Part 04
-
Base Configuration
Of Your Raspberry Pi

Version: 2020-12-23

Upfront security information

The default user name for Raspbian is **pi** with its password **raspberry**
The root password is disabled.

When using SSH (port 22), select an Xterm terminal in color, set white on black as screen colors, use UTF-8 as character set.

Basic Configuration (keyboard, Memory split, SSH, ...)

With the launch of Raspbian Jessie there is one significate difference users are going to notice, and that's the fact that your Pi will boot straight up into a GUI (instead of the CLI, like Raspbian Jessie).

That's great, but what if you just simply prefer the Command Line Interface, or don't want the added overhead of the Graphical User Interface? Well you can quite easily tell your Pi to boot to CLI. Here's how:

Start off by turning your Pi on and letting it boot to the GUI
Once booted, you'll want to click on "Menu" (top left of the screen)
then "Preferences"
and finally "Raspberry Pi Configuration"

This will load up the raspi-config menu. From the "System" tab, you can simply click the radio button next to "To CLI" to change the boot preference. Hit "OK" and then "YES" to reboot.

The Raspberry Pi will reboot, but this time you'll be presented with the all too familiar CLI login!

If you want to change back to booting to the GUI, you can still change it using raspi-config

Please note that Raspbian start with the 102key US QWERTY keyboard layout loaded !!! See last page for the layout of such a keyboard.

Now type at command prompt

```
sudo raspi-config
```

and change, as needed, the basic configuration for your region & language

1. Expand Filesystem
2. Internationalisation Options
    a. change locale
    b. timezone
    c. keyboard layout

Now reboot

```
sudo reboot
```

After reboot, type again at command prompt

```
sudo raspi-config
```

and change the other settings as required. Consider to change the user password right now
If you are going to add a camera, enable camera here
Check out also the advanced options. See suggestions a bit further
Consider to re-organize its memory. The Raspberry Pi has a "memory_split" option that divides the memory between the CPU and GPU.

- If you use it headless, you don't need that much GPU (Graphics) memory so change it to 16 or 32M.
- If you are going to use a camera, you need at least 128.
- If you are going to use Flask, you need trim this down to 32 if you have a model A or B with only 512MB of RAM.

Set or change your screen resolution. Edit this file

```
sudo nano /boot/config.txt
```

Inside of the file, there are a few lines commented out with hashes. If you want to change this, uncomment and change as required

```
#console-width=1024
#console-height=700
```

When you have finished press [Ctrl] + X. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

If you have already connected your Raspberry Pi to your network, find your IP address

```
hostname -I
```

or use

```
ifconfig
```

where you can see also its MAC address. Eg. Eth0 = wired network interface

```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:5a:a5:0b
```

After finishing all the changes, reboot.

```
sudo reboot
```

Note: If you are using an wired ethernet connection (with a router) there should be no configuration required and your PI should connect to your network out of the box.

Suggestions

1. Change the default password

   Changing the default password on your Raspberry Pi is important, especially if you're connected to the internet. Since SSH is configured by default (more on this later), anyone with access to your network could take full control of your Raspberry Pi. Your first level of defence in this regard is the password. Either use the 'Change User Password' option in raspi-config or on the command line, simply type

   ```
   passwd
   ```

   You'll be asked to enter your current password, which by default is set to `raspberry`. Hit Return again and you'll be asked for your new password.
   You'll need to enter it twice, hitting Return both times, to set it correctly.

2. Set a new hostname

   By default all Raspberry Pis are called `raspberrypi`. In Linux-speak this is its hostname, not to be confused with your user name, which defaults to `pi`.
   Either use the 'Advanced Options' -> 'Hostname' option in raspi-config or at the command prompt, you can view your hostname by typing

   As it is with all Debian-based systems your hostname can be permanently changed by editing a file called `hostname`.

   ```
   sudo nano /etc/hostname
   ```

   simply use the arrow keys on your keyboard to jump to the end of the word '`raspberrypi`' so you can delete and replace it with whatever hostname you'd like.
   When you have finished press [Ctrl] + X. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

   Now change also `/etc/hosts`, so local address(es) resolves with the new system name.

   ```
   sudo nano /etc/hosts

   127.0.1.1       <hostname>
   ```

   When you have finished press [Ctrl] + X. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

   A reboot is required for the change to take effect.

3. SSH into your Raspberry Pi (go headless)

   SSH stands for Secure Shell. It's a secure means of accessing the command line interface of your Raspberry Pi from a separate computer anywhere in the world but usually from within your own home network.
   Using SSH you can update your Pi, tweak settings or run command line tools and applications in a 'headless' way = without the need for a monitor, keyboard or mouse to be attached.
   Since it's already configured by default, we'll demonstrate how to do it, but also how to make finding your Pi on any network infinitely easier than it is by default…
   Let's say you have another Linux or Mac computer and you want to access your Raspberry Pi from. Assuming your Pi is powered up and connected to the same network as your computer, all you'd need to do is type a simple command into the terminal window of your other computer to access it:

   ```
   ssh pi@192.168.0.9
   ```

   Typing SSH tells your computer you want to create a Secure Shell connection, 'pi' is the user name of the system you wish to access, and the numbers that follow is the IP address of the system you want to access.
   It's quite straightforward, but it does assume you already know the IP address. In this world of dynamic IP assignment, it's not always easy to know without defeating the object of running 'headlessly', but setting up your Raspberry Pi and typing

   ```
   ifconfig
   ```

   into a terminal window to find out what the IP address is in the first place.

   Recent versions of Raspbian, which use dhcpcd, allow ssh to work over a link-local address and avahi (which is a zeroconf implementation) enables programs to discover hosts running on a local network. This means you can plug the Pi into a computer with an Ethernet cable or a local network router and connect without knowing the IP address.

You can easily connect from Linux and OS X with ssh pi@hostname.local (the default hostname is raspberrypi) This should work with popular GUI ssh programs. This is sometimes problematic with some versions of Windows and networks which use .local in a non-standard way. (See https://en.wikipedia.org/wiki/.local)
NOTE .local resolution does not always work e.g. in rsync. The following should resolve IP (and can be included in bash scripts)
If you have multiple Pi you need to make sure each Pi has a unique hostname.

While it's relatively easy to set a static IP address for your Pi, or easier still to reserve the IP address for the Pi on your router's settings page, why bother?! You can actually connect to your Pi by name, not number, circumventing all the looking up, writing down and generally mucking around you otherwise need to do.
To achieve this all you need is a little tool installed via the command line like so:

```
sudo apt –y install avahi-daemon
```

Since you already know your Raspberry Pi's hostname, assuming both the computer and the Raspberry Pi are on the same network you now just need to type the following to SSH into your Pi:

```
ssh pi@hostname.local
```

Again, we type SSH to tell your computer you wish to make a Secure Shell connection and '`pi`' to denote the username you'd like to connect as, but instead of having to worry about IP addresses, we simply replace 'hostname' in the above command with the actual hostname of your Pi and append the `.local`.

Once you've hit Return, you'll be asked to type 'yes' if you want to continue. Then you'll be asked to input your Raspberry Pi's password. Once complete you'll be met by the same command prompt you'd see if you were sat in front of your Raspberry Pi.

Once you've finished doing whatever you want with your Raspberry Pi, you can end the SSH session by typing Control + D.
Accessing your Raspberry Pi via SSH on a Windows computer is a little different since you'll need to use a dedicated piece of software like Putty
http://www.chiark.greenend.org.uk/~sgtatham/putty/. That said, it's very easy to set-up and use once installed.
With the Avahi daemon set-up as we've demonstrated you can also enjoy IP address-less access to your Raspberry Pi in just about any network-aware application you like, not just SSH!

Give root access via SSH

If you want to login as root using SSH or WinSCP you need to edit the config of SSHD, do this:

- Login, and edit this file:

```
sudo nano /etc/ssh/sshd_config
```

- Find this line

```
PermitRootLogin without-password
```

  and change it to

```
PermitRootLogin yes
```

- Close and save file. Reboot or restart sshd service using:

```
/etc/init.d/ssh restart
```

```
or
```

```
service ssh restart
```

- Set a root password if there isn't one already:

```
sudo passwd root
```

Now you can login as root, but I recommend you using a strong password or ssh-keys

How to check what Pi board Revision you have

There have now been a number of revisions to the Raspberry Pi PCB so the device you have in front of you could be one of a number of variants. The changes include mounting holes, modifications to the power supply circuitry, different GPIO headers and varying numbers of USB ports. The Pi 2 introduced a new CPU and additional memory.
The variants currently available are:

In order to find out what hardware revision you have you can run the following command at the command prompt or via a terminal window :

```
cat /proc/cpuinfo
```

This will give you a text output something like this :

```
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 847.05
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        : 0002
Serial          : 000000000abc0ab1
```

In this example I've got a PCB with a Revision code of 0002. That is a plain old "Model B Revision 1.0".

Note : The Revision number given in cpuinfo file is the hardware revision number. This is not the same as the Raspberry Pi Revision. In this example I have a Revision 1.0 with a hardware revision code of 0002. This is not a Revision 2 board!
The differences between the board revisions are minor but it now makes sensible to identify which board revision you are using when creating hardware or software.
Important Note: As of the 4.9 kernel, all Pi's report BCM2835 as Hardware, even those with BCM2836 and BCM2837 processors. You should not use this string to detect the processor.

Old-style revision codes

The first set of Raspberry Pi revisions were given sequential hex revision codes from 0002 to 0015:

Code	Model	Revision	RAM	Manufacturer
0002	B	1.0	256MB	Egoman
0003	B	1.0	256MB	Egoman
0004	B	2.0	256MB	Sony UK
0005	B	2.0	256MB	Qisda
0006	B	2.0	256MB	Egoman
0007	A	2.0	256MB	Egoman
0008	A	2.0	256MB	Sony UK
0009	A	2.0	256MB	Qisda
000d	B	2.0	512MB	Egoman
000e	B	2.0	512MB	Sony UK
000f	B	2.0	512MB	Egoman
0010	B+	1.2	512MB	Sony UK
0011	CM1	1.0	512MB	Sony UK
0012	A+	1.1	256MB	Sony UK
0013	B+	1.2	512MB	Embest
0014	CM1	1.0	512MB	Embest
0015	A+	1.1	256MB/512MB	Embest

New-style revision codes

With the launch of the Raspberry Pi 2, new-style revision codes were introduced. Rather than being sequential, each bit of the hex code represents a piece of information about the revision:

`uuuuuuuuFMMMCCCCPPPPTTTTTTTTRRRR`

Part	Represents	Options
uuuuuuuu	Unused	Unused
F	New flag	1: new-style revision 0: old-style revision
MMM	Memory size	0: 256MB 1: 512MB 2: 1GB 3: 2GB 4: 4GB
CCCC	Manufacturer	0: Sony UK 1: Egoman 2: Embest 3: Sony Japan 4: Embest 5: Stadium
PPPP	Processor	0: BCM2835 1: BCM2836 2: BCM2837 3: BCM2711
TTTTTTTT	Type	0: A 1: B 2: A+ 3: B+ 4: 2B 5: Alpha (early prototype) 6: CM1 8: 3B 9: Zero a: CM3 c: Zero W d: 3B+ e: 3A+ f: Internal use only 10: CM3+ 11: 4B
RRRR	Revision	0, 1, 2, etc.

Resulting in

Code	Model	Revision	RAM	Manufacturer
900021	A+	1.1	512MB	Sony UK
900032	B+	1.2	512MB	Sony UK
900092	Zero	1.2	512MB	Sony UK
900093	Zero	1.3	512MB	Sony UK
9000c1	Zero W	1.1	512MB	Sony UK
9020e0	3A+	1.0	512MB	Sony UK
920092	Zero	1.2	512MB	Embest
920093	Zero	1.3	512MB	Embest
900061	CM	1.1	512MB	Sony UK
a01040	2B	1.0	1GB	Sony UK
a01041	2B	1.1	1GB	Sony UK
a02082	3B	1.2	1GB	Sony UK
a020a0	CM3	1.0	1GB	Sony UK
a020d3	3B+	1.3	1GB	Sony UK
a21041	2B	1.1	1GB	Embest
a22042	2B (with BCM2837)	1.2	1GB	Embest
a22082	3B	1.2	1GB	Embest
a220a0	CM3	1.0	1GB	Embest
a32082	3B	1.2	1GB	Sony Japan
a52082	3B	1.2	1GB	Stadium
a22083	3B	1.3	1GB	Embest
a02100	CM3+	1.0	1GB	Sony UK
a03111	4B	1.1	1GB	Sony UK
b03111	4B	1.1	2GB	Sony UK
c03111	4B	1.1	4GB	Sony UK
d03114	4B	1.4	8GB	Sony UK
c03130	Pi 400	1.0	4GB	Sony UK

Update and upgrade your Pi's firmware

You can determine the revision of your current firmware with the following command:

```
$ uname –a
```

The firmware revision is the number after the #:

```
Linux kermit 3.12.26+ #707 PREEMPT Sat Aug 30 17:39:19 BST 2014 armv6l GNU/Linux
                        /
                       /
  firmware revision --+
```

On Raspbian, the standard upgrade procedure should keep your firmware up to date:

```
sudo apt update
sudo apt upgrade
```

To update and upgrade to the (optional) change(s) from the older Raspbian desktop/user-interface to the current layout/behaviour etc.:

```
sudo apt install raspberrypi-ui-mods
```

Note: Previously, documents have suggested using the `rpi-update` utility to update the Pi's firmware; this is now discouraged. If you have previously used the `rpi-update` utility to update your firmware, you can switch back to using `apt` to manage it with the following commands:

```
sudo apt update
sudo apt install --reinstall libraspberrypi0 libraspberrypi-{bin,dev,doc}
raspberrypi-bootloader
sudo rm /boot/.firmware_revision
```

You will need to reboot after doing so.

```
sudo reboot
```

Update and upgrade your Pi's OS

```
sudo apt -y update
sudo apt –y dist-upgrade
sudo reboot
```

or as of Raspbian Buster

```
sudo apt -y update
sudo apt –y full-upgrade
sudo reboot
```

Notes:

`upgrade`

upgrade is used to install the newest versions of all packages currently installed on the system from the sources enumerated in /etc/apt/sources.list. Packages currently installed with new versions available are retrieved and upgraded; under no circumstances are currently installed packages removed, or packages not already installed retrieved and installed. New versions of currently installed packages that cannot be upgraded without changing the install status of another package will be left at their current version. An update must be performed first so that apt knows that new versions of packages are available.

`Dist-upgrade / full-upgrade`

dist-upgrade in addition to performing the function of upgrade, also intelligently handles changing dependencies with new versions of packages; apt has a "smart" conflict resolution system, and it will attempt to upgrade the most important packages at the expense of less important ones if necessary. So, dist-upgrade command may remove some packages. The /etc/apt/sources.list file contains a list of locations from which to retrieve desired package files. See also apt_preferences(5) for a mechanism for overriding the general settings for individual packages.

Less writing

To prevent your Raspberry Pi's from writing a lot of data, and thus, wearing the SD card, you can do a couple of things.

The first one is to mount a few folders in RAM as `tmpfs`. The folders are the folders where temp files and logging is written to. This means that you won't have syslog available, but most of the time that is not a problem.
Edit `/etc/fstab` and add the following:

```
tmpfs  /tmp      tmpfs defaults,noatime,nosuid,mode=1755,size=100m   0      0
tmpfs  /var/tmp  tmpfs defaults,noatime,nosuid,mode=0755,size=10m    0      0
tmpfs  /var/log  tmpfs defaults,noatime,nosuid,mode=0755,size=10m    0      0
tmpfs  /run      tmpfs defaults,noatime,nosuid,mode=0755,size=100m   0      0
```

This will mount the above folders in RAM, with a max size of 1 megabyte. The `noatime` option means that the access time of a file is not updated, saving a lot of writes as well. You should also add the `noatime` option to your other partitions, for example on a standard Raspbian:

```
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    ro,noatime        0       2
/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
```

Here the `/boot` partition is also mounted read only (`ro`). The `noatime` option is added.
Issue a `mount -a` command or reboot the machine to make this active.

Linux divides its physical RAM (random access memory) into chucks of memory called pages. Swapping is the process whereby a page of memory is copied to the preconfigured space on the hard disk, called swap space, to free up that page of memory. The combined sizes of the physical memory and the swap space is the amount of virtual memory available.
Swappig causes a lot of writes to the SD card. You would want to turn it off to save writes. The downside of this is that when there is not enough RAM available the linux OOM killer will randomly kill processes to save RAM.
Raspbian by default has a swap file, dynamically managed by the `dphsys-swapfile` utility. You can turn off this utility by issueing the following commands:

```
dphys-swapfile swapoff
dphys-swapfile uninstall
update-rc.d dphys-swapfile remove
```

After a reboot the swap will be gone, which you can check with the `free -m` command:

```
             total       used       free     shared    buffers     cached
Mem:           484        243        241          0         42        162
-/+ buffers/cache:         38        446
Swap:            0          0          0
```

My Raspberry Pi's have a cronjob which reboots them once every seven days. This to apply kernel updates and just a general good procedure to see if all still works after a reboot. By default, `fsck` checks a filesystem every 30 boots (counted individually for each partition). I decided to change this to every boot, so problems will be found and possibly fixed earlier.
To set up an `fsck` at every boot, execute the following command:

```
tune2fs -c 1 /dev/mmcblk0p2
```

Where `/dev/mmcblk0p2is` the Linux Raspbian partition.

Update or upgrade Python 2 & 3

```
sudo apt -y install python-dev
sudo apt -y install python-setuptools
sudo apt -y install python-pip
sudo apt -y install python-tk

sudo apt -y install python3-dev
sudo apt -y install python3-setuptools
sudo apt -y install python3-pip
sudo apt -y install python3-tk

sudo reboot
```

Add some interesting and usefull modules for your Pi

```
sudo apt -y install avahi-daemon
sudo apt -y install ca-certificates git-core
sudo apt -y install git

sudo reboot
```

How to check what RPI.GPIO version you have

This works for all versions of RPi.GPIO

```
find /usr | grep -i gpio | grep –i egg-info
```

And the output will look something like this

```
...
/usr/lib/python2.7/dist-packages/RPi.GPIO-0.6.2.egg-info
...
/usr/lib/python3/dist-packages/RPi.GPIO-0.6.2.egg-info
...
```

You can see all those lines with **0.6.2.egg-info** telling me I have version 0.6.2.

You can also check the RPi.GPIO version in Python (works from 0.4.1a onwards)

It's quite useful to be able to find out what version of RPi.GPIO you have using Python. Some features require a recent version, whereas others work with older versions. For example, interrupts were added at 0.5.1, PWM at 0.5.2 and with 0.5.3 some bugs have been squashed. You don't have to write or run any programs to find out which version you have. You can do it very quickly in a live Python session. In the command line, type…

```
sudo python

Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the next line at the >>> prompt

```
>>> import RPi.GPIO as GPIO
```

which loads the RPi.GPIO library. Watch out for RPi.GPIO. **The i in RPi, is lowercase.** All the rest is uppercase. Get it wrong and it won't work. So we can use it

```
>>> GPIO.VERSION
```

this returns the version number of RPi.GPIO

```
'0.6.2'
>>>
```

`[Ctrl]+D` ends the python session cleanly.

Installing, updating or upgrading RPi.GPIO module

To install or upgrade the RPi.GPIO module, run following:

```
sudo apt –y install python-rpi.gpio python-pigpio
sudo apt –y install python3-rpi.gpio python3-pigpio
```

Extra security : change root password?

If you buy Raspberry Pi and download Raspbian image then put it on your SD card, you will be allowed to login using username **`pi`** and password **`raspberry`**.

Using sudo will enable you to do all you want. If you are annoyed to use sudo every time, and would like to have root access then, do below

```
sudo vipw
```

change line

```
root:x:0:0:root:/root:/bin/bash
```

to

```
root::0:0:root:/root:/bin/bash
```

When you have finished press [Ctrl] + X. This will ask if you want to save the modified files.
Press 'Y' and then hit [Return] to save the file with the same name.
After this change reboot and after boot, enter root as username, press enter and you will have root access.

After this, change root password and you can now login directly as root.

```
passwd root
```

Now, open a SSH session and from the prompt, type

```
su root
```

and enter the password for `root`
Other method, is to login as pi and at command prompt, type

```
su -
```

and enter the password for `root`

To go back as pi user, just type

```
su pi
```

Get all the info you can get

There is a very usefull tool `inxi` available to list all sorts of information about your PI. To install this tool, type the following in the command line:

```
sudo apt install –y inxi
```

Once installed, you can type e.g.

```
sudo inxi –F
```

or

```
sudo inxi –h
```

to get all command line switches. Here is a short list of them

```
Output Control Options:
-A     Audio/sound card information.
-b     Basic output, short form. Like inxi -v 2, only minus hard disk names.
-c     Color schemes. Scheme number is required.
-C     CPU output, including per CPU clockspeed (if available).
-d     Optical drive data. Same as -Dd. See also -x and -xx.
-D     Full hard Disk info, not only model, ie: /dev/sda ST380817AS 80.0GB.
-f     All cpu flags, triggers -C. Not shown with -F to avoid spamming.
-F     Full output for inxi. Includes all Upper Case line letters, plus -s and -n.
-G     Graphic card information
-i     Wan IP address, and shows local interfaces (requires ifconfig network tool).
-I     Information: processes, uptime, memory, irc client (or shell type), inxi version.
-l     Partition labels. Default: short partition -P.
-M     Machine data. Motherboard, Bios, and if present, System Builder (Like Lenovo).
-n     Advanced Network card information. Same as -Nn. Shows interface, speed, mac id, etc.
-N     Network card information. With -x, shows PCI BusID, Port number.
-o     Unmounted partition information (includes UUID and LABEL if available).
-p     Full partition information (-P plus all other detected partitions).
-P     Basic partition information (shows what -v 4 would show, but without extra data).
-r     Distro repository data. Supported repo types: APT; PACMAN; PISI; YUM; URPMQ; Ports.
-R     RAID data. Shows RAID devices, states, levels, and components, and extra data
-s     Sensors output (if sensors installed/configured.
-S     System information: host name, kernel, desktop environment (if in X), distro
-t     Processes. Requires extra options: c (cpu) m (memory) cm (cpu+memory).
-u     Partition UUIDs. Default: short partition -P. For full -p output, use: -pu (or -plu).
-v     Script verbosity levels. Verbosity level number is required. Should not be used with
-w     Local weather data/time. To check an alternate location, see: -W <location>.
-W     <location> Supported options for <location>: postal code; city, state/country;
-x     Adds the following extra data (only works with verbose or line output,
-xx    Show extra, extra data (only works with verbose or line output, not short form)
-y     Required extra option: integer, 80 or greater. Set the output line width max.
-z     Security filters for IP/Mac addresses, location, user home directory name.
-Z     Absolute override for output filters. Useful for debugging networking issues in irc
-h     Help menu.
-H     Help menu, plus developer options.
-V     inxi version information. Prints information then exits.
```

Note however that this tool is written for Linux running on standard computers. So it might well not recognise all on your Raspberry Pi.

Cleanup packages

When you install a package, `apt` retrieves the needed files from the hosts listed in /etc/apt/sources.list, stores them in a local repository (/var/cache/apt/archives/), and then proceeds with installation.

In time the local repository can grow and occupy a lot of disk space. Fortunately, `apt` provides tools for managing its local repository: apt's clean and autoclean methods.

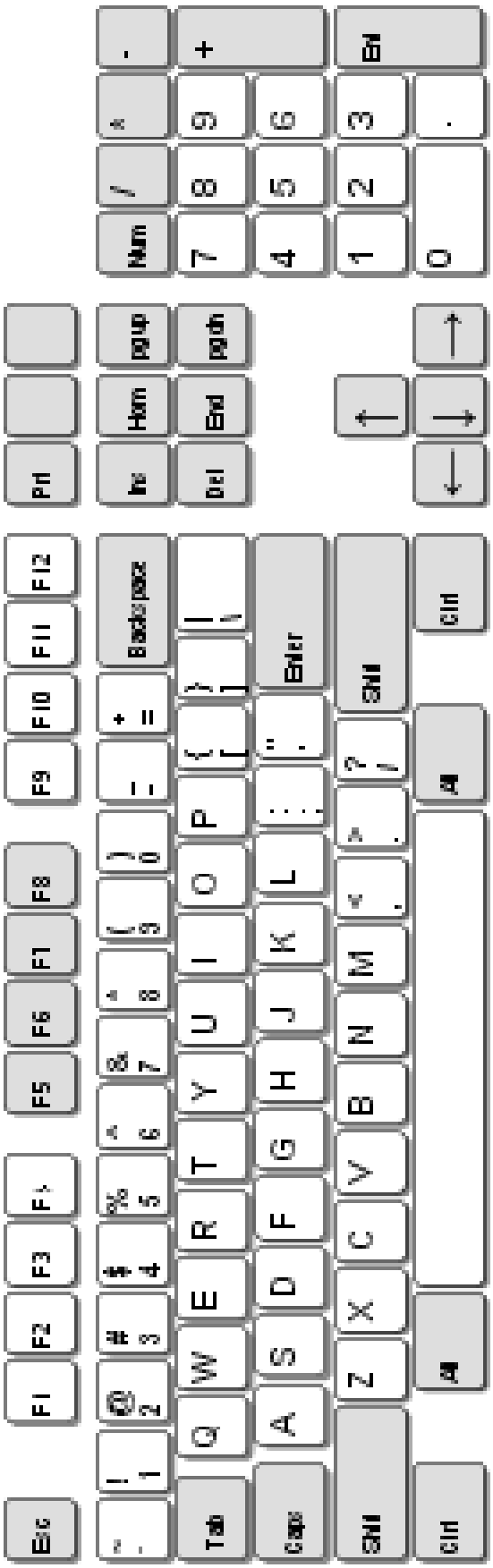
`apt clean` removes everything except lock files from /var/cache/apt/archives/ and /var/cache/apt/archives/partial/. Thus, if you need to reinstall a package `apt` should retrieve it again.

`apt autoclean` removes only package files that can no longer be downloaded.

```
sudo apt -y clean
sudo apt -y autoclean
sudo apt -y autoremove
```

Appendix

1. 102 key US QWERTY keyboard lay-out



2. Incorporate RPI_REVISION in your future programs

We could write some code to check the cpuinfo and extract the bit we want, compare it with known revision codes etc. But we don't need any of that because, from RPi.GPIO 0.4.0a onwards (September 2012) we can use a built-in RPi.GPIO variable which does it all for us.
It's called **GPIO.RPI_REVISION.** We can try this out in a live python session…

```
sudo python
...
>>> import RPi.GPIO as GPIO
>>> GPIO.RPI_REVISION
2
>>>
```

Mine returns a 2 because it's a Rev 2 Pi. Possible answers are
0 = Compute Module
1 = Rev 1
2 = Rev 2
3 = Model B+/A+

So then, when you're making program with some GPIO work that uses any of the ports 0, 1, 2, 3, 21, 27, 28, 29, 30, 31, you can be sure to control the correct ports.

```
import RPi.GPIO as GPIO

if GPIO.RPI_REVISION == 1:
    ports = [0, 1, 21]
else:
    ports = [2, 3, 27]

print "Your Pi is a Revision %s, so your ports are: %s" %
(GPIO.RPI_REVISION, ports)
```

As long as no further changes to GPIO pin allocations are made, the above should work with all future Pi revisions. If further changes are made, they will need to be incorporated like this

```
import RPi.GPIO as GPIO

if GPIO.RPI_REVISION == 1:
    ports = [0, 1, 21]
elif GPIO.RPI_REVISION == 2:
    ports = [2, 3, 27]
else:
    ports = ["whatever the new changes will be"]

print "Your Pi is a Revision %s, so your ports are: %s" %
(GPIO.RPI_REVISION, ports)
```

3. Determine your PI model and related memory

```
import re

def getPiInfo():
    try:
        with open('/proc/cpuinfo', 'r') as infile:
            for line in infile:
                # Match a line of the form "Revision : 0002"
                # while ignoring extra info in front of the revsion
                # (like 1000 when the Pi was over-volted).

                # look first for 6-character word
                match = re.match('Revision\s+:\s+.*(\w{6})$', line)
                if match and match.group(1) in ['900092']:
                    return "Zero", "512 MB"
                elif match and match.group(1) in ['a01041', 'a21041']:
                    return "2 Model B", "1 GB"
                elif match and match.group(1) in ['a02082', 'a22082']:
                    return "3 Model B", "1 GB"

                # now look for 4-character word
                match = re.match('Revision\s+:\s+.*(\w{4})$', line)
                if match and match.group(1) in ['0000', '0002', '0003']:
                    return "Model B Revision 1.0", "256 MB"
                elif match and match.group(1) in ['0004', '0005', '0006']:
                    return "Model B Revision 2.0", "256 MB"
                elif match and match.group(1) in ['0007', '0008', '0009']:
                    return "Model A", "256 MB"
                elif match and match.group(1) in ['000d', '000e', '000f']:
                    return "Model B Revision 2.0", "512 MB"
                elif match and match.group(1) in ['0010']:
                    return "Model B+", "512 MB"
                elif match and match.group(1) in ['0011']:
                    return "Compute Model", "512 MB"
                elif match and match.group(1) in ['0012']:
                    return "Model A+", "256 MB"
                elif match:
                    return "Unlisted model", ""

            # we could fall throught to here meaning we
            # couldn't find the revision, assume revision 0
            # like older code for compatibility.
            return "Failed finding ‘revision’", ""

    # failed to open /proc/cpuinfo'
    except:
        return "Failed to open", ""
```