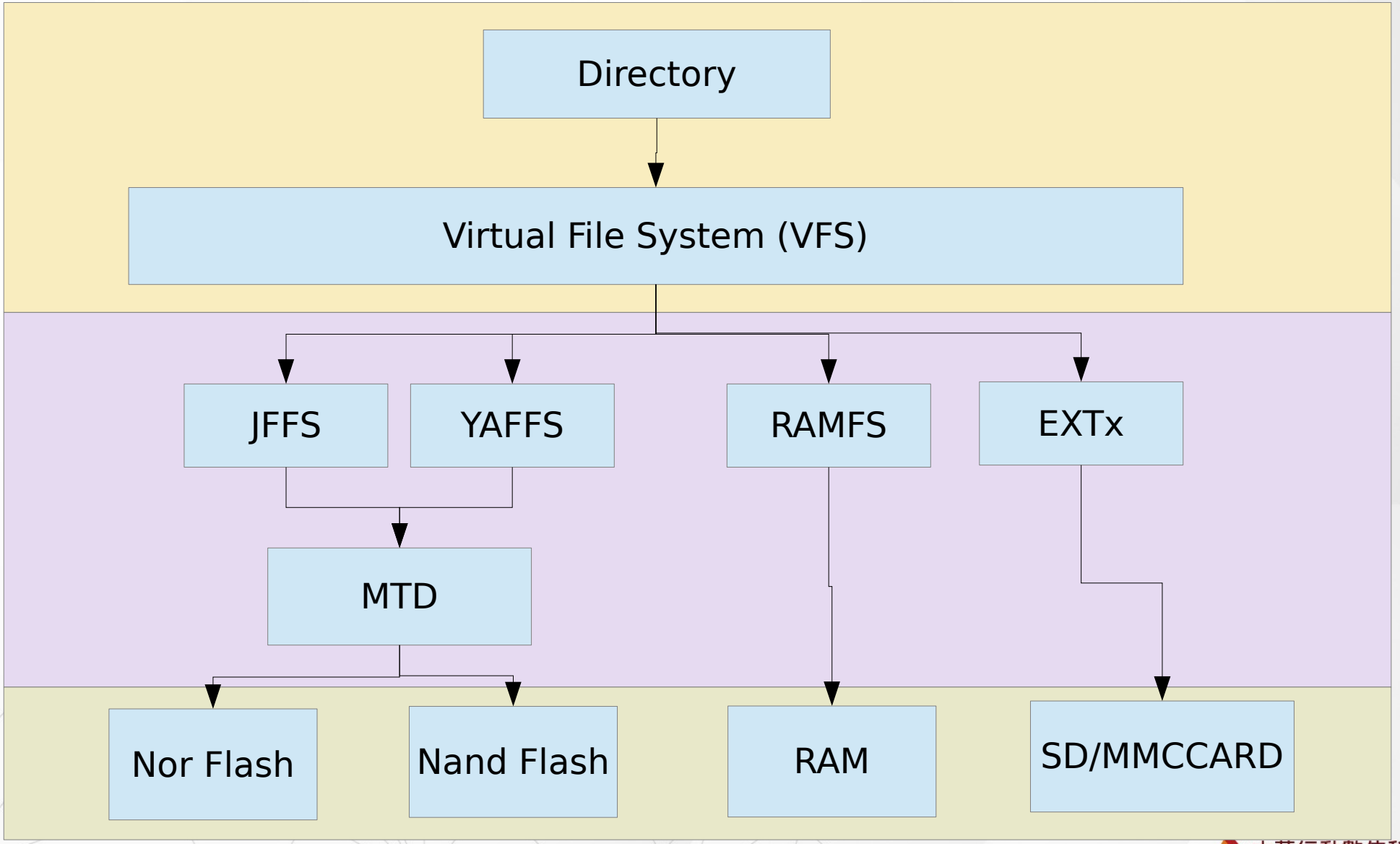
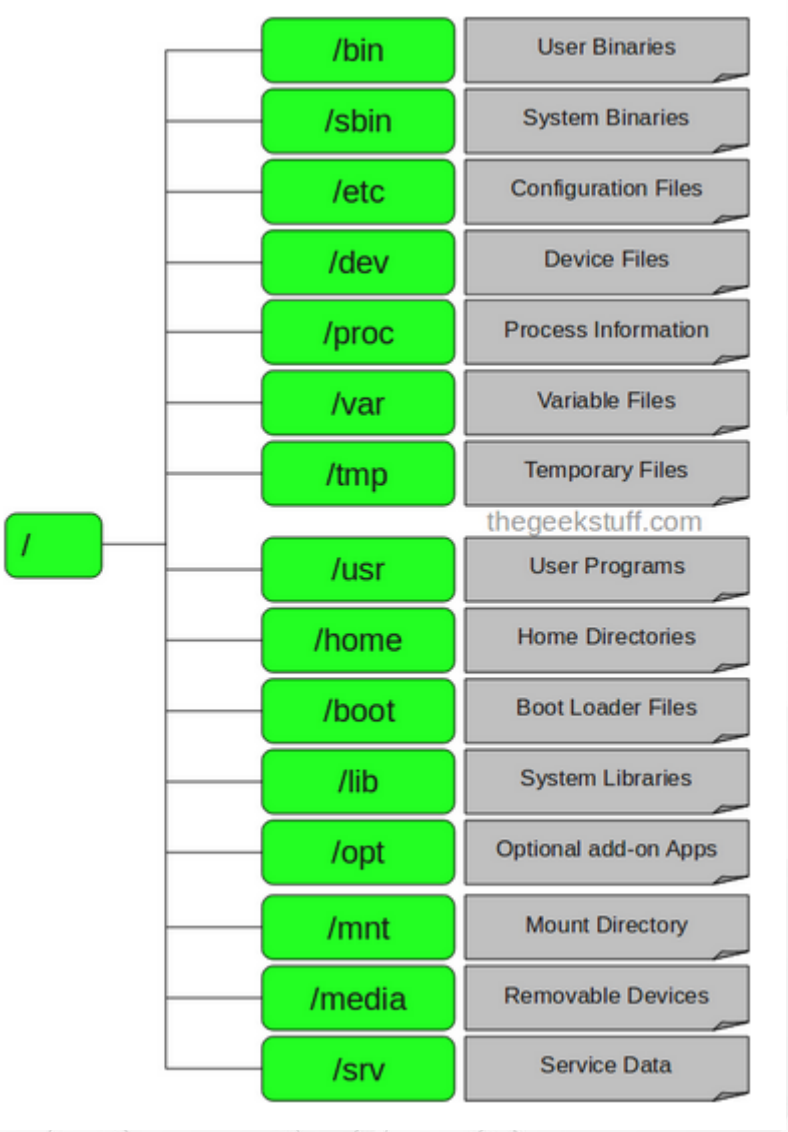
CH 7 Linux Root File System

File System in Linux



Linux RootFS Structure



Root

- Every single file and directory starts from the root directory
- Only root user has write privilege under this directory
- Please note that /root is root user’s home directory, which is not same as /

/bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

/sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

/etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

/dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

/proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

/var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

/tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

/usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can’t find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can’t find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

/home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

/boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot

/lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*

/opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

/mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

/media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

Linux System V

- AT&T developed
- The first initial process is → init
- /etc/inittab
- /etc/init.d/ and /etc/init.d/rcS

Linux System V

- Inittab

→ terminal initial setting table

```
# Startup the system
::sysinit:/bin/mount -t proc proc /proc
::sysinit:/bin/mount -o remount,rw /
::sysinit:/bin/mkdir -p /dev/pts
::sysinit:/bin/mkdir -p /dev/shm
::sysinit:/bin/mount -a 2>/dev/null
::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::respawn:-/bin/sh
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
#ttyFIQ0::respawn:/sbin/getty -L  ttyFIQ0 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
::shutdown:/etc/init.d/rcK
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
```

Linux System V

- respawn

  → The process will be restarted whenever it terminates

- sysinit

  → sysinit actions are started first, and init waits for them to complete

- askfirst

  → For askfirst, before running the specified process, init displays the line "Please press Enter to activate this console

- shutdown

  → shutdown actions are run on halt/reboot/poweroff, or on SIGQUIT

https://git.busybox.net/busybox/tree/examples/inittab

Linux System V

- /etc/init.d/S*

  → initial system script

```
[root@rk3399:/etc/init.d]# ls
S01logging      S22hdmion    S50link_iq            S80dnsmasq
S10init         S30dbus      S50sshd               S99input-event-daemon
S10udev         S40network   S50telnet             rcK
S20urandom      S41dhcpcd    S50usbdevice          rcS
S21mountall.sh  S50launcher_ S66load_wifi_modules
```

Linux System V

- /etc/profile
- /etc/profile.d/

  → environment initial script

```
[root@rk3399:/etc/init.d]# ls
S01logging      S22hdmion     S50link_iq            S80dnsmasq
S10init         S30dbus       S50sshd               S99input-event-daemon
S10udev         S40network    S50telnet             rcK
S20urandom      S41dhcpcd     S50usbdevice          rcS
S21mountall.sh  S50launcher_  S66load_wifi_modules
```

Rockchip Launch Script

- /etc/init.d/S50launcher

  → launch rockchip or customer applications

```
start)
            printf "Starting launcher: "
            export LC_ALL='zh_CN.utf8'
            export QT_QPA_PLATFORM=wayland

            # music
            /usr/bin/audioservice &

            # bt
            /usr/libexec/bluetooth/bluetoothd --compat &

            #for kmssink
            #export QT_GSTREAMER_WINDOW_VIDEOSINK=kmssink
```

BusyBox

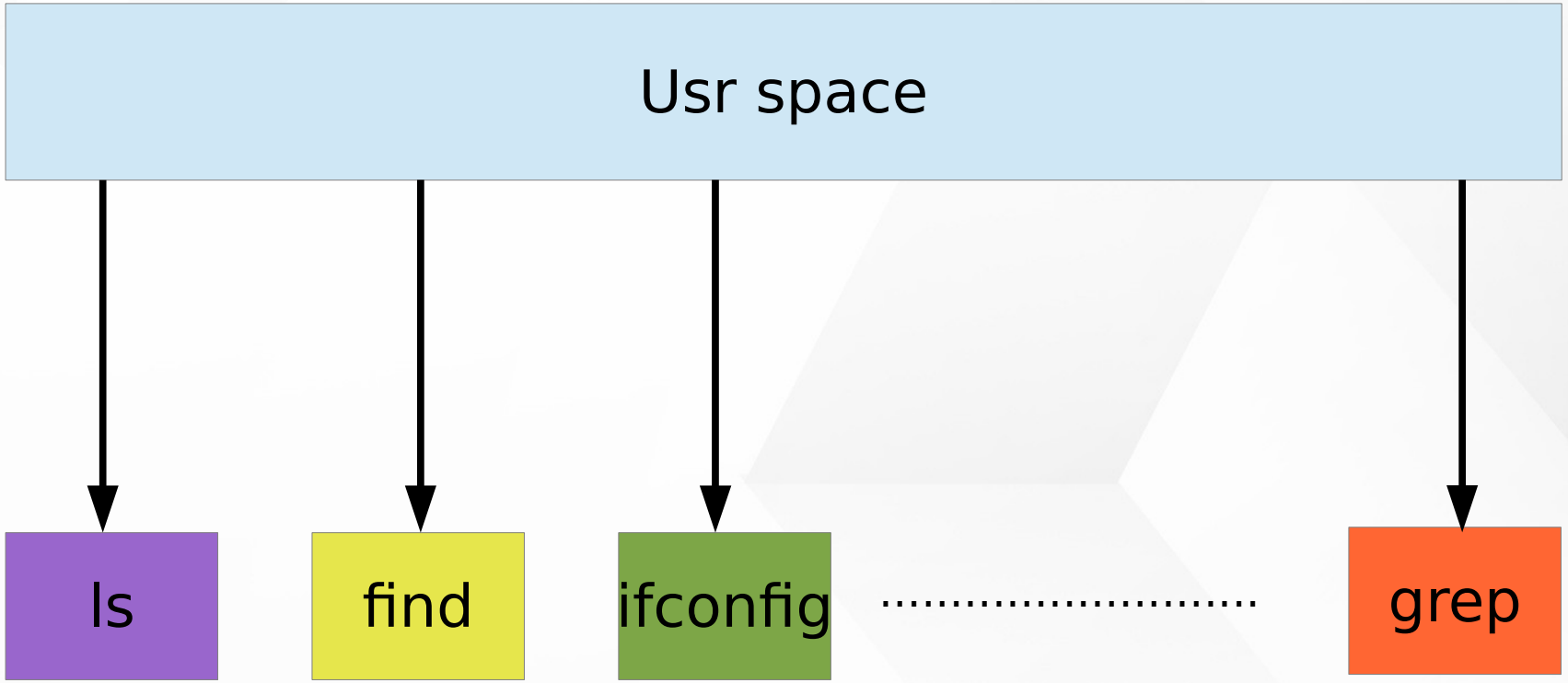
BusyBox

- Linux system needs a basic set of programs to work
  - Init program
  - shell
- In normal Linux systems, command are independent programs
  - Binary size large
- BusyBox down size these program

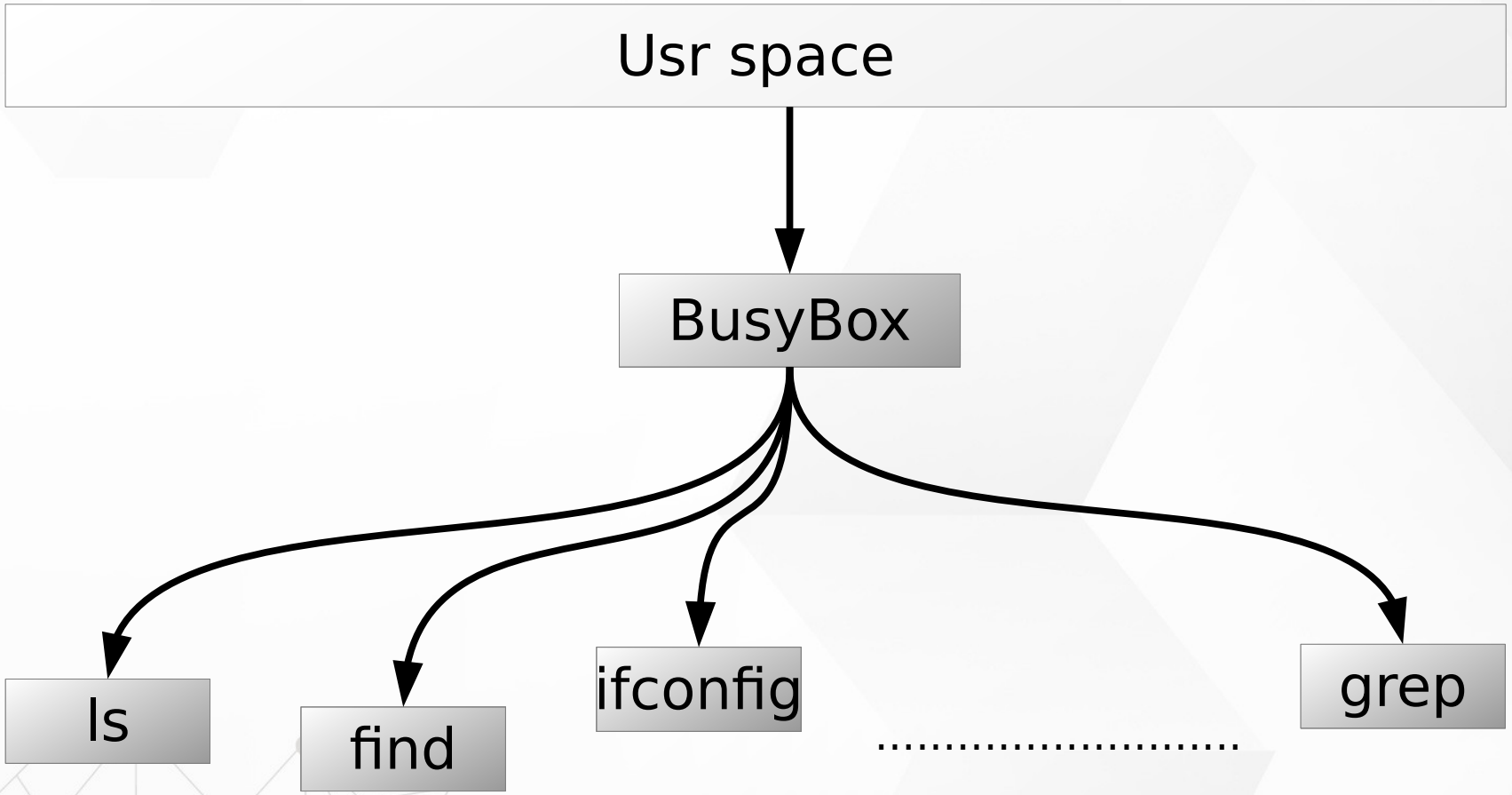
BusyBox

- Rewrite of many useful Unix command
  - Down size
- All the program are compiled into a single executable, /bin/busybox
- It can configure all command
- http://www.busybox.net/

RootFS no BusyBox



RootFS on BusyBox



Busybox init

- Busybox provides an implementation of an init program
- A single configuration file: /etc/inittab
  - Each line has the form <id>::<action>:<process>
- Check examples/inittab in Busybox for details on the configuration

Configuring BusyBox

- http://busybox.net

  → wget https://busybox.net/downloads/busybox-1.30.0.tar.bz2

- Configure BusyBox

  → make menuconfig

  → make defconfig

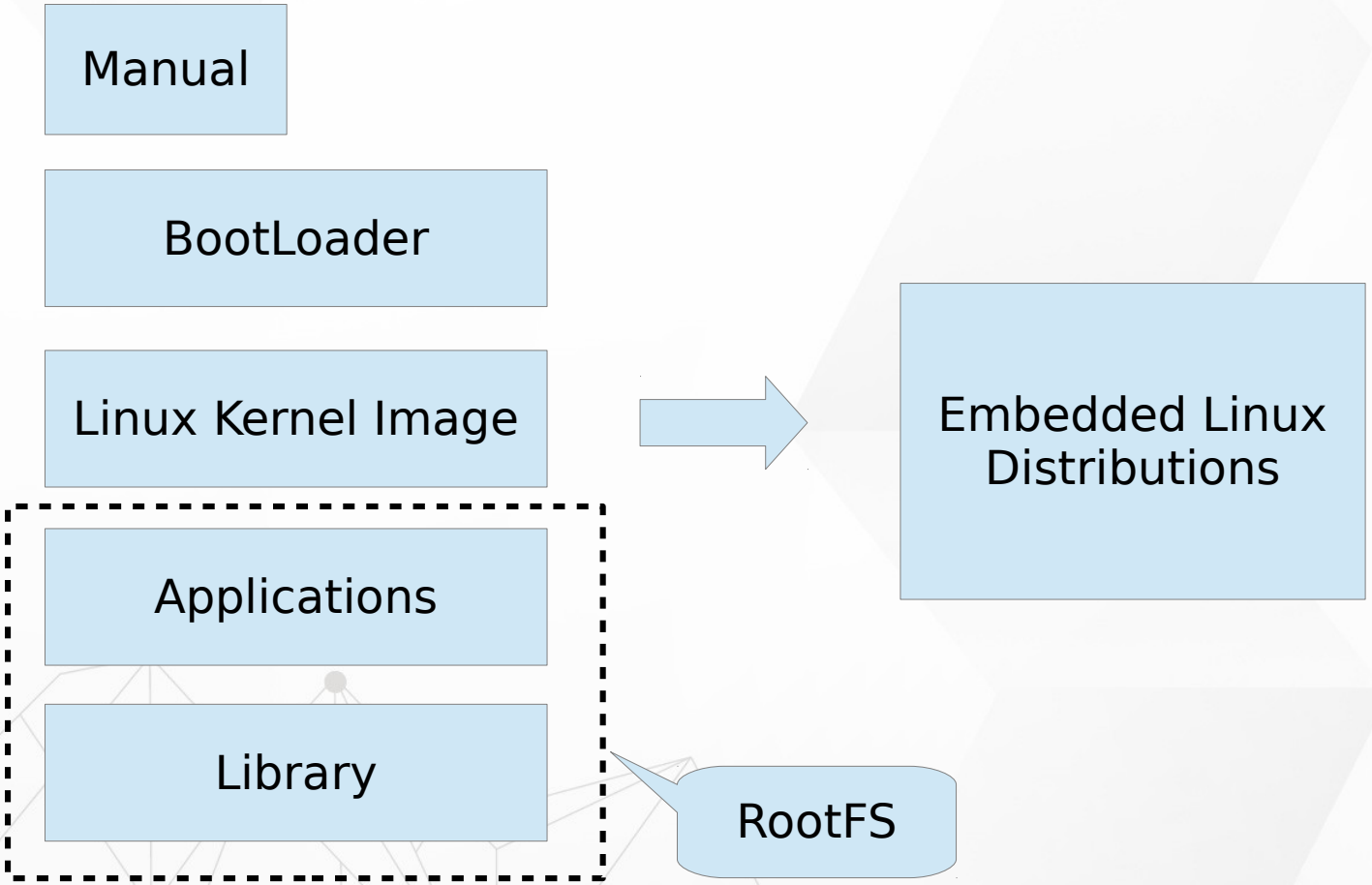
- Install it

  → make install

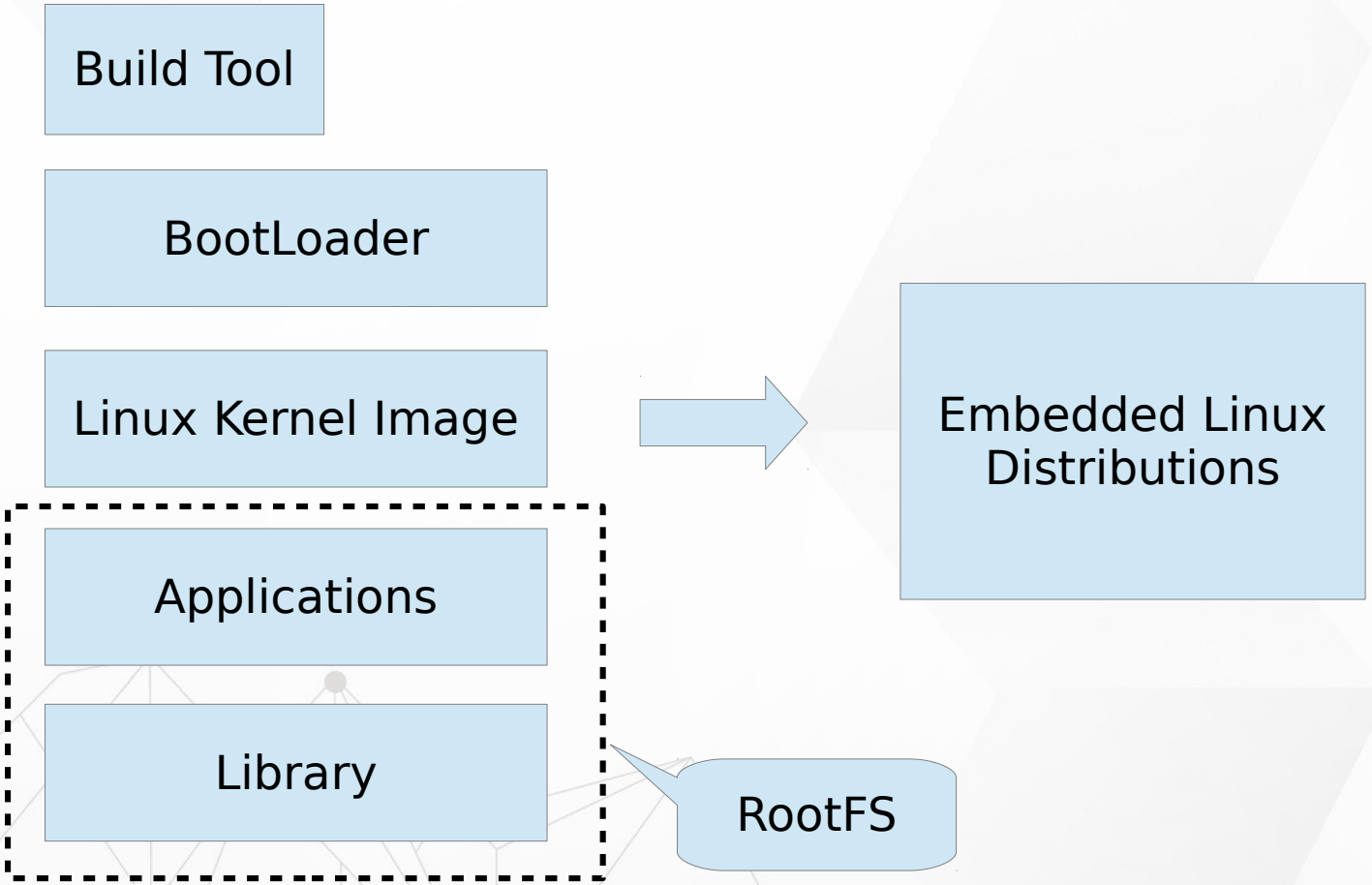
Linux Distribution

- Boot-loader
- Linux kernel
- RootFS
- Application
- Library
- Linux driver moudes

Build Distribution by Manual

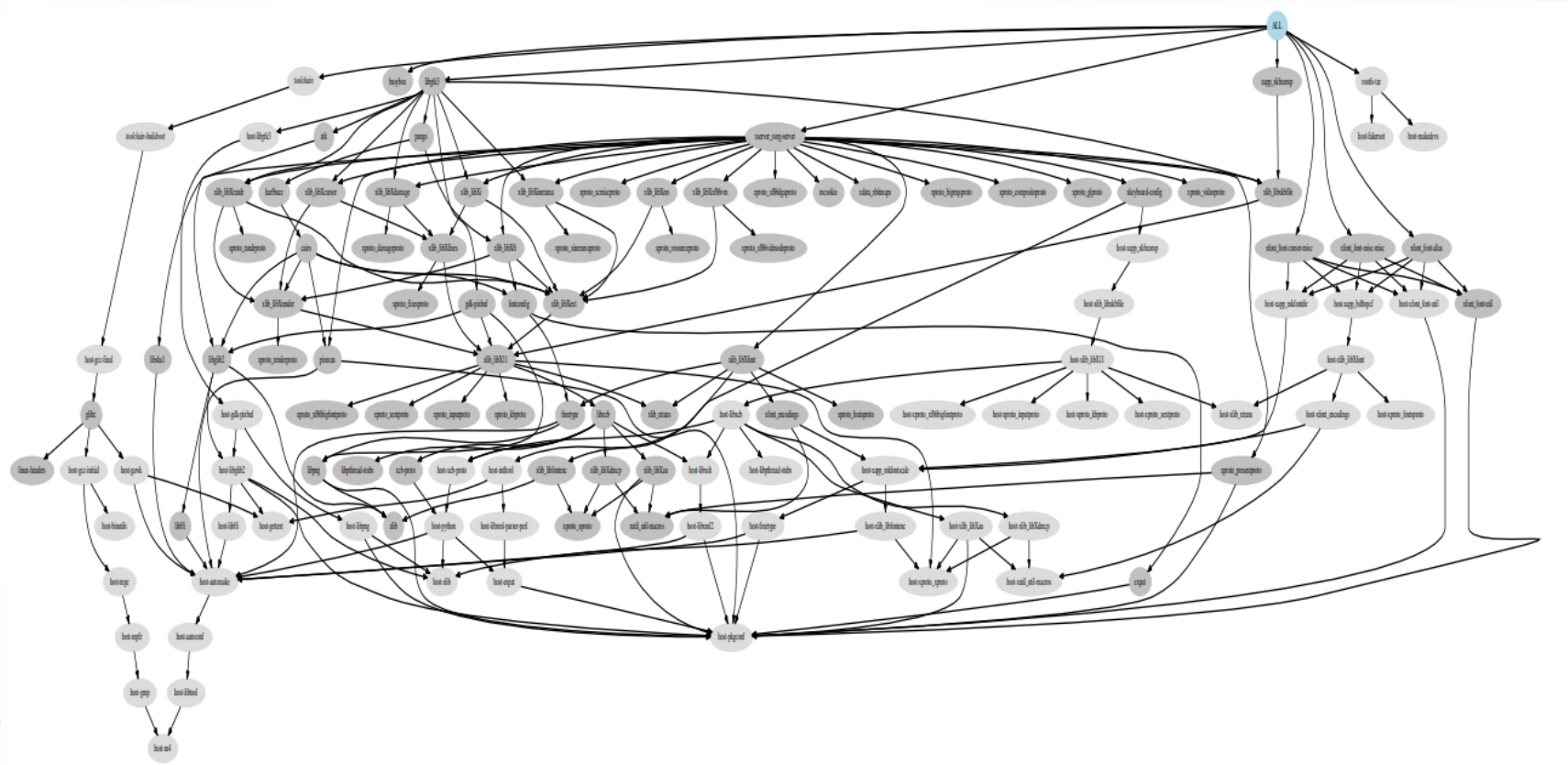


Build Distribution by Tool



Complexity of user space depend

https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf



System integration: several possibilities

https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well-defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

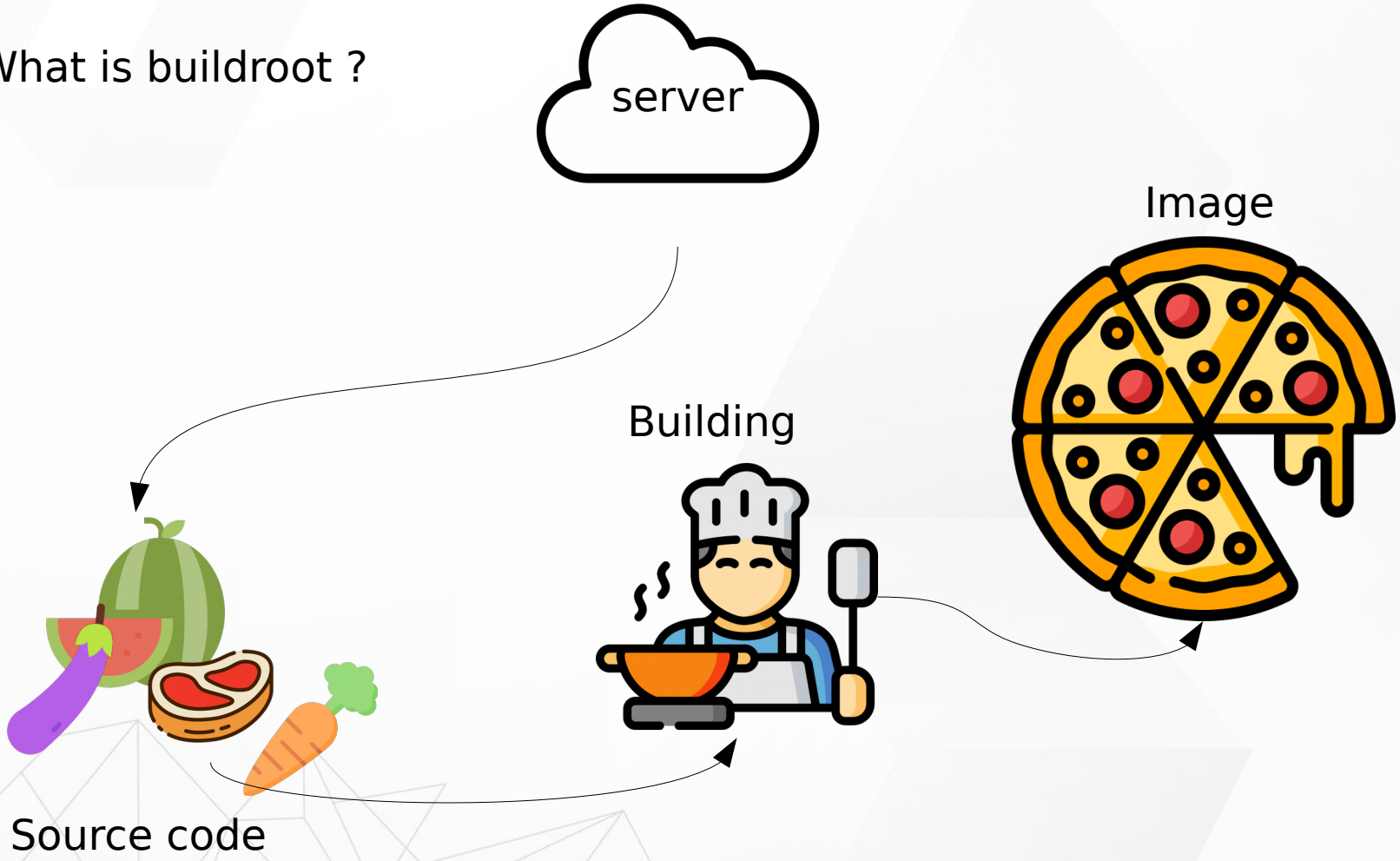
Buildroot

Buildroot

- https://buildroot.org/

What is Buildroot

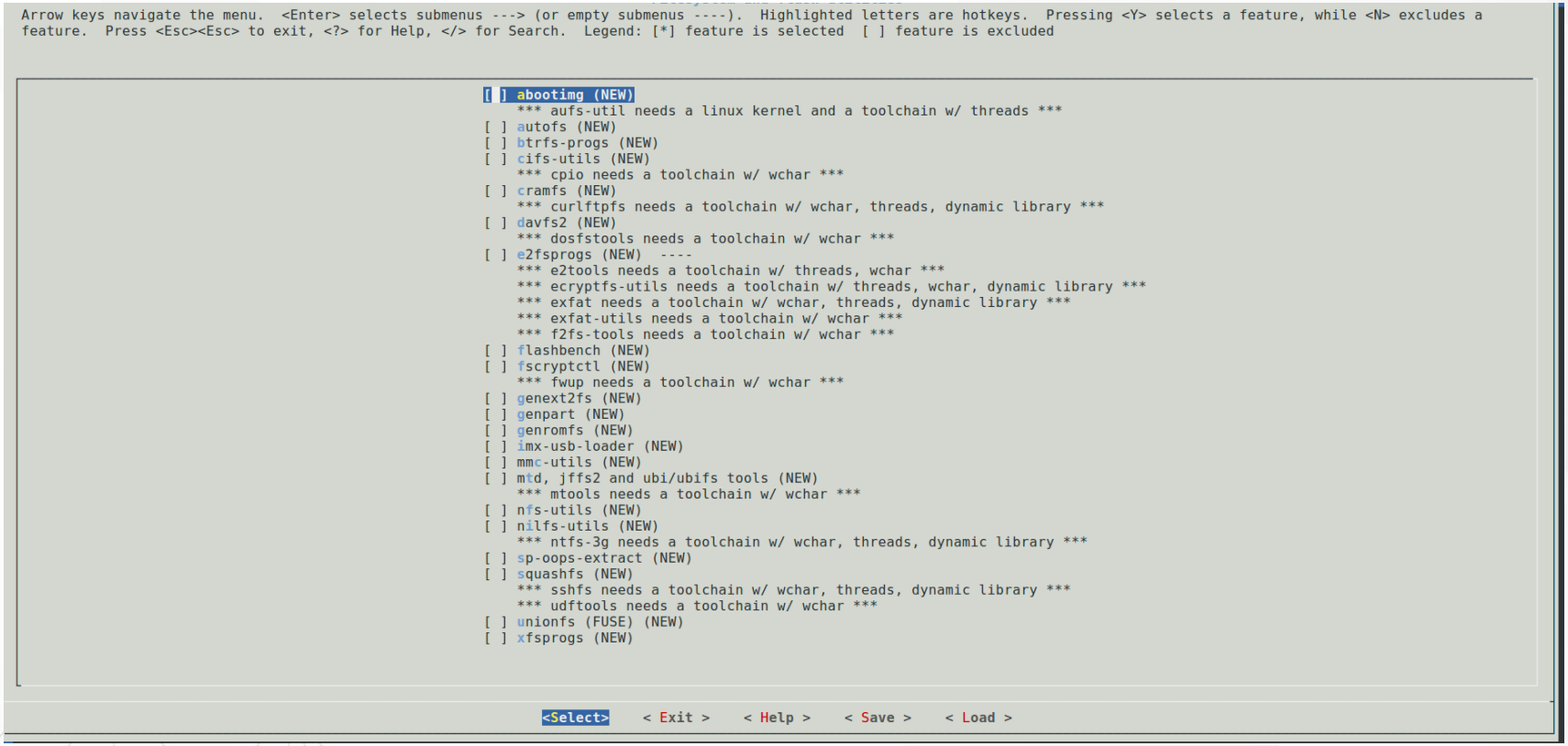
What is buildroot ?



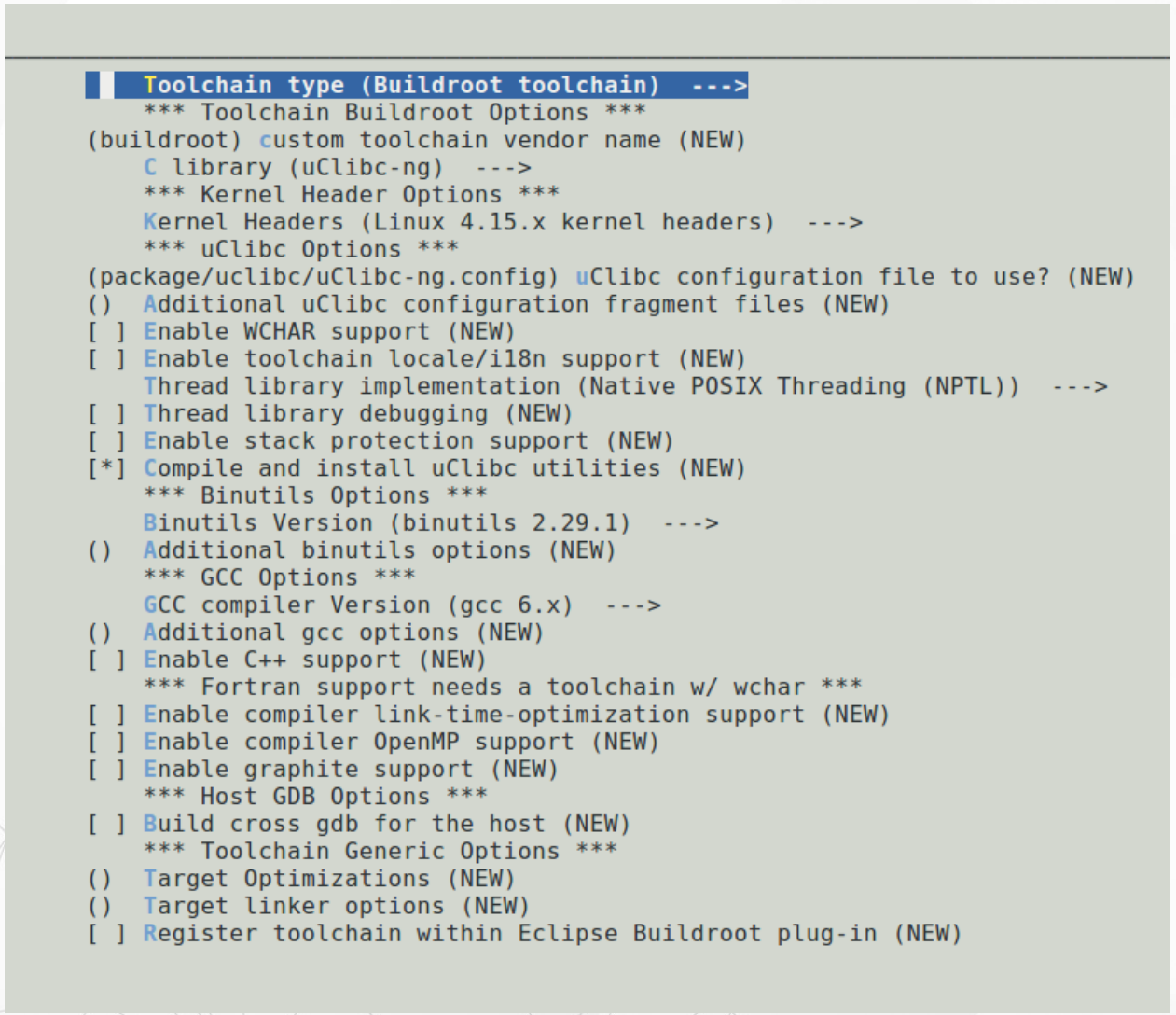
Get Buildroot

- wget https://buildroot.org/downloads/buildroot-2020.02.1.tar.gz
- tar -xvzf buildroot-2-2-.02.1.tar.bz2
- cd buildroot-2-2-.02.1.tar.bz2
- make menuconfig

Configure



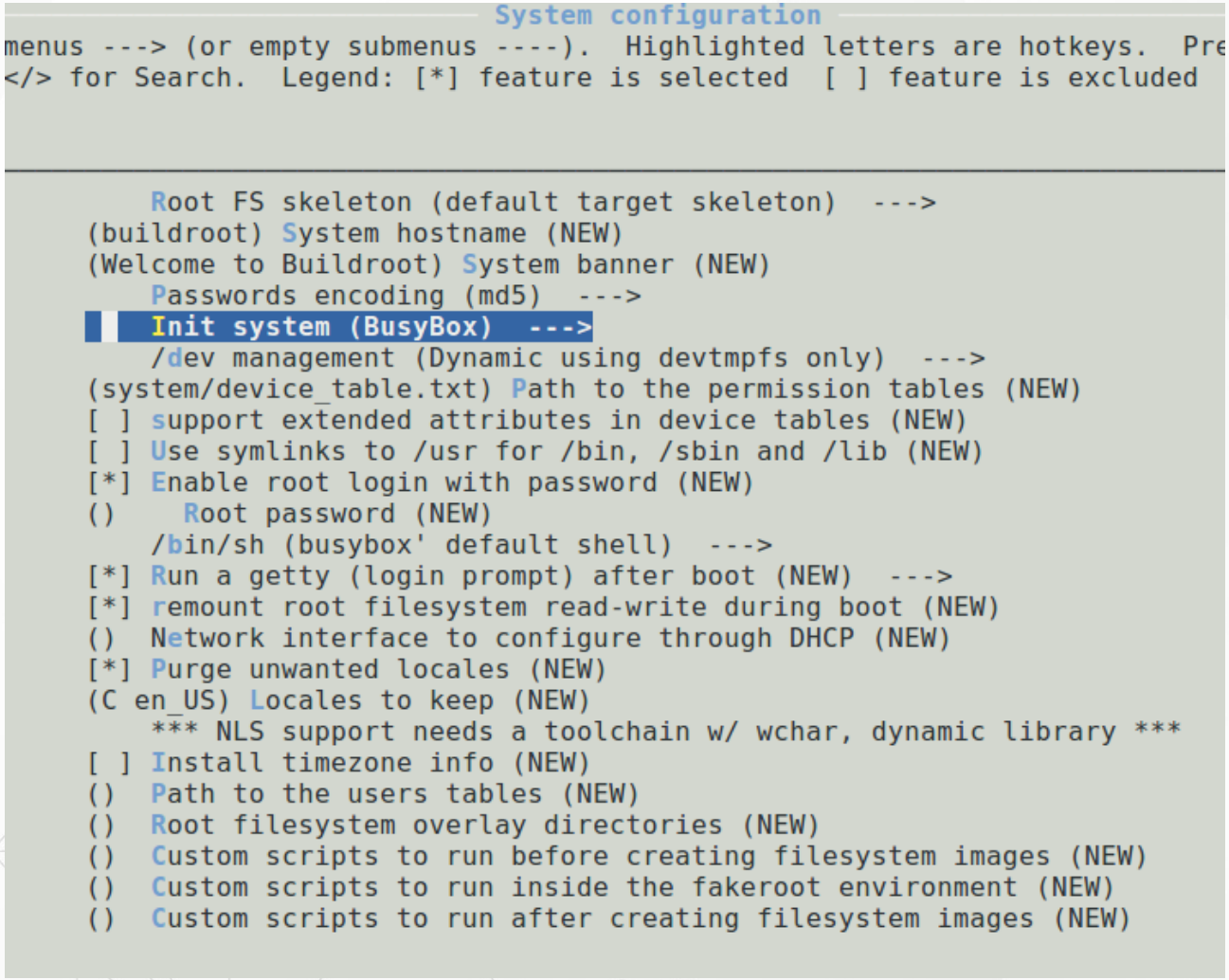
Toolchain



Toolchain

- Easy install different platform toolchain
- Easy change toolchain version
- Easy custom toolchain

System Configure



System Configure

- Init system
  - BusyBox
  - SystemV
- Password setting
- Login setting
- System banner
- RootFS skeleton

Target packages

```
-*- BusyBox
(package/busybox/busybox.config) BusyBox configuration file to use? (NEW)
()    Additional BusyBox configuration fragment files (NEW)
[ ]   Show packages that are also provided by busybox (NEW)
[ ]   Individual binaries (NEW)
[ ]   Install the watchdog daemon startup script (NEW)
[ ] rockchip BSP packages (NEW)  ----
    Audio and video applications  --->
    Compressors and decompressors  --->
    Debugging, profiling and benchmark  --->
    Development tools  --->
    Filesystem and flash utilities  --->
    Fonts, cursors, icons, sounds and themes  --->
    Games  --->
    Graphic libraries and applications (graphic/text)  --->
    Hardware handling  --->
    Interpreter languages and scripting  --->
    Libraries  --->
    Mail  --->
    Miscellaneous  --->
    Networking applications  --->
    Package managers  --->
    Real-Time  --->
    Security  --->
    Shell and utilities  --->
    System tools  --->
    Text editors and viewers  --->
    Libretro cores and retroarch  --->
```

Target packages

```
-*- BusyBox
(package/busybox/busybox.config) BusyBox configuration file to use? (NEW)
()    Additional BusyBox configuration fragment files (NEW)
[ ]   Show packages that are also provided by busybox (NEW)
[ ]   Individual binaries (NEW)
[ ]   Install the watchdog daemon startup script (NEW)
[ ] rockchip BSP packages (NEW)  ----
    Audio and video applications  --->
    Compressors and decompressors  --->
    Debugging, profiling and benchmark  --->
    Development tools  --->
    Filesystem and flash utilities  --->
    Fonts, cursors, icons, sounds and themes  --->
    Games  --->
    Graphic libraries and applications (graphic/text)  --->
    Hardware handling  --->
    Interpreter languages and scripting  --->
    Libraries  --->
    Mail  --->
    Miscellaneous  --->
    Networking applications  --->
    Package managers  --->
    Real-Time  --->
    Security  --->
    Shell and utilities  --->
    System tools  --->
    Text editors and viewers  --->
    Libretro cores and retroarch  --->
```

Configure File

- ${BUILDROOT}/configs

```
rockchip_rk3326_recovery_defconfig
rockchip_rk3326_robot32_defconfig
rockchip_rk3326_robot64_defconfig
rockchip_rk3326_robot64_no_gpu_defconfig
rockchip_rk3328_defconfig
rockchip_rk3328_recovery_defconfig
rockchip_rk3399_defconfig
rockchip_rk3399pro_defconfig
rockchip_rk3399pro-npu_defconfig
rockchip_rk3399pro-npu-multi-cam_defconfig
rockchip_rk3399pro_recovery_defconfig
rockchip_rk3399_recovery_defconfig
rockchip_rv1108_defconfig
roseapplepi_defconfig
s6lx9_microboard_defconfig
sheevaplug_defconfig
snps_aarch64_vdk_defconfig
snps_arc700_axs101_defconfig
snps_archs38_axs103_defconfig
snps_archs38_haps_defconfig
snps_archs38_vdk_defconfig
socrates_cyclone5_defconfig
solidrun_macchiatobin_mainline_defconfig
solidrun_macchiatobin_marvell_defconfig
stm32f429_disco_defconfig
```

Output Directory

- ${BUILDROOT}/output/rockchip
  - build
  - **Host → Toolchain**
  - **Image → Image file**
    - rootfs.cpio, rootfs.cpio.gz
    - rootfs.ext2 , rootfs.ext4
    - rootfs.squashfs
    - rootfs.tar
    - Linux kernel , u-boot Image
  - target

Nanopi-m4 SDK

Nanopi-m4 Build System

- Reference Buildroot
  - http://wiki.friendlyarm.com/wiki/index.php/Buildroot_for_RK3399
- It will build and output
  - Kernel Image
  - Boot-loader image
  - RootFS
    - System setting file (/etc)
    - Driver modules
    - Application
    - library

Nanopi-m4 Build System

```
./linuxsdk-friendlyelec
├── [Mar 31  8:40]  app
├── [Mar 31  8:40]  buildroot
├── [Mar 31  8:40]  build.sh -> device/rockchip/common/build.sh
├── [Mar 31  8:40]  device
├── [Mar 31  8:40]  distro
├── [Mar 31  8:40]  docs
├── [Mar 31  8:40]  envsetup.sh -> buildroot/build/envsetup.sh
├── [Mar 31  8:41]  external
├── [Mar 31  8:41]  friendlyelec
├── [Apr 20 16:15]  kernel
├── [Mar 31  8:40]  Makefile -> buildroot/build/Makefile
├── [Mar 31  8:40]  mkfirmware.sh -> device/rockchip/common/mkfirmware.sh
├── [Mar 31  8:41]  out -> friendlyelec/rk3399/sd-fuse_rk3399/out
├── [Mar 31  8:41]  prebuilts
├── [Mar 31  8:41]  rkbin
├── [Mar 31  8:40]  rkflash.sh -> device/rockchip/common/rkflash.sh
├── [Mar 31  8:41]  rootfs
├── [Mar 31  8:42]  tools
├── [Mar 31  8:42]  u-boot
└── [Mar 31  8:42]  yocto
```

Setup Compilation Environment

Install Package on host machine

sudo apt-get install repo git-core gitk git-gui u-boot-tools device-tree-compiler mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs lib32stdc++6

repo

- Install repo Utility

```
Step 1 :
   git clone https://github.com/friendlyarm/repo

Step 2 :
   cp repo/repo /usr/bin/
```

Download Source Code

- Fix a version

Step 1 :
tar xvf linuxsdk-friendlyelec-YYYYMMDD.tar

Step 2 :
cd linuxsdk-friendlyelec

Step 3 :
repo sync -l --no-clone-bundle

Download Source Code

Retrive Repo Package from Github

Step 1 :
mkdir linuxsdk-friendlyelec

Step 2 :
cd linuxsdk-friendlyelec

Step 3 :
repo init -u https://github.com/friendlyarm/buildroot_manifests -b master -m rk3399_linux_release.xml --repo-url=https://github.com/rockchip-linux/repo –no-clone-bundle

Step 4 :
repo sync -c --no-clone-bundle

Compilation

- Build Image
  - Build ALL -> ./build.sh
  - Build Kernel -> ./build.sh kernel
  - Build u-boot -> ./build.sh uboot
  - Build RootF -> ./build.sh rootfs

Create Image

- SDCard Image
  - sudo ./build.sh sd-img

Create Image

- eMMC Image
  - sudo ./build.sh emmc-img

Create Image

- EMMC Image
  - sudo ./build.sh emmc-img

Build Configure File

- Configure file
  - device/rockchip/rk3399/BoardConfig.mk
- Use setting build environment variable
  - # Target arch
    - export RK_ARCH=arm64
  - # uboot defconfig
    - export RK_UBOOT_DEFCONFIG=rk3399_defconfig
  - # Kernel defconfig
    - export RK_KERNEL_DEFCONFIG=nanopi4_linux_defconfig

Rockchip Driver

- external
- Put many depend hardware software package

```
external/
├── [Mar 31  8:40]  alsa-config
├── [Mar 31  8:40]  audioservice
├── [Mar 31  8:40]  bluez-alsa
├── [Mar 31  8:40]  camera_engine_cifisp
├── [Mar 31  8:40]  camera_engine_rkisp
├── [Mar 31  8:40]  ffmpeg
├── [Mar 31  8:40]  gst-plugins-rockchip
├── [Mar 31  8:40]  gstreamer-camera
├── [Mar 31  8:40]  gstreamer-rockchip
├── [Mar 31  8:40]  libdrm
├── [Mar 31  8:40]  libmali
├── [Mar 31  8:40]  linux-rga
├── [Mar 31  8:40]  minigui
├── [Mar 31  8:40]  mpp
├── [Mar 31  8:40]  mpv
├── [Mar 31  8:40]  powermanager
├── [Mar 31  8:40]  recovery
├── [Mar 31  8:40]  rknn_demo
├── [Mar 31  8:40]  rknn-toolkit
├── [Mar 31  8:40]  rk_pcba_test
├── [Mar 31  8:40]  rkscript
├── [Mar 31  8:40]  rkssd
├── [Mar 31  8:40]  rkupdate
├── [Mar 31  8:40]  rkwifibt
├── [Mar 31  8:40]  security
├── [Mar 31  8:41]  softapDemo
├── [Mar 31  8:41]  softapServer
├── [Mar 31  8:41]  tensorflow
└── [Mar 31  8:41]  uvc_app
```

Toolchain

- Prebuilts/

```
slash@slash-HD631-Q87CRM:linuxsdk-friendlyelec$ prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/home/slash/work/special_task/cadtc/rk3399/linuxsdk-friendlyelec/prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linu
64-linux-gnu/6.3.1/lto-wrapper
Target: aarch64-linux-gnu
Configured with: '/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/snapshots/gcc.git~linaro-
bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x
-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-
g-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --wit
le-libmudflap --enable-lto --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id
le-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --enable-fix-cortex-a53-83
43419 --with-arch=armv8-a --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --enable-gnu-indirect-function --with-build-sysroot=/home/tcwg-bu
elease/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/sysroots/aarch64-linux-gnu --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-
4/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu/aarch64-linux-gnu/libc --enable-checking=release --disable-boot
,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=aarch64-linux-gnu --prefix=/home/tcwg-buildslave/workspace/tcwg-make-relea
wg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)
slash@slash-HD631-Q87CRM:linuxsdk-friendlyelec$
```

Other Folders

- kernel, u-boot
- friendlyelec
  - Nano Pi M4 build distribution folder
- rootfs
  - Build debian and ubuntu RootFS
- app
  - Special user space application
- Buildroot
  - Linux distribution build reference

Prebuild Image and Setting File

- friendlyelec/rk3399/sd-fuse_rk3399/prebuilt
  - idbloader.img
  - trust.img
  - boot.img
  - uboot.img
  - generic
    - param4sd.txt
    - partmap.txt

param4sd.txt

FIRMWARE_VER: 6.0.1
MACHINE_MODEL: RK3399
MACHINE_ID: 007
MANUFACTURER: RK3399
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3399
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
#KERNEL_IMG: 0x00280000
#FDT_NAME: rk-kernel.dtb
#RECOVER_KEY: 1,1,0,20,0
#in section; per section 512(0x200) bytes
CMDLINE: root=/dev/mmcblk0p1 rw rootfstype=ext4
mtdparts=rk29xxnand:0x00002000@0x00002000(uboot),
0x00002000@0x00004000(trust),
0x00002000@0x00006000(misc),
0x00006000@0x00008000(resource),
0x00010000@0x0000e000(kernel),
0x00010000@0x0001e000(boot),
-@0x00030000(rootfs)

partmap.txt

```
flash=mmc,1:loader:idb:0x8000,0x280000:idbloader.img;
flash=mmc,1:env:env:0x3F8000,0x8000;
flash=mmc,1:parm:parm:0x400000,0x0400000:param4sd.txt;
flash=mmc,1:uboot:raw:0x800000,0x0400000:uboot.img;
flash=mmc,1:trust:raw:0xC00000,0x0400000:trust.img;
flash=mmc,1:misc:raw:0x1000000,0x0400000;
flash=mmc,1:resc:raw:0x1400000,0x0C00000:resource.img;
flash=mmc,1:kern:raw:0x2000000,0x2000000:kernel.img;
flash=mmc,1:boot:raw:0x4000000,0x2000000:boot.img;
flash=mmc,1:rootfs:ext4:0x6000000,0x0:rootfs.img;
```

Update Tool

sd_update

linuxsdk-friendlyelec/friendlyelec/rk3399/sd-fuse_rk3399/tools

```
Usage: sd_update [ARGS]

Options:
  -d <device  node>       default: none
  -p <partmap file>       default: ./partmap.txt
  -i <images path>        default is "partmap.txt" directory
  -r <raw image file>
  -s                      show device partitioin only
  -f                      force to no warning
  -h                      print this help text

partmap file:
flash=<device>.<dev no>:<partition>:<fstype>:<start>,<length>[:file name];
  <device>                device name
  <dev no>                device number
  <partition>             partition name
  <fstype>                filesystem type, MBR = raw, fat, ext4
  <start>                 partition start address (hex)
  <length>                partition length (hex)
  <file name>             write file to partition
```