Ace Guide to installing and setting up Arch/Multi-boot system

I have spent over the past 3 month going from an avid Ubuntu user to an experienced Arch user. I have installed Arch Linux literally more than 30-40 times now, always referring to this tutorial to speed up the process. This obviously isn't meant for regular users, not even people who just use Ubuntu for gcc/g++. This tutorial is meant for people who want to know Linux inside out, and are enthusiastic enough to change their daily driver OS to Arch Linux.

This tutorial isn't a LFS guide, I personally found LFS to be tiring, and requiring too much time. But instead, at the end of this tutorial, you would have a powerful operating system, whose all relevant portions YOU installed. You would know how to handle partitions like a pro, even when you have multiple OS. You would perhaps not even want to install Ubuntu again (consider elementaryOS if you want a better looking version of Ubuntu, installs exactly the same way, but in the end its still the same.)

I'm going to format all my Linux partitions right now (I'm not a fan of removing OS again and again, but somehow I always end up wanting some change). So here goes:

1. Setting up partitions:

Back up all your data first. Especially from the partitions you want to modify. I had already done that. Then open your linux distro (whichever you used before this), and make a bootable Arch linux USB Stick as follows:

- Download the Arch iso from https://www.archlinux.org/download. I used the IIT Kanpur Mirror, and downloaded the latest 64 Bit (x86-64) iso. 32 Bit users use i386, but I highly doubt anyone wants to use that.
- Open a terminal.
- Do lsblk and note the 'Name' of your USB Stick. In my case it was /dev/sdb (Don't use /dev/sdb1 etc).
- Do: sudo dd bs=4M if=/pathtoyourArch.iso of=/dev/sdX && sync

Now boot to Windows.

Let me describe my case:

I have a 1 TB hard disk.

I have kept 400 GB for my Windows partition (which is excessively large, I know. But I couldn't shrink it further initially, even after defragmenting (which basically moves all your data to the start of the partition, so that all the data is now in a line, with no blank memory spaces in between. This speeds up the partition, and lets you cut the partition from the end. Besides, I need that space for games ;))

I have a 240 GB partition which contains my Data I share between OS. My music library, tutorials, files and documents, Images etc.

I kept 236 GB unallocated. This would be my playground. So, Arch goes at the top (so that I can give it more space from Data drive if ever needed). 110GB seems fine. The remaining seems just fine for my future experiments (I plan to test out Gentoo, MacOS, LFS, BSD in due time. 'Wink'). You can give more space to your Data drive/Ubuntu/Arch if you aren't as adventurous as me (or don't want to make your laptop cry).

To accomplish this, I used the default Windows partition manager (bring it up by typing 'partition' in the charms bar in Windows 8.) I would recommend AOMEI Partition Assistant Standard Edition to people who require any more work on their partitions (it helps in almost every case where the default one fails, people who happen to be stuck on a dynamic hard-drive can try the paid Professional version of it to switch back to basic if they need to). But the default windows partition manager ensures that if you mess up, you at least have a standard way to get things normal.

Now format the partitions as NTFS (doesn't matter).

2. Booting with your USB Stick and getting things ready

Turn off fast-boot in windows. Restart, and open the BIOS settings, and turn off secure-boot.

Then boot with your pen-drive, it would display as 'EFI' something in the boot menu. Select the first option in the Arch menu. Wait a little, and there you have, an Arch installation running on your pen drive. It runs in root at the moment.

Do lsblk and identify the partition you want to install your arch on. In my case it was /dev/sdb7.

Format it as EXT4 by:

➔ mkfs.ext4 /dev/partition

Mount it in /mnt as:

➔ mount /dev/partition /mnt.

Now here comes the challenge, connecting to the internet. Read this fully before starting.

For most laptops, the WiFi drivers should work by stock. If they do, and if you have a DHCP WiFi connection nearby (or if you can get hold of a WiFi hotspot) , there is a simple way: Run wifi-menu. Select your network and type your password. Then run

→ ping –c 3 www.google.com

It should return the 3 packets. If it works, then congrats. You have a working net connection.

If no, and you have static IP configuration in your area, you can configure it using dhcpcd as follows.

Type vi /etc/dhcpcd.conf #(non-vim users please type 'nano' in place of 'vi')

Add new lines in the end:

```
profile new_profile
static ip_address=172.***.***.***
static routers=***.***.***.***   #Gateway
static domain_name_servers=***.***.***.***

interface enp9s0          #would vary. Check with 'ip link'
fallback static new_profile
```

To start the network, type:

→dhcpcd start enp9s0 #change the adapter name

To restart, first kill dhcpcd with 'pkill dhcpcd', then do the above again.

If you happen to be in a campus which has its own Arch Linux server, like mine, you just go to a place with a DHCP LAN connection (CSE building works best in IITk, with speeds touching 100 MBPS), and try pinging the local network right away (I ping webmail.iitk.ac.in).

Now you better select an awesome mirror near your location, it really helps, especially if you have a server right in your campus (Like IIT Kanpur). For that, type

➔ vi /etc/pacman.d/mirrorlist

I searched for India, cut those two lines, and placed them directly at the top of list, just before the France entry. To ensure that it doesn't fall back to a slow mirror, I also moved the other Indian entry (iitm) to the top below the iitk one. Do :wq to exit.

3. Installing the OS

Here comes the cool part, we are going to strap-up our new OS with Arch. Install the base and base-devel packages with:

➔ pacstrap /mnt base
➔ pacstrap /mnt base-devel

You now need to generate an fstab file, used for mounting partitions on boot. Use:

➔ genfstab –p /mnt >> /mnt/etc/fstab

It basically adds an entry to /mnt/etc/fstab that tells it to mount this partition on boot as root.

And now, chroot, we access the root of our in-the-making Arch right from this USB:

➔ arch-chroot /mnt /bin/bash

I added bash because by default it uses sh, and it doesn't suit my keyboard.

Set the name of your PC:

➔ echo SakshamPC > /etc/hostname

Set your time zone. In my case it was:

➔ ln –sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime

Press tab twice at any stage to see all the options available.

Open /etc/locale.gen and uncomment the needed locales. A safe bet is to uncomment these two

- en_US.UTF-8 UTF-8
- en_US ISO-8859-1

Here the part before the space is the localename. Then generate the locales with:

➔ locale-gen

Put your preferred locale into the /etc/locale.conf:

- ➔ echo LANG=en_US.UTF-8 > /etc/locale.conf

Create an initial ram-disk environment (people who want to know what it is, just go to https://wiki.archlinux.org/index.php/Mkinitcpio):

- ➔ mkinitcpio –p linux

I got a warning 'possibly missing firmware for module: aic94xx'

Set up your root password with

- ➔ passwd

Install dependencies for wifi-menu, you don't want an OS without WiFi.

- ➔ pacman –S dialog
- ➔ pacman –S wpa_supplicant

Also install wicd if you want to use other network interfaces (even if you don't, install it nonetheless.)

- ➔ pacman –S wicd wicd-gtk

4. Getting ready to boot

Now there are 2 different types of booting-environments. Most new laptops have UEFI with BIOS support disabled. Windows is by default on UEFI. It is particularly convenient to boot Arch from BIOS using syslinux. Follow the wiki to the word and it would be simple (I would put my steps here soon).

You can also boot some other linux installation using grub you have, and run 'sudo update-grub'. Next time you restart, you should see Arch listed too.

Note: Your username at the moment is 'root'.

5. Getting stuff up

Firstly, connect back to the WiFi. Then I decided to install some basics first, they being vim and git.

- ➔ pacman –S vim
- ➔ pacman –S git

Also get ranger for file managing.

Now at this stage, I install lynx web browser using 'pacman –S lynx' to sign in to my proxy on campus. Then I pull my git repository, that is, my rc files, and run my Vim setup scripts too at this stage.

Now, we'll make a user for you, give it a home folder, and 'own' it.

I would be using my Arch with arduino too, so I also add my user to the groups other than 'wheel'.

- ➔ useradd –m –G wheel –G games –G rfkill –G uucp –G tty –s /bin/bash saksham

The '-m' flag creates a home folder and –s defines login shell.

You can use usermod to add your user to groups later on.

- ➔ usermod –aG <groups> <username>
- ➔ gpasswd -a <username> <groupname>

  You can also create new groups later as
  - ➔ groupadd groupname

  You can give folder permissions to a group using
  - ➔ chown -R :groupname /path/to/folder
  - ➔ chmod -R g+w /path/to/folder #for write permissions for current user.

Then do passwd to set a new password.

- ➔ passwd <username>

Now to give your user sudo rights. Login as root (or do 'su') and run:

- ➔ EDITOR=vim visudo (you can do directly visudo)

Uncomment the two lines at the end

- ❖ Defaults targetpw
- ❖ ALL ALL=(ALL) ALL

Now your user can go sudo.

Configuring mount permissions

Install ntfs-3g.

Then open /etc/fstab and add the following line to mount ntfs partition /dev/sdaX:

- ➔ /dev/sdaX /media/Data ntfs-3g user,r,umask=111,dmask=000 0 1

The flag 'user' allows users to access the mounted partition.

Package managing

1. ABS, wget #Arch Build Repository, downloader
2. pacman –S base-devel fakeroot jshon expac wget
3. wget https://aur.archlinux.org/packagespa/packer/PKGBUILD
4. pacman –U packer-*.pkg.tar.gz

6. Installing the bare necessities

If you want a simple straightforward GNOME installation, just do:

```
sudo pacman -S gnome Xf86-input-synaptics      #Synaptics touchpad driver
```

You'll be good to go. Install the default graphics drivers (mesa).

If you don't want a pre-built desktop environment and would want to be experimental (I use i3wm), you should read below, else just skip it, it can get scary.

Here are some packages I install at this stage:

Configuring graphics

1.	Xorg-server	#Allows using GUI
2.	Xorg-server-utils	#Some more meta-packages
3.	Xorg-apps	#Better get them now
4.	Xorg-xinit	#startx
5.	Xterm	#Terminal
6.	Xorg-xclock	#Clock required for the basic window manager

Type startx now. You should get a cool (don't puke) looking GUI. Exit all the terminals. Get a local copy of .xinitrc now:

➔ cp /etc/skel/.xinitrc ~/

Add 'exec xterm' at its end to test this.

Now install a simple-installing WM (which BTW, is my favourite too):

➔ pacman –S i3 dmenu conky

Append 'exec i3' to .xinitrc now.

Install 'feh' now to set a wallpaper.

Configuring Audio

Now to set up Audio drivers, install

➔ alsa-utils alsa-plugins pulseaudio-alsa

Unmute the channels with

➔ amixer sset Master unmute #Didn't work for me

If you plug in headphones or plug them out, you would experience problems. Type this to redetect the plugged in audio device:

➔ alsactl-restore

Getting a nice GUI based OS up

Let's start with a proper terminal. I personally like gnome-terminal, although it brings with itself a lot of other stuff too, not all of which is useless/useful, according to whether the glass is half empty or full.

Then we have the good old firefox, followed by the complete gnome group. It comes handy. Installed gedit (optional), Guake drop-down terminal. Then synapse application launcher. Then libreoffice (I don't install it anymore, I configured Microsoft Office 2010 to work with Wine on my Arch).

Add custom keyboard shortcuts. If your touchpad doesn't work as it should, try 'synclient' and modify its parameters.

For example if you cannot do the left click by tapping with you finger, enter 'synclient TapButton1=1'

Add any changes to your .xinitrc to make them load every boot. This would be done as, putting "exec <command>&" before "exec i3" in .xinitrc

// Tasks done, redundant list for my convenience:

Cmake, file-roller, docky, gnome-tweak-tool

I3, dmenu, wicd-gtk, python-numpy, python2-numpy, python2-pip, python-pip

Curl

Wget

Gnome-terminal and nautilus and evince (big bad packages with dependencies)

Zsh

Conky

Git cloned my-rc-files repository

Installed vundle with script in .myscripts

Opened vim and did :PluginInstall

Installed powerline fonts

Ran SetupZSH script.

Lxappearance

Jshon expac (for packer)

Ran my packer script.

Installed packer from the directory in home.

Installed feh

Guake (changed its settings and added it to i3 startup)

Gedit

Cmake

File-roller

Powertop

Ntfs-3g

Set up mounting of data drive

Synapse

Openssh (aur)

Flatter icon theme and flatstudio theme (aur)

Numix icon theme and sable gtk theme (aur)

Libreoffice

All audio stuff written above.

Create_ap for hotspot (aur)

Xbacklight

Gparted

Dosfstools

VLC

GIMP

ZSH-completion

devmon and eject

libmtp andgbvfs-mtp for my android phone drivers.

Wine and Android studio (after enabling multilib)

PlayOnLinux (for Office)

Java

Eclipse