ArchPi cheat book

Jérémy Bardon

December 28, 2014

Contents

Preface

Howto book to learn you a few things you need to know about ArchLinux ARM on RPi. From basic setup of the system to side packages installation to turn your Raspberry into a music sharing or even a versioning control server.

Structure of book

The first part of this book will be focused on system setup and basic settings as keyboard language, user account and others. The second part will describe how to install some third party sofwares as git and mpd server.

Author words

I am not an expert in linux system as ArchLinux and even less in electronic stuff. However, as a developer I like to tinker with my new toy which is a Raspberry Pi.

I had a lot of troubles when I decided to find uses for it and tried to install some third party software. As a result, I am glad to write this “book” to help you to install things on your RPi with ArchLinux.

Chapter 1

Introduction

1.1 Are you interested?

This book is written by a non-specialist of ArchLinux with basic knowledge of linux system so I will try to made it as simple as possible for people who have no idea about what is console. Indeed, all commands will be explained for a better comprehension and an index will be available for you.

No matter if you are an expert or a novice, you will be able to find how to install stuff on your your Pi plus tips which includes all the problems I encounter during my first installation.

1.2 What is a Raspberry Pi

If you succeed to find this book I guess you allready know but some people buy a Raspberry with OpenELEC[1] pre-installed so here is a little explaination.

The Raspberry Pi is a credit-size computer with low performances if you compare with a common PC. Nevertheless, it means its power consumption is very low (1W for B+ version[2]) so it is not a problem to let it on forever.

[1]Tiny linux system based on XBMC media center. More details on openelec.tv

[2]Most robust version of RPi with 512MB of RAM and 4 usb ports

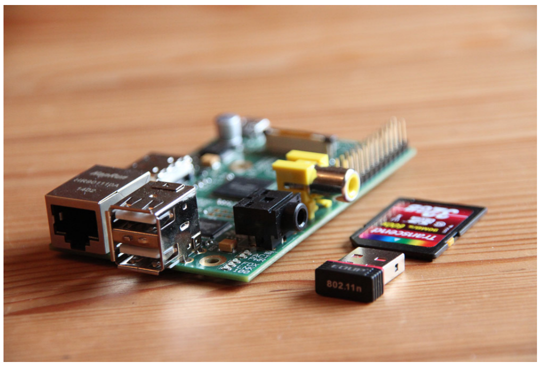


Figure 1.1: Raspberry Pi B+

Finally if you install a good linux distribution on it you can turn this old computer into a cheap server on which you will have the control. You can use it at home for file sharing, media player or others but it is also possible to host a website which will be available on the internet[3].

1.3 ArchLinux versus Raspbian

The operating system recommended by the Raspberry fundation is Raspbian – a custom version of the famous Debian[4] system – optimized for RPi hardware.

In general it will be the default choice for an inexperienced user to get a user interface and most common softwares allready installed at the first boot. However we forget the limited performances of the Raspberry and you will be able to realize that for yourself if you decided to install Raspbian.

A server does not need a user interface except a terminal which is enough to manage it everyday from anywhere. As a result, my choice has been focused on ArchLinux which is a pretty light and fast system. In addition, system updates are based on rolling release [5] model, so it means you do not have one version of the system. You will just receive updates frequentely – as soon as their availability – and it will be not necessary to reboot during the upgrade process.

[3]An example of website hosted by a Rpi on raspberrypi.goddess-gate.com
[4]One of the most popular linux system. See debian.org for more details
[5]Definition on wikipedia/Rolling_release

Chapter 2

ArchLinux installation

2.1 We are Noobs

There are two ways to install ArchLinux on a Raspberry Pi: the first is the ArchLinux way – no idea if it is the same with other systems – and the second is an official manager which works with many systems.

- follow instructions from archlinuxarm.org[1] which requires to allready have a linux system
- use NOOBS, an operating system install manager provided by the Raspberry fundation[2]

I choose NOOBS because it is the easiest way to install a system on a RPi and in addition you get an extra "boot manager" which is usefull. Moreover, you can complete the full setup of your SD card on any system in few simple steps described in the next part.

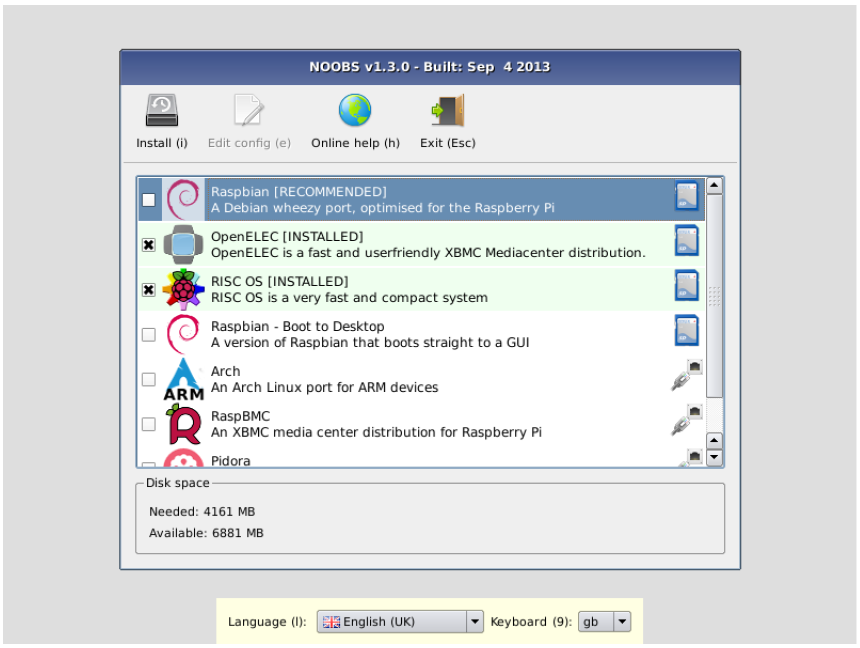
NOOBS is available on two forms: one for offline installation and the other – the smallest one – downloads automatically the last release of the system online. The offline installer contains many systems – which takes a large space – but you can just keep ArchLinux and remove others (in `os` folder). Anyway, you need to know that other systems files will be keeped after the installation so it is lost space. If you still want to use the offline way because you have no choices, you will have to find an older version of NOOBS because ArchLinux has been removed since the last release.

[1]Specific instruction on archlinuxarm.org/platforms/armv6/raspberry-pi
[2]Details on www.raspberrypi.org/help/ noobs-setup

2.2 Installer usage

According to the Noobs documentation, you just have to download the last Noobs release and format your SD card before unzip and copy Noobs files on the card.

After putting the card into the RPi and power on, you will get the installer interface with a list of all systems you can install. If you choose the online way you have to connect your ethernet cable in the Raspberry even if you want to use a wifi dongle later.



- [ ] Raspbian [RECOMMENDED] — A Debian wheezy port, optimised for the Raspberry Pi
- [✓] OpenELEC [INSTALLED] — OpenELEC is a fast and userfriendly XBMC Mediacenter distribution.
- [✓] RISC OS [INSTALLED] — RISC OS is a very fast and compact system
- [ ] Raspbian - Boot to Desktop — A version of Raspbian that boots straight to a GUI
- [ ] Arch — An Arch Linux port for ARM devices
- [ ] RaspBMC — An XBMC media center distribution for Raspberry Pi
- [ ] Pidora

Disk space
Needed: 4161 MB
Available: 6881 MB

Language (l): English (UK) Keyboard (9): gb

Figure 2.1: Noobs installer menu

From the menu, select ArchLinux – with $\boxed{\uparrow}$ and $\boxed{\downarrow}$ – and press $\boxed{\text{space}}$ to valid your choice. Then, you can change menu and keyboard language with respectively $\boxed{\text{L}}$ and $\boxed{\text{0}}$ before pressing $\boxed{\text{I}}$ to begin ArchLinux installation on your SD card.

Chapter 3

Basic setup

3.1 Change language

There are no default languages selected in ArchLinux but the keyboard map is set to qwerty[1] at the beginning. To choose your language you have to complete two steps and then the system will be able to use it for characters encoding and some softwares as `nano`.

3.1.1 Enable yours

Before choosing yours it is necessary to enable it in `/etc/locale.gen` file with `locale` tools.

You will need to use `nano` to edit the configuration file – ctrl+W can help you to search – and remove "#" before the language you want to enable (fr_FR.UTF-8 for example), to save your changes press ctrl+X.

```
$ locale # Current language settings

# Edit the /etc/locale.gen file
$ nano /etc/locale.gen

$ locale-gen # Update available languages
$ locale -a  # See available languages
```

Listing 3.1: Enable your language

[1] Most common layout for keyboards

3.1.2 Change your settings

The second step is to set the language and configure your keyboard map. Notice that you will have to logout for the system to take into account changes you made in language setup.

```
$ localectl status
   System Locale: n/a  # System language
       VC Keymap: n/a  # Virtual console
      X11 Layout: n/a  # Graphic interface

# Change system language (choose enabled one)
$ localectl set-locale LANG=fr_FR.UTF-8

# List of keymaps, choose the one you want "fr-pc" for example
$ localectl list-keymaps

# Change settings
# no-convert not update VC with X11 and vice versa
$ localectl set-keymap --no-convert fr-pc      # VC Keymap
$ localectl set-x11-keymap --no-convert fr-pc # X11 Layout

# Logout to apply changes
$ exit
```

Listing 3.2: Change language settings

3.2 Configure wifi connexion

3.2.1 Check your dongle

The first thing you can do is checking if your dongle has been recognized by the system and can be used.

```
$ ifconfig -a wlan # All wireless interfaces (also disabled)
```

Listing 3.3: Check wifi device

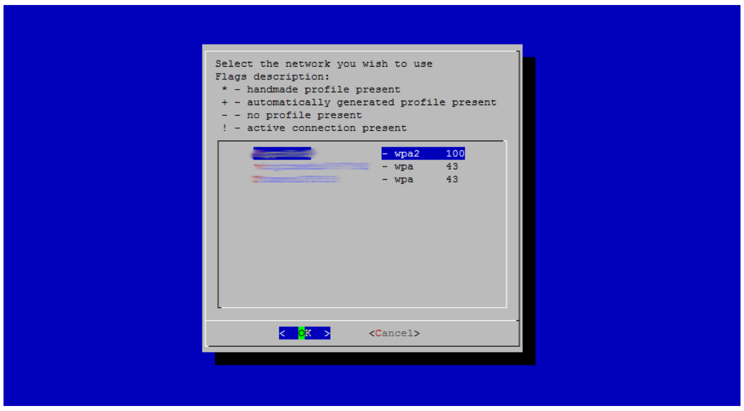
There are a lot of ways to connect your RPi to a network using a wifi dongle but all of them requires to install a package before – wifi-menu needs dialog, iw and wpa_supplicant are not installed – so it is necessary to use an ethernet wire to install them.

```
# pacman is the package manager in ArchLinux
#          -S install a new package
#
# dialog to get wifi-menu interface
# wpa_supplicant for wireless network protected with wpa keys
#
$ pacman -S dialog wpa_supplicant
```

Listing 3.4: Install wireless dependencies

3.2.2 Searching the internet

Once you installed all the packages – synonym for software – that wifi-menu needed to run you can launch it with “`wifi-menu`” command.



```
Select the network you wish to use
Flags description:
 * - handmade profile present
 + - automatically generated profile present
 - - no profile present
 ! - active connection present

     [redacted]          - wpa2     100
     [redacted]          - wpa      43
     [redacted]          - wpa      43
```

Figure 3.1: wifi-menu interface

Then, select your network – with ↑ and ↓ – , type your password (if required) and you will be connected to your wireless network.

3.3 Create yourself

3.3.1 Simple user

The default username and password for ArchLinux are `root/root`, this user got all right on the system it means he can do anything – even break the system – so it is not recommanded to use it.

However, you should change this default password for security purposes and only use your new account.

```
$ passwd root        # Changes root password (for security)

$ useradd -m jeremy  # Creates user jeremy and his home folder
$ passwd jeremy      # Changes jeremy password
```

Listing 3.5: Create a new user called jeremy

Now we have a new user account for everydays usage. You can see all users on your system with "`cat /etc/passwd`" which will display the content of the config file for users.

3.3.2 Very important user

It is possible to specify user rights with `visudo` command, the general syntax for one user is the following "`username machine=(targetuser) commands`", let's look at some details:

username name you gave to `useradd` command

machine machine on where rights are applied, ALL in general

target user user that we takes the rights

command allowed commands separated with one coma – no spaces –, use exclamation mark for banned commands

`visudo` is the command which prevent you from blocking the system with bad changes in the config file `/etc/sudoers`. The default text editor used by `visudo` is `vim` but before we used `nano` so to keep using it you have tell it to the system.

```
$ pacman -S sudo   # visudo is inside

# For this session set nano as default editor
# EDITOR is an environement variable
$ export EDITOR=/usr/bin/nano

# To check your changes, use echo which print a message
# in the console and variable with "$" before
$ echo $EDITOR

# Add your account rights after root
# for example "jeremy ALL=(ALL) ALL"
$ visudo
```

Listing 3.6: Specify user rights

Now you are a regular user with root permissions but some commands requires to be root – as making changes in systems files – so if a command want not works you can try to add `sudo` before.

By doing this the system will ask you for your password – even if you are logged – to get root permissions temporarily.

```
# Logged as jeremy (regular user)
$ visudo
visudo: /etc/sudoers: Permission denied

$ sudo visudo  # Launches visudo with root rights
```

Listing 3.7: sudo command usage

This example shows you that `visudo` command requires to be root so if you are logged with a regular account you can not use it. Instead of logout and login with root user – remember do not use this account – just add `sudo` before your command.

3.4 Setup remote connexion

If you want to keep working with your regular computer on the RPi – without two screens and two keyboards – it possible to setup a remote connexion with `ssh`[2].

On the system there are many programs as `ssh` or web server launched

[2]Remote secure shell access

in background. These programs are regularly started at system startup and are called *daemons* or *services*.

Before trying to connect to your Raspberry you should check if `sshd` – for ssh daemon – is running.

```
$ systemctl status sshd  # Check service state

# If sshd is not running, launch it
$ systemctl start sshd   # Start service

# If after a reboot it is not started
# It can be disabled
$ systemctl enable sshd  # Enable service
```

Listing 3.8: Check service state

Now that sshd is running, you will be able to connect to it remotely from any other system. Here is a list of the methods you should use depending on your own machine:

Linux and OSX Use the build-in terminal

Windows PuTTY[3] is the most used tool for this kind of work

Whathever the system you are using, two informations are necessery to perform an ssh connexion : remote account – created in section 3.3.1 – and remote machine IP or domain name.

```
# Format: username@IP (or domain)
$ ssh jeremy@192.168.0.2          # Initiate ssh connexion
jeremy@192.168.0.23's password:  # Type your account password

$ exit  # Stop ssh connexion and go back to local terminal
```

Listing 3.9: SSH with command line

[3]SSH client available on putty.org

Chapter 4

Web server

4.1 Nginx versus Apache

In web servers world there are three major actors: Apache, nginx and IIS (by Microsoft). If you look at the number of production uses[1] you will see that I followed the distribution from above it means ~55% for our old Apache and ~15% for the two others.

However, it appears like Apache decreases by 5% and nginx get these same 5% all that between January 2014 and February 2014. Now, I can tell you I choosed to use nginx instead of Apache for some reasons and the nginx rise is one of them.

Let's sum up some advantages and drawbacks of using nginx instead of apache. Few of them are extracted from raspbian-france.fr[2] but others are personnal.

[1]Netcraft survey available on news.netcraft.com/archives/2014/02/03/february-2014-web-server-survey.html

[2]Article on the same topic (in french), see raspbian-france.fr/installer-nginx-raspbian-accelerez-serveur-web-raspberry

+	Asynchronous server[3] More scalable[4] Less RAM usage (think about our poor RPi) Config file syntax (XML vs readeable)
-	Less mature (1995 vs 2004) Some PHP behaviours may differ from Apache .htacess not work directly with nginx

Table 4.1: Advantages and drawbacks of nginx

4.2 Installation

To run a non-static website, nginx is not sufficient we also need to install PHP – for dynamic pages – and a mySQL database to save our precious data. In addition, we will install phpMyAdmin which is a tool – written in PHP – to manage a mySQL database through a web interface.

```
# php-fpm is the nginx module for PHP
# mariadb is a mySQL database
$ pacman -S nginx php-fpm mariadb phpmyadmin

# Enable and run services
$ systemctl enable nginx php-fpm mysqld
$ systemctl start nginx php-fpm mysqld
```

Listing 4.1: Install a full web server

Nginx server works well but php is not enabled in configuration files. Moreover, there are some incompatibilties troubles in initial setup.

```
# First problem: welcome file directory is not set in
# availables directories for php
#
# Update php config file
# Search open_basedir, add ":/usr/share/nginx/html/" at the end
#
$ nano /etc/php/php.ini

# Second problem: php not enabled and contains bad setup
# in nginx configuration file
#
# Uncomment (remove #) on the following block and change it
# to fit the example
#
$ nano /etc/nginx/nginx.conf

# pass the PHP scripts to FastCGI server listening on
    127.0.0.1:9000
```

```
#
location ~ \.php$ {
    root    /usr/share/nginx/html; # Fix path to php files

# Use sockets
#   fastcgi_pass    127.0.0.1:9000;
    fastcgi_pass    unix:/var/run/php-fpm/php-fpm.sock;

    fastcgi_index   index.php;

# Use fastcgi.conf file
#   fastcgi_param   SCRIPT_FILENAME   /scripts$fastcgi_script_name
    ;
#   include         fastcgi_params;
    include         fastcgi.conf;
}

# Third problem: index pages must be .html or .html
# want to add index.php
#
location / {
    root            /usr/share/nginx/html;

    # Add index.php
    index           index.html index.htm index.php;
}
```

Listing 4.2: Enable php in nginx

Php does not enable mysql extensions by default in configuration files. Therefore, to be able to perform SQL queries on local database through php scripts you need to enable these extensions.

```
$ nano /etc/php/php.ini

# Remove ";" before mysql and mysqli extensions
;extension=mysql.so
;extension=mysqli.so
```

Listing 4.3: Enable php in nginx

The last thing to do for security purposes is to change mySQL default root password that is empty.

```
# -u: considerer specified user
# password: update user password
# NewPassword: your password for root user
#
$ mysqladmin -u root password NewPassword
```

Listing 4.4: Change mySQL root password

Each time you change nginx or php configuration files you have to reload and restart the corresponding service for it to take into account your changes.

```
# Restart only considered service
$ systemctl reload nginx php-fpm
$ systemctl restart nginx php-fpm
```

Listing 4.5: Reload and restart a service

4.3 Basic verifications

The next step is to check our web server with basic verifications to test whether the installation worked well. We will simply try to access to the default nginx page remotely, create a php page and display it and finally access to local database.

4.3.1 Nginx and php

During installation process, nginx places a welcome page called *index.html* in `/usr/share/nginx/html` directory. However Php gives no sample to test with nginx therefore we will create a new php script in welcome page directory.

The content of this new script will simply display php configuration on your server.

```
<!-- /usr/share/nginx/html/index.php -->

<?php
    phpinfo();
?>
```

Listing 4.6: Php simple script

The goal of these test is to access to *index.html* and *index.php* from a remote web browser to check if nginx and php work well on your server.

To perform that, type the IP address of your RPi on your PC web browser – followed by /index.html and then /index.php – and you will see nginx welcome page and php config respectively.

4.3.2 MySQL

There are two points to check with mySQL installation, first the database itself and then communication between mySQL and PHP.

```
$ mysql -uroot      # Connect to local database with root user
> show databases; # List of databases
> use mysql;        # Go into mysql database
> show tables;      # List of tables in current database

# Perform SQL query to list users
> select host, user, pasword from user;
> exit; # Logout from local database

# If nothing works try to launch mysql install script
# user: user account used by mysql service
# ldata: path to MariaDB data directory
# basedir: path to MariaDB installation directory
$ mysql_install_db --user=mysql --ldata=/var/lib/mysql
    --basedir=/usr
```

Listing 4.7: Test mySQL installation

Yet you suceed to list database users with the dedicated command, the last step is to check if we can do the same thing through php.

```
<!-- /usr/share/nginx/html/checkdb.php -->

<?php

if (!$link = mysql_connect('localhost', 'root', '')) {
    echo 'Could not connect to mysql';
    exit;
}

if (!mysql_select_db('mysql', $link)) {
    echo 'Could not select database';
    exit;
}

$sql    = 'SELECT host, user, password FROM user';
$result = mysql_query($sql, $link);

if (!$result) {
    echo "DB Error, could not query the database\n";
    echo 'MySQL Error: ' . mysql_error();
    exit;
}

echo '<table><tr><th>host</th><th>user</th><th>password</th></tr
    >';
while ($row = mysql_fetch_assoc($result)) {
    echo '<tr><td>' .
         $row['host'] . '</td><td>' .
```

```
        $row['user'] . '</td><td>' .
        $row['password'] . '</td></tr>';
}
echo '</table>';

mysql_free_result($result);

?>
```

Listing 4.8: Test php with mySQL

4.4 Domains and custom directory

4.4.1 Custom web directory

Nginx default setup set this directory as the root of your server but websites are usually hosted in `/var/www` so I will explain you how to move the welcome page into that new file.

```
# Create /var/www directory and move into it
$ cd /var
$ mkdir www # Create a folder named "www"
$ cd mkdir

# Copy directory (with -r) source destination
$ cp -r /usr/share/nginx/html/ /var/www/

# Move a directory but if source and destination
# are in same directory rename source
$ mv /var/www/html /var/www/home
```

Listing 4.9: Move nginx welcome page

Before trying to access to your website it will be necessary to configure nginx and php for the new path. The sample nginx configuration file contains some help to do specific things so we will save it and create a new one with the following example.

```
# Save and change nginx file
$ mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.old
$ nano /etc/nginx/nginx.conf # See following example

# Update php config file
# Search open_basedir and add ":/var/www/" at the end
$ nano /etc/php/php.ini

# Do not forget to reload and restart services
```

Listing 4.10: Configure nginx and php

```
# /etc/nginx/nginx.conf

worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include        mime.types;
    default_type  application/octet-stream;

    access_log  /var/log/nginx/access.log;

    sendfile          on;
    keepalive_timeout  65;

    server {
        listen 80;
        root /var/www/home; # New directory
        index index.html;   # nginx welcome page
    }
 }
```

Listing 4.11: nginx configuration

4.4.2 Subdomains and site setup

Nginx uses the same principles as Apache in sites management. Indeed, in nginx configuration directory – `/etc/nginx/` – there are two folders called respectively *site-available* and *site-enabled* and in *nginx.conf* file there must be a line which loads sites in *site-enabled*.

```
# First check: site-available and site-enabled exist
#
# ls (list segment): list of files in given directory
# "|": pass result to the next command
# grep site: display lines which contains "site"
#
$ ls /etc/nginx | grep site

# Create these folders if there are not displayed
#
# mkdir: create directory with specified name
```

```
#
$ mkdir /etc/nginx/site-available /etc/nginx/site-enabled

# Second check: site-enable is included in nginx.conf
#
# cat: display specified file in console
#
$ cat /etc/nginx/nginx.conf | grep site

# If not in nginx.conf add the following line:
#   include /etc/nginx/site-enabled/*;
# in html block
#
$ nano /etc/nginx/nginx.conf
```

Listing 4.12: Nginx domains architecture

Once the nginx setup completed, you can follow with the creation of a subdomain. As an example, we would like to access to phyMyAdmin interface – see 4.5 for installation – with `http://pma.local.pi` instead of using `http://RpiIP/phpmyadmin/`.

```
# Create phpmyadmin file with the next example as content
$ nano /etc/nginx/site-available/phpmyadmin

# Create a link in site-enable to declare it as enabled
$ ln -s /etc/nginx/site-available/phpmyadmin /etc/nginx/site-
   enabled/

# Check the link
$ ls -l /etc/nginx/site-enabled/

# Do not forget to restart service
```

Listing 4.13: Create a subdomain

```
# /etc/nginx/site-available/phpmyadmin

server {
    root /usr/share/webapps/phpMyAdmin;
    index index.php;

    server_name pma.local.pi;

    location ~ \.php$ {
        fastcgi_pass unix:/run/php-fpm/php-fpm.sock;
        fastcgi_index index.php;
        include fastcgi.conf;
    }
}
```

Listing 4.14: PhpMyAdmin domain example

The final step to be able to access to `http://pma.local.pi` is to match this domain with the RPi IP address. Usually, there two ways to perform that kind of things :

DNS[5]server A service with matches domain names with IP address. If your router is the server it will work for all connected devices.

hosts file Local file in your machine which overrides DNS server and thus works only for your machine.

For the first method it depends on your material but for the second I can tell you that on Linux and Mac OSX *hosts* file is located in `/etc/hosts` and on Windows in `C:\Windows\system32 \drivers\etc\hosts`.

Add a line like "`RpiIP pma.local.pi`" in this file opened with root access (for Windows launch text editor with administrator rights).

4.5 PhpMyAdmin setup

The installation package puts phpMyAdmin files in `/usr/share/webapps/phpMyAdmin` directory so if you want to access to the web interface you need to move this directory in `/usr/share/nginx/html`. However, it would be more logical to let application files where there are, so we will create a link.

```
# ln create link to source in destination
# -s symbolic link (not change link if source has been moved)
#
$ ln -s /usr/share/webapps/phpMyAdmin/ /usr/share/nginx/html/
    phpmyadmin
```

Listing 4.15: Configure phpMyAdmin

Now, you can try to connect to `RPiIP/phpmyadmin` from a remote browser and use the root account of mySQL (remember you changed the default password).

If you get a warning message which tells you it can not access *config.inc.php* it is because phpMyAdmin puts this file in it's installation directory and also in `/etc/webapps/phpmyadmin` but it has no rights on the second one.

[5]DNS for Domain Name Service

```
# Update php config file
# Search open_basedir and add ":/etc/webapps/" at the end
$ nano /etc/php/php.ini

# Do not forget to reload and restart php service
```

Listing 4.16: Fix phpMyAdmin troubles