# Linux tips

Some tips and little tutorials for linux. This is mainly tips I wrote for myself to not forget things I discovered through my travel inside the Linux world.

# Summary

# I. Linux tips

# 1. Mounting a File System of a remote computer

It can always be useful to be able to mount a File System from another computer. In here, we'll use `sshfs` to do it (other softwares like `ftpfs` exist).

## Preparing the field

First, you need to be able to connect on the remote computer with the `ssh` command:

```
> ssh user@some_computer
# if it doesn't work, try with the configured ssh port like this (let's say it's 2564):
> ssh user@some_computer -p 2564
```

If everything worked fine, we can move forward.

## Mounting the File System

In general, it's preferable to mount the remote directory inside the `/tmp` directory. So first:

```
mkdir /tmp/remote
```

Ok, now we just have to mount it:

```
sshfs user@some_computer:/the/remote/folder /tmp/remote/
```

And that's all! However, it's possible you might encounter rights issue. For this, we'll need to add more options. First, you'll need to get your remote folder user id and group id:

```
> ssh user@some_computer
> ls -l /the/remote/folder
drwxr-xr-x  3 imperio imperioland 4096 Dec 23 11:33 /the/remote/folder
> ls -ln /the/remote/folder
drwxr-xr-x  3 1000 1000 4096 Dec 23 11:33 /the/remote/folder
```

Create two files on your computer (not the remote one!) like this:

```
echo imperio:1000 > uidmap
echo imperioland:1000 > gidmap
```

Don't forget to enter your information and not mine! Then, we can retry to use `sshfs`:

```
sshfs -o allow_other,idmap=file,uidfile=`pwd`/uidmap,gidfile=`pwd`/gidmap,nomap=ignore user@some
```

Here you go!

# 2. Playing with sudoers file

This is "just" some explanations and examples about the sudoers file and how it works. For more information, don't hesitate to read the sudoers manual `man sudoers`.

Be **very** careful when you modify this file, it could create really BIG issues. To prevent most of them, it is **very** recommended to edit the sudoers file (located in `/etc/sudoers`) with the `visudo` command.

## Let's start!

For these explanations, we'll test with `insults` parameter (when you enter an invalid password for sudo command, it'll display some "messages").

If you want to modify defaults for everyone:

```
Defaults insults
```

For one (or more) user:

```
Defaults:user1 insults # for one
Defaults:user1,user2,user3 insults # for more
```

When you run as an other user (for example, when you use 'sudo -u user1 bash'):

```
Defaults>user1 insults
Defaults>user1,user2,user3 insults
```

You can as well do it with an host list (with '@') and a command list (with '!'):

```
Defaults@localhost insults # /!\ if you connect from another computer (with ssh for example), th
Defaults!/bin/ls insults
```

Just like for hosts/users/commands/run as, you can set multiple parameters, separated by ',':

```
Defaults@localhost insults,targetpw
```

However you can't mix default types:

```
Defaults@localhost:user1 insults # doesn't work!
```

## User specification

Now, if you want a more precise control over a specified user, you can use user specification. The base syntax is the following: `who where = (as_whom) what`. Example:

```
user1 ALL = (user2) /bin/ls
```

Here, the user `user1` on/from every computer may run `/bin/ls` only as `user2` (equivalent of `sudo -u user2 /bin/ls`).

You can also specify one (or more) group:

```
user1 ALL = (user2 : some_group, some_other_group) /bin/ls
```

Which means that the command can be run with either `user1` current group or `some_group` or `some_other_group`. So the following commands are all correct:

```
sudo -u user2 /bin/ls
sudo -u user2 -g some_group /bin/ls
sudo -g some_group /bin/ls
sudo -u user2 -g some_other_group /bin/ls
```

Just like groups, you can set multiple users and multiple commands at once:

```
user1 ALL = (user2, user3 : some_group, some_other_group) /bin/ls, /bin/grep
```

Or just one group without specifying a user:

```
user1 ALL = (: some_group, some_other_group) /bin/ls, /bin/grep
```

And you can also set specific commands for differents users/groups:

```
user1 ALL = (user2 : some_group) /bin/ls, (user3) /bin/grep
```

`/bin/ls` may be launched as `user2` or `some_group` and `/bin/grep` may be launched as `user3`.

And also, you can set those parameters for a group, a "run as" or a host:

```
%sudo ALL=(ALL) ALL # so every member of the sudo group has root access
```

## Tags

From the man: "A command may have zero or more tags associated with it. There are ten possible tag values: NOPASSWD, PASSWD, NOEXEC, EXEC, SETENV, NOSETENV, LOG_INPUT, NOLOG_INPUT, LOG_OUTPUT and NOLOG_OUTPUT. Once a tag is set on a Cmnd, subsequent Cmnds in the Cmnd_Spec_List, inherit the tag unless it is overridden by the opposite tag (in other words, PASSWD overrides NOPASSWD and NOEXEC overrides EXEC)."

I'll let you read the manual to know what each tag does. Now let's take some examples:

```
user1 ALL = NOPASSWD: /bin/ls, /bin/grep
```

In here, `user1` on/from any computer will be able to run `/bin/ls` and `/bin/grep` as `root` without authenticating himself.

Just like user specification, you can set multiple tags on one line:

```
user1 ALL = NOPASSWD: /bin/ls, PASSWD: /bin/grep
```

## Alias

You can declare four types of alias:

- User alias (wrote `User_Alias`)
- Run as alias (wrote `Runas_Alias`)
- Host alias (wrote `Host_Alias`)
- Command alias (wrote `Cmnd_Alias`)

An example will better than a speech:

```
User_Alias PEOPLE = user1, user2, user3

Defaults:PEOPLE insults
PEOPLE ALL = NOPASSWD: ALL # however, you'd better avoid to write this
```

Now, I guess you could easily give rights for `user1` to log as `root`, no?

```
user1 ALL=(ALL) ALL
```

## The '!' operator

Adding '!' before a command/variable/tag/whatever means it'll revert its effets. So `!ALL` means no one. You can put more than one, of course (so '!!' will very likely do nothing whereas '!!!' will certainly do the same as '!'). Example:

```
!PEOPLE ALL=(ALL) ALL # so all people not in PEOPLE will be able to log as root
```

## Mixup

This part will be short. You can mix tags and user specifications. Example:

```
user1 ALL=NOPASSWD: /bin/ls, (root)
```

However, be careful. In some cases, I had to do it in two steps (each operation on its own line) because `visudo` didn't recognized it (why? the mystery remains complete!).

## Add external files/directories

You can add other files and/or directory with `#include` and `#includedir`. It can be quite convenient if you don't want to modify original sudoers file. Example:

```
#include /etc/sudoers.local # include the sudoers.local file
#includedir /etc/sudoers.d # all files in this folder will be added
```

**Warning!!**: for `#includedir`, every files which end with '~' or contain a '.' character will be ignored. They also need to have `0440` rights.

For more details, read `man sudoers`, "Including other files from within sudoers" part.

## Check your sudoers files

Once you have finish to modify it (without `visudo`), you can check it with the following command:

```
sudo visudo -cf path/to/sudoers
```

Or you could use `sudo -l`.

Enjoy!

# 3. Setup a bridge connection with a linux VM

This was tested on VirtualBox and VMWare, I can't confirm for other VMs.

First, go to your targetted VM into "network settings". Create two interfaces (one host-only and one Bridge adapter) or just one (the Bridge adapter one seems to be the easier one). Once this is done, open a terminal and write:

```
> sudo route del default
> sudo dhclient eth1
```

Then try to ping your host from your VM (and vice-versa) to check if everything went correctly.

## Reminder

To get your ip (of the VM or of your computer): just `ifconfig`.

# 4. Having remote graphic applications with ssh

After failing to setup correctly openserver to get graphic access from another computer, I discovered the *X* option of ssh. All you need to do is to run a ssh server on the computer you want to connect on and that's good! You also have to allow the X11 forwarding on both server and client sides.

## Client side

Go to the *~/.ssh/config* file and set `ForwardX11` parameter to `yes` like this:

```
# some other parameters...

ForwardX11 yes

# ...
```

## Server side

Go to the */etc/ssh/sshd_config* file and set `ForwardX11` parameter to `yes` like this:

```
# some other parameters...

ForwardX11 yes

# ...
```

You also need to have `xauth` installed (but it's very uncommon to not have it).

And finally, you have to restart ssh service. In case you forgot how to do so:

```
> sudo /etc/init.d/ssh restart
# or:
> sudo service ssh restart
# on systemd:
> systemctl restart ssh
```

Then you can connect to the remote computer like this:

```
> ssh login@ip -X
```

For now, nothing different should have happened. But if you start a graphic application, it'll start directly on your computer.

# 5. Change user default login command

I needed to change the login command to execute a script of my own when a user logged in for the first time. At first, I changed the `.bashrc` file to execute it from there, but it had some bothering disadvantages. So I decided to go lower and finally found what I was looking for by changing directly `/etc/passwd`.

## /etc/passwd

This file contains a lot of useful information (such as encrypted password, users' login, user id (if 0, it's supposed to be root), group, login commands, home directory, etc...). It looks like this:

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

Each column of this file is separated by ':' and a line equals a row. So first is the login and last is the login command (I think you can guess pretty well what are the other fields). You can edit this file by hand if you want (even if I don't recommend it) or use commands which will do it for you (and avoid doing crap).

## chsh

`chsh` is certainly the best I know for doing this. I'll show how to use it with `sudo` command:

```
sudo chsh -s /bin/my_super_script the_user_i_want_to_modify
```

All login commands paths have to be **absolute**! Also, if you don't specify an user, `chsh` will be executed for the current user (so `root` here).

## useradd

In this case, it's only if you create a new user. `useradd` command has also a `-s` option which allow to change the login command as well. Example:

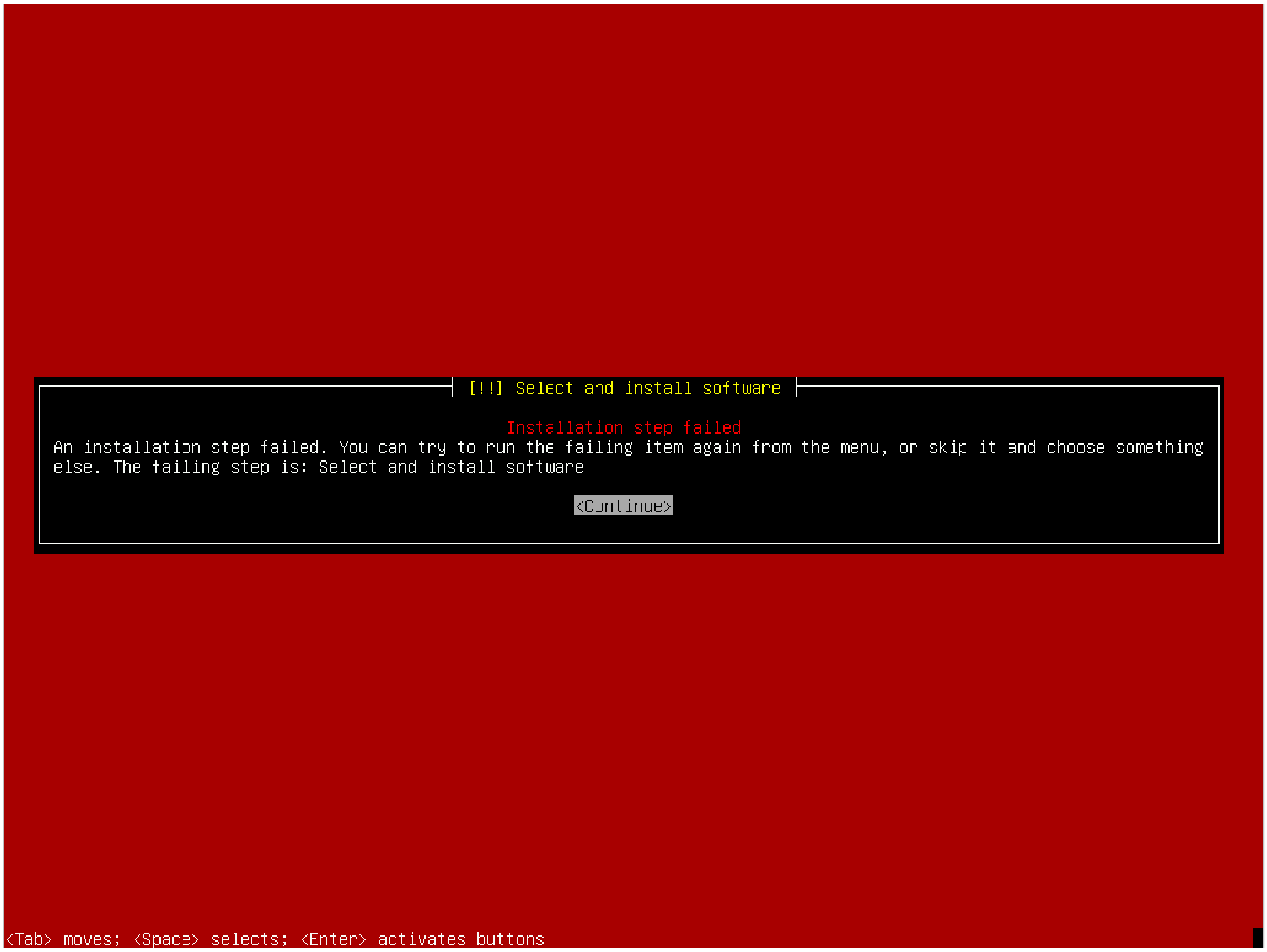
```
useradd -s /bin/my_super_script the_user_i_want_to_create
```

The login command path still must be **absolute**!

If you want more information, don't hesitate to take a look at their manual.

# 6. Get syslog when your linux installation fails

Sometimes, while modifying a linux installer, you might break it (it happened to the best of us) and get a wonderful red screen:

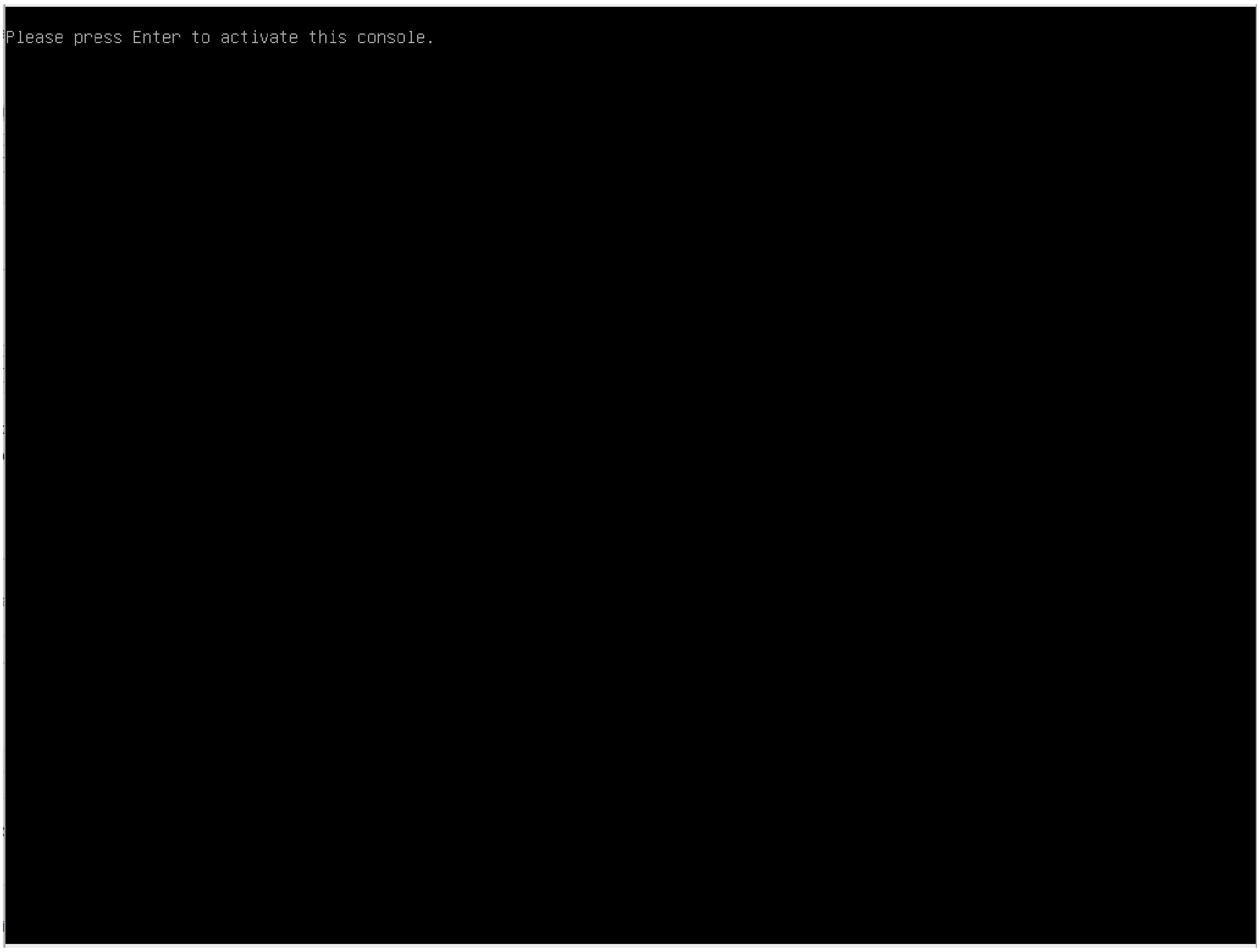


```
[!!] Select and install software

Installation step failed
An installation step failed. You can try to run the failing item again from the menu, or skip it and choose something
else. The failing step is: Select and install software

<Continue>

<Tab> moves; <Space> selects; <Enter> activates buttons
```

Problematic isn't it? Let's try to get the installation logs! To do so, I'll show two ways of doing it.

## Access to console

Once you get the red screen, you have to possibility to access another console (on mine, I have two). You can access it with the keys ALT+F2 (and F3 for the second console for me):



```
Please press Enter to activate this console.
```

From there, you'll have a minimal shell which is more than enough for what we need. The logs *should* be located in `/var/log/syslog`. You should at least have an access to `nano` text editor (once again, it's minimal but should be enough) if you want to explore your log file (`less` and `vim` weren't available for me).

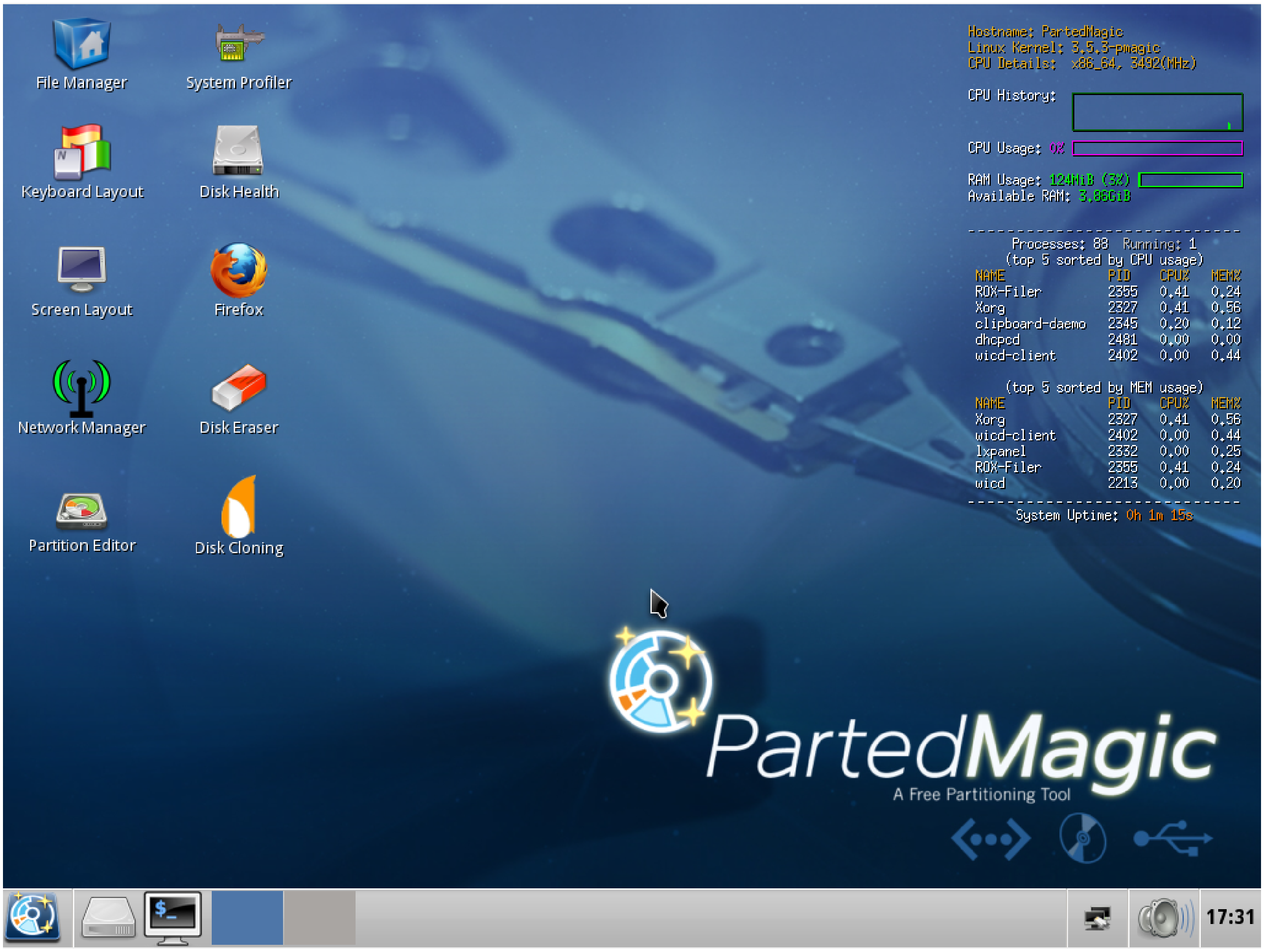
If you want to take a look to what the installer installed, you need to go into the `/target` folder. Also, if you want a bit more "control" or a better shell. You can do it like this:

```
> chroot /target
> bash
```

And here you go!

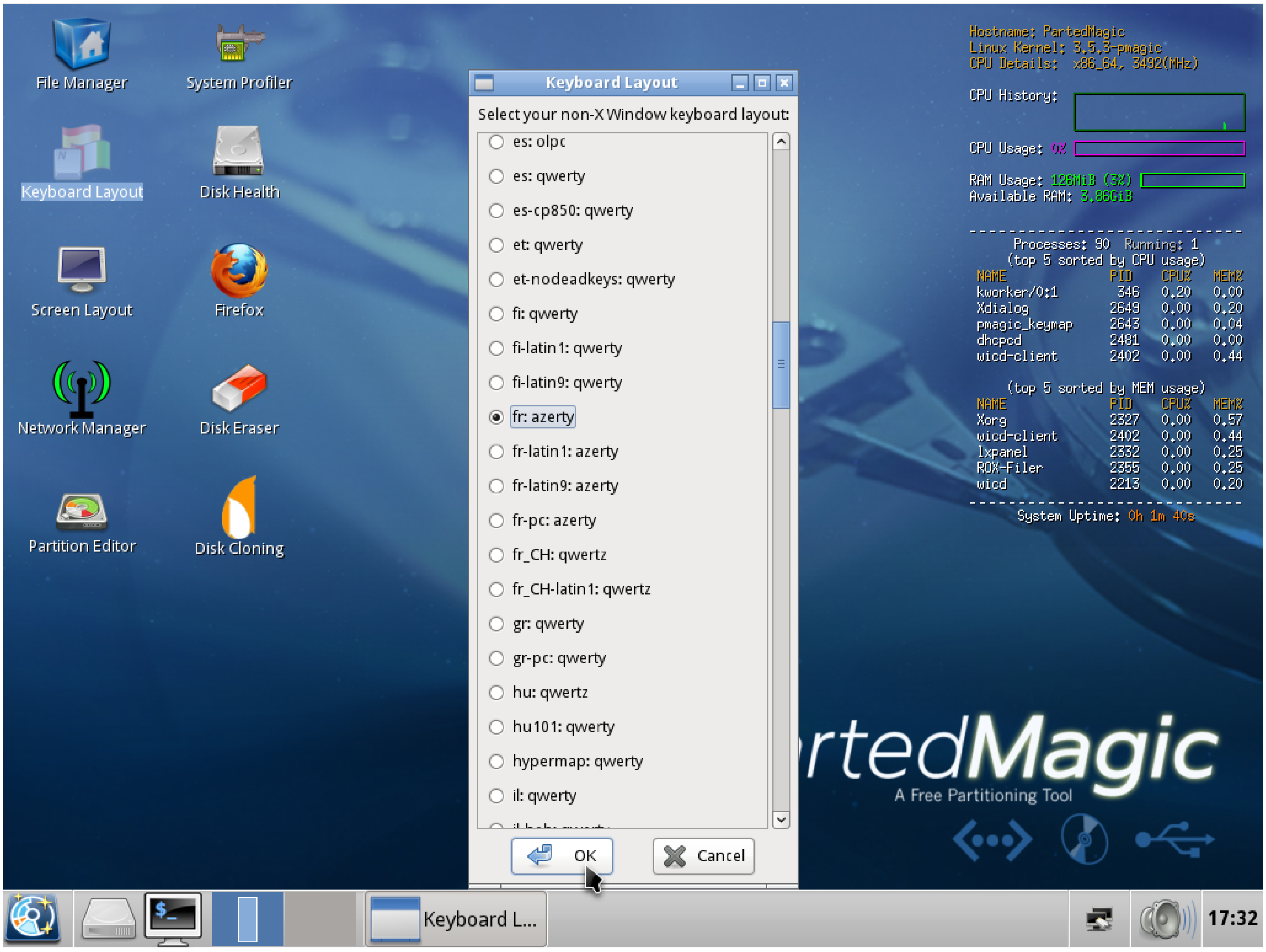
## Using a live-cd

Any live-cd would do the trick, but I like PartedMagic, so let's go for it. I use this version of it (if you're too lazy to search :p). An old one, but it works fine.



```
Hostname: PartedMagic
Linux Kernel: 3.5.3-pmagic
CPU Details:  x86_64, 3492(MHz)

CPU History:

CPU Usage: 0%

RAM Usage: 124MiB (3%)
Available RAM: 3.86GiB

------------------------------------
      Processes: 88  Running: 1
     (top 5 sorted by CPU usage)
 NAME               PID    CPU%    MEM%
 ROX-Filer          2355   0.41    0.24
 Xorg               2327   0.41    0.56
 clipboard-daemo    2345   0.20    0.12
 dhcpcd             2481   0.00    0.00
 wicd-client        2402   0.00    0.44

     (top 5 sorted by MEM usage)
 NAME               PID    CPU%    MEM%
 Xorg               2327   0.41    0.56
 wicd-client        2402   0.00    0.44
 lxpanel            2332   0.00    0.25
 ROX-Filer          2355   0.41    0.24
 wicd               2213   0.00    0.20
------------------------------------
       System Uptime: 0h 1m 15s
```

First thing to do (if needed), change the keyboard layout:



Keyboard Layout

Select your non-X Window keyboard layout:

- ○ es: olpc
- ○ es: qwerty
- ○ es-cp850: qwerty
- ○ et: qwerty
- ○ et-nodeadkeys: qwerty
- ○ fi: qwerty
- ○ fi-latin1: qwerty
- ○ fi-latin9: qwerty
- [✓ fr: azerty]
- ○ fr-latin1: azerty
- ○ fr-latin9: azerty
- ○ fr-pc: azerty
- ○ fr_CH: qwertz
- ○ fr_CH-latin1: qwertz
- ○ gr: qwerty
- ○ gr-pc: qwerty
- ○ hu: qwertz
- ○ hu101: qwerty
- ○ hypermap: qwerty
- ○ il: qwerty

```
Hostname: PartedMagic
Linux Kernel: 3.5.3-pmagic
CPU Details:  x86_64, 3492(MHz)

CPU History:

CPU Usage: 0%

RAM Usage: 126MiB (3%)
Available RAM: 3.86GiB

-----------------------------------
     Processes: 90  Running: 1
    (top 5 sorted by CPU usage)
 NAME             PID    CPU%   MEM%
 kworker/0:1       346   0.20   0.00
 Xdialog          2649   0.00   0.20
 pmagic_keymap    2643   0.00   0.04
 dhcpcd           2481   0.00   0.00
 wicd-client      2402   0.00   0.44

    (top 5 sorted by MEM usage)
 NAME             PID    CPU%   MEM%
 Xorg             2327   0.00   0.57
 wicd-client      2402   0.00   0.44
 lxpanel          2332   0.00   0.25
 ROX-Filer        2355   0.00   0.25
 wicd             2213   0.00   0.20
-----------------------------------
      System Uptime: 0h 1m 40s
```

Once you find the good one, just press "Ok"/"Yes" buttons until it stops asking.

Time to get into our system! To see what available partitions you have, run the `lvscan` command into a console. Then just mount it and find the log file. So for example, in my case I'll do:

```
> mkdir /tmp/t # any other name would be fine
> mount /dev/vg00/lvroot /tmp/t
> cd /tmp/t
> less var/log/syslog
```

You can also check around what your installer installed and stuff. PartedMagic live-cd also provides useful tools to work on system things (partitions for example). Don't hesitate to take a look to what it offers!

# 7. Some bash tips

I know there are other (better) shells, but bash is certainly the most common one so I'll show tips for it! You can make all the changes that'll be presented permanent by adding them into the file ".bashrc".

## Aliases

Quite simple, I guess everyone knows how it works. To make it simple:

```
alias [alias_name]="[command]"
```

Which gives:

```
alias ll="ls -l"
```

Aliases can use other aliases as well:

```
alias ls="ls --color=auto"
alias ll="ls -l" # so here, "ls" will become "ls --color=auto"
```

And I think it's more or less all there is to say for this part.

## History

Command history can have some really nice features. For example, you use oftenly the "ls" command and you'd prefer that it wouldn't get reentered in the history? This is possible. Just do the following:

```
export HISTCONTROL=erasedups
```

And so, if the command you just entered is already inside your history, the old occurence will be removed.

## Prompt

To change your shell prompt, you have to change PS1 variable (there is also PS2, PS3 and PS4, but only PS1 and PS2 are really used). Example:

```
export PS1="> "
```

And now your prompt should be "> ".

Nice right? Now what about displaying your name and the current directory?

```
export PS1="\u @ \w > "
```

Just like you certainly guessed, "\u" is to display the current user and "\w" is for the current directory.

A lot of other "tools" like this are available. For example, you can also see what was the value returned by the last executed command with "$?". Example:

```
export PS1="\$? > "
export PS1='$? > ' # this line is the same as the previous one, the difference is that the simpl
```

Just look around on the Internet what flags bash provides.

Now, let's see how to put some colors! First, test this:

```
export PS1='\[\033[01;32m\]Hello!\[\033[00m\]> ' # and use simple quotes!
```

As you can see, "Hello!" is green. For more information about colors, take a look to "man console_codes". In our previous example, '[\033[01;32m]' means bold green ("01" is for bold and "32" is for green"), '[\033[00m]' is for going back to normal. If you don't put it at the end of your prompt, the text you'll enter will have this color as well.

Now you know the basics for some bash tips. Before leaving you, I'll show you the prompt I'm using:

```
"\[\033[01;31m\]\u\[\033[01;32m\]@\[\033[01;33m\]\$PWD >\[\033[00m\] " # ("\h" stands for hostna
```

## Redirect all outputs

It can be convenient to get all outputs (stdio and stderr) in one stream. You can do it with `|&`. Example:

```
> find . |& grep "not found"
```

## Adding colors to man pages

Pretty easy to do:

```
#
#   L E S S   C O L O R S   F O R   M A N   P A G E S
#

# CHANGE FIRST NUMBER PAIR FOR COMMAND AND FLAG COLOR
# currently 0;33 a.k.a. brown, which is dark yellow for me
export LESS_TERMCAP_md=$'\E[0;33;5;74m'  # begin bold

# CHANGE FIRST NUMBER PAIR FOR PARAMETER COLOR
# currently 0;36 a.k.a. cyan
export LESS_TERMCAP_us=$'\E[0;36;5;146m' # begin underline

# don't change anything here
export LESS_TERMCAP_mb=$'\E[1;31m'       # begin blinking
export LESS_TERMCAP_me=$'\E[0m'           # end mode
export LESS_TERMCAP_se=$'\E[0m'           # end standout-mode
export LESS_TERMCAP_so=$'\E[38;5;246m'    # begin standout-mode - info box
export LESS_TERMCAP_ue=$'\E[0m'           # end underline

#########################################
# Colorcodes:
# Black       0;30     Dark Gray     1;30
# Red         0;31     Light Red     1;31
# Green       0;32     Light Green   1;32
# Brown       0;33     Yellow        1;33
# Blue        0;34     Light Blue    1;34
# Purple      0;35     Light Purple  1;35
# Cyan        0;36     Light Cyan    1;36
# Light Gray  0;37     White         1;37
#########################################
```

# 8. Playing with Debian packages

First, let's take a look to a Debian package:

## Inside .deb files

So, I think you guessed it but `.deb` file are Debian package files. Now let's see a bit how they work (let's take libm.deb for this example):

```
> dpkg-deb -x libm.deb libm
```

This line will extract data from `libm.deb` into a `libm` folder. Now we need to extract Debian content as well:

```
> dpkg-deb -e libm.deb libm/DEBIAN
```

No we have access to all the content. First thing to know, folders (except DEBIAN) will be copy as-is in the system (from "/"). So if you have `./libm/usr/lib/libm.so`, it means the `libm.so` file will be copied into `/usr/lib/libm.so`. Nothing complicated in here.

The interesting part is in `./libm/DEBIAN`. You should have at least a `checksums` and a `control` files. The first one is just a file containing checksums of files provided by this package (but this isn't mandatory) whereas `control` contains all information the package manager might need to know about this package. I let you take a look to this manual for more information about it.

## Merging packages or add more files into it

The interesting thing in here is that you can totally merge different packages into one by just extracting other packages folders (except the `DEBIAN` one!) into the first one. Don't forget to add `DEBIAN/control` useful information like dependencies or conflicts!

The same goes if you want to add other libraries that aren't in the package. Just copy-paste them in the correct folder (the one into you want them to be installed) and that's all.

## Creating a debian package from original package and a patch

It's common to have an "original" package file and a Debian patch provided alongside. It's not very complicated to build it in order to then install it but it's always good to know! So let's start with `libm_orig.tar.gz` and `libm_debian.tar.gz`.

First, let's extract `libm_orig.tar.gz`:

```
> tar xzf libm_orig.tar.gz
```

Now let's extract `libm_debian.tar.gz` into it:

```
> cd libm_orig
> tar xzf libm_debian.tar.gz
```

Now you just need to build the package:

```
> dpkg-buildpackage -us -uc
```

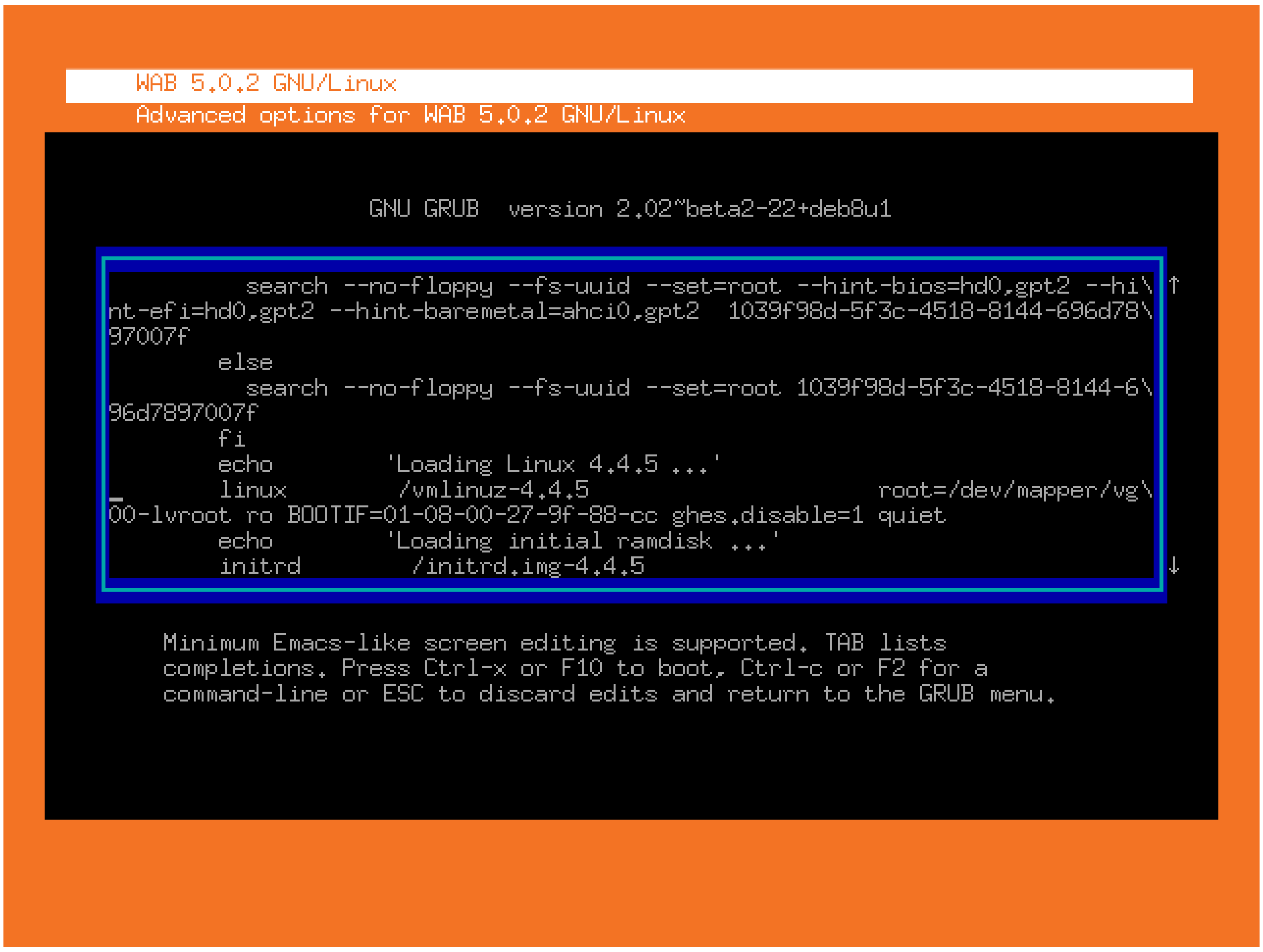
The `us` and `uc` options are for generating non-signed package. Read the manual if you want to build a signed package. :p

# 9. Modify files of your linux without booting on it

If you broke your linux and need to undo a change, you can do it by either using a bootable key and change the file through it like showed in here or avoid running `/sbin/init`. You wonder why? Then read what follows!

## Change boot line in grub

The first thing to do is to edit the boot line of your distribution when you're on the grub. Generally, it's the key 'e' but it might change, depending on your system. Then go to the line starting by "linux".



```
      WAB 5.0.2 GNU/Linux
      Advanced options for WAB 5.0.2 GNU/Linux

                        GNU GRUB  version 2.02~beta2-22+deb8u1

          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt2 --hi\
nt-efi=hd0,gpt2 --hint-baremetal=ahci0,gpt2  1039f98d-5f3c-4518-8144-696d78\
97007f
        else
          search --no-floppy --fs-uuid --set=root 1039f98d-5f3c-4518-8144-6\
96d7897007f
        fi
        echo         'Loading Linux 4.4.5 ...'
        linux        /vmlinuz-4.4.5                        root=/dev/mapper/vg\
00-lvroot ro BOOTIF=01-08-00-27-9f-88-cc ghes.disable=1 quiet
        echo         'Loading initial ramdisk ...'
        initrd        /initrd.img-4.4.5

     Minimum Emacs-like screen editing is supported. TAB lists
     completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a
     command-line or ESC to discard edits and return to the GRUB menu.
```

In this line, add at the end (with a whitespace before):

```
init=/bin/sh
```

(Or another shell, like `bash` if you know its binary path.)

Or even more simple:

```
break
```

Then press the button to boot (in my case, this is `CTRL-X` or `F10`). Once done, you should be on a command line. Congrats, we have prevent the `init` program to run!

## Change files

We can now explore our root partition but not modify files. If you want to modify files, you'll have to remount the root partition with write rights:

```
mount -o remount,rw /
```

We can now edit files as we want. If you want to mount other partitions, run (but you'll need a write access):

```
vgchange -ay
```

Then check the format of the partition you want to mount with:

```
blkid
```

So it'll display in my case:

```
> blkid
/dev/sda1: PARTUUID="c3554sd-sdfdsq-a454-01154875qs"
/dev/mapper/vg00-lvroot: UUID="05547454-c54d-4999-bfbf-cc54557574" TYPE="ext3"
/dev/mapper/vg00-lvopt: UUID="5af4d5ee-aaa4-9ef6-e254-dde475s47d" TYPE="ext3"
/dev/mapper/vg00-lvvar: UUID="4599fab2-a4b7-f65e-5d4e-c47d6654ec" TYPE="ext4"
/dev/mapper/vg00-lvsub: UUID="19fab277-47b1-5dfe-4235-d66dfe54ec" TYPE="ext4"
```

And then mount them:

```
mount -t [format] /dev/vg00/[partition] /[where]
```

So in my case (`sub` is a child of `var`):

```
mkdir /mnt
mount -t ext3 /dev/vg00/lvroot /mnt
mount -t ext3 /dev/vg00/lvopt /mnt/opt
mount -t ext4 /dev/vg00/lvopt /mnt/var
mount -t ext4 /dev/vg00/lvopt /mnt/var/sub
```

Now you have everything you want in `/mnt`. Then you `chroot` in it to make your life a bit easier:

```
chroot /mnt
```

# 10. Low level partition formatting

It can be useful to perform a low level partition formatting. However, it might be problematic if you want to format some specific partition (the one where `/home` is mounted for example). In here, I'll just show you how to do it without the "full boot". First, you need to boot just like explained in the Modify files of your linux without booting on it chapter. Once done, we can start!

## Get partitions' list and their sizes

This is the easy part. Just use the `fdisk` command like this:

```
> fdisk -l /dev/sda
# or just:
# > fdisk -l
```

At the end of the output, you should have the following:

```
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x136b232d

Device     Boot    Start      End  Sectors Size Id Type
/dev/sda1  *        2048 77596671 77594624  37G 83 Linux
/dev/sda2       77598718 94369791 16771074   8G  5 Extended
/dev/sda5       77598720 94369791 16771072   8G 82 Linux swap / Solaris
```

The important pieces of information here are:

```
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

And the `Sectors` column of the partitions.

## Formatting

Now, let's see how to format one of the previous partitions. However: *BE VERY CAREFUL!!!*. Once done, you cannot get back the erased data!

In this example, I'll format the `/dev/sda1` partition.

First, get the `Sectors` number corresponding to the partition you want to format (77594624 for me). Then take the I/O optimal size (512 here) and the Sector logical size (512 once again).

Now that you have your 3 numbers, you'll need to do 2 things:

- First, divide the I/O optimal size by the Sector logical size (so for me: 512 / 512).
- Second, divide the Sectors number by the result of the previous operation (so for me: 77594624 / 1).

All good? Great, now we can actually start the formatting!

First let's take a look to what we'll do:

```
dd bs=[I/O optimal size] [result of the second division (77594624 / 1)] if=[device on which we'l
```

And now in practice:

```
dd bs=512 count=77594624 if=/dev/zero of=/dev/sda1
```

If you want a bit more security, you can replace `/dev/zero` by `/dev/urandom` if you want to have your partition filled with random numbers instead of zeros.

This operation can take quite a while depending on your hard drive writing speed, so don't worry if it takes time!

# 11. Setting up an IRC bouncer

In here, I'll explain how to install an IRC bouncer (`znc` to be precise). Let's go!

## Getting pieces

First, install `znc` (I'll use `apt` in this tutorial):

```
# try this line first:
sudo apt-get install znc znc-extra
# if you have an error about znc-extra, then try this one:
sudo apt-get install znc
```

If everything in here went fine, then great!

## Setting up

Now that you have `znc`, you need to configure it. To make things easier, I'll comment each step:

```
> znc --makeconf
[ .. ] Checking for list of available modules...
[ >> ] ok
[ ** ] Building new config
[ ** ]
[ ** ] First let's start with some global settings...
[ ** ]
[ ?? ] What port would you like ZNC to listen on? (1025 to 65535): 3333
```

In here, you need to pick a port on which your bouncer will listen (it doesn't have to match the irc servers you want to connect to). I chose `3333`.

```
[ ?? ] Would you like ZNC to listen using SSL? (yes/no) [no]: yes
```

For security reasons, I enabled `SSL`.

```
[ ?? ] Would you like ZNC to listen using both IPv4 and IPv6? (yes/no) [yes]:
```

Doesn't matter so I used `yes`.

```
[ .. ] Verifying the listener...
[ >> ] ok
[ ** ]
[ ** ] -- Global Modules --
[ ** ]
[ ** ] +-----------+----------------------------------------------------------+
[ ** ] | Name      | Description                                              |
[ ** ] +-----------+----------------------------------------------------------+
[ ** ] | partyline | Internal channels and queries for users connected to znc |
[ ** ] | webadmin  | Web based administration module                          |
[ ** ] +-----------+----------------------------------------------------------+
[ ** ] And 13 other (uncommon) modules. You can enable those later.
[ ** ]
```

This part is about the modules. I'll let you choose yourself and just put what I chose.

```
[ ?? ] Load global module <partyline>? (yes/no) [no]: yes
[ ?? ] Load global module <webadmin>? (yes/no) [no]:
[ ** ]
```

The interesting part is now starting. You need to setup the user that'll be used to connect to your bouncer:

```
[ ** ] Now we need to set up a user...
[ ** ]
[ ?? ] Username (AlphaNumeric): imperio
[ ?? ] Enter Password:
[ ?? ] Confirm Password:
[ ?? ] Would you like this user to be an admin? (yes/no) [yes]:
[ ?? ] Nick [imperio]:
[ ?? ] Alt Nick [imperio_]:
[ ?? ] Ident [imperio]:
[ ?? ] Real Name [Got ZNC?]:
```

So until this point, just basic information. Try to not mess with them! The following options aren't *that* useful so you can keep default values, it won't have much impact...

```
[ ?? ] Bind Host (optional):
[ ?? ] Number of lines to buffer per channel [50]: 1000
[ ?? ] Would you like to clear channel buffers after replay? (yes/no) [yes]: no
[ ?? ] Default channel modes [+stn]:
[ ** ]
```

We're now back in the modules, but on the user's side now:

```
[ ** ] -- User Modules --
[ ** ]
[ ** ] +-------------+-----------------------------------------------------------------------
[ ** ] | Name        | Description
[ ** ] +-------------+-----------------------------------------------------------------------
[ ** ] | chansaver   | Keep config up-to-date when user joins/parts
[ ** ] | controlpanel | Dynamic configuration through IRC. Allows editing only yourself if you'r
[ ** ] | perform     | Keeps a list of commands to be executed when ZNC connects to IRC.
[ ** ] | webadmin    | Web based administration module
[ ** ] +-------------+-----------------------------------------------------------------------
[ ** ] And 21 other (uncommon) modules. You can enable those later.
[ ** ]
[ ?? ] Load module <chansaver>? (yes/no) [no]: yes
[ ?? ] Load module <controlpanel>? (yes/no) [no]:
[ ?? ] Load module <perform>? (yes/no) [no]: yes
[ ?? ] Load module <webadmin>? (yes/no) [no]:
[ ** ]
```

The next part is very important. You'll now fulfill the IRC server(s) on which you want your bouncer to connect to. Do it very carefully!

```
[ ?? ] Would you like to set up a network? (yes/no) [no]: yes
```

So obviously you answered `yes`. Now the next thing is simple: if you want to connect to `freenode`, put `freenode`, otherwise put `efnet`.

```
[ ?? ] Network (e.g. `freenode' or `efnet'): efnet
[ ** ]
```

Other modules, but for network this time.

```
[ ** ] -- Network Modules --
[ ** ]
[ ** ] +-------------+------------------------------------------------------------------------
[ ** ] | Name        | Description
[ ** ] +-------------+------------------------------------------------------------------------
[ ** ] | chansaver   | Keep config up-to-date when user joins/parts
[ ** ] | keepnick    | Keep trying for your primary nick
[ ** ] | kickrejoin  | Autorejoin on kick
[ ** ] | nickserv    | Auths you with NickServ
[ ** ] | perform     | Keeps a list of commands to be executed when ZNC connects to IRC.
[ ** ] | simple_away | This module will automatically set you away on IRC while you are disconne
[ ** ] +-------------+------------------------------------------------------------------------
[ ** ] And 20 other (uncommon) modules. You can enable those later.
[ ** ]
[ ?? ] Load module <chansaver>? (yes/no) [no]: yes
[ ?? ] Load module <keepnick>? (yes/no) [no]: yes
[ ?? ] Load module <kickrejoin>? (yes/no) [no]: yes
[ ?? ] Load module <nickserv>? (yes/no) [no]:
[ ?? ] Load module <perform>? (yes/no) [no]: yes
[ ?? ] Load module <simple_away>? (yes/no) [no]: yes
[ ** ]
```

We're now going into the interesting stuff: setting up the IRC servers we want to connect to:

```
[ ** ] -- IRC Servers --
[ ** ] Only add servers from the same IRC network.
[ ** ] If a server from the list can't be reached, another server will be used.
[ ** ]
[ ?? ] IRC server (host only): irc.mozilla.org
[ ?? ] [irc.mozilla.org] Port (1 to 65535) [6667]:
[ ?? ] [irc.mozilla.org] Password (probably empty):
[ ?? ] Does this server use SSL? (yes/no) [no]:
[ ** ]
[ ?? ] Would you like to add another server for this IRC network? (yes/no) [no]:
[ ** ]
[ ** ] -- Channels --
[ ** ]
[ ?? ] Would you like to add a channel for ZNC to automatically join? (yes/no) [yes]: #rust-docs
[ ?? ] Would you like to add a channel for ZNC to automatically join? (yes/no) [yes]:
[ ?? ] Channel name: #rust-docs
[ ?? ] Would you like to add another channel? (yes/no) [no]: yes
[ ?? ] Channel name: #rust-fr
[ ?? ] Would you like to add another channel? (yes/no) [no]: yes
[ ?? ] Channel name: #rust-internals
[ ?? ] Would you like to add another channel? (yes/no) [no]: yes
[ ?? ] Channel name: #servo
[ ?? ] Would you like to add another channel? (yes/no) [no]: yes
[ ?? ] Channel name: #winapi
[ ?? ] Would you like to add another channel? (yes/no) [no]: yes
[ ?? ] Channel name: #hyper
[ ?? ] Would you like to add another channel? (yes/no) [no]:
[ ?? ] Would you like to set up another network? (yes/no) [no]:
[ ** ]
[ ?? ] Would you like to set up another user? (yes/no) [no]:
[ .. ] Writing config [/home/pi/.znc/configs/znc.conf]...
[ !! ] This config already exists.
[ ?? ] Would you like to overwrite it? (yes/no) [no]: yes
[ .. ] Overwriting config [/home/pi/.znc/configs/znc.conf]...
[ >> ] ok
[ ** ]
[ ** ]To connect to this ZNC you need to connect to it as your IRC server
[ ** ]using the port that you supplied.  You have to supply your login info
[ ** ]as the IRC server password like this: user/network:pass.
[ ** ]
[ ** ]Try something like this in your IRC client...
[ ** ]/server <znc_server_ip> +3333 imperio:<pass>
[ ** ]And this in your browser...
[ ** ]https://<znc_server_ip>:3333/
[ ** ]
[ ?? ] Launch ZNC now? (yes/no) [yes]:
[ .. ] Opening config [/home/pi/.znc/configs/znc.conf]...
[ >> ] ok
[ .. ] Loading global module [partyline]...
[ >> ] [/usr/lib/znc/partyline.so]
[ .. ] Binding to port [+3333]...
[ >> ] ok
[ ** ] Loading user [imperio]
[ ** ] Loading network [efnet]
[ .. ] Loading network module [chansaver]...
[ >> ] [/usr/lib/znc/chansaver.so]
[ .. ] Loading network module [keepnick]...
[ >> ] [/usr/lib/znc/keepnick.so]
[ .. ] Loading network module [kickrejoin]...
[ >> ] [/usr/lib/znc/kickrejoin.so]
[ .. ] Loading network module [perform]...
[ >> ] [/usr/lib/znc/perform.so]
[ .. ] Loading network module [simple_away]...
[ >> ] [/usr/lib/znc/simple_away.so]
[ .. ] Adding server [irc.mozilla.org 6667 ]...
[ >> ] ok
[ .. ] Loading user module [chansaver]...
[ >> ] [/usr/lib/znc/chansaver.so]
[ .. ] Loading user module [perform]...
[ >> ] [/usr/lib/znc/perform.so]
[ .. ] Forking into the background...
[ >> ] [pid: 2648]
[ ** ] ZNC 1.4 - http://znc.in
```

And that's mostly all.

## Connecting to the bouncer

To connect to you bouncer, you'll need to do a few things in your IRC client:

- Enable SSL.
- Accept invalid SSL certificate.
- Set server password.

For the last point, it has to follow a precise format. So in my case, my bouncer user is `imperio`, I chose the `efnet` network (since I'm not connecting to freenode) and then I chose `poney` as password. So my server password will have to be:

```
imperio/efnet:poney
```

Simple!

## Accessing to the bouncer from outside

If your bouncer has been set in a local network, I guess you want to make it accessible from outside? The only way to do it is to open a port from your internet box provider. I'll let you look into it.

## znc commands

A few commands from znc can be useful like:

- `znc --makepass`: to reset the bouncer user password.
- `znc --makeconf`: to generate a config file (or overwrite it) interactively.

It's important to note that the configuration of your znc bouncer is generally located into `~/.znc/configs/znc.conf`. Try to not mess this file!

## Automatic startup

In case you want to have znc to start when you boot and you lost the config file, here it is:

```
[Unit]
Description=ZNC - IRC Bouncer
Requires=nss-user-lookup.target
After=network-online.target nss-user-lookup.target

[Service]
User=pi
ExecStart=/usr/bin/znc --foreground
Restart=on-failure
KillSignal=SIGINT

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

Put it into `/etc/systemd/system/znc.service`, then:

```
sudo systemctl enable znc
sudo systemctl start znc
```

# 12. Store passwords (debian based solution)

It can be (really) annoying to re-enter your password everytime you want to push a commit with git (or use another application using passwords). Luckily, there is a solution for this.

## Install libgnome-keyring-dev

```
sudo apt-get install libgnome-keyring-dev
sudo make --directory=/usr/share/doc/git/contrib/credential/gnome-keyring
```

Now let's setup the credential:

```
git config --global credential.helper /usr/share/doc/git/contrib/credential/gnome-keyring/git-cr
```

You should be all set now!