ArchPi cheat book

Jérémy Bardon

January 1, 2015

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

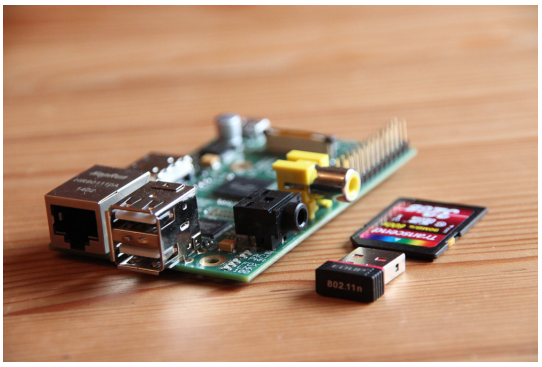

Figure 1.1: Raspberry Pi B+

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.

Figure 2.1: Noobs installer menu

From the menu, select ArchLinux – with ↑ and ↓ – and press space to valid your choice. Then, you can change menu and keyboard language with respectively L and 0 before pressing 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.

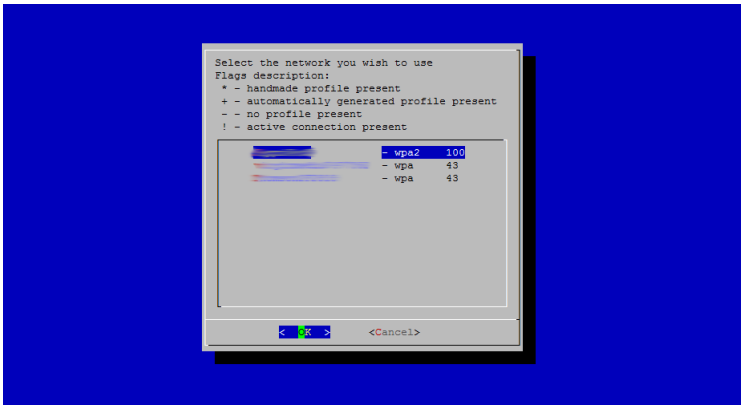

Figure 3.1: wifi-menu interface

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 System update and timezone

As I said before – in ArchLinux presentation1.3–, the system updates are based on rolling system principle. Therefore, you can update your system at anytime without reboot.

```
# S: synchronize packages list
# y: force refresh
# u: upgrade all outdated packages
$ pacman -Syu
```

Listing 3.8: Update ArchLinux

Each time you get a new device (smartphone, PC, ..) you have to set your timezone to have your system on time. It is much important on a server especially for logs feeds by your services.

```
# Current timezone
$ timedatectl status

# List available timezones
# Can add "| grep Eur" to limit result
# Only lines which contains "Eur" are displayed
#
$ timedatectl list-timezones

# Set your timezone (Europe/Paris for example)
# You must be root (use sudo)
$ timedatectl set-timezone Europe/Paris
```

Listing 3.9: Set your timezone

3.5 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 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
```

[2]Remote secure shell access

Listing 3.10: 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

Figure 3.2: PuTTY connexion interface

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.

In the case you did not have domain name for your RPi4.4.2, you can only use it's IP address. The `ifconfig` command will help you to get this address but be carefull about how your RPi is connected to your network: ethX means wired and wlanX is wireless.

```
# On your RPi
# ifconfig (interface configurator): list interfaces
# Here the IP address of my RPi is 192.168.0.23
#
$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ...

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    ...

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.23  netmask 255.255.255.0  broadcast
    192.168.0.255
```

[3]SSH client available on putty.org

```
    ...

# On your Remote PC
# 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 3.11: SSH with command line

3.6 Remote access to your files

If you want to manage your server files from a remote computer you can use FTP[4] which will be a service on your RPi.

There are a lot of ways to add an FTP access to your server and I choose to use *vsftpd* – for very secure ftp daemon – a secure solution to our problem. With this protocol you can connect yourself to the server as an anonymous user – no username and no password – or you can use your account on the server.

In my configuration, I enabled access to logged users only but feel free to enable anonymous access (think about security).

```
$ pacman -S vsftpd  # Install package

# Change settings in /etc/vsftpd.conf
$ nano /etc/vsftpd.conf

# Enable anonymous access uncomment
anonymous_enable=YES
anon_upload_enable=YES
anon_mkdir_write_enable=YES

# Enable logged users access uncomment
local_enable=YES
write_enable=YES
local_umask=022

# Enable and start service
$ systemctl enable vsftpd
$ systemctl start vsftpd
```

[4]FTP for File Transfer Protocol

Listing 3.12: VsFTPd setup

Your FTP server is running, it is time for remote connexion. The most used multiplatform tool is FileZilla an FTP client with a graphic interface but some Windows users like WinSCP.

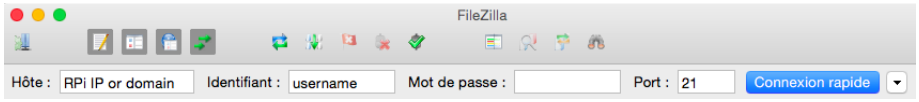

Figure 3.3: FileZilla connexion interface

As for SSH connexion you need your RPi IP address (or domain name, see 3.5) and your account informations – username and password – if you did not enabled anonymous access to your server.

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 creates a 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

Chapter 5

Jukebox

5.1 Do I need that?

Maybe you want to use your Raspberry as a jukebox controlled remotely, so your music will be played on sound devices – through HDMI or DVI port – attached to your RPi.

A MPD[1] server is exactly what you need and in addition it can manage playlists – not explained here – and it exists a plenty of applications[2] which are dedicated or compatible with.

In following sections you will learn how to setup your server and then you will discover four clients for various platforms (PC and smartphones).

5.2 MPD setup

There are two things you need to take care of when you want to install a mpd server: how to play music on the RPi and where your music will be stored.

In the example, it is a "classic" situation wherein you music is located in a folder called *music* placed in a user home directory.

```
# mpd is the server
# alsa-utils manage your RPi audio outputs
#
$ pacman -S mpd alsa-utils
```

[1] MPD for Music Player Daemon

[2] A list of client are available on mpd wiki, see mpd.wikia.com/wiki/Clients

```
# Update the mpd config file to work with
# the RPi audio (see the next example)
$ nano /etc/mpd.conf

# Mpd files (including music database) are located
# in /var/lib/mpd so you have to change it's owner
# for the mpd user to be able to access it
#
# chown: change file or diretory owner
# "mpd:mpd": new owner (user and his group)
#
$ chown mpd:mpd /var/lib/mpd

# My music is located in the SD card, where the system
# is installed: /home/jeremy/music
#
# In default configuration, your home directory can only
# be crossed by you but the mpd user must access to your
# music inside
#
# chmod: changes rights on a file or a directory
# o+x: anyone can cross the specified directory
# or execute the file (if it is a file)
#
$ chmod o+x /home/jeremy
```

Listing 5.1: Install mpd server

```
# /etc/mpd.conf

pid_file "/run/mpd/mpd.pid"
db_file "/var/lib/mpd/mpd.db"
state_file "/var/lib/mpd/mpdstate"

playlist_directory "/var/lib/mpd/playlists"

# Path to your musics
music_directory "/home/jeremy/music"

bind_to_address "any"
audio_output {
    type "alsa"
    name "rpi"
    device "hw:0,0"
}
```

Listing 5.2: Mpd configuration file

A RPi has two audio outputs (HDMI and DVI) so you can play your music on your screen (or TV, or audio system plugged in your TV) and in a

headset or any device which provides a jack input.

Do not forget the power of a Raspberry, is modularity can help you to setup bluetooth or airplay wireless network to play your music. However, these interesting solutions will be not covered in this book.

```
# Set audio output to HDMI
# Last number: 0 (auto, avoid it), 1 (DVI), 2 (HDMI)
$ amixer cset numid=3 2

# Fix HDMI troubles
# Uncomment hdmi_force_hotplug=1 and hdmi_drive=2
$ nano /boot/config.txt

# You must reboot for HDMI changes
$ reboot

# Enable, start and then check if the service started
$ systemctl enable mpd
$ systemctl start mpd
$ systemctl status mpd
```

Listing 5.3: Set audio output

5.3 Music directory on usb device

5.3.1 Connect an usb device

Connect a device to an linux system as ArchLinux is called *mount*, once you connected you device the system will be able to see it with no ways to access it.

It is after the *mount* operation that the system get an access to the device and programs as mpd will be able to read on.

```
# List of connected devices
$ fdisk -l
# One device: mmcblk0, zero because you can have others sd cards
# called mmcblk1, mmcblk2, ...
Disk /dev/mmcblk0 : 7,5 GiB, 8017412096 bytes, 15659008 sectors

# Divided in 2 parts (partitions)
# Note: this table was truncated in lines and columns
Device          Size    Id  Type
/dev/mmcblk0p1  116,2M   e  W95 FAT16 (LBA)
/dev/mmcblk0p2    7,3G  85  Linux extended
```

```
# Create a directory in /mnt where you
# will see your device files
$ mkdir /mnt/sdcard

# Mount a device (mmcblk0 for example)
#
# Can add some options:
# -t ntfs-3g: format type of the device
# -o uid=user,gid=group: owner of mounted directory
#
$ mount /dev/mmcblk /mnt/sdcard

# Unmount a device: use the target directory
# choosed on mounting operation
#
# Some devices may be used by programs or services
# Be careful you are not in your device directory (use pwd)
#
$ umount /mnt/sdcard

# List mounted directories
$ mount
/dev/sda1 on /mnt/hd1 type vfat
```

Listing 5.4: Mount an external device

If you use some devices with a Windows system they are maybe formated with the *ntfs* format that is not recognize by default on a linux based system.

To solve this trouble you have to install *ntfs-3g* which contains necessary drivers and an easiest way to mount ntfs devices.

```
# Install the package with ntfs drivers
$ pacman -S ntfs-3g

# For ntfs-3g works you need to reboot
$ reboot

# Mount your ntfs device
# Short for "mount -t ntfs-3g /dev/mmcblk /mnt/sdcard"
$ ntfs-3g /dev/mmcblk /mnt/sdcard
```

Listing 5.5: Mount ntfs devices

5.3.2 Change directory in mpd

To change your music directory in MPD this is only one file to update mpd.conf (refer to 6.14) which contains the path to your music on *music_directory* line.

Nevertheless, this is one thing you need to take care of: rights on every directory in your path. In linux system, there are three access types : read, write and execute knowing execute means "can cross" for directories.

```
$ ls -l / | grep home
drwxr-x---  3 root root  4096 18 dec.  23:48 home
```

Listing 5.6: Analyze rights on home

Let's analyze the previous result for `/home` directory to able to understand rights easily.

Rights (modes)	**Links number**	**Owner**	**Group**	**Size (in bytes)**	**Last update**	**Folder name**
`drwxr-x-x`	`3`	`root`	`root`	`4096`	`18 dec. 23:48`	`home`

Table 5.1: Details on `ls -l` syntax

Now, we can decrypt `ls -l` results and identify who is the owner of a file and of which group it belongs. With the first part of the output it is possible to deduce what things a user can do on the given file.

Indicators	
d	File type: directory (d), symbolic link (l), file (-)

Rights – nothing(-) read(r) write(w) execute(x)	
rwx	Owner, got all rights usually
r-w	Users who belongs to group
–w	Others

Table 5.2: Rights on a file

Remember, the initial purpose was to change our mpd music directory so let's change the directory to "`/home/jeremy/music`" with good rights to ensure the *mpd* user will be able to access to the *music* directory and read files into.

```
# We want to check that mpd user can acces
# to /home/jeremy/music, we have to check the three directories
#
```

```
$ ls -l / | grep home

# Anyone can cross home: OK
drwxr-xr-x  3 root root  4096 18 dec.  23:48 home

$ ls -l /home | grep jeremy
# Nobody can cross this directory exept jeremy: NOT OK
drwx------ 4 jeremy jeremy 4096 30 dec.  11:56 jeremy

# Add execute right for everyone because mpd do not
# belongs to jeremy group
#
# Format: level+/-right
#   level: u (owner), g (owner group), o (anyone)
#   +/-: + to add right, - to remove it
#   right: r (read), w (write), x (execute)
#
$ chmod o+x /home/jeremy

$ ls -l /home | grep jeremy
# Anyone can cross this directory: OK
drwx------x 4 jeremy jeremy 4096 30 dec.  11:56 jeremy

$ ls -l /home/jeremy/ | grep music
# Anyone can read music files into this directory: OK
drwxr-xr-x 2 jeremy jeremy 4096 29 dec.  12:18 music

# Now the full path has been verified: mpd can cross each
# directories from root and read in music folder
```

Listing 5.7: Check rights on path

5.4 Classic and web clients

There many mpd clients available and it is necesarry to distinguish thoses which are installed on the server – as web clients – and others that are on a remote machine owned by the client.

5.4.1 External clients

I will present you two applications which are mpd client and as much as possible multiplatform or at least compatible with most common devices.

The first one is `MPDroid` an Android application available on the play store[3] which enable you to manage your mpd server depending on the wireless network you are connected.

GMPC[4] is the second client that I will talk about because it's strong point is that it is multiplateform so compatible with Windows, Mac OSX and Linux systems.

To setup the connexion between your application and the mpd server on the RPi the only thing you need is it's IP address or domain name. That is why we did not change the default port – 6600 – and set a password to access it but feel free to do it if it is necessary for you.

5.4.2 YMPD web client

YMPD is a lightweight and easy to install web client interface build over the bootstrap. It means the interface is really responsive and it can be used either on the PC or on a smartphone through a web browser.

```
# wget: tool to download files from network
$ pacman -S wget

# Use wget to download ympd release
$ wget http://www.ympd.org/downloads/ympd-1.2.3-armhf.tar.bz2

# Unpack ympd from the downloaded archive
$ tar -xvf ympd-1.2.3-armhf.tar.bz2

$ rm ympd-1.2.3-armhf.tar.bz2 # Remove unused archive
$ mv ympd /usr/bin            # Move executable to the "classic"
    directory

# Run ympd server and choose the port you will
# use to connect to it
$ ympd --webport 2020 # Any port > 1024
```

Listing 5.8: ympd setup

Now, you can connect to the ympd web interface from any web browser access to http://RPiIP:2020 but pay attention to use the port you choosed when you launched ympd your server.

[3]Download MPDroid on the play store at play.google.com/store/apps/details?id=com.namelessdev.mpdroid

[4]Gnome music player client

5.4.3 Ampache

If you are looking for a more complex and customizable solution for your mpd server web client, Ampache can be the answer to your problem.

It is php based application (similar to phpMyAdmin) which allows you to manage music catalogs – on your server and on remote machines – and play them on the server or locally in your browser (with streaming).

```
# You will need unzip to unpack ampache
$ pacman -S unzip

# Download last ampache release
$ wget https://github.com/ampache/ampache/archive/develop.zip

# Unpack ampache, remove zip file and rename ampache directory
$ unzip develop.zip
$ rm develop.zip
$ mv ampache-develop ampache

# Create a domain with nginx by using the following
# config file (too long but necessary) available at :
# http://github.com/ampache/ampache/wiki/Installation#nginx
#
# See Subdomains and site setup in Web server chapter of this
    book
# if you have some troubles
```

Listing 5.9: Download Ampache

Ampache offers a php script called *install.php* that you can reach by going to http://RPiIP/install.php but in my case it allways redirect me to */test.php* which tells me that I do not respect prerequires.

Finally, once I launched install script after some setup, I have realized that pretty much all things I have done before had to be done by this script.

As a result either you are lucky and the install script works or follow me for some setup to fill requirements and tell to the install script to do nothing because we have done it's work.

```
# The test page show me 6 errors at the begin
# here are the "name" of theses tests plus the
# ways to pass them

# MySQL and PHP iconv extension
# Problem: php extensions not enabled
#
```

```
# Remove ";" before: extension=pdo_mysql.so
#                    extension=iconv.so
#
$ nano /etc/php/php.ini
$ systemctl reload php-fpm

# Configuration file readibility and validity
# Problem: wrong name for ampache config file
#
$ cd /usr/share/webapps/ampache/config
$ mv ampache.cfg.php.dist ampache.cfg.php

# Database connexion
# Problem: user account for database not set in
# ampache configuration file
#
# Fill following fields with right values
# (remember you may changed the root password)
#         database_username = root
#         database_password = password
#
$ nano /usr/share/webapps/ampache/config/ampache.cfg.php


# Database table
# Problem: ampache database and tables not created
#
$ mysql -u root -ppassword # no space between -p and password
> create database ampache;
> use ampache;
> source /usr/share/webapps/ampache/sql/ampache.sql;
> exit
```

Listing 5.10: Ampache setup

At this point, Ampache let you run the install script but it says some tests about the configuration files fails, ignore them and then skip database and configuration file steps. The last thing to do is creating an admin account to connect yourself to ampache web client.

Chapter 6

Git server

6.1 Why you would use Git

For some reasons, you need to keep several versions of document so you make copies – can be long and space consumer – but it still difficult to get the last version when more than one person works on one document.

Git is a versionning tool which can do this work for you and in addition keep traces of who modified what on each document on a project. This tool will be not intrusive and can be used at every moments of a projet – even if it is allready started – it ensure that you have a project which is "good" and up-to-date version.

Largely used in programming projects it can be usefull for any types of projects which contains files readable by a simple text editor. You can get an idea of what is git by browsing projects on github.com which is the most famous git projects host because it is free for public (open source) projets.

6.2 Install Git and add projects

To visualize git projects on web interface the first thing is need to have git project. In order to create our own projet we will download an existing one to have a real history and some things to visualize.

I choose to use an open source project – Firefox OS[1] – because it is quite fast

[1]Firefox OS is the operating system developed by the Mozilla foundation for smartphones. More informations on www.mozilla.org/ en-US/firefox/os/, sources availables on github.com/mozilla/firefox-ios

to download and have some folders with enough commits to be interesting.

```
# Install git package
$ pacman -S git

# Create repositories(projects) directory
$ mkdir /srv/git

# Get an existing project on github to test our web
# interface, Firefox OS in this example.
$ cd /srv/git
$ git clone https://github.com/mozilla/firefox-ios.git
```

Listing 6.1: Install git and add projects

There are a few directories that are most used for repositories but I retained two of them: `/home/git/` and `/srv/git/`.

In ArchLinux, websites and ftp are by default in */srv* and */home/git* is the home directory for *git* user – which is not a physic user – so to keep a kind of consistency – in my opinion – I prefer the second alternative.

6.3 Host repositories

6.3.1 Create a new project

Host a git project is quite easy because it is really close from creating a local git repository. You just must think about adding *–bare* option to specify that your project is not a working copy.

```
# Let's create a new git project called "project"
# Naming convention: add .git at the end of your project name
# to specify it is not a working copy
#
$ mkdir /srv/git/project.git
$ cd /srv/git/project.git
$ git init --bare # --bare: not a working copy
```

Listing 6.2: Create a repository on your server

On a remote machine on which you want to work, clone the project and do a first commit on your project.

```
# Clone project
# username is your username for local account on the server
$ git clone username@RPiIP:/srv/git/project.git

# Create a new file and commit
```

```
$ cd project
# New file called README.md with "New project" text inside
$ echo "New project" > README.md
$ git commit -am"First commit"
$ git push origin master # First time, after just "git push"
```

Listing 6.3: Clone repository to work

6.3.2 Use an existing project

If you want to host a git project which allready it is possible to do. There two possible situations:

- The project is allready hosted by someone else or websites like GitHub or BitBucket.
- It is a local project so hosted locally

Allready hosted

As you can imagine, the first situation is the easier to solve, you just have to clone the repository from the server with a specific option.

```
$ git clone username@RPiIP:/srv/git/project.git --bare
```

Listing 6.4: Mpd configuration file

Note that the folder you will get will be named *project.git* so it is not a working copy. In addition, notice that is the same *–bare* option as when you create a new repository on the server.

Local project

The first thing to do is retrieve the local project on the server. There are many ways to it, I advice to use FTP – if you setup an FTP – but for others here is a simple way if you have a linux system.

```
# Your project is named project
$ scp username@RPiIP:/srv/git /local/path/to/project
```

Listing 6.5: Mpd configuration file

With the working copy on your server, the last thing to do is clone the repository as *bare* locally and remove the working copy.

```
# Extract project informations
$ git clone --bare /srv/git/projet /srv/git/project.git
$ rm -r /srv/git/project
```

Listing 6.6: Mpd configuration file

6.4 Web application for viewing

6.4.1 GitWeb: Build-in solution

Gitweb is the web interface offered by git when you install it. Based on cgi technology, it is a bit slow but very refined so easy to use and understand.

You can see a live usage of git web at repo.or.cz/w?a=project_list, browse projets, commit and all things you want and before abandoning GitWeb keep in mind that we will improve a little bit the visual.

Installation and configuration

```
# Gitweb is a cgi script, you need to install
# some packages
$ pacman -S fcgiwrap spawn-fcgi

# Update php configuration to accept .cgi files
$ nano /etc/php/php-fpm.conf
# Uncomment and add .cgi
security.limit_extensions = .php .php3 .php4 .php5 .cgi

# Set your repositories folder in gitweb.conf
# Look at the next example
$ nano /etc/gitweb.conf  # Is created of not exist
```

Listing 6.7: Gitweb setup

```
# path to git projects (<project>.git)
$projectroot = "/srv/git/";

# directory to use for temp files
$git_temp = "/tmp";

# target of the home link on top of all pages
#$home_link <nowiki>= $my_uri || "/";

# html text to include at home page
```

```
#$home_text = "indextext.html";

# file with project list; by default, simply scan the
    projectroot dir.
#$projects_list = "/home/git/projects.list";

# stylesheet to use
#$stylesheet = "/usr/share/gitweb/static/gitweb.css";

# logo to use
#$logo = "/usr/share/gitweb/static/git-logo.png";

# the 'favicon'
#$favicon = "/usr/share/gitweb/static/git-favicon.png";

# change default git logo url
$logo_url = "http://gitweb.naushika.pi";
$logo_label = "Local Git Repository";

# This prevents gitweb to show hidden repositories
#$export_ok = "git-daemon-export-ok";
#$strict_export = 1;

# This lets it make the URLs you see in the header
#@git_base_url_list = ( 'git://www.deimos.fr/git' );

$feature{'blame'}{'default'} = [1];
$feature{'highlight'}{'default'} = [1];
```

Listing 6.8: gitweb.conf file

Configure nginx and install cgi tools

According to the ArchLinux wiki, it is necessary to change the *fcgiwrap.service* file because it uses spawn-fcgi so trust the wiki. Note you have to restart fcgiwrap service after your updates.

```
# /usr/lib/systemd/system/fcgiwrap.service

[Unit]
Description=Simple server for running CGI applications over
    FastCGI
After=syslog.target network.target

[Service]
Type=forking
Restart=on-abort
PIDFile=/var/run/fcgiwrap.pid
```

```
ExecStart=/usr/bin/spawn-fcgi -s /var/run/fcgiwrap.sock -P /var/
    run/fcgiwrap.pid -u http -g http -- /usr/sbin/fcgiwrap
ExecStop=/usr/bin/kill -15 $MAINPID

[Install]
WantedBy=multi-user.target
```

Listing 6.9: fcgiwrap.service from wiki.archlinux.org/index.php/gitweb

The standard subdomain file presented before will not work in this case because gitweb does not contains php scripts but only one cgi script.

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

server {
    server_name gitweb.local.pi;
    root /usr/share/gitweb/;

    autoindex on;
    index gitweb.cgi;

    location /gitweb.cgi {
        include fastcgi_params;
        gzip off;

        fastcgi_param SCRIPT_NAME /usr/share/gitweb/gitweb.cgi;
        fastcgi_param GITWEB_CONFIG  /etc/gitweb.conf;
        fastcgi_param GIT_PROJECT_ROOT /srv/git;
        fastcgi_pass  unix:/var/run/fcgiwrap.sock;
    }
}
```

Listing 6.10: Gitweb configuration for nginx

Finishing touch

After that, you will be able to access to your gitweb interface with the domain you defined – `gitweb.local.pi` in the example – but if do not see any projects although you set some of them in the directory you specified in *gitweb.conf*, try to launch git daemon.

```
$ /usr/lib/git-core/git-daemon --inetd --export-all --base-path
    =/srv/git
```

The last thing you can do is replace the default theme the one developed by

kogakure[2] which is a little bit more appealing.

```
# Save default theme
$ cp -r /usr/share/gitweb/static/ /usr/share/gitweb/static.old

# Retrieve kogakure theme, copy it in right folder
# and delete unused files
$ git clone https://github.com/kogakure/gitweb-theme.git
$ cp gitweb-theme/git* /usr/share/gitweb/static
$ rm -r gitweb-theme
```

Listing 6.11: Install kogakure gitweb theme

6.4.2 GitList: Simple GitHub clone

If you are looking for a git web interface which look like GitHub – with less side features – GitList is probably a good alternative for you.

This project is hosted by GitHub at github.com/klaussilveira/gitlist but as said in the README file, do not clone the repository and prefer download the packaged version on the GitList website at gitlist.org.

```
# Download
$ cd /usr/share/webapps
$ wget https://s3.amazonaws.com/gitlist/gitlist-0.5.0.tar.gz

# Extract and remove unused files
$ tar -xvf gitlist-0.5.0.tar.gz
$ rm gitlist-0.5.0.tar.gz
```

Listing 6.12: Download GitList

Let's declare a new subdomain for GitList, a sample is available in it's directory – `/usr/share/webapps/gitlist/INSTALL.md` – but here I will propose you a shorter solution. Note you have to use at least one of these two sample because GitList uses url rewriting.

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

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

    server_name gitlist.local.pi;

    location ~ \.php$ {
```

[2]More informations on his web site: kogakure.github.io/gitweb-theme/

```
        fastcgi_pass unix:/run/php-fpm/php-fpm.sock;
        fastcgi_index index.php;
        include fastcgi.conf;
    }

    location / {
        try_files $uri @gitlist;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
    add_header Vary "Accept-Encoding";
        expires max;
        try_files $uri @gitlist;
        tcp_nodelay off;
        tcp_nopush on;
    }

    location @gitlist {
        rewrite ^/.*$ /index.php;
    }
}
```

Listing 6.13: GitList configuration for nginx

Now you can access to GitList interface but you may have some errors messages and your are forced to fix these troubles.

```
# Message 1
# The "/usr/share/webapps/gitlist/cache" folder must be writable
# for GitList to run.
#
$ mkdir /usr/share/webapps/gitlist/cache # If not exist
$ chown http:http /usr/share/webapps/gitlist/cache

# Message 2
# Please, create the config.ini file.
$ cd /usr/share/webapps/gitlist/
$ mv config.ini-example config.ini

# Message 3
# Please, edit the config file and provide your repositories
    directory
# Solution: set repositories[] = '/srv/git/'
#
$ nano /usr/share/webapps/gitlist/config.ini
```

Listing 6.14: Fix GitList troubles

6.5 Gitolite: manage projects

Soon in libraries