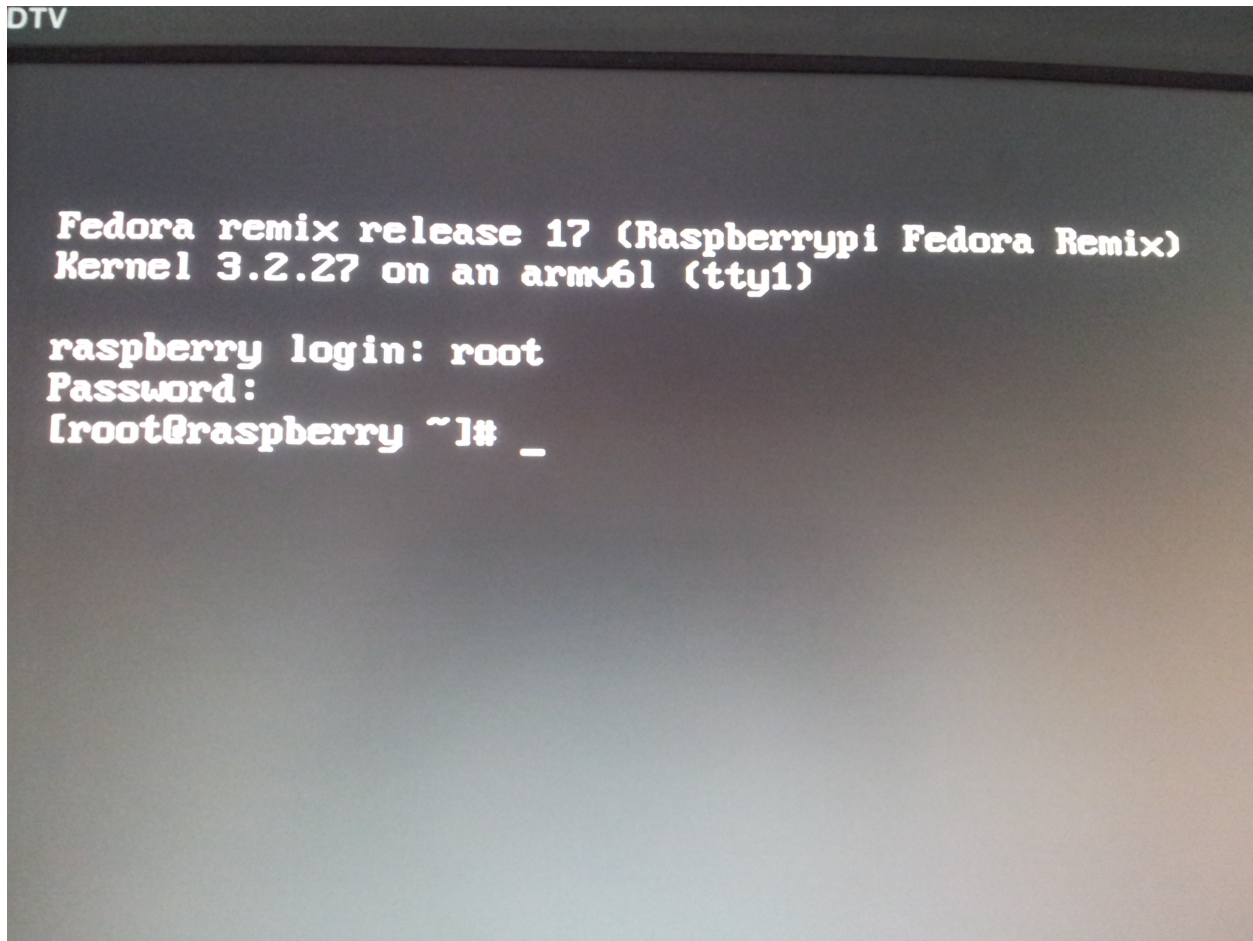
Root user:



System Settings:



Login:



1.2 Setting up for remote access

SSH server should already be installed and started in your Fedora installation. To check:

```
# service sshd status
Redirecting to /bin/systemctl status  sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Thu, 01 Nov 2012 12:06:23 +1000; 9h ago
     Main PID: 223 (sshd)
    CGroup: name=systemd:/system/sshd.service
            223 /usr/sbin/sshd -D
```

and then:

```
# ssh localhost
root@localhost's password:
Last login: Thu Nov  1 21:13:07 2012 from zion.gateway
# who
root    pts/0      2012-11-01 21:13  (zion.gateway)
root    pts/3      2012-11-01 22:07  (localhost)
```

We are setup for remote login now. So you may now disconnect the TV jack from your Raspberry Pi. The network

cable must stay on.

1.3 Web-based Access

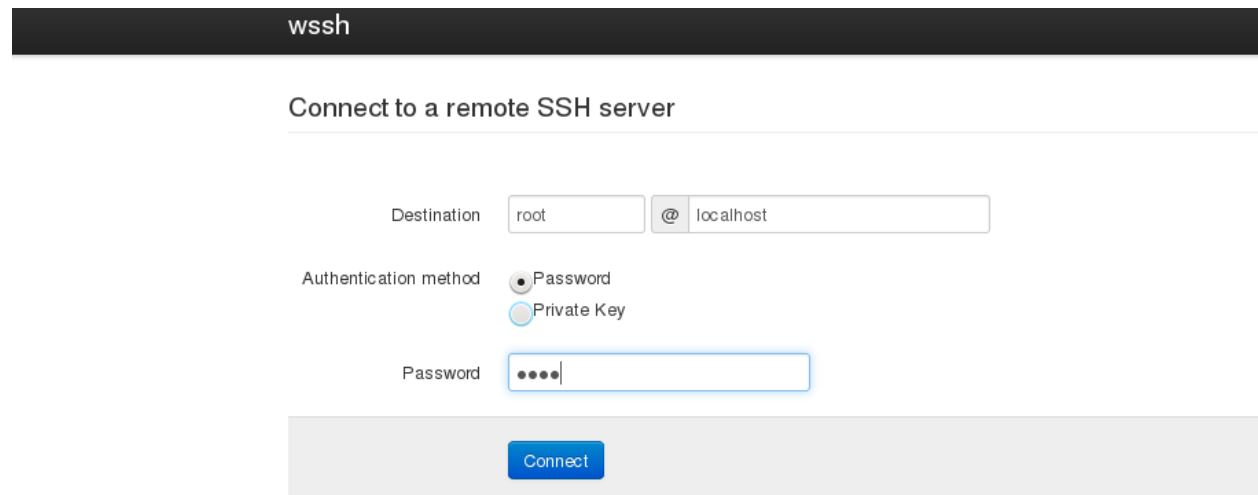
Using `wssh` you can SSH into your Pi using a browser. Here's how. Install the dependencies

```
# yum -y install python-devel libevent-devel python-setuptools
# pip-python install gevent flask paramiko gevent-websocket wssh
```

Now, start `wsshd`

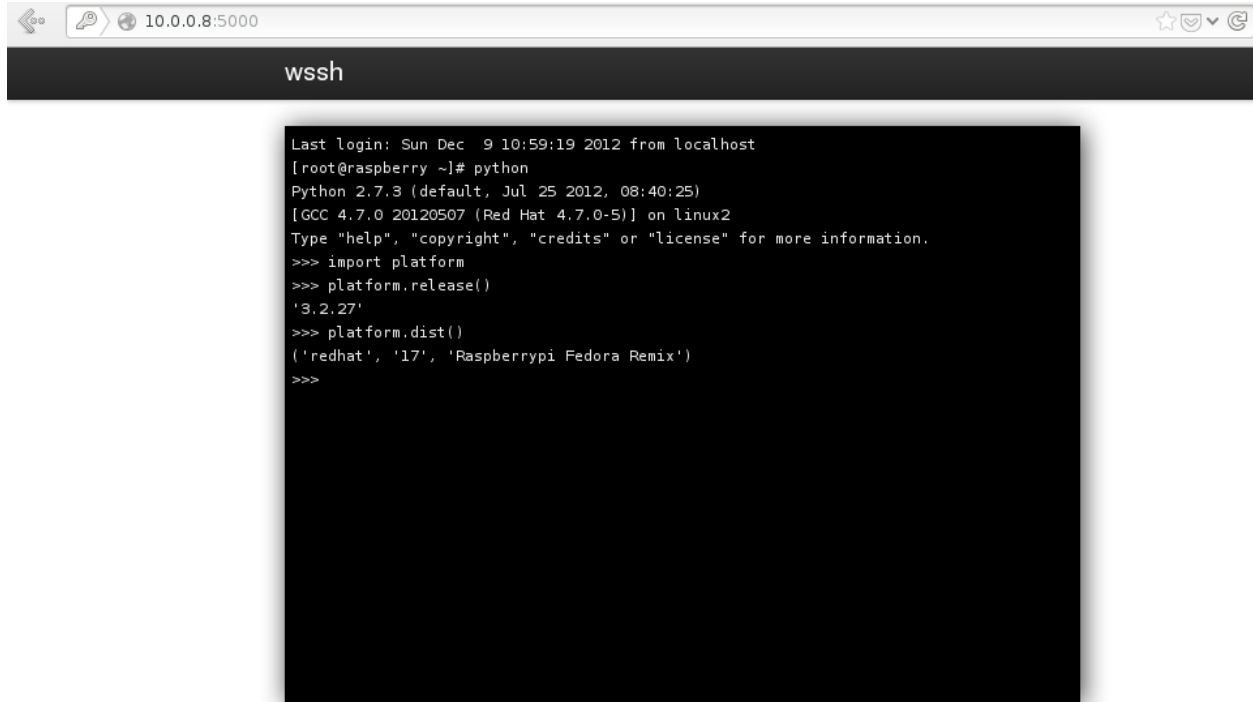
```
# wsshd
wsshd/0.1.0 running on 0.0.0.0:5000
```

Now, from any other computer on your network, visit the URL, `<raspi-ip>:5000` and you will see a login screen where you can enter the username, hostname/IP you want to connect to and use the password/private key authentication to connect to the remote host. Since the `wssh` daemon is running on your Pi, specifying `localhost` would mean the Pi itself and use the IP address for any other computer on your network. The login screen looks like this:



The screenshot shows a web browser window with the title "wssh". The main heading is "Connect to a remote SSH server". Below this, there are three input fields: "Destination" with "root" entered, a separator "@", and another field with "localhost" entered. Underneath is the "Authentication method" section with two radio buttons: "Password" (which is selected) and "Private Key". Below that is a "Password" field with four dots indicating masked text. At the bottom of the form is a blue "Connect" button.

And then you can use the terminal, like you would normally do:



It is a good idea to run `wsshd` in the background using a daemonizer program like `zdaemon` (more on this later). Install `zdaemon` using

```
# pip-python install zdaemon
```

Run `wsshd` using `zdaemon`

```
# zdaemon -p wsshd start
. . .
daemon process started, pid=5277
```

Effectively, this gives you an entry point to your local home network from the Web browser. To make this possible outside your home network and being able to access it from anywhere, literally, you can setup tunnelling. You can either play with your router configuration to set up port forwarding or simply use a tunnelling service such as [PageKite](https://pagekite.net/). Install it using

```
# curl -s https://pagekite.net/pk/ |sudo bash
```

Before starting to use this service, you should make yourself familiar with the pricing of the service.

Before you can start using the service, you have to sign up using

```
# pagekite.py --signup
```

Once you are done with that, you can setup your Pi for exposing the 5000 port using

```
# pagekite.py 5000 mypi.pagekite.me
```

And you should have your SSH login screen at your chosen kitename: <http://mypi.pagekite.me> in this case.

1.4 Updating and Adding RPM fusion

Install RPMFusion repositories:

```
# yum localinstall --nogpgcheck
  http://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-stable.noarch.rpm
  http://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-stable.
↪noarch.rpm
```

Update system:

```
# yum -y update
```

Kernel version:

```
# uname -r
3.2.27
```

1.5 Time Synchronization

The Pi does not have a real time clock, which means the Pi cannot keep track of time across reboots. The solution is to run the ntpd daemon and use a NTP server to synchronize the time everytime you reboot.

Install the ntpd and ntpdate utilities:

```
# yum -y install ntp ntpdate
```

Although, the default NTP servers works, you may want to edit the /etc/ntp.conf to include NTP servers closer to your geographic [location](#).

And start the ntp service

```
# service ntp start
```

The date and time should now be reflected to be your local time. To start the ntp service on boot

```
# chkconfig ntpd on
```

You will see that few seconds after the boot, the time is always set to the local time.

If you have the need to add one, see Adafruit's [tutorial](#).

Know your Raspberry Pi

We shall use simple Linux commands to know more about the Raspberry Pi.

2.1 Exploring /proc

The /proc file system is the place to look for all sorts of information. For example:

```
# cat /proc/cpuinfo
Processor      : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS      : 697.95
Features       : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xb76
CPU revision   : 7

Hardware      : BCM2708
Revision     : 0003
Serial        : 00000000f90cbb6b
```

The output tells us that we are on a system with an ARM processor with a BogoMIPS of 697.95. Among the CPU features..

You can obtain the information about the memory usage from the file /proc/meminfo:

```
# cat /proc/meminfo
MemTotal:      234968 kB
MemFree:       56840 kB
Buffers:       10380 kB
Cached:        126004 kB
..
..
```

Various other information about your system can be obtained such as:

```
# cat /proc/sys/kernel/hostname
raspberrypi
# cat /proc/sys/kernel/ostype
Linux
# cat /proc/sys/kernel/osrelease
3.2.27
# cat /proc/sys/kernel/pid_max
32768
# cat /proc/sys/kernel/poweroff_cmd
/sbin/poweroff
```

2.2 USB Devices

To explore the USB devices connected to your Pi, we can use the `lsusb` command

```
# lsusb -t

/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/lp, 480M
   |__ Port 1: Dev 2, If 0, Class=hub, Driver=hub/3p, 480M
      |__ Port 1: Dev 3, If 0, Class=vend., Driver=smc95xx, 480M
```

The above output corresponds to no USB devices connected to the Pi. If you connect any other device(s), the command's output will change accordingly. For example, when I plugged in two external storage disks, here is the what the output corresponds to

```
# lsusb -t

/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/lp, 480M
   |__ Port 1: Dev 2, If 0, Class=hub, Driver=hub/3p, 480M
      |__ Port 1: Dev 3, If 0, Class=vend., Driver=smc95xx, 480M
      |__ Port 2: Dev 6, If 0, Class=stor., Driver=usb-storage, 480M
      |__ Port 3: Dev 5, If 0, Class=stor., Driver=usb-storage, 480M
```

2.3 Mount points and mounting disks

The `df` and `mount` commands can be used to obtain information about the secondary storage disks attached to your Raspberry Pi:

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          3.7G  2.1G  1.4G  61% /
/dev/root        3.7G  2.1G  1.4G  61% /
devtmpfs        115M    0  115M    0% /dev
tmpfs           115M    0  115M    0% /dev/shm
tmpfs           115M  1.1M  114M    1% /run
tmpfs           115M    0  115M    0% /sys/fs/cgroup
tmpfs           115M    0  115M    0% /media
/dev/mmcblk0p1   51M   16M   36M   31% /boot

# mount
/dev/root on / type ext4 (rw,noatime,user_xattr,barrier=1,data=ordered)
devtmpfs on /dev type devtmpfs (rw,relatime,size=117396k,nr_inodes=29349,mode=755)
```

```
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
..
..
```

The mount command is also used to mount external disks. Insert your external USB disk and note the output of dmesg

```
# dmesg

[79827.845946] usb 1-1.3: new high-speed USB device number 5 using dwc_otg
[79827.947579] usb 1-1.3: New USB device found, idVendor=1058, idProduct=1111
[79827.947611] usb 1-1.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[79827.947628] usb 1-1.3: Product: My Book 1111
[79827.947641] usb 1-1.3: Manufacturer: Western Digital
[79827.947655] usb 1-1.3: SerialNumber: 574341563535323139373832
[79827.954851] scsi0 : usb-storage 1-1.3:1.0
[79828.953242] scsi 0:0:0:0: Direct-Access      WD           My Book 1111      1032 PQ: 0_
↪ANSI: 4
[79828.954734] scsi 0:0:0:1: CD-ROM           WD           Virtual CD 1111    1032 PQ: 0_
↪ANSI: 4
[79828.956757] scsi 0:0:0:2: Enclosure         WD           SES Device        1032 PQ: 0_
↪ANSI: 4
[79828.966489] sd 0:0:0:0: [sda] 1952151552 512-byte logical blocks: (999 GB/930 GiB)
[79828.968510] sd 0:0:0:0: [sda] Write Protect is off
[79828.968547] sd 0:0:0:0: [sda] Mode Sense: 23 00 10 00
...
[79829.068725] sda: sda1
[79829.075284] sd 0:0:0:0: [sda] No Caching mode page present
[79829.095857] sd 0:0:0:0: [sda] Assuming drive cache: write through
[79829.102133] sd 0:0:0:0: [sda] Attached SCSI disk
```

The device that you inserted corresponds to the device file `/dev/sda`. Now let us see what partitions are there on this device

```
# fdisk -l /dev/sda

Disk /dev/sda: 999.5 GB, 999501594624 bytes
255 heads, 63 sectors/track, 121515 cylinders, total 1952151552 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002ae3f

Device Boot      Start         End      Blocks   Id  System
/dev/sda1        2048     952151551    976074752    7  HPFS/NTFS/exFAT
```

As you can see, there is only one partition on this disk. If you again run the mount command, you will see that this partition hasn't yet been mounted. The mount command can be used for this purpose

```
# mount -t ntfs /dev/sda1 /var/disk1/
```

If you now rerun the mount command, you will see this line in the output

```
/dev/sda1 on /var/disk1 type fuseblk (rw,relatime,user_id=0,group_id=0,allow_other,
↪blksize=4096)
```

Now, you can read/write from this disk by going to `/var/disk1`. One drawback of this mechanism is that you will have to do this manually everytime you restart the Pi. Assuming that you will have the disk always connected to your Pi, you can add the following entry to your `/etc/fstab` file

/dev/sda1	/var/disk1	ntfs	defaults	↵
↵ 0 0				
/dev/sdb1	/var/disk2	ntfs	defaults	↵
↵ 0 0				

I have another disk connected to the Pi and hence I add both entries to the file. Save the file and you will see that when you reboot your Pi, these disks will be ready to use.

Now you are ready to devise a file sharing/backup solution using one of the various possible mechanisms.

CHAPTER 3

Programming

Fedora Linux for the Raspberry Pi already comes with Python, Ruby and Perl installed

```
# python --version
Python 2.7.3

# ruby --version
ruby 1.9.3p194 (2012-04-20 revision 35410) [armv5tel-linux]

# perl --version

This is perl 5, version 14, subversion 2 (v5.14.2) built for arm-linux-thread-multi
Copyright 1987-2011, Larry Wall
```

Vi is installed

```
# vi --version
VIM - Vi IMproved 7.3 (2010 Aug 15, compiled Aug 29 2012 04:04:56)
```

3.1 Python

Getting to know your system using some of the standard library modules

```
# python
Python 2.7.3 (default, Jul 25 2012, 08:40:25)
[GCC 4.7.0 20120507 (Red Hat 4.7.0-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> import platform
>>> platform.linux_distribution()
('Fedora remix', '17', 'Raspberrypi Fedora Remix')
```



```
>>> import os
>>> os.uname()
('Linux', 'raspberrypi', '3.2.27', '#1 PREEMPT Mon Oct 1 22:37:41 UTC 2012', 'armv6l')
>>> os.name
'posix'
>>> os.getlogin()
'root'
>>> os.getpid()
30378
>>> os.getppid()
29788
>>> os.pathsep
':'
>>> os.sep
 '/'
>>> os.linesep
 '\n'
>>> os.devnull
 '/dev/null'

>>> import sys
>>> sys.platform
linux2
>>> sys.byteorder
'little'
```

Use the subprocess module to access Linux commands

```
>>> import subprocess

>>> subprocess.call(['cat', '/proc/cpuinfo'])
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 697.95
Features        : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xb76
CPU revision    : 7

Hardware       : BCM2708
Revision      : 0003
Serial        : 00000000f90cbb6b
0

>>> subprocess.call(['ifconfig'])
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.0.0.8  netmask 255.255.255.0  broadcast 10.0.0.255
inet6 fe80::ba27:ebff:fe0c:bb6b  prefixlen 64  scopeid 0x20<link>
ether b8:27:eb:0c:bb:6b  txqueuelen 1000  (Ethernet)
RX packets 44678  bytes 41925811 (39.9 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 27311  bytes 4467866 (4.2 MiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 16436
inet 127.0.0.1  netmask 255.0.0.0
```

```
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 0 (Local Loopback)
RX packets 9366432 bytes 468467070 (446.7 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9366432 bytes 468467070 (446.7 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
0
```

The `pylinux` package aims to provide a Python interface to the most common system information on a Linux system. Install it by cloning it from the project website and using the `setup.py` script to install the package:

```
# git clone https://github.com/amitsaha/pylinux.git
# python setup.py install
```

You can then use it, like so:

```
>>> import pylinux.pylinux as pylinux
>>> pylinux.distro_name()
'Fedora remix'
>>> pylinux.arch()
'armv6l'
>>> pylinux.freemem()
'44356 kB'
```

Install pip

```
# yum -y install python-pip
```

3.2 Web application using Flask

Flask can be used to create web applications on your Pi. Its simple, lightweight and really easy to get started.

Install Flask using

```
#yum -y install python-flask
```

Let us now write an example web application using Flask. Create a file `app.py` with the following contents

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():

    return ''' <center><h1> Flask on Raspberry Pi </h1></center> \n
           Time now ''' + time.ctime()

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

Now start the web application using:

```
# python app.py
```

You should see the following message

```
* Running on http://0.0.0.0:5000/  
* Restarting with reloader
```

Now, if you visit the URL <http://raspi-ip:5000> from your browser, you will see the following page (where raspi-ip is the IP address of your Raspberry Pi which you can find out as we saw earlier)

Flask on Raspberry Pi

Time now Sun Nov 4 02:44:39 2012

The `@app.route("/")` indicates that this is the function which will be called when you visit the URL: <http://raspi-ip:5000>. You could write your web application such that when you visit the URL: <http://raspi-ip:5000/uptime>, it gives you the time for which your Raspberry Pi has been switched on. Rewrite the `app.py` file as

```
import pylinux.pylinux as pylinux  
  
from flask import Flask  
import time  
  
app = Flask(__name__)  
  
@app.route("/uptime")  
def uptime():  
    return 'Raspberry Pi up for %s hours' % str(pylinux.uptime())  
  
@app.route("/")  
def index():  
  
    return ''' <center><h1> Flask on Raspberry Pi </h1></center> \n  
        Time now ''' + time.ctime()  
  
if __name__ == "__main__":  
    app.run(host='0.0.0.0', debug=True)
```

As you can see, we use the `pylinux` package to get the uptime in hours. Run the web application and you will see that when you visit the URL: <http://raspi-ip:5000/uptime> in your browser, you will get back the uptime for your Raspberry Pi.

Raspberry Pi up for 2.36551111111 hours

The `pylinux` package includes a Flask web application which uses the package to display system statistics. It also uses jQuery and smoothie.js to display dynamic data. It can be found in the `examples/flask_app` directory.

3.3 Ruby

The Process class's methods can be used to get useful information about the Ruby interpreter

```
irb(main):002:0> Process.pid()
=> 31224
irb(main):003:0> Process.ppid()
=> 29788
```

We can use Kernel's system method to execute external Linux commands and get useful information

```
irb(main):001:0> system('whoami')
root

irb(main):013:0> system('uname -a')
Linux raspberry 3.2.27 #1 PREEMPT Mon Oct 1 22:37:41 UTC 2012 armv6l armv6l armv6l_
↳ GNU/Linux
=> true

irb(main):002:0> system('cat /proc/cpuinfo')
Processor      : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS       : 697.95
Features       : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xb76
CPU revision   : 7

Hardware       : BCM2708
Revision       : 0003
Serial         : 00000000f90cbb6b
=> true

irb(main):003:0> system('ifconfig')
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.0.8 netmask 255.255.255.0 broadcast 10.0.0.255
inet6 fe80::ba27:ebff:fe0c:bb6b prefixlen 64 scopeid 0x20<link>
ether b8:27:eb:0c:bb:6b txqueuelen 1000 (Ethernet)
RX packets 71698 bytes 72360222 (69.0 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 40831 bytes 5926797 (5.6 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 16436
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 0 (Local Loopback)
RX packets 11999249 bytes 600570906 (572.7 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 11999249 bytes 600570906 (572.7 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

=> true
```

Fedora 17 comes with gem installed

```
# gem --version
1.8.24
```

3.4 Web application using Sinatra

3.5 CouchDB

Install couchdb and couchdb library for Python

```
# yum -y install couchdb
# pip-python install couchdb
```

Start Couchdb

```
# service couchdb start
```

Create a simple database and store data

```
>>> import couchdb
>>> couch = couchdb.Server('http://127.0.0.1:5984/')
>>> db = couch.create('test')
>>> doc={'name':'raspi'}
>>> db.save(doc)
('2a94bdde4f092c50be2ec8ab68000baa', '1-07a810f328653abedf230bb8321d3d4c')
>>> doc
{'_rev': '1-07a810f328653abedf230bb8321d3d4c', '_id':
↳ '2a94bdde4f092c50be2ec8ab68000baa', 'name': 'raspi'}
>>> db['2a94bdde4f092c50be2ec8ab68000baa']
<Document '2a94bdde4f092c50be2ec8ab68000baa'@'1-07a810f328653abedf230bb8321d3d4c' {
↳ 'name': 'raspi'}>
```

3.6 Daemonizing applications

Install zdaemon

```
# pip-python install zdaemon
```

3.7 GPIO Pins

The command line utility gpio can be used to access the GPIO pins on the Raspberry Pi. This command is provided by the wiringpi package which is already installed on Fedora

```
# rpm -q wiringpi
wiringpi-1-4.rpfr17.armv5tel
```

To get the basic idea of accessing the GPIO pins using gpio, setup a simple LED circuit. Connect the anode of the LED to the GPIO pin 18 (pin 12 in real life) and the cathode to the Ground pin (6 in real life) via a resistor. The LED will not glow. We will now use gpio utility to switch on the LED

```
# gpio -g mode 18 out
# gpio -g write 18 1
```

The LED should now be switched on. You can switch it off using

```
# gpio -g write 18 0
```

Next, we switch to Python for GPIO control using the RPi.GPIO package (<http://pypi.python.org/pypi/RPi.GPIO>). Install gcc and python-devel packages first and then install it using pip-python

```
# yum -y install gcc python-devel
# pip-python install RPi.GPIO
```

Use RPi.GPIO to switch on/off the LED

```
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(18, GPIO.OUT)
>>> GPIO.output(18, GPIO.HIGH)
>>> GPIO.output(18, GPIO.LOW)
>>> GPIO.output(18, GPIO.HIGH)
```

Now, connect another LED such that its anode is connected to GPIO pin 23 (pin 14 in real life). Now that we have two LEDs with each capable of being in an ON or an OFF state, we can use them to display numbers (upto 3 in decimal) as their binary equivalent:

```
import sys
import RPi.GPIO as GPIO

# GPIO pins
pins = [18,23]

def setup_gpio():

    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    for pin in pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)

if __name__=='__main__':

    if len(sys.argv) == 1:
        print 'Usage: dec2bin_led.py <decimal integer>'
        sys.exit()

    dec = int(sys.argv[1])
    if dec < 0 or dec > 2**len(pins)-1:
        print 'Please enter a decimal integer between 0 and %s (both inclusive)' \
            % str(2**len(pins)-1)
        sys.exit()

    binary = bin(dec).lstrip('0b')

    setup_gpio()

    for i,bit in enumerate(binary):
        if bit=='1':
```

```
        GPIO.output(pins[i], GPIO.HIGH)
    else:
        GPIO.output(pins[i], GPIO.LOW)
```

Run the program as follows

```
# python dec2bin_led.py 1
# python dec2bin_led.py 3
# python dec2bin_led.py 10
Please enter a decimal integer between 0 and 3 (both inclusive)
```

Depending on the decimal integer you enter, you will see none, one or both the LEDs glowing. You can also increase the number of LEDs and update the list of pins accordingly to increase the range of decimal input.

You could also consider writing a web application using Flask to control your LEDs via HTTP.

Half-baked Projects with Full Potential

4.1 Fedora Infrastructure Bus Scraper

Install python-zmq

```
# yum -y install python-zmq
```

A client program

```
#!/usr/bin/env python
'''
fedbus_client.py

Fedora Infrastructure Messaging Bus consumer
without using fedmsg.

Install python-zmq to use this.

Developed upon Ralph Bean's example code at
http://threebean.org/blog/zeromq-and-fedmsg-diy/
'''
import json
import pprint
import zmq

def listen_and_print():
    # You can listen to stg at "tcp://stg.fedoraproject.org:9940"
    endpoint = "tcp://hub.fedoraproject.org:9940"
    # Listen to all messages being published
    topic = 'org.fedoraproject.'

    ctx = zmq.Context()
    s = ctx.socket(zmq.SUB)
    s.connect(endpoint)
```



```
s.setsockopt(zmq.SUBSCRIBE, topic)

poller = zmq.Poller()
poller.register(s, zmq.POLLIN)

while True:
    evts = poller.poll() # This blocks until a message arrives
    topic, msg = s.recv_multipart()
    print topic, pprint.pformat(json.loads(msg))

if __name__ == "__main__":
    listen_and_print()
```

When you run this program, you may have to wait, but will eventually see messages like

```
org.fedoraproject.prod.git.receive.jack-audio-connection-kit.master
{'certificate':
u'LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUVSekNDQTdD
.....
.....
Z0
u'i': 1,
u'msg': {'commit': {'branch': u'master',
u'email': u'brendan.jones.it@gmail.com',
u'message': u'Add audio group in README dropped in merging.\n',
u'name': u'Brendan Jones',
u'rev': u'b81b6f46a94c9603c5f1b199f22ec7d2589bfe07',
u'stats': {'files': {'jack-audio-connection-kit-README.Fedora': {'deletions': 1,
u'insertions': 1,
u'lines': 2}},
u'total': {'deletions': 1,
u'files': 1,
u'insertions': 1,
u'lines': 2}},
u'summary': u'Add audio group in README dropped in merging.',
u'username': u'bsjones'}}},
u'signature': u
↪ 'vNqmuPiwvVSEQGqL3Q4dlMGQ1VfSJmaB5yTjCg3K0M+GVgvvZVX7D1LbZjKeYpM2zyd893px4LDF\nt7kustfGPXFfN156Tmt
↪ WQTqAH9QnpbFX93PAIupLeKEcM16\nUIweYYhtj/DCEB85pZ0=\n',
u'timestamp': 1353227420.312382,
u'topic': u'org.fedoraproject.prod.git.receive.jack-audio-connection-kit.master'}
```

4.2 Network Sniffer

Libpcap using ctypes

```
# yum -y install libpcap libpcap-devel
```

4.3 Zer0mq Broadcaster

CHAPTER 5

Next Things to do

6.1 Source for the book

The book and the source is available [here](#) under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported license. See LICENSE file for the complete description.

6.2 Contact

Errors/suggestions/corrections may be emailed to me at amitsaha.in@gmail.com

Other places:

- [Blog](#)
- [Twitter](#)
- [ReadTheDocs](#)

Very much a work under construction. Comments/suggestions welcome.