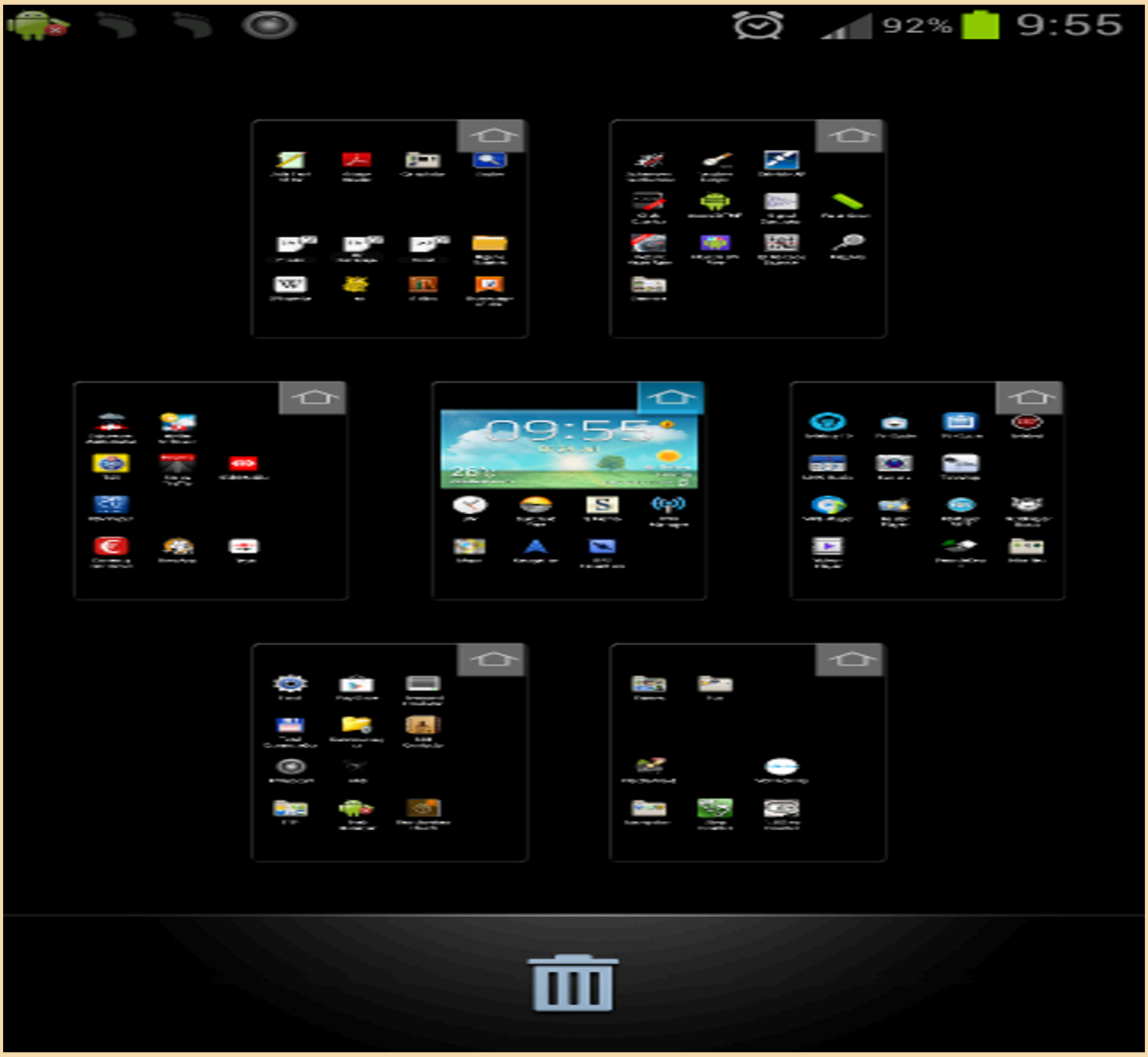
android



https://www.linurs.org

Android for Linux users

Android for Linux users

Table of Contents

List of Figures

Chapter 1. Copyright

Chapter 2. Introduction

THE DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, IN NO EVENT SHALL I BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY.

Android is considered a open source project https://source.android.com/

In most cases people buy a device with Android set up by the device manufacturer. The device manufacturers however take an Android setup from Google and add their device specific SW.

The standard way to look for apps is the built in Google play store, however also a web browser on a PC with a big screen using the link https://play.google.com/store/apps can be used. Additionally any apk package file can be installed, if it is trusted.

Chapter 3. Find Phone

There are different ways to find an android device. If the device is known by a google account then https://www.google.com/android/find?u=1 will find it if

Chapter 4. Package manager

This section is not meant for hackers. It is for the understanding and help for keep the android device clean.

The package manager installs a package and can also remove it.

Unfortunately it is naive to think that after removing the package everything is cleaned and the device is as before the installation. Maybe after installing and immediate removing this is the case, but when an application gets started it can create data on the android device that is unknown to the package manager and therefore remains on the device after the un-install.

Luckily root restriction prevent that those data can not be spread into the system.

Unfortunately, but with good reason looking into package management on the device is restricted to root access. There seem to be a `/data/system/packages.xml` file that has registered all apps. Luckily the restrictions are not just an obstacle to the user but also to the applications, so applications can not just dump files where they want.

Therefore applications that need to store some data create usually a directory under `/mnt/sdcard` having the name of the app and then putting the data there. If the app is no more installed such directories (if identified) can be deleted manually. Good apps might let you choose or show where the data is stored, so it can be put in a directory where you make backup. Bad apps might just write it somewhere where the permission allows it and use arbitrary file and directory names.

When programming an Android device there is the PackageManager class that can be used to retrieve more from the device.

Chapter 5. Data Exchange

The Android devices are made for data exchange, but there are too many interests in the data. The interest of the Android user is just one out of many and sometimes it seems that the user has low priority. Other interests are: Knowing all about the user and sell whatever to whoever and make money with it.

The user gets stuff for free or almost free, but it is like fishing, maybe there is a hook in the bait. Luckily the android market is open and bad apps get bad comments and might therefore disappear.

Therefore the following describes data exchange in a maybe a bit hard way:

Data exchange via FTP

Android as FTP server

Apps that can act as FTP server are:

1. Wifi FTP Server (free version that wants to sell pro version)

2. Ftp server (The Olive Tree)

A root directory can be specified that holds the data to be backed up or/and synchronized.

On the other side a FTP client needs to be installed.

1. **curlftpfs** mounts a FTP server in a local filesystem

2. Filezilla (snyc must be done manually, but differences are shown)

Android as FTP client

The file manager Total commander with its plugin FTP-Plugin is a FTP client.

Data exchange via USB

When plugged to USB older devices act as expected and pop up as memory devices. Under Windows they get a drive letter.

Since Android seems to be really attracted to Microsoft, newer devices do not support anymore this expected behavior. Reasons for that might be data security. Those devices support instead of the commonly used memory device behavior MTP. MTP is the Microsoft media transfer protocol (alternatively the Picture Transfer Protocol PTP can be selected). Under Windows those devices pop up as multimedia device where no drive letter is assigned. This has a side effect, most application can not save a file to the device. So two steps are required store the file somewhere and then use the file manager to move it to the device.

mtp under linux

Under Linux the library **libmtp** needs to be installed **libmtp**. libtmp comes with simple commands as **mtp-detect** that should work before going ahead. Linux is able using **libmtp** to communicate to such devices without mounting them and using the mtp protocol. **gmtp** is a graphical front end for it.

On the android device it should made sure that USB gets the permission to exchange file (not just charge or exchange photos)

Modern featured Linux desktop environments have file managers that allow to exchange data with mtp devices (but this is not via a mounted file system).

For gentoo there is the mtp useflag that puts mtp functionalities to programs

Note

When having directories with more than thousands files them mtp seems to block. Using ftp might then be the way to go

Samba

Samba uses Common Internet File System (CIFS) and is very commonly used to bring Windows PC's into a Linux network. Samba can also be used to exchange data with Android.

SMBSync2 allows to synchronize a Android devices directory with a directory on a Samba server.

Sms from Android

To receive or send an sms plug phone (e.g. samsung galaxy s2, old phones as nokia supporting enough AT commands should also understand it) via USB to computer and check that a file as `/dev/ttyACM0` is created. Then open terminal program as **gtkterm** and connect it 9600baud.

To receive sms: Type in the following (replace *nnn* with your phone number) after AT +CNMI=1,2,0,0,0 wait for an incoming sms.

```
AT
OK
AT+CMGF=1
OK
AT+CNMI=1,2,0,0,0
OK

+CMT: "+4179nnnn",,"15/06/12,17:31:13+08"
This is a sms
```

To send CTRL+Z (ASCII char 26) will terminate the sms text and sends it out

```
AT
OK
AT+CMGF=1
OK
AT+CMGS="+4179nnnn"
> This sms goes out

+CMGS: 6

OK
```

Chapter 6. Apps

WhatsApp

To get WhatsApp on a Linux device open in a browser https://web.whatsapp.com/ then open whatsApp on the phone and select Linked Device and scan the QR code.

Camera

If accidentally deetedd a photo the Gallery App might have an album where recently deleted photos appear and can be restored.

Also File Manager Apps might have a similar feature.

Motion Photos

Motion Photos are photos with a small video sequence.

Important

Don't use Motion Photos when high quality photos are required.

Motion Photos might be device specific implemented. There is a high quality picture and a more compressed video sequence. Pictures can be exported from the video sequence but have a much lower quality than expected (video quality versus picture quality).

The video can usually be exported from the motion pictures.

More tricky is to get just the high quality picture out of the motion picture. The video sequence might be device specific embedded in the EXIF (EXchangable Image Format).

exiftool -G1 -a -u -n <*name*>.jpg might show unknown markers and segments where the video is inside

Ways to remove the video are:

- Open the picture in Gimp and store it without the EXIF data.
- **exiftool -all= -overwrite_original <*picture*>.jpg** removes all EXIF
- Create a file with known EXIF data and remove all EXIF data from the jpg, then re-add the known EXIF data back to the jpg. Those 3 steps can be done using the following **jpg_exif_cleaner.sh** script:

```
#!/bin/bash

# Show help message
show_help() {
    echo "Usage: $0 /path/to/jpgs"
    echo
    echo "Cleans EXIF data in jpg files."
    echo "Creates cleaned JPGs in the same directory, prefixed with 'clean_'."
    echo
    echo "Options:"
    echo "  -h, --help    Show this help message and exit"
}
```

```
# Check for help option
if [[ "$1" == "-h" || "$1" == "--help" ]]; then
    show_help
    exit 0
fi

# Check if a directory was given
if [ -z "$1" ]; then
    echo "Error: No directory provided."
    show_help
    exit 1
fi

# Change to target directory
cd "$1" || { echo "Directory not found: $1"; exit 1; }

# Process JPG files
for file in *.jpg; do
    [ -f "$file" ] || continue  # skip if no matching files
    base="${file%.jpg}"
    clean_file="${base}_clean.jpg"
    echo "$clean_file"

    # Step 1: Save original EXIF
    exiftool -exif:all -tagsFromFile "$file" -o "${base}.exif"

    # Step 2: Strip metadata and write cleaned file
    exiftool -all= -o "$clean_file" "$file"

    # Step 3: Restore EXIF to cleaned file
    exiftool -tagsFromFile "${base}.exif" -exif:all -overwrite_original "$clea

    # Clean up
    rm "${base}.exif"

done
```

Observe the different file sizes.

Google Photos

The concept behind Google Photos is that the Photos are stored on Google Servers that might have more memory space than the Smart Phone. It supports also multiple Smart Phones and allows also Desktop PC's to see the photos. It is full of other features. Obvious Google likes to earn money for additional memory space and service.

Deleting photos on the device as in the gallery app does not delete the photo in Google Photos.

Make first photo backups from the gallery app before deleting photos in Google Photos.

Important

Google Photos wants to be the master place for the photos.

It might look also in other than the `DCIM/` directory. On a Honor device files got deleted from a user created directory. But are actually renamed to hidden files as `.trashed-<filename>`

> When taking lots of photos and then check later in the Gallery what should be deleted and what kept will not impact Google Photos. Google Photos might act as a photo magnet and keep unwanted photos.

The free space on Google Drive is limited, to see how much is used go to Manage Google Account => Payments and Abo => Manage Memory

It is easier to find how to pay more money then to find out how the memory is used.

Chapter 7. USB

USB seems to be well know but Android devices have some additional USB features. The original USB connector has 4 pins and has a clear direction host to slave devices. Originally Android devices acted as slave devices. USB connector are relatively simple and therefore robust and are almost the only connector on a android device beside the headset connector. Having a tabled device rises the issue, that sometimes it should act as a host (when connection a memory device and sometimes acting as a slave when connection to a PC. A solution would have to be adding two USB connectors for that. But an other implementation OTG (on the go) https://pdfserv.maximintegrated.com/en/an/AN1822.pdf allows acting on demand as slave or host. Mini and micro USB connectors have an additional pin and have therefore 5 pins. The device that has set this pin to ground acts as host.

Note

It is not clear how a cable having a standard USB 4 pin connector on one side and a mini or micro connector on the other side is wired. Some have the fifth pin connected to ground others have it unconnected. Such cables not having the fifth pin connected to ground can not be used to charge a modern OTG device, since the device might act as host.

An other problem is that android devices are usually not rooted. This means no changes to the kernel and no additions of device drives can be added. On a PC attaching a USB device as a DVB-T USB receiver requires a driver to be installed, but this is not allowed not having root permission. Additionally the Android device might be a USB slave and the device driver is written for a USB host. To have a solution android has the Accessory Mode. The Android device kernel supports this and acts therefore as a link to the applications. On the USB level the Android device acts still as slave, and therefore the peripheral device connected to the android device needs to act as USB host. Additionally the peripheral device needs to have the accessory mode protocol implemented. This sounds complicated an is probably. However there are solutions available as the FT311D chip from FTDI http://www.ftdichip.com/Products/ICs/FT311D.html that acts as Android accessory device and allows to attach electronic components without any need of doing programming on the peripheral device.

The fifth pin of the micro USB plug can have more functionalities by having a resistor toward ground, it can identify additional states. An interesting state is an android device might enter accessory mode acting as USB slave but still supply power to the accessory.

Some other standards make use of the USB Plug to use it as audio, video or headset output.

Newer devices have USB Type C that is more robust (higher lifetime, no polarity to be plugged) and has more functionalities (automatic OTG) and a higher performance.

Chapter 8. Filesystems

It is disappointing when knowing that Android is Linux and knowing what features Linux filesystems support (links, UTF8 filenames, ...) and then Android tells that it supports just FAT for the memory cards.

Fortunately it can be checked what is really supported by installing a Terminal Emulator app and type **cat /proc/filesystems**. Its quite obvious that the internal stuff does not run on FAT and therefore also the memory card inserted can have a Linux friendly filesystem as ext4.

Unfortunately this does not mean that the system mounts automatically filesystems other than FAT. The next hurdle is that **mount** can needs root permission.

Important

As used in Linux files as `.<filename>` that start with a "`.`" character are hidden files and will not be shown. Android makes use of that so consider to check the settings in data manager app to show also hidden files.

Data manager Apps might support the feature clean up duplicate in the local data.

Data manager Apps might have a place where last deleted files are stored for the next 30 days

Chapter 9. Encrypted files

There might be lots of programs that do that but nice is it if the data is encrypted in a standard way that also allow other programs and operating systems to decrypt it.

A standard method is using **encfs**. It is actually not an encrypted filesystem, since each file and directory can be seen. However the directory and file names and obviously the data inside the file is encrypted. This allows easily to copy the data from the phone to a computer as for backup reasons and then decrypt it and work on the data on the pc.

Install on the android phone apps as:

1. EDS Lite

2. Encdroid

On the linux pc install **encfs** and a gui front-end as **gencfs** https://www.linurs.org/gencfs.html.

gencfs allows to do the necessary setup using its gui. For manually doing it at the command line do:

mkdir ~/Phone/Encfs/.crypt to create the directory where the encrypted stuff is

mkdir ~/Phone/Encfs/crypt the directory that serves ans mounting point

encfs ~/Phone/Encfs/.crypt ~/Phone/Encfs/crypt to create the password the encryption method

When done **fusermount -u ~/Urs/Phone/decfs** unmounts it.

Now copy/synchronize the directories to the phone and start encdroid and import the volume.

Chapter 10. Calendar

It is important to understand that calendars under android have back-ends and front ends. Front ends can show more than on calendar back-end. An important topic is calendar synchronization among different devices. Google calendar works well and is also be able to synchronize itself with the Linux thunderbird calendar when having installed the necessary plugins (lightning and provider for google calendar).

Chapter 11. Wireguard

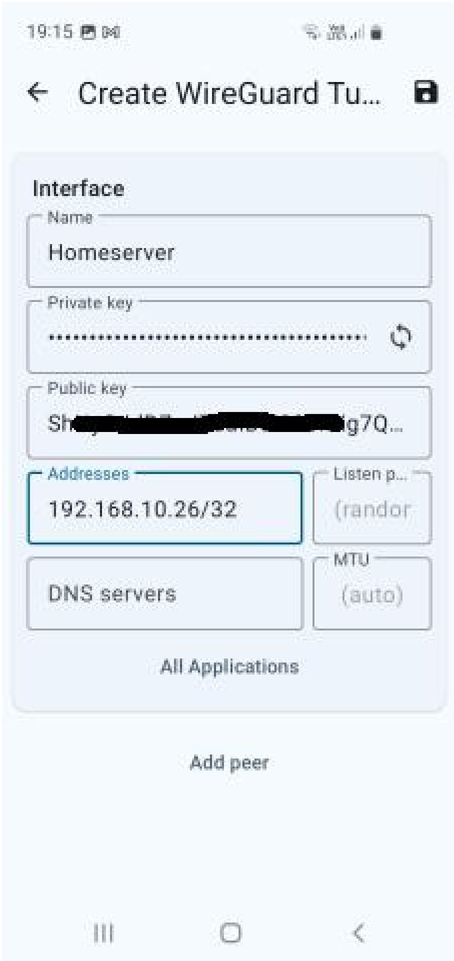
Wireguard creates a VPN (Virtual Private Network) tunnel between a the Android device acting as client and a Wireguard server.

Install the Wireguard App and create from scratch a new tunnel

Figure 11.1. Create the Tunnel

Give a name to the VPN tunnel and define the Interface by creating the private key and getting the public key, add the address received from the wireguard server administrator.

Figure 11.2. Define the Interface



Important

Check what applications are allowed to use Wireguard: All Applications can use the tunnel or just the ones that get selected.

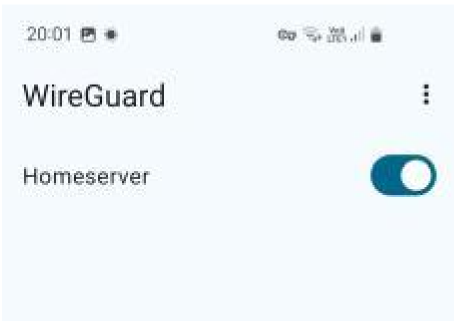
Copy and past the public key into a mail to the wireguard server administrator

Define the Peer (the connection information of wireguard server) with the values from the wireguard server administrator. In the example 3 IP address ranges are listed. The Android device is allowed to be (DHCP) in those networks.

Figure 11.3. Define the Peer

Turn on the VPN tunnel

Figure 11.4. Turn on the VPN Tunnel



Test the VPN tunnel using a browser do not use https use just http or simply type in the IP address

Figure 11.5. Test the VPN Tunnel

Chapter 12. SSH

SSHDroid is a server on the android device. On a Linux computers console, login is done with **ssh -p 2222 root@192.168.1.35** and **exit** (maybe multiple **exit** commands are required) to log out.

It is also possible to login via a public key file that needs to be first transferred from the android device to the Linux computer. The manual says any user name will work, however it must be a user that exists on the android device (as system)

ConnectBot is a ssh client

Termux a terminal program where

pkg update && pkg upgrade Update packages

pkg install openssh gets ssh comand

pkg install inetutils gets ping, ifconfig, etc.

Important

If used on a Wireguard VPN connection make sure those applications are listed as allowed applications

Chapter 13. GPS

GPS Essentials

GPS Essentials does what a dedicated outdoor hiking, biking GPS does. Tracks can be recorded and is continuously recorded in background also when other stuff is made. When done, the data can be exported to a file. The openly documented and pure xml format gpx is recommended for that. Alternatively there are the google kml formats, but they are not well supported by google earth. It has a portable map application that can downloads maps that can be used when offline.

Chapter 14. Multimedia

Dailyme

Dailyme allows to subscribe to TV series and downloads them to the phone (per default WLAN only). Then they can be viewed offline. Unfortunately the films are stored in an mp4 format that no other player understands. Probably this is intended. It has a restriction to not use a HDTV screen, however some Andoid devices with build in HDMI or video output as the Meteorit TV box or Tables do not handle that so it works perfect. Dailyme just runs on WLAN but not on Ethernet.

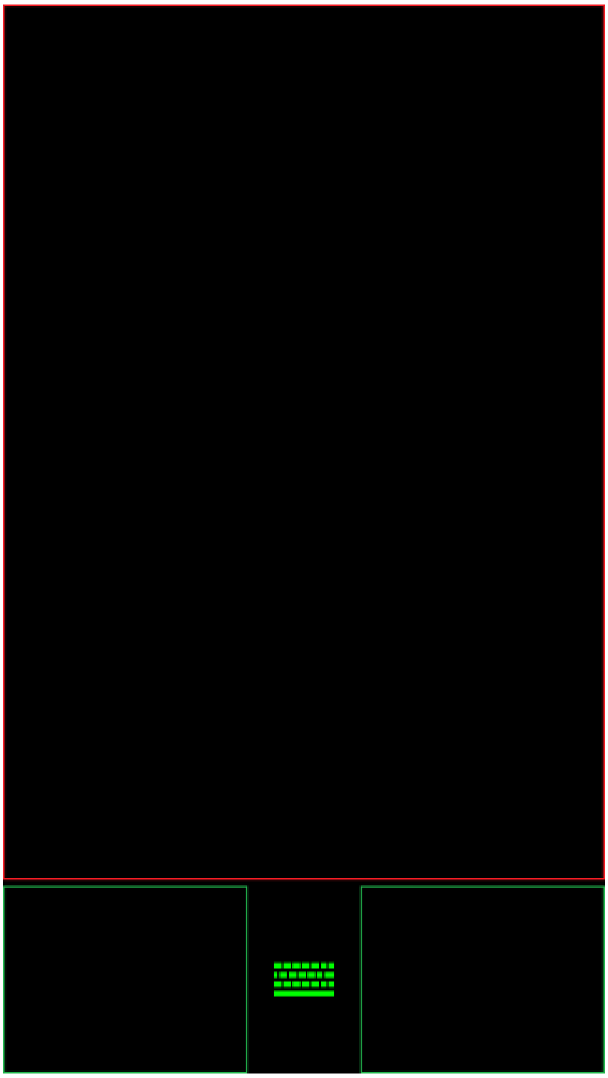
Chapter 15. Remote control a Linux PC

RemoteDroid is an app that can act as a keyboard and mouse. A server needs to be run on the Linux PC and is available via http://www.remotedroid.net. This is a java archive that can be run directly and needs just a java setup and can be run directly as **java -jar ./RemoteDroidServer.jar**

Most desktop system as gnome have a Startup application setting where the command can be added so it starts automatically. Unfortunately on the desktops the RemoteDroid window pops up. To avoid this a script can be started that calls first RemoteDroidServer and than the application:

```
#!/usr/bin/python3
import subprocess
import time
subprocess.Popen(["java","-jar","~/RemoteDroidServer.jar"])
time.sleep(5)
subprocess.Popen(["<add here the application>"])
```

Figure 15.1. RemoteDroid



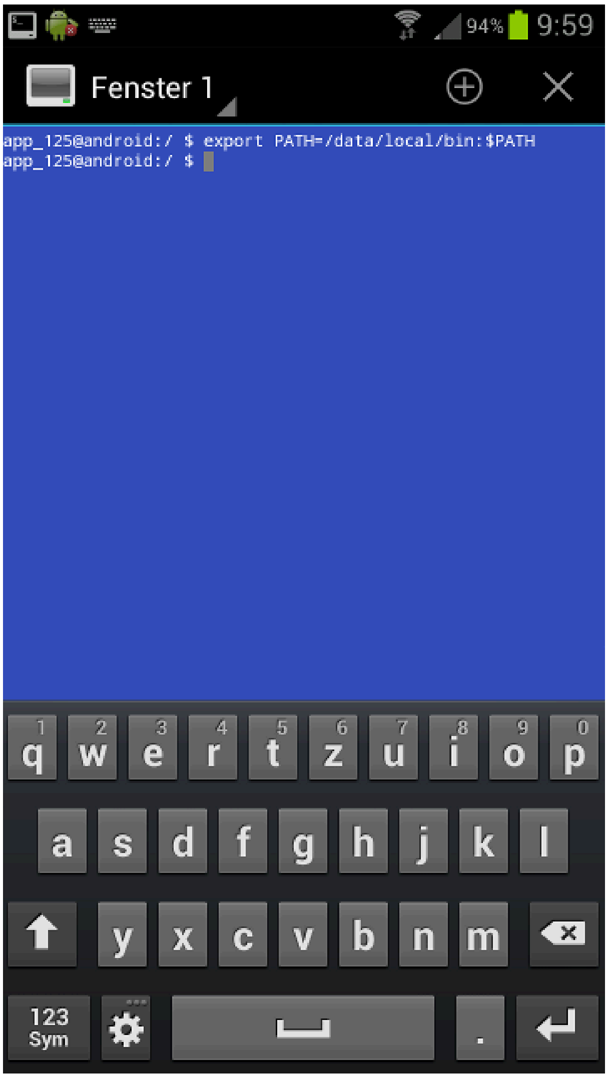
Chapter 16. Inside Android

Android is Linux, but something as a standard Linux does not exist. Linux means "just" the kernel. To have a working system the Linux kernel needs some more packages and setup. Android went it's own ways that is very different from a regular Linux desktop setup (boot process, X server), but taking over a lot of the standard features to protect the device against all unwanted things that could be happen. One fundamental thing is the root permissions (Administrator). Every file, program or process belongs to a user and can be restricted. Android devices make use of that. In a console type **id** to see who you are and to what groups you belong. You probably will not be root and therefore a lot of system commands and modifications are not allowed to do. This is OK like that and prevents you from damaging your Androids software setup.

There are ways to root the device to be able to work as root, but this is just for experienced professionals and people that know how to recover the physical device. To run such tests it is recommended to install Android on a PC as in virtual box where snapshots can be made and going back to the installation as it was before a screw up.

There are Linux console apps that allow you to observe Linux:

Figure 16.1. Terminal



As any other Linux system the kernel is loaded and starts the program **/init** (this might not always true since the program started can be passed via boot loader). `/init.rc` and `/init.<platform>.rc`

The C++ programmed system process **/system/bin/app_process** zygote is started by **/init** and is the key, to run other processes.

Regular applications are written in java, however using special byte code requiring the register based Dalvik virtual machine instead of the stacked based standard java virtual machine. Due to this small difference standard java byte code can be converted to Dalvik byte code. This allows to use any standard Java environment for development. When finished the Dalvik compiler **dx** can convert java byte code in dex-code (Dalvik Executable).

Chapter 17. Developer Options

The developer Options are usually turned off and hidden. The Developer Options menu can be enabled by going to the System settings: Select "About". Tap "Build Number" seven times. Go back. Go to "System" and "Developer Options" should be visible. There are some small differences between the different Android versions ans Devices.

Android Debug Bridge

adb the Android Debug Bridge establishes the connection to the device as via USB. It is used for development purposes and to control and manipulate files on the device via PC. It usually has to confirm on the devices screen that **adb** is permitted to be used.

Under Gentoo Linux it can be installed using **emerge android-tools**

It brings the following applications :

- /usr/bin/simg2img
- /usr/bin/append2simg
- /usr/bin/lpdump /usr/bin/adb
- /usr/bin/img2simg
- /usr/bin/lpunpack
- /usr/bin/lpflash
- /usr/bin/fastboot
- /usr/bin/lpmake
- /usr/bin/ext2simg
- /usr/bin/lpadd
- /usr/bin/mke2fs.android
- /usr/bin/e2fsdroid

Note

Usually **adb** is disabled. Settings, System, Developer Options might be the place to enable it.

adb requires to be connected with the android device typically via USB.

See https://wiki.lineageos.org/adb_fastboot_guide.html and https://wiki.archlinux.org/index.php/Android_Debug_Bridge

lsusb shows the devices **lsusb -s 2:5** just the device on bus2 device 5 that has previously been identified as the phone **lsusb -s 2:5 -v** gives then the verbose info where idVendor and idProduct can be spotted:

If done, as regular used do **adb devices** where the phone should be listed.

adb shell gives a Linux shell on the PC

adb shell ls -la /sdcard/DCIM/Camera lists photos

Use a desktop PC to see the androids device screen

Install **scrcpy**

With **adb** and therefore USB debugging activated on the android device plug in the USB connection to phone. In the PC as regular user start **scrcpy** in a console window.

A window pops up with the phones screen content and mouse navigatio support.

It can also use other displays

scrcpy --new-display=1920x1080

RNDIS USB-Ethernet

In the Developer Options RNDIS (Remote Network Driver Interface Specification) can be enabled. RNDIS allows a remote computer to use the Android device as Ethernet interface (WLAN or Mobile data).

Rooted Android

Open/Install a terminal application.

If **#** appears as prompt then the device is rooted. If **$** then the terminal works as not root (it might still be a rooted device). Try typing **su** to see if the prompt changes to **#**.

Different apps as root checker allow to see if the device is rooted or not.

On rooted devices in the System menu under Developer Option root permissions for apps can be enabled. If enabled and **su** is entered in the terminal. a question will pop up asking to grant root permission for the app. Additionally such granting events are counted.

Chapter 18. Custom Firmware

Special Boot Modes

Recovery with TWRP

The recovery mode can be activated from Samsung Galaxy devices by a special key combination at power up **home**, **volume up** and **power**. Alternatively ./**adb reboot recovery** goes to download mode without need to press all the buttons.

In this mode different options are possible. To have more flexibility as required for custom OS TWRP can be installed.

TWRP is the TeamWin Recovery Project a GUI based custom touch recovery from https://twrp.me/. It allows to backup/restore partitions, install Custom Firmware and more.

heimdall flash --RECOVERY recovery.img --no-reboot

heimdall flash --RECOVERY twrp-3.6.2_9-0-i9300.img --no-reboot will install it.

Important

There is also the twrp app. This is a regular app and not a recovery. It is used to deal with recoveries (update, download, restore). To be useful the twrp app needs to be run with root privileges.

Important

After running it, the device needs to unplugged from the PC and then **home**, **volume up** and **power**, to have twpr persistent installed

Important

Having twrp, it is recommended to backup the devices original ROM to an internal SD card, before attempting to install a custom ROM.

Download with heimdall

In the download mode the device listens to commands to download data in the flash device.

It can be activated from Samsung Galaxy devices by a special key combination at power up **home**, **volume down** and **power**. Alternatively ./**adb reboot download** goes to download mode without need to press all the buttons.

heimdall is a tool to flash firmware (aka ROMs) onto Samsung Galaxy S devices. https://glassechidna.com.au/heimdall/Source is hosted on Github: https://github.com/Benjamin-Dobell/Heimdall/. **heimdall** is a command line tool but the separate **heimdall-frontend**. **heimdall** might be installed as regular package of the desired Linux distribution. On gentoo **emerge app-mobilephone/heimdall**

PIT files hold the Partition Information Table of the device and can be read or written to the device.

heimdall print-pit reads the partition table from the device. **heimdall-frontend** allows to read and write the pit to a binary file.

LineageOS

If the Android is outdated on a phone and refuses to install the newest apps, then LineageOS https://lineageos.org/ might be a solution.

Other reasons are: Getting rid of all annoying bloated, pre-installed applications and features, creating a simple phone as for elderly people, ...

Android consists mainly of the following parts: The open source part, device specific drivers and the google part containing playstore, gmail and others. `linage-<version-device>.zip` is the image to be "sideloaded" into the file.

Important

The steps installing LineageOS can brick the device when something goes wrong and makes it permanently unusable. No guarantee can be given within this guide that this does not happen.

Due to issues the google part is not part of LineageOS but can be installed as well.

Important

To have Google play store, it needs to be installed before LineageOS starts the first time. So watch out the LineageOS installation. Depending on the LinageOS version this file is called as `MindTheGapps-<version and arch>.zip`

LineageOS has its wiki https://wiki.lineageos.org/

The supported devices are https://wiki.lineageos.org/devices/

Mostly newer devices are supported, older as the Samsung Galaxy S III (International) (i9300) get no more maintained. Luckily under the wiki device list also those devices are listed where a guide how to install them can still be found.

LineageOS has for the supported architectures (as arm or arm64) two zip files under Download Extras https://download.lineageos.org/extras. One is to get root and one to remove root. So no strange untrusted apps need to be used. In the recovery mode and for example using twrp those zip files can be installed.

Important

LinageOS has also a `addonsu-<arch>.zip` file that gets **su** support.

LinageOS comes with Trust that shows if root access is disabled. If Trust claims that SELinux is not enabled. It can be enabled starting in Rocovery mode then under Advanced there is a terminal in this rooted terminal. **sestatus** or **getenforce** shows the SELinux status, **setenforce Enforcing**

Installing Apps without Google Play Store

Yes, this is possible. It might be necessary for self developed aps and if Google Play Store is not installed. If Google Playstore is installed then apps should be installed with it so the possibility of getting malware is lower and updates to the apps are announced.

To install it without Google Play Store, the apps need to be available as `<filename>.apk` A good source to get such files is https://www.apkmonk.com/

Install custom ROM

Having installed the recovery twrp and having adb the installation goes easy.https://wiki.lineageos.org/devices/i9300/install

Download a LineageOS zip file as Lineage 16.0 that comes with Android 9 (alternatively as the lineage guide shows you might want to try to build one yourself) to your computer

Download the google apps https://wiki.lineageos.org/gapps.html to your computer

Download the Lineage extras https://download.lineageos.org/extras to your computer

Put the android device in recovery mode using the special button press power up sequence or ./**adb reboot recovery** to get twrp

Consider to backup the existing system ROM

Do Wipe and Format Data

Go to Advanced Wipe and select Cache and System and then wipe

Do advanced ADB Sideload on the device

On the computer do **adb sideload *`<filename>`*.zip** to move the file to the device and repeat this for all files

Use twrp to reboot the system

e OS

https://doc.e.foundation/devices https://doc.e.foundation/devices/i9300/install

The e foundation creates open source, privacy-enabled smartphone operating systems https://e.foundation/. /e/OS is a complete, fully “deGoogled”, mobile ecosystem.

There is no google play store. it uses the https://microg.org/ project that includes an app store with the most popular android apps. The is a built in function to request adding apps into the store.

It is a fork from LinageOS and might have better support for particular old phones https://doc.e.foundation/devices

The installation as for a Samsung i9300 follows the same procedure using TWPR and Heimdall seehttps://doc.e.foundation/devices/i9300/install

postmarketOS

postmarket OS is an initiative to port Linux onto devices and therefore get better support for updates of the operating system https://postmarketos.org/ and https://wiki.postmarketos.org/wiki/Main_Page

The goal is: Let's keep our devices useful and safe until they physically break!

Chapter 19. USSD

Instead of calling a number a Unstructured Supplementary Service Data (USSD) code can be called.

`*#<number>#`

Some codes are sent to the mobile network operator as money related things. Other codes handles the phone itself.

Chapter 20. Emergency mode

Pressing on/off a couple of time fast send an SMS to your emergency contacts (if configured)

Chapter 21. Ethernet

An USB-C to Ethernet device adds Ethernet to a android device. It can work in two direction, to connect an android device to the wired Ethernet or with Ethernet-tethering (personal hot-spot, advanced) provide wired Ethernet to other devices.

Chapter 22. WiFi bridge

Wi-Fi Bridge makes that the android device appears as access points for other devices. It is up to the android device to handle the Internet connection WiFi or Mobile data or even wired Ethernet.

WLAN devices can then log in and have Internet connectivity.

For wired Ethernet, a device being able to work as Wi-Fi Bridge can be used. So the android phone acts as first Wi-Fi bridge, the second Wi-Fi bridge logs in and creates the wired Ethernet that can be fed to a Ethernet switch.

Chapter 23. Development

Note

This section is outdated

Development tools can be found under http://tools.android.com/. In the past many different tools using eclipse were interacting with each other. This needs deep knowledge but it is also easier to locate a problem in the tool chain.

The Android Studio should bring an uniform development environment.

Android Studio

The early version of Android studio can be found under http://developer.android.com/sdk/installing/studio.html. There is a migration guide how to move Android eclipse Projects to Android Studio.

ADK and NDK

There is the Android software development kit (SDK) for java based applications. Typically and the recommended way to program apps is making use of java.

But there is also the Native Development Kits (NDK) available to support other languages as C and convert them into hardware dependent binary code. Reasons for using the NDK are:

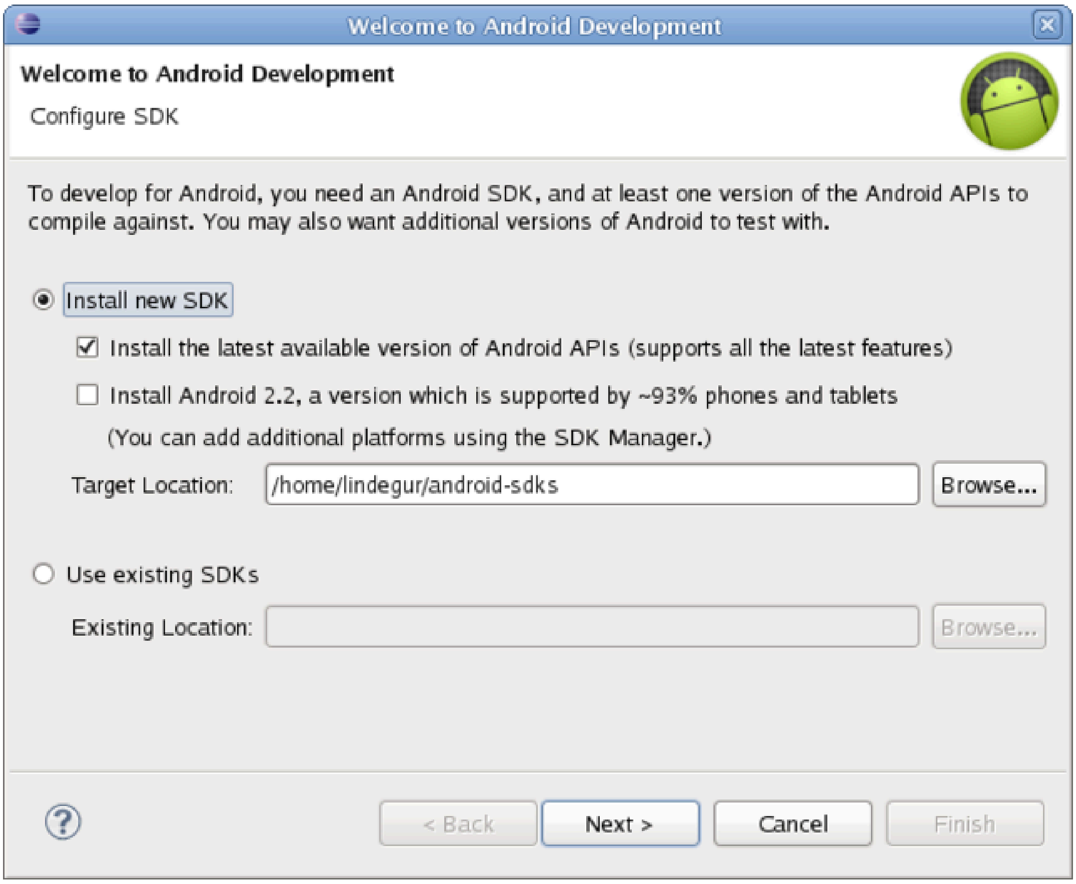
- Re-use of existing code
- speed improvement

The SDK could be obtained by http://developer.android.com/sdk/index.html but it will be better to install it via packet manager of your Linux distribution. The SDK has a GUI where the different android versions and packages can be downloaded. The SDK has all the necessary command line tools to build an android application and could therefore be all you need.

However the preferred way is using launching it via GUI instead of the command line tools. The supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) plug-in using the following link https://dl-ssl.google.com/android/eclipse/. Be aware that this plug in, will call the SDK and is not a replacement nor contains the SDK. You still need to install the SDK and now you have even an option more. The options to install the SDK are:

1. Use the package management system of your Linux distribution to install it system wide.
2. Download it manually and install it manually somewhere, hopefully to not break your system in your home directory.
3. Use eclipse and the ADT to guide you through the steps.

Figure 23.1. Link to Android SDK



Eclipse must be configured to find the SDK. The straight forward way is installing first the SDK and then the ADT. If you do it the other way you get a nice demonstration about what the two packages are.

Anyway the SDK is quite autonomous and can be updated via its gui, so your linux package management system does not get aware about those frequent updates. However regular users can not write to /opt/android-sdk-update-manager. An alternative would be to install the SDK into your home directory and keep your system untouched.

The android SDK comes with the android virtual device manager AVD. So you can create virtual devices with various options as different screen sizes. Then you can start them. The purpose of those AVD's is to use them for development and testing. **Ctrl**+**F11** turns the emulator 90°.

The configuration goes in the directory ~/.android to your system.

Installation under Gentoo Linux

Make sure your system has java installed and check the required version of eclipse.

Install the android SDK under gentoo Linux this is done with **emerge =dev-util/android-sdk-update-manager-20**. See https://forums.gentoo.org/viewtopic-t-919462-highlight-android.html

To work, the user must be put in the android group, this is done with: **gpasswd -a *<username>* android** . Unfortunately the user must be log out and log in after that to take effect.

It will be installed in /opt/android-sdk-update-manager/tools/ where the SDK manager could be started via command line as **./android**. However usually this is done via Gui of eclipse. Select at least on API (android version).

The classic Eclipse 3.6.2 or higher needs to be installed unfortunately eclipse in gentoo is too old and the gentoo maintainers struggle with the eclipse sources.

Luckily there are overlays

layman -a seden gets the necessary ebuild for a newer (but still old) eclipse. **emerge eclipse-sdk**..

layman -a java-overlay has binary versions of the newest eclipse versions, so there is no hassle with struggling compiling this complex software. **emerge eclipse-sdk-bin** and **/opt/eclipse-sdk-bin-4.2/eclipse** is then the magic command.

It is usually installed under /opt/android-sdk-update-manager there is a subdirectory called tools /opt/android-sdk-update-manager/tools where **./android** can be started using the command line.

Developing Apps with the ADT and SDK

To work with the SDK some concepts of eclipse should be understood. Additionally basic java knowledge is required. Things as how to inherit an object, overriding methods and interfaces should be known. The applications are developed within a directory structure. Being new to android application development, this structure looks a bit complicated, but when experienced, the structure gives a familiar touch to all apps available and the individual files can be found easily.

Tutorials

Follow the tutorial http://developer.android.com/training/basics/firstapp/index.html or follow the tutorial but open a sample project as HelloCompute (I know the letter r is missing but thats the case) under File->New Project->Android->Sample Project and jump over the non relevant sections in the tutorial to see if your environment works. Later and as written in the tutorial you have to click on File->New Project->Android->Android Application Project. To debug and run it a virtual device needs to be created, started and unlocked. Then select the project in android and click the run icon.

If this works then redo the tutorial again but concentrate to the project data structure and the eclipse environment your eclipse/android environment will probably be newer then the one described in the tutorial, so it is good that you know that it works with the sample. A good article about android programming can be found under http://www.vogella.com/tutorials/Android/article.html

Note

During editing code, the code is checked and when errors appear a list of proposals to solve it appears too. The tutorial does not mention for instance that some more libraries need to be imported. This can be easily fixed when the errors pop up. Obviously all error must be removed when running an app.

Structure

The goal is to not go and program, but be lazy and use what is already there. The work goes more into configuring and app than doing a lot of programming. This means you must get familiar with the infrastructure android has, so more learning than doing is the way.

The res/AndroidManifest.xml describes the applications requirements.

Activity

To create the GUI of an app, Android uses xml files to describe them. The file /res/layout/activity_main.xml is the one that appears at start. Activity means a single window or since android can just show one window at a time it is also a screen. After your program gets more complex you migt get additional Activities and therefore additional xml files as Help-Activity and Setting-Activity. In eclipse you can switch between xml text view or rendered graphical view to observe this file. This file can be edited in both views.

Those xml files are not left alone there is an associated java file under /src directory. To have you data consistent eclipse allows to create activities for you so don't create those files manually.

Android is and must be quite flexible if it comes to display sizes, it must be usable on any screen size.

An Android App Window is an activity and runs using an activity class that on its onCreate method imports an XML file containing all widgets (buttons and stuff).

Runtime concerns

The main activity object is called and shows up. As with any other multitasking system, loops waiting for an event have to be avoided. However sometimes it needs to be waited, this does not mean that the system can not do something useful as running other apps in parallel. Android uses for that callback methods. The app does its work and then leaves, but it sets up a callback method. This is usually an already existing method of an object. If the event happens later, then this callback method is called. Often those methods are empty and do nothing, however the methods can be overridden with the desired code. In Android those methods have a name that starts with the word on as onCompletion used by the mediaplayer to be called when the audio file has been played. Looking closer also the activities make use of callback methods as onCreate that creates the activity.

Strings

Android needs to support multiple languages. Therefore all strings are located in /res/values/strings.xml and should be referred in the gui.

Indent

If a button is clicked an event is produced, this event needs to be linked to an action. Android uses the Indent class for that. Indents point from the source event to a destination where an action needs to be done. Indents can also be considered as messages. When creating an indent object (an instance of the indent class) the source and destination needs to be passes. The source is in most cases simply the word "this" since source is mostly the current context.

New -> Other -> Android lets to create or overwrite different items.

Icons

To change the icon edit the default icon ic_launcher-web.png then go to New -> Other -> Android -> Android Icon Set and recreate the icon set.

Distribute the app

Android applications are packaged in .apk format and are created in the bin subdirectory of the project. Using an archive manager the contents of the .apk file can be observed. The directories res and META-INF and the files AndroidManifest.xml, classes.dex (dalvik executable byte code) and resources.arsc (binary file containing the android resource tables) appear.

Testing should of course be done before distribution via Google Play or alternative channels. To directly install the apk file found in the project directory is possible. Copy it to a real android device and enable Settings>Security>Unknown sources, use a file manager and click on it. Un-install work as well.

As shown in the tutorial and documentation it is also possible to build android applications via command line. This sounds really out fashioned, but it would allow to automatically create applications for android as Java on Linux to Java on android conversion tool.

Settings

For settings there is the Preference class that can be fed from an xml file /res/xml/preferences.xml" and produces a SharedPreference file as default as /data/data/org.linurs.irsend/shared_prefs/

org.linurs.irsend_preferences.xml that holds the data of the preferences. So the work to do is creating the xml file making the system aware of it and read out the data. File -> New -> Others -> Android -> Android XML File is the best way to create the /res/xml/preferences.xml it includes also a graphical editor so for beginning there is no need to go into XML. The strings used in the preference file should not be hard coded but taken out of the /res/values/strings.xml file, there are many reason for that as allowing to modify and translate them easily.

Settings have to be included in a PreferenceFragment. Fragments are user interfaces running inside the Activity.

Help

It is always nice to tell something to the user. This does not mean a lot of programming work. A good way is creating a dedicated help activity and just open a local html file that is placed in the assets subdirectory.

Money Making

Open Source and software for free is obviously a good thing to users and allow many hobby programmer to learn and do programing. However all people need some money to survive and most of them need to earn money.

Therefore it is desirable to convert the many hours of doing programming work into money to get something back. In android there are actually to ways:

1. Selling the apps as through Play Store

2. Adding advertising. This can be done using the AdMob SDK

Finally when working in release mode proguard can be enabled that makes it difficult to reverse engineer the code.

Chapter 24. Infrared

Note

This chapter is outdated

What is missing on most android devices is an infrared interface. Exceptions are phones as from Huawei.

One solution is to use the 3.5mm audio socket to connect an IR transmitter. Having this, many things can be done with the android device:

1. Remote controller for TV and other multimedia devices
2. Door opener
3. Camera remote controller

LIRC the Linux Infra Red Controller package http://www.lirc.org/ has been taken over to Android. Since most android devices lack on IR receiver capabilities the required config files can not be made. To made them a Linux computer with Lirc is the option.

The simplest hardware driver is http://www.lirc.org/ir-audio.html it is limited to send out about 3m. More advanced designs use extra power as from a battery to improve this as or use a bluetooth to audio converter where the android device communicates with the bluetooth devices via bluetooth and the IR transmitter is plugged to the bluetooth device. Since the bluetooth device has some power and it is stationary, any kind of IR transmitters and distances can be handled.

Irdroid sells transmitter (IR and bluetooth) and has it remote controller SW https://www.irdroid.com/

Conversal is a remote controller using LIRC (or/and for Linux multimedia centers SSH)

Androlirc is the straight forward porting of LIRC http://code.google.com/p/androidinfrared/ and is considered a legacy application replaced by Conversal.

Androlirc will probably no more further developed and Conversal seems to go into a different direction than just be a remote controller.

PhotoIRmote (not free) is controlling a camera http://www.wegroo.com/photoirmote/ . It is actually very easy to control a camera. Since a remote controller has maybe just two buttons, two audio files can be created that can be sent out using a IR transmitter. It is so simple that I will probably write my first app for that. Unfortunately my camera collection is not so big therefore I can not test and support many cameras. The problem is to know the ir wave forms and that are not well or not at all documented in public. I plan to do it sound file based, so new cameras can be added without any necessary programming.

Unfortunately Androlirc and Conversal are not stable and have a lot of issues on my Samsung Galaxy S3 sending audio to a Belkin bluetooth device. I guess that something (probably Belkin bluetooth device or bluethooth sw) handles the lirc ir audio as sound and disturbs it using maybe something as a sound compression algorithms. Indication for that came from observation recording ir signals with a native Linux PC using the soundcard with alsa and observing them using audacity or using an real oscilloscope. The signals look disturbed. Everything from a native Linux PC works but almost every thing from Samsung Galaxy S3 towards the Belkin bluethooth device does not. I did not do a lot of tests with directly connection the ir transceiver to my Galaxy S3, since it is too new. An other obstacle is that the voltage amplitude is too small to make the simple ir transmitter working.

Chapter 25. Specialties on different devices

USB types

Most newer devices do not act as a USB storage when connected to USB, It acts as MTP device or PTP.

For MTP there is libmtb that needs to run and different programs make use of that lib as jmtpfs.

For PTP there is libptp2 and gtkam.

However FTP remains a real alternative.

Google Discover

Google discover feeds the news to the device.

Huawei devices might have as default feed HUAWEI Assistant-Today that does not really provide interesting news.

Touching the home screen and going in the screen setting allows change from HUAWEI Assistant-Today to google discover.

android