Installing and Configuring Arch Linux on your 2012 Nexus 7

A step by step guide

- Introduction
  - Why Arch Linux
  - Why This Guide
- Preparation of the host computer and the Nexus
  - Enable Android Debug Bridge (adb) on your Nexus
    - Windows
  - Setting up fastboot (and adb) on your computer
    - Windows
    - Mac OS X
    - Linux
  - Unlock the bootloader
- Installing MultiROM
  - Installing a custom recovery
  - Installing MultiROM from the Android App
    - Installing SuperSU (root)
    - Installing the MultiROM manager from the play store
  - Installing MultiROM manually from the command line
- Installing the Arch Image
- Connecting to your Nexus
  - Windows
  - Linux
  - Mac OS X
- Getting WiFi working
- Create a user and setup your passwords
- Install and configure sudo
- Install and configure SSH
- Setting up Bluetooth
- Setting up X (the windowing system)

Onscreen keyboard
Graphical Login
Logging in without a keyboard
On screen keyboard in LightDM
Automatic login
Login without password
Tweaking touch
Flash
Kernel plans and other notes
Changelog

Introduction

TL;DR Use the table of contents to skip to the section you need.
Go here to discuss this document

I've also enabled comments since I've noticed a lot more traffic coming here, then realized this document is a top google hit so the traffic may be coming from google not xda. Feel free to comment on this document, or e-mail me.

Why Arch Linux

Arch Linux is kind of a mixed bag. Upon installing the system you will have little more than a command line with base *nix functionality. Unlike Ubuntu or some other popular distributions it does not come ready to go out of the box. The system is however quite easy to set up to meet your needs exactly, if you are willing to spend a little extra time to set it up. There are advantages and disadvantages to this approach.

The great thing about a system like this, especially when you have limited resources (like on your Nexus) is that there is absolutely no bloat. Things should run more smoothly and quickly, allowing you to use your system more efficiently. Since you set everything up, you also have a much better understanding of how the system works. This is great since you will be able to much more easily fix things when they inevitably break. The system is also very easy to configure and work with.

The downside is that you spend longer setting up your system. Things don't just work out of the box. Want a specific program? You need to install it. Want networking to start on boot? You'll have to tell it to. Nothing is automatic. Unless you manually set up the automatic functionality ;-) You will also need to be able to use a command line and edit text files to configure the system.

There is a wealth of documentation on Arch Linux, and if you aren't proficient with *nix systems already you will learn a lot by running Arch. You will not however have a system that just works 30 seconds after you install.

More information about the philosophy and goals of Arch can be found here.

Why This Guide

The Arch Linux Wiki is a fantastic resource. However, currently there is no information specific to the Nexus 7. Furthermore, the wiki's install guide targets desktops and laptops. This guide is intended to bridge that gap.

This guide has three goals. Firstly to be a walkthrough setup guide for anyone who has no experience with Arch. Secondly, to be a reference guide to setting up features that are specific to the Nexus 7 for those who have already gone through the walkthrough, or have more experience with Linux and Arch on more traditional computers. Thirdly, to provide a bridge that will allow novices and experienced users alike to get comfortable enough with Arch on their Nexus to branch off from this guide and use the Arch Wiki to further customize their installation and find answers to questions beyond the scope of this guide.

Preparation of the host computer and the Nexus

If your Nexus is already rooted, you can actually do the install without ever plugging your Nexus into your computer. Just skip to Installing the MultiROM manager from the play store. If you are not rooted, carry on.

Preparing for the installation is actually much more involved than the actual installation. First you will need to get the fastboot and adb programs onto your computer. Then you'll need to unlock your bootloader. Thirdly, you will need to install MultiROM by Tassadar. Finally you will do the actual install. If you have already done any of these steps, you may skip them. Additionally, fastboot is only required for the unlock and the installation of MultiROM, you don't need it for the actual installation of Arch.

Enable Android Debug Bridge (adb) on your Nexus

If your Nexus isn't booted up, go ahead and start it up. Go to “Settings->About tablet”. Tap on “Build Number” until it says you are a developer (about seven times). Go back to the main Settings screen. You should now have a “Developer options” choice. Go into “Developer options” and enable “USB debugging”.

Plug your Nexus into your computer using the USB cable.

Windows

If you are running Windows, you'll need to install some additional drivers at this point. You can get them from Asus. Choose "Others" for the OS. Then pick the global option under USB. Unzip the resulting file to create a `usb_driver` folder. Follow these steps to get the driver installed:

1. Open Device manager. The steps to get there may vary slightly depending on which version of Windows you have, but they should be similar to below.
    a. Find "Computer" on your desktop or in your start menu.
    b. Right click computer.
    c. Choose properties.
    d. Choose device manager.
2. Right click Nexus 7.
3. Choose "Update Driver Software..."
4. Choose "Browse my computer for driver software"
5. Click "Browse..."
6. Choose the `usb_driver` folder you created earlier
7. Click "Next"
8. Choose "Install"

Setting up fastboot (and adb) on your computer

The fastboot utility is required to unlock the Nexus, and install the MultiROM utility. These instructions will be similar for Windows and Macs.

Windows

You'll need to get the Android SDK from here. Go ahead and choose to "Download the SDK/ADT Bundle for Windows". I'll assume you unzip the resulting download into your `C:\` drive resulting in a folder named `C:\adt-bundle-windows-x86_64-20130522\`. Adjust accordingly if you put it somewhere else or if the folder is named differently.

Open up a command line by clicking on Start and type `cmd.exe` under the "Run.." option or the "Search programs and files" option. Use the following command to change to the proper directory:

```
cd c:\adt-bundle-windows-x86_64-20130522\sdk\platform-tools
```

Mac OS X

You'll need the Android SDK, just as in windows. Get it here. Open the Terminal program. Assuming you unzipped the to your home directory type the following command in your terminal:

```
$ cd ~/adt-bundle-mac-x86_64-20130522/sdk/platform-tools
```

Linux

Your distribution probably includes a package for these tools. Simply install them from your package manager. Examples follow (source)

Debian based (Ubuntu, Mint, Sid, etc)

```
$ sudo apt-get install android-tools-fastboot android-tools-adb
```

Fedora

```
$ sudo yum install android-tools
```

Arch

```
$ sudo pacman -S base-devel
$ wget
https://aur.archlinux.org/packages/an/android-sdk-platform-tools/android-sdk-pl
atform-tools.tar.gz
$ tar -xzf android-sdk-platform-tools.tar.gz
$ cd android-sdk-platform-tools
$ makepkg -s
$ sudo pacman -U android-sdk-platform-tools*.tar.xz
```

Alternatively you may use your favorite AUR manager to install the android-sdk-platform-tools package.

Open up a terminal emulator and you are good to go for the next steps.

Unlock the bootloader

Obviously if you've already unlocked your bootloader you can skip this step.

WARNING: Unlocking the bootloader will erase any data you have on your Nexus. Backup if you care about anything on the device.

Type the following command into your terminal window:

```
adb reboot bootloader
```

Attention: If you are running **Mac** you may have to change `adb` to `./adb` and `fastboot` to `./fastboot`. In **Linux** you may have to prepend `sudo` to make it `sudo adb` and `sudo fastboot`.

After your Nexus reboots to the screen with the Android on his back, type in:

```
fastboot oem unlock
```

Use the volume key to highlight "Yes" then the power button to select it. At the bottom of your screen it should now say in red letters “Lock state - unlocked”. Now is a good time to go ahead and set up Android again. Use fastboot to reboot your computer and come back once you are done and you have enable USB Debugging again. I’ll wait.

```
fastboot reboot
```

Installing MultiROM

This is the MultiROM thread. You can get the files you’ll need by scrolling down to the second post where it says “Download”. Alternatively you may get the files from Tassadar’s goo.im mirror.

Installing a custom recovery

You need to be in the bootloader for this step. You may type the following command into your terminal window on your computer to get to the bootloader:

```
adb reboot bootloader
```

We will install the version of TWRP that has been modified to work with MultiROM. You can get it from the MultiROM thread or from Tassadar’s goo.im mirror. The file you need is named TWRP_multirom_grouper_*.img. Download this to your computer. In the following, replace `<path to download>` with the location you downloaded TWRP_multirom_grouper_*.img to. For example, `C:\Users\ylixir\Downloads` or `~/Downloads`. Remember you may have to modify the fastboot command on Linux or Mac, see the Unlock the bootloader section for more information.

```
fastboot flash recovery <path to download>/TWRP_multirom_grouper_*.img
```

After this has finished you may reboot.

```
fastboot reboot
```

You may now proceed to Installing MultiROM from the Android App or Installing MultiROM manually from the command line. Choose your own adventure!

Installing MultiROM from the Android App

First we need to install root, if you are already rooted, you can skip to Installing the MultiROM manager from the play store.

Installing SuperSU (root)

You should be in the TWRP recovery for this step. If you are not then type in the following command into the terminal on your computer:

```
adb reboot recovery
```

TWRP will actually offer to root for you, simply choose the “Reboot” option followed by the

"System" option. It will ask you if you wish to install SuperSU. Just swipe the blue dot with the white arrow to root your Nexus.

After the Nexus finishes rebooting, find the "SuperSU Installer" in your app drawer and run it. Choose the "Play" option. Choose "Update" from the Play Store and "Accept".

Go back to your app drawer and open the new SuperSU app. You will probably need to update the binary now. I see no reason to get your recovery involved, simply choose the "Normal" option.

"Installation success !" You can reboot as recommended now if you like.

Installing the MultiROM manager from the play store

If you are reading this guide from your computer you can follow this link and to install the MultiROM manager to your Nexus (make sure you have your Nexus 7 selected after you click "Install" if you do it from your computer). Alternatively, you may also install it from the Play Store on your Nexus. Search for "multirom manager" and it should come right up. The author of the app you want is named "Vojtech Bocek"--incidentally a.k.a. Tassadar, the person that made this all possible.

Open the "MultiROM Manager" app after you have installed it. Choose "Grant" when it asks for superuser permissions. It should give you the options to install MultiROM itself (basically the boot menu), Recovery (a modified TWRP), and Kernel (a modified kernel that enables the magic).

You must install MultiROM and a kernel. If you have updated to Kit Kat then choose "Stock 4.4" for your kernel. Otherwise pick the one that fits what you have on your Nexus. If you already did the Installing a custom recovery step above you don't need to choose the "Recovery" option. If you haven't done this step, then choose "Recovery".

After you have made your selections, choose "Install". It will prompt you to reboot, go ahead.

Now you may skip to Installing the Arch Image.

Installing MultiROM manually from the command line

You need the `multirom-*-grouper.zip` and the `kernel_kexec_grouper_*.zip` that is appropriate for your system. You can get them from the MultiROM thread or from Tassadar's goo.im mirror.

You may download the files directly with your Nexus, or onto your computer and copy them over. Anywhere is fine--I put mine in the `Download` folder. Now reboot your Nexus into the bootloader again.

```
adb reboot bootloader
```

Once it has come up choose “Install”. Then find the multirom-*-grouper.zip. Choose “Add More Zips” and also select the kernel_kexec_grouper_*.zip. Then swipe to confirm the installation. When you have the blue “Successful” message, tap the “Home” button.

Installing the Arch Image

You’ll need Arch itself from this post. The link you want is the mediafire one at the bottom of the post. The file is arch_20130626.mrom.

You may download the file directly with your Nexus, or onto your computer and copy it over. Anywhere is fine--I put mine in the `Download` folder. Reboot your Nexus into the bootloader again.

```
adb reboot bootloader
```

Now we install Arch by choosing “Advanced->MultiROM->Add ROM”. Pick the “MultiROM installer” option and tap “Next”. Choose `arch_20130626.mrom`. Swipe to confirm. Now choose “Reboot System”.

When MultiROM comes up, choose “arch_20130625” to boot into Arch Linux. If you choose “Internal” instead, you will boot into Android.

Congratulations on your newly installed (and fairly useless as of now) Arch Linux system!

Connecting to your Nexus

Wiki link for more information

Arch boots to a command line by default, so you’ll need to get a keyboard attached to do anything useful. One option is to connect a USB keyboard with an OTG cable. They can be had for very cheap on Amazon and elsewhere. Another (free) option is to make your computer into a serial console and connect with that. The following instructions are operating system specific instructions on how to do that.

Note: After you’ve connected either via keyboard or via USB serial you type in “root” and “root” again for your username and password to log in.

Windows

If you have access to a Linux system, I recommend that over Windows, but if you must use Windows here are the instructions. I have Windows 7 64 bit so the following instructions may need to be adjusted depending on what version you have.

You'll need some kind of terminal program. PuTTY will work for both serial communication and ssh. Get it here.

Plug in the Nexus to your USB port. The driver will probably fail to install properly. It did not work out of the box for me. The working driver is just one inf file that just tells Windows to pretend your Nexus is a serial console. Get the driver from here.

Follow these steps to get the driver installed:

9. Open Device manager. The steps to get there may vary slightly depending on which version of Windows you have, but they should be similar to below.
   a. Find "Computer" on your desktop or in your start menu.
   b. Right click computer.
   c. Choose properties.
   d. Choose device manager.
10. Right click gadget serial v2.4
11. Choose properties
12. Click "Update driver"
13. Choose "Browse my computer for driver software"
14. Let me pick from a list of device drivers on my computer
15. Choose "Have disk..."
16. Click "Browse..."
17. Choose the "linux-cdc-acm.inf" that you downloaded earlier
18. Select "Gadget Serial"
19. Click "Next"
20. Choose to "Install anyway"

Take note of the COM number for the device. Mine is COM3 so I'll use that in the instructions, replace the 3 with whatever number you have. Open the putty program you downloaded earlier. Choose "Serial" under "Connection type:". Change the "Serial line" to "COM3". Click open.

Linux

Plug in the Nexus to your USB port. Open up a terminal and type:

```
$ sudo screen /dev/ttyACM0 115200
```

The screen program may not be installed. If it isn't already installed then get it with your package manager of choice. In a ubuntu, mint, or other debian flavored distribution the command below should install it:

```
$ sudo apt-get install screen
```

Mac OS X

I don't have a Mac to test this with, but Linux and Mac both share a unix heritage so the process should be similar to Linux. You'll need to open the "Terminal" program to get the unix command line. The device you are looking for is /dev/cu.usbmodem* where * is some value. Type ls /dev/cu.usbmodem* to find the correct device.
For example:

```
$ sudo screen /dev/cu.usbmodemfa1331 115200
```

Thanks for the tip Aiden Foxx.

Getting WiFi working

Wiki link for more information

If you don't like netctl for managing your wifi, options not covered here include wicd and Network Manager.

As of this writing (July 16, 2013) the system comes with the netcfg package for networking. Netcfg is, however, being phased out in favor of netctl. So to properly set up networking we will have to first configure your wifi, then install netctl, and configure your wifi again :-/

First get wifi working:

```
# wifi-menu
```

After you've configured wifi, we'll need to get our system up to date, then download the netctl packages, and finally remove netcfg and install netctl

```
# pacman -Syu

# pacman -Sw netctl wpa_actiond iproute2 iptables

# pacman -Rs netcfg

# pacman -S netctl wpa_actiond
```

Now we need to stop the interface and move the netcfg profile to netctl. We will then need to edit the profile configuration. I use vi, but feel free to use nano or whatever floats your boat. You can get the wlan0-profile-name by doing a `$ls /etc/network.d/`.

```
# netctl stop wlan0-profile-name
# mv /etc/network.d/wlan0* /etc/netctl/
# vi /etc/netctl/wlan0-profile-name
```

Now rename the variables that are in all capitals to UpperCamelCase. So "CONNECTION" becomes “Connection”. Leave ESSID and IP in capital. Also remove the single quote character from around the values after the equal signs, *but* leave the single quote around the value for Description and ESSID. You may also want to add a line with the following to fix some connection issues:

```
TimeoutDHCP=30
```

At this point I couldn’t start the network, but I could ping google, so let’s just reboot and see what we see. First lets set it to start automatically.

~~`# netctl start wlan0-profile-name`~~

```
# systemctl enable netctl-auto@wlan0.service
# reboot
```

That gets your wifi working, but you may want to follow the next couple of steps to make things a little more secure.

Change permissions:

```
# chmod 600 /etc/netctl/wlan0-profile-name
```

Passphrase Obfuscation
First get an obfuscated passphrase with the following command:

```
# wpa_passphrase [ssid] [passphrase]
```

Now edit your profile configuration file. As always feel free to replace vi with nano.

```
# vi /etc/netctl/wlan0-profile-name
```

Replace the `Key=your_key` with `Key=\”wpa_passphrase_output`

Create a user and setup your passwords

Wiki link for more information

Use the following commands to set up a username for yourself and to set up your passwords. Change “ylixir” to whatever username you want.

```
# useradd -m -g users -s /bin/bash ylixir
```

```
# chfn ylixir
```

```
# passwd ylixir
# passwd root
# exit
```

You may now type in your new username and password to log in.

Install and configure sudo

[Wiki link for more information]

The sudo command allows you to run commands as root (superuser do). It’s basically the same as running something as an Administrator on Windows. Logging in as root generally considered a bad idea. However you often need to be root to perform maintenance on your system. This convenience utility makes it easy to perform root commands while logged in as another user.

You need to switch to root to install packages. The `su` command does this. When it asks for a password, use your `root` password. After you are root you may install `sudo`.

```
$ su
# pacman -S sudo
```

Okay, now that sudo is installed we need to configure it.

```
# visudo
```

or

```
# EDITOR="nano" visudo
```

Find the following line:

```
# %wheel ALL=(ALL) ALL
```

Remove uncomment it by removing the # and space from the beginning so it becomes:

```
%wheel ALL=(ALL) ALL
```

Save and exit out of the editor.

Okay, now any user of the `wheel` group can use `sudo`. You better add your own username to the `wheel` group. And then exit out of superuser mode. You will also have to log out of your own user in order for the changes in your group membership to take effect.

```
# gpasswd -a ylixir wheel
```

```
# exit
$ exit
```

Install and configure SSH

Wiki link for more information

If you wish you may install `ssh` to allow you to log in over a network among other things.

```
$ sudo pacman -S openssh
```

It’s probably a bad idea to allow root logins from over a network. Lets disable that by editing the configuration file. You may use `nano` instead of `vi` if you wish.

```
$ sudo vi /etc/ssh/sshd_config
```

Add the following line to the configuration file.

```
PermitRootLogin no
```

Now it’s all set up, start it, and tell the system to start it automatically on boot.

```
$ sudo systemctl start sshd
```

```
$ sudo systemctl enable sshd.service
```

If you wish to find out what IP address your Nexus is using type in:

```
$ ip addr show dev wlan0
```

Now you may use PuTTY if on Windows, or the `ssh` command from a terminal on Linux or Mac to log into your Nexus over the network.

Setting up Bluetooth

Wiki link for more information
Another wiki link
I stole some scripting from Ubuntu and from this guy and I think I have bluetooth up and running.

First, install some packages we need.

```
$ sudo pacman -S rfkill bluez-utils base-devel gstreamer0.10-base libsndfile
$ wget -c https://aur.archlinux.org/packages/bl/bluez4/bluez4.tar.gz
$ tar -xzf bluez4.tar.gz
$ cd bluez4
```

you can use nano instead of vi in the next command if you wish

```
$ vi PKGBUILD
```

Change the arch line to include ‘armv7h’

```
$ makepkg -s
$ sudo pacman -U *-armv7h.pkg.tar.xz
```

Next we need to create the scripts to start bluetooth on boot.

```
$ sudo vi /usr/bin/load_bcm4330.sh
```

Enter the following into this file:

```
#!/bin/bash

#
# BCM4330 BT Adapter upload rampatch

# env Variables
logfile=/var/log/rampatch.log

# logfile check
if ! test -f ${logfile} ; then
    # logfile does not exist, so create it
    touch $logfile
fi

echo "--------------------------------------------------------">>$logfile
entrytime=$(date +"%Y-%m-%d %H:%M:%S")
echo $entrytime " applying rampatch to BCM4330 bluetooth adapter" \
>> $logfile

# wait for hci0 to become available and usable
#sleep 1
/usr/bin/rfkill unblock bluetooth
/usr/bin/brcm_patchram_plus --patchram /lib/firmware/bcm4330.hcd \
--no2bytes \
--scopcm=0,2,0,0,0,0,0,0,0,0 \
--enable_hci \
--enable_lpm \
--baudrate 3000000 --use_baudrate_for_download \
--tosleep=50000 \
/dev/ttyHS2 >> $logfile

entrytime=$(date +"%Y-%m-%d %H:%M:%S")
echo $entrytime " ..done" >> $logfile
echo "---------------------------------------------------”>>$logfile
```

Next we need to create the scripts to start bluetooth on boot Source.

```
$ sudo vi /usr/bin/connect_bt.sh
```

Enter the following into this file:

```
#!/bin/bash


address="AA:BB:CC:DD:EE:FF"


while (sleep 1)
do
```

```
        connected=`hidd --show` > /dev/null
        if [[ ! $connected =~ .*${address}.* ]] ; then
        hidd --connect ${address} > /dev/null 2>&1
        fi
        done
```

Now we set the permissions for those files, and edit the systemd files.

```
$ sudo chmod u+x /usr/bin/load_bcm4330.sh
$ sudo chmod u+x /usr/bin/connect_bt.sh
$ sudo vi /etc/systemd/system/loadfw.service
```

And enter the following into this file:

```
        [Unit]
        Description=Load firmware into BCM4330 bluetooth USB adapter
        After=basic.target
        After=suspend.target
        After=hibernate.target

        [Service]
        Type=simple
        ExecStart=/usr/bin/load_bcm4330.sh

        [Install]
        WantedBy=basic.target
        WantedBy=suspend.target
        WantedBy=hibernate.target
```

And another file

```
$ sudo vi /etc/systemd/system/connectbt.service
```

Which contains

```
        [Unit]
        Description=Connect to bluetooth devices
        After=basic.target
        After=suspend.target
        After=hibernate.target

        [Service]
        Type=simple
        ExecStart=/usr/bin/connect_bt.sh

        [Install]
        WantedBy=basic.target
        WantedBy=suspend.target
        WantedBy=hibernate.target
```

Create new file /etc/udev/rules.d/10-local.rules Source

```
        # Set bluetooth power up
        ACTION=="add", KERNEL=="hci0", RUN+="/usr/bin/hciconfig hci0 up"
```

And finally set it to start on boot:
```
$ sudo systemctl start loadfw.service
$ sudo systemctl enable loadfw.service
$ sudo systemctl start bluetooth.service
$ sudo systemctl enable bluetooth.service
```

At this point you may want to do an `hidd --search` to find out the addresses of your devices and edit `connect_bt.sh` accordingly
```
$ sudo systemctl start connectbt.service
$ sudo systemctl enable connectbt.service
```

Note that if you wish to add more than one device you can adjust the connect_bt.sh script to add an address2, address3 etc, and add the if...fi block for each device.

Setting up X (the windowing system)

Install the `xorg` package, and remove the bogus `xorg.conf`
```
$ sudo pacman -S xorg xorg-xinit
$ sudo rm /etc/X11/xorg.conf
$ sudo pacman -S xfce4 xfce4-goodies gamin
$ sudo pacman -S kde
$ sudo pacman -S enlightenment17
$ sudo pacman -S lxde
```

Next make a new `/etc/X11/xorg.conf.d/01-nexus7-input.conf`
Enter the following into this file (Cribbed but modified from here):
```
Section "ServerFlags"
        Option        "AllowEmptyInput" "False"
EndSection

Section "InputClass"
        Identifier      "Nexus 7 Touchscreen"
        MatchProduct    "elan-touchscreen"
        Driver          "evdev"
        Option          "Ignore" "off"
#        Option          "SwapAxes" "yes"
#        Option          "InvertX"  "yes"
#        Option          "InvertY"  "no"
EndSection

# Configure gpio keys
Section "InputClass"
        Identifier      "HW keys"
        MatchProduct    "gpio-keys"
        Driver          "evdev"
        Option          "Ignore" "off"
```

```
      EndSection
```

Also make a file for the video drivers. Note that you have two options for devices here. The fbdev and the tegra. Tegra is faster but really glitchy. The fbdev on the other hand renders much better, but is slow. Tegra is basically unusable for me, so I am currently using fbdev. This needs to be fixed eventually

```
$ sudo vi /etc/X11/xorg.conf.d/10-nexus7-screen.conf
      Section "Module"
          Disable     "dri"
          Disable     "dri2"
          Disable     "glx"
          SubSection  "extmod"
              Option  "omit xfree86-dga"
          EndSubSection
      EndSection

      Section "Device"
              Identifier "Tegra"
              Driver "tegra"
      #       Option "ARGBHWCursor" "true"
      EndSection

      Section "Device"
          Identifier "Framebuffer"
          Driver "fbdev"
      # The rotate option makes the screen landscape mode
      # put a # in front of this line if you want portrait mode
          Option    "Rotate" "CW"
      EndSection

      Section "Monitor"
          Identifier    "Monitor"
      # The rotate option makes the screen landscape mode
      # put a # in front of this line if you want portrait mode
          Option    "Rotate" "right"
      EndSection

      Section "Screen"
          Identifier    "Screen"
      # remove the # from the driver you want to use
      # add a # to the driver you don't want to use
      #     Device"Tegra"
            Device"Framebuffer"

          Monitor       "Monitor"
      EndSection

$ vi .xinitrc
      #exec startxfce4
      #exec startkde
      #exec enlightenment_start
      exec startlxde
```

```
$ startx
```

A mouse will help make things useable but kind of defeats the point. A cheap stylus is smaller, cheaper, and helps hit those tiny buttons.

Mer (plasma active) References
Nvidia references note that to get the newer X ABI 14 drivers, change the domain name to developer.nvidia.com
Getting KDE working
Enlightenment 17 is also reported to be working

Onscreen keyboard

These are only really working for me on fbdev, the graphics are just so glitchy :-/

Pick one:

```
$ sudo pacman -S matchbox-keyboard
$ sudo pacman -S xvkbd
$ sudo pacman -S caribou
$ sudo pacman -S onboard
```

Caribou and Onboard both give errors starting up, but matchbox and xvkbd seem fine.

Graphical Login

I'll cover installing lightdm. You can use a different one if you like, see the Arch Linux Wiki for more information.

```
$ sudo pacman -S lightdm lightdm-gtk3-greeter
$ sudo systemctl enable lightdm
```

Upon reboot, lightdm will start automatically.

Logging in without a keyboard

You basically have three options here. The first is to get an on screen keyboard showing up on the login screen. The second is setting up automatic login. And the third is setting up your display manager to not require a password. I don't have passwordless logins working yet. I'm only going to cover lightdm in this guide. See the wiki/google for information on other display managers.

On screen keyboard in LightDM

You can tweak this to whichever keyboard you choose. You only need to edit /etc/lightdm/lightdm-gtk-greeter.conf and change the keyboard= line. The keyboard is available from the accessibility menu. I used:

```
keyboard=/usr/bin/xvkbd -geometry 800x300-0-0 -compact
```

Automatic login

You need to add yourself to the autologin group.

```
$ sudo groupadd autologin
$ sudo gpasswd -a USERNAME autologin
```

Edit the configuration file for lightdm:

```
$ sudo vi /etc/lightdm/lightdm.conf
```

In the `[SetDefaults]` section, add/edit the `autologin-user=` line. Make sure there is no # at the beginning of the line. Also the timeout for good measure.

```
autologin-user=your-username

autologin-user-timeout=0
```

Login without password

Don't quite have this working yet. With automatic login working anyway, multiple users is probably a small use case so I'm going to leave it for now and come back later. The following should get one close if one decides to try to get it working before I do.

You need to add yourself to the nopasswdlogin group.

```
$ sudo groupadd nopasswdlogin
$ sudo gpasswd -a USERNAME nopasswdlogin
```

Edit the configuration file for lightdm:

```
$ sudo vi /etc/pam.d/lightdm
```

Add the following line to the end of the file.

```
auth  sufficient  pam_succeed_if.so  user ingroup nopasswdlogin
```

Tweaking touch

We'll install touchégg to get touch working better. First we need the development tools, then we'll download the package specification, create the package, and install it.

```
$ sudo pacman -S base-devel
$ wget -c https://aur.archlinux.org/packages/fr/frame/frame.tar.gz
$ tar -xzf frame.tar.gz
$ cd frame
$ vi PKGBUILD
```

Change the arch line to include 'armv7h'

```
$ makepkg -s
$ sudo pacman -U *-armv7h.pkg.tar.xz
$ cd ..
$ wget -c https://aur.archlinux.org/packages/gr/grail/grail.tar.gz
$ tar -xzf grail.tar.gz
$ cd grail
$ vi PKGBUILD
```

Change the arch line to include ‘armv7h’

```
$ makepkg -s
$ sudo pacman -U *-armv7h.pkg.tar.xz
$ cd ..
$ wget -c https://aur.archlinux.org/packages/ge/geis/geis.tar.gz
$ tar -xzf geis.tar.gz
$ cd geis
$ vi PKGBUILD
```

Change the arch line to include ‘armv7h’

```
$ makepkg -s
$ sudo pacman -U *-armv7h.pkg.tar.xz
$ cd ..
$ wget -c https://aur.archlinux.org/packages/to/touchegg/touchegg.tar.gz
$ tar -xzf touchegg.tar.gz
$ cd touchegg
$ vi PKGBUILD
```

Change the arch line to include ‘armv7h’

```
$ makepkg -s
$ sudo pacman -U *-armv7h.pkg.tar.xz
$ cd ..
$ mkdir -p ~/.config/touchegg
$ cp /usr/share/touchegg/touchegg.conf ~/.config/touchegg/touchegg.conf
$ vi ~/.config/touchegg/touchegg.conf
```

Tweak to your preferences. More info here. Then add touchegg to your xinitrc if you use startx to start X, or xprofile if you use a display manager such as lighdm.

```
$ vi ~/.xinitrc
      touchegg &
      #exec startxfce4
      exec startkde
      #exec enlightenment_start
      #exec lxde
```

or

```
$ vi ~/.xprofile
```

And add the line:

```
      touchegg &
```

Flash

I haven't tried this yet, but I found this. There is also apparently a package in AUR.

Kernel plans and other notes

Currently there are some issues with the kernel. It's old, which is a problem for bluez5. ~~It is also not configured to play nice with systemd, which arch has migrated to qq~~. So we need a new one. I'm currently have one building, ~~but it's basically just the same one we have~~. Here are my plans for this. I have basically two projects. One is a set of shell scripts to automate the building of the kernel. These scripts will ideally work for any linux out there. The second is the actual kernel tree. I'm putting my work up on github so anyone can follow along. The kernel scripts are kernel-shell-game. The kernel tree is grouper-kernel.

You can try out a new kernel if you like. Check out the kernel shell games scripts. Run sudo builder.sh with the following command line options:

1. setup
2. kernel get
3. kernel build
4. kernel boot

You will need fastboot and abootimg installed on your system for the last step and your nexus connected to the PC in fastboot/boot loader mode. You will also need the initramfs file from the boot directory in arch to be in the kernel_target directory

The process will likely be as follows:

1. ~~Get the scripts to fire up a kernel from fastboot, so I can directly test my progress without messing about with transferring files about and horking my arch setup.~~
2. ~~Get a working kernel building without using the debianized process canonical has setup.~~
3. Fork the kernel tree right before canonical started messing with it and cherry-pick only the ubuntu stuff that is absolutely necessary. For sure leave out all the debianized stuff.
4. ~~Update to include systemd support.~~
5. Make a package that can be installed.

Once the kernel is straight, then I'll focus on getting a clean bootstrap of the root filesystem rolling. I'll also hopefully be simplifying the instructions by providing packages to take care of a lot of the grunt work.

As far as the graphics work, the video seems fine with ubuntu so it's probably not a kernel issue. I'll have to take a look at their packages. Maybe we are just missing some libraries or something.

And I'll take it from there. Maybe make a setup program so people don't have to plug in a keyboard or a computer to get things functional. Maybe try to modify the base to replace systemd with something less gross (and less cpu->battery hungry). Maybe work on getting the kernel more vanilla to setup keeping it current instead of being tied to google. Maybe something else, who knows.

Anyhow, just wanted people to know what I was thinking, since the guide hasn't been updated in a while. Weigh in if you like.

NG Wireless configuration

```
pacman -S wpa_supplicant
cat > /etc/wpa_supplicant.conf
network={
	ssid="SSID_NAME"
	psk="bleahbleah"
}
<CTRL-D>
cat > /etc/systemd/system/network-wireless@.service
[Unit]
Description=Wireless network connectivity (%i)
Wants=network.target
Before=network.target
BindsTo=sys-subsystem-net-devices-%i.device
After=sys-subsystem-net-devices-%i.device

[Service]
Type=oneshot
RemainAfterExit=yes

ExecStart=/usr/bin/ip link set dev %i up
ExecStart=/usr/bin/wpa_supplicant -B -i %i -c /etc/wpa_supplicant.conf
ExecStart=/usr/bin/dhcpcd %i

ExecStop=/usr/bin/ip link set dev %i down

[Install]
WantedBy=multi-user.target
<CTRL-D>
pacman -S dhcpcd
pacman -S iproute2
systemctl enable network-wireless@wlan0.service
systemctl start network-wireless@wlan0.service
```

Changelog

2013-07-20

1. Added instructions for installing the platform tools on an arch host pc
2. Added references to wicd and network manager as alternatives to netctl

2013-10-20

1. Updated instructions to reflect new *-grouper.zip naming scheme for MultiROM files
2. Added instructions for updating the kernel to include kexec
3. Got bluetooth 90% working. Interface comes up and can find devices with scan. Can't get it to pair yet though.

2013-10-24

1. Got the bluetooth device to power up on boot by adding a udev rule. Unfortunately the bluetooth daemon seems to be failing as it starts.

2013-10-28

1. My keyboard is pairing on boot now. Ugly ugly things to get it working, but it works.

2013-11-03

1. Updated the guide to get X working.

2013-11-04

1. Tweaked xorg.conf files to get kde and e17 working.
2. Added instructions to get Touchégg working to provide more tablety experience
3. Stubbed out getting the display manager going and a section for an on screen keyboard

2013-11-05

1. More xorg.conf tweaks. Also instructions for automatic login and onscreen keyboards.

2013-11-17

1. fixed typo in bluetooth section (Thanks Lord Socky)
2. tweaked the fastboot instructions to help people notice that they may need to use sudo depending on how their box is setup (Thanks Сергей Глита)
3. Added a note about the MultiROM Manager in the app store (Thanks Lord Socky)
4. removed some redundant steps in the "Tweaking Touch" section (Thanks Сергей Глита)
5. slight rewording of setup for wifi (Thanks Сергей Глита)

2013-11-18

1. breakage with wifi passphrase obfuscation. Need to add \" to your key. Fixed instructions
2. clarified a citation (thanks Сергей Глита)

2013-11-24

1. Upgraded my Nexus to Kit Kat, thought it was a good time to update the guide to include instructions for installing multirom from the android app. Thanks again for the tip Lord Socky

2013-12-18

1. Fixed a "typo" with the Touchegg instructions. (Thanks Alex Fuhr)
2. Added instructions for connecting with a serial terminal in OS X. (Thanks Aiden Foxx)

2013-12-22

1. Tweaked bluetooth instructions to get bluez4 from AUR (Thanks Lord Socky)