**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**

Logo, company name

Description automatically generated

**BÁO CÁO**

**THỰC TẬP NGOÀI TRƯỜNG**

**HỌC KỲ 223 NĂM HỌC 2022-2023**

* **NGÀNH: Kỹ Thuật Máy Tính**
* **CHƯƠNG TRÌNH ĐÀO TẠO: OISP**
* **ĐƠN VỊ/ DOANH NGHIỆP NHẬN THỰC TẬP:**
* **Big Dolphin Co., Ltd.**
* **CÁN BỘ HƯỚNG DẪN**
* **Thầy Lê Trung Khanh**
* **SINH VIÊN THỰC HIỆN (SVTH)**

**Lê Tuấn Hưng MSSV: 2052508**

TP. HỒ CHÍ MINH, THÁNG 4, NĂM 2023

Contents

[Week 1: Linux and GCC 3](#_Toc132968663)

[1. Operating System 3](#_Toc132968664)

[a. Definition of OS: 3](#_Toc132968665)

[b. Structure of OS: 3](#_Toc132968666)

[2. Linux 4](#_Toc132968667)

[a. History of Linux: 4](#_Toc132968668)

[b. Linux Directory Structure: 4](#_Toc132968669)

[c. Linux File system: 5](#_Toc132968670)

[d. Properties of common linux filesystem: 5](#_Toc132968671)

[e. Linux permission for files and directories. 6](#_Toc132968672)

[f. Linux basic command. 6](#_Toc132968673)

[3. Open source license 7](#_Toc132968674)

[Week 2: Makefile and bootloader 8](#_Toc132968675)

[1. Makefile 8](#_Toc132968676)

[2. C/C++ wrapper 9](#_Toc132968677)

[3. Bootloader of OS: 11](#_Toc132968678)

[Week 3: Advanced Makefile and pthread programming 13](#_Toc132968679)

[1. String Substitution: 13](#_Toc132968680)

[2. Pthread programming 14](#_Toc132968681)

[3. Memory sharing between threads 17](#_Toc132968682)

[Week 4-5-6: Build Linux: 18](#_Toc132968683)

[1. Requirements 18](#_Toc132968684)

[2. Configuration and build 19](#_Toc132968685)

[3. Install the OS 20](#_Toc132968686)

[4. Ch2root.sh 21](#_Toc132968687)

[Week 7-8: Build OS for Rockchip RK3588 23](#_Toc132968688)

[1. Export Configuration 23](#_Toc132968689)

[2. Yocto SDK 25](#_Toc132968690)

[Reference 26](#_Toc132968691)

# Week 1: Linux and GCC

## Operating System

### Definition of OS:

The program that, after being initially loaded at computer by boot program, will manages all the other application programs in the computer.

Act intermediary between a user and computer hardware.

Using to allocate resource and control program.

The main different of OS from firmware is that OS will control the CPU to do tasks when CPU free.

### Structure of OS:

There are various way to structure: Simple, Monolithic, Layered, Micro-Kernel.

Simple Structure:

* Include: Application program, System program, MS – DOS device driver, and ROM BIOS device driver.

Monilithic (UNIX):

* Include: System program and **Kernel** (above hardware, below system-call interface, which provides file system, CPU scheduling, memory management, etc.)

Layer:

* The OS is devide into a number of layers.
* Level 0: hardware, level N: user interface, Only higher level can access to the lower level. => security.

Microkernel:

* Bring non-essential components to the user interface.
* Each Micro-Kernels are inpendently isolated from the others.

## Linux

### History of Linux:

Linux is a free and open-source operating system that was created by Linus Torvalds in 1991, which base on the UNIX operating system.

List of popular Linux distribution: Ubuntu, Fedora, Debian, Kali linux, CentOS, etc.

### Linux Directory Structure:



Fig 1. Linux file directory

Linux use the ‘/’ to split the directory, the root folder start from ‘/’, and use the Filesystem Hierarchy Standard (FHS) is a reference describing the conventions used for the layout of a UNIX system.

### Linux File system:

* The method and data structure that the operating system uses to control how data is stored and retrieved.

| **Folder** | **Function** |
| --- | --- |
| /bin | Basic programs |
| /boot | Contains the Linux kernel to boot and system maps and second-stage boot files. |
| /dev | Contains device files (CDRom, HDD, FDD….). |
| /etc | Contains system configuration files. |
| /home | Directory for users other than root. |
| /lib | Contains shared libraries for commands located in /bin and /sbin.  And this directory also contains kernel modules. |
| /mnt hoặc /media | Mount point is the default for external file systems. |
| /opt | Directory containing additional software installations. |
| /sbin | System programs |
| /srv | Data used by servers on the system. |
| /tmp | Directory containing temporary files. |
| /usr | Directory containing permanent or important files to serve all users. |
| /var | Variable data handled by the daemon. This includes log files, queues, caches, caches, etc. |
| /root | Administrator personal files (root) |
| /proc | Used for the Linux kernel. They are used by the kernel to output data to user space. |

### Properties of common linux filesystem:

ext3: have an journaling (error-detection), can support from 2TiB to 32TiB.

ext4: can help defracmentation, and large size volume, provide fast transfering.

btrfs: better file system, Btrfs is a copy-on-write (COW) filesystem that focuses mainly on ease in repair and administration, have data integrity, remove duplicate data and handle more size. \*performance

### Linux permission for files and directories.

3 type of permission: READ (r), WRITE (w), EXECUTE (x)

Users are Owner, Group, Public.

* - - Regular file.
* b - Block special file.
* c - Character special file.
* d - Directory.
* l - Symbolic link.
* n - Network file.
* p - FIFO.
* s - Socket.

### Linux basic command.

**pwd**: show current directory.

**cd**: navigate folder

**ls**: see the information of the folder.

**cat**: create file, concatenate 2 file.

**cp**: copy file to another folder

**rmdir**: delect empty directory

**rm**: remove

**touch**: create a file

**grep**: find the “word” inside the file.

**sudo**: super user do

**df:** disk space used

**du**: disk usage

**head, tail**: see first, last line of the file.

**chmod**: change the permission of the directory

**chown**: change owner of directory

**kill**: kill the PID

**wget**: download from internet from URL.

**echo**: add data to the file.

## Open source license

Open-source licenses are categorized as copyleft or permissive.

4 popular OSL: GPL, Apache, MIT, BSD

* MIT License:

A permissive free and open-source software license. It allows users to use, copy, modify, merge, publish, distribute, sublicense, and sell copies of the software, either in source code or binary form, without restriction. The only real requirement of the MIT License is that users must include a copy of the license in any distribution of the software.

* Apache License:

A permissive free and open-source software license, one of the key features of the Apache License is its patent grant, which gives users a license to any patents that may be associated with the software.

* BSD License:

A permissive free and open-source software license, the key point that can be used in proprietary software, meaning that users are free to incorporate the software into proprietary products without releasing the source code.

* GPL License:

A copyleft license, which means that any derivative work or modification of GPL-licensed software must also be released under the GPL license and the requirement that source code need to be available.

# Week 2: Makefile and bootloader

## Makefile

Makefileis a file that it defines a set of tasks to be executed .

There are 2 parts: target and recipe.

* Target: might be a binary file that depends on prerequisites (source files). On the other hand, a prerequisite can also be a target that depends on other dependencies. \*target will not run if there exists the file with the same name\*
* Recipe: a command to be run.

“.PHONY” make the target always run.

“.DEFAULT\_GOAL :=” will define the default target, which mean that when we use “make” only, that target will be executed.

We can use variable ‘$(name)’ to simplify the makefile file.

Text

Description automatically generated

Fig 2. Example of a makefile file

## C/C++ wrapper

**C compiled process:**

* Preprocessed source file:
  + gcc -E [-o hello.i] hello.c
* Assembly code:
  + gcc -S [-o hello.S] hello.c
* Binary file
  + gcc -c [-o hello] hello.c
* Executeable file
  + gcc [-o hello] hello.c

**\*Extra:** How to compile C/C++ in the source code with g++.

First, we need to understand the mangle.

In C++, there is a name mangling, which mean that the function name are decoded => C++ support overloading function.

But in C, the funtion name are kept the same. So to combine C and C++, we will use the extern “C” to notice that these function do not need to be mangled.

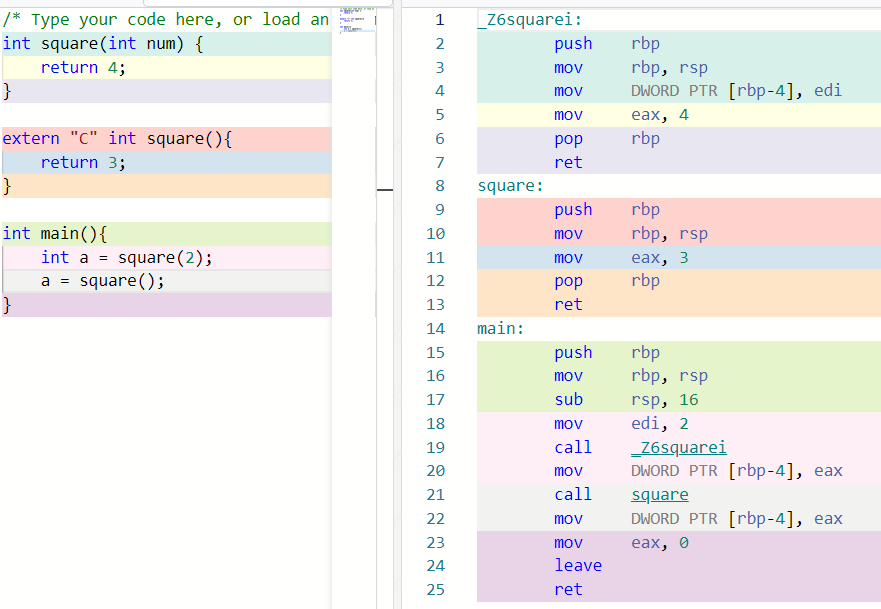


Fig 3. Function mangled in C++

Then, to compile the C/C++ we need to use extern “C” both when calling:

* C from C++: tell g++ to expect unmangled symbols produced by gcc
* C++ from C: tell g++ to generate unmangled symbols for gcc to use

The example code should be:

#ifdef \_\_cplusplus

extern "C" {

#endif

int square();

#ifdef \_\_cplusplus

}

#endif

## Bootloader of OS:

Bootloader isa program that loads an operating system when a computer is turned on.

In linux, there are 6 distinct stages in typical booting process.

* 1: POST and BIOS (Basic Input/Output System).

POST do the integrity check (hardware check)

BIOS stored on ROM will search the MBR (independent with OS).

* 2: MBR (Master Boot Record) locate and load the bootloader into RAM.
* 3: GRUB, LILO, LOADIN: Load the kernel into memory.
* 4 - 5: Kernel initialization:
* 6: Run level programs

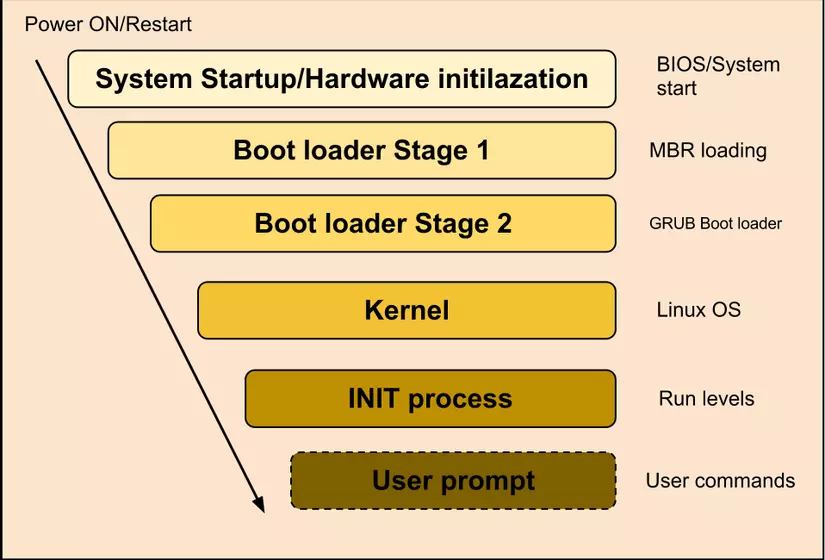


Fig 4. Bootloader progress

Embedded Linux: Usually use u-boot, the process is a bit same with above.

* 1: BootRom: the program stored on ROM, to initialize the system at hardware level.
* 2: SPL: same with MBR, locate and load bootloader into RAM.
* 3: uboot: load kernel into RAM

4: Linux kernel: The remain part of initialize the OS.

**Compare between GRUB and LILO:**



Fig 5. GRUB and LILO comparison

U-boot can be first and second stage of booting.

# Week 3: Advanced Makefile and pthread programming

## String Substitution:

$(subst from, to, text): replace a part of text.

$(patsubst pattern, replacement, text):

Ex:

$(patsubst %.c, %.o, abc.c ok.c) => abc.o ok.o

Which mean that, for the 3rd argument, anything have suffix .c will create an file with suffix.o.

Or we can use $(objects:.c=.o)

Then, when we want to create an object for each .c or .cpp file. We can write like:

SOURCES := $(wildcard \*.c)

OBJECTS := $(patsubst %.c, %.o, $(SOURCES))

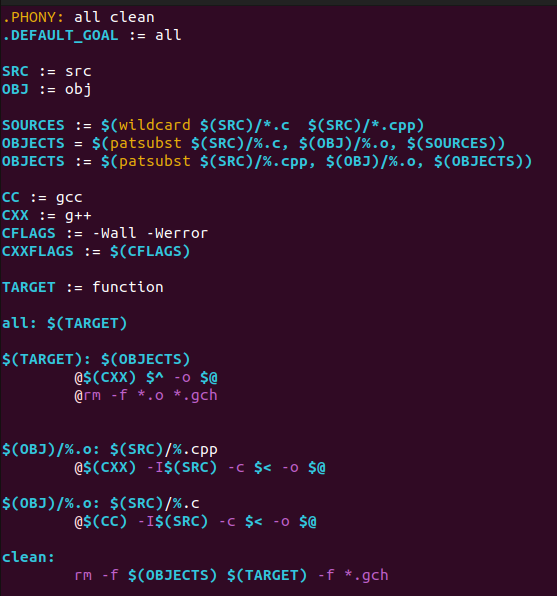


Fig 6. Makefile file with wildcard

There are something to explain:

First %.o : %cpp, and %.o : %c. which means, for each prerequisite with suffix .c and .cpp, has a corresponding target with.o. $< and $@ is the symbol that it will automatic match the gcc and g++ command.

## Pthread programming

Diagram

Description automatically generated

Fig 7. Single-Threaded vs Mutilthreaded

We can separate the process into multithread, that the resources are shared between threads with many benefits.

* Responsiveness: the execution may continue if a part of process are blocked (user interface)
* Resource sharing: threads share resources of process, then don’t need share memory
* Economy: we need less resources than create a new process.

To create a thread:

int pthread\_create(pthread\_t \* thread,   
 const pthread\_attr\_t \* attr,  
 void \* (\*start\_routine)(void \*),   
 void \*arg);

* thread - returns the thread id. (unsigned long int defined in bits/pthreadtypes.h)
* attr - Set to NULL if default thread attributes are used.
* void \* (\*start\_routine) - pointer to the function to be threaded. Function has a single argument: pointer to void.
* \*arg - pointer to argument of function. To pass multiple arguments, send a pointer to a structure.

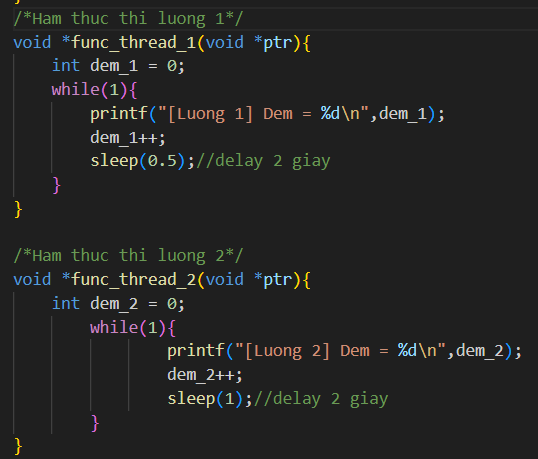


Fig 8. Example of creating multithread

Text

Description automatically generated

Fig 9. The result of 2 threads

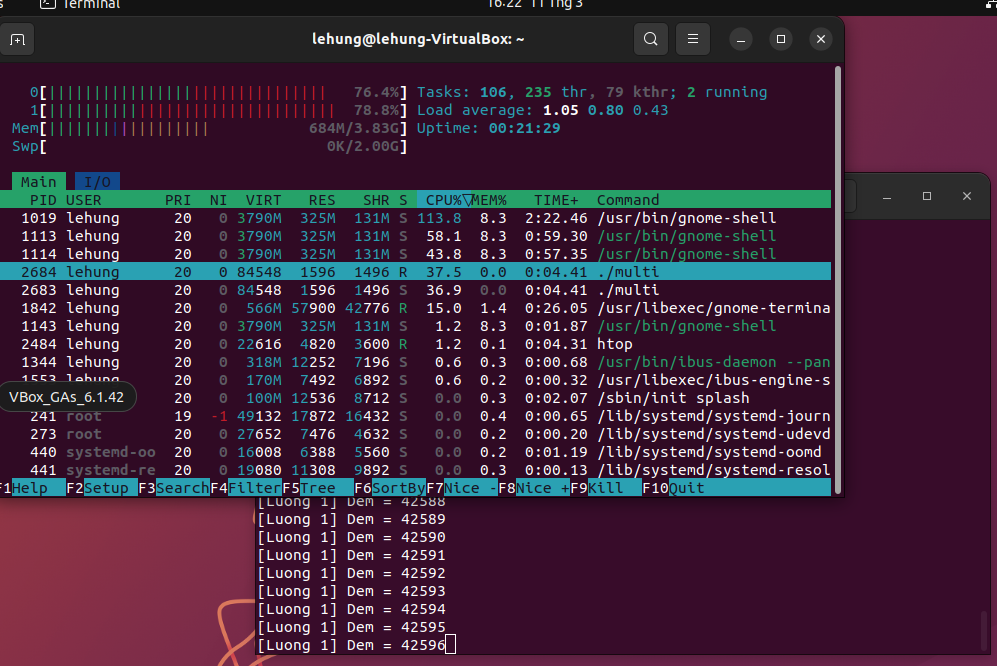


Fig 10. CPU ultilization when the delay is set at high value

## Memory sharing between threads

Data race: occurs when

* two or more tasks access in a shared memory, at least 1 access for writing and no exclusive lock.

Race condition: occurs when

* the error of execution that make unexpected output. (because the interruption).

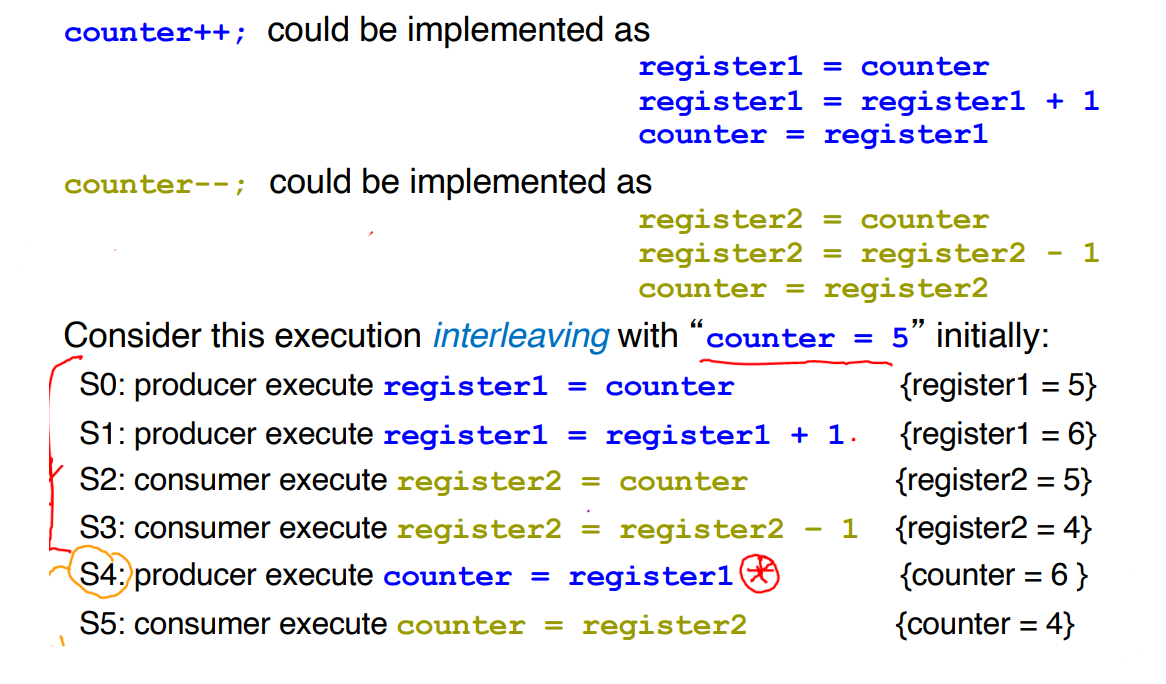
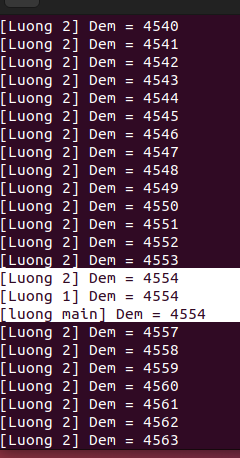


Fig 11. Brief explaination of data race

\***Critical Section:** a process may contains the changing in common variables, updating table, writing file, etc. When one process in the critical section, no other may in the critical section.

When I set 3 thread access into 1 global variable (shared memory), it may have the unexpected output.



We can use the mutex lock (and binary semaphore) to solve that problem.

# Week 4-5-6: Build Linux:

## Requirements

Step 1: Prepare packet for build.

sudo apt-get update && sudo apt-get dist-upgrade

#Instaill required package.

sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping libsdl1.2-dev xterm libyaml- dev libssl-dev python3-git zstd liblz4-tool

**#**Clone poly and Yocto SDK

git clone git://git.yoctoproject.org/poky

git checkout -t origin/mickledore -b my-mickledore

git pull

cd poky

**git clone git://git.yoctoproject.org/poky.git poky-mickledore**

**git clone git://git.openembedded.org/meta-openembedded**

**git clone git://git.yoctoproject.org/meta-raspberrypi**

**git clone https://github.com/meta-qt5/meta-qt5**

#Start to pass the parameter to build

source oe-init-build-env

## Configuration and build

Because the machine is raspberry pi 4, then we need to customize the configuration file.

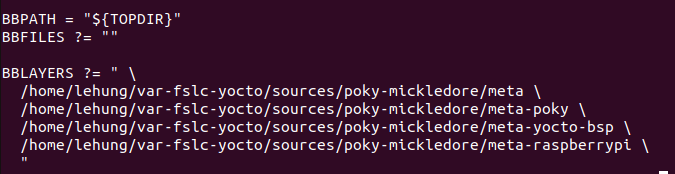


Fig 12. In conf/bblayer.conf

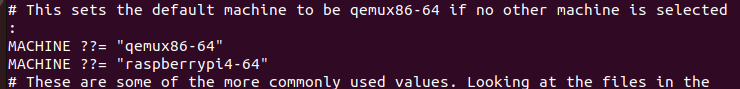


Fig 13. In conf/local.conf

We also need to edit the size of rootfs to make sure it have enough size to install essential packages.

IMAGE\_ROOTFS\_SIZE = "2900000"

IMAGE\_ROOTFS\_EXTRA\_SPACE = ""

IMAGE\_OVERHEAD\_FACTOR = ""

Text

Description automatically generated

Fig 14. Modifying the size of rootfs

\*Note: by default, the overhead\_factor is 1.3, then the size of rootfs is

2,900,000\*1.3 = 3,770,000 KB + x (the size of kernel, with about of 100-200MB).

Finally, we start to build the uboot and kernel

**bitbake core-image-minimal**

The process may take 5-12 hours, based on the internet connection. When we need to stop, please press CTRL+C **One time**, and wait until the task remain in queue finish, if the task is forced to stop, it may cause errors when continue.

## Install the OS

Then after finish, in ./tmp/deploy/images/raspberrypi4-64.wic.bz2

We will unzip the core-image-minimal-raspberrypi4-64.wic.bz2

bzip2 -d -f core-image-minimal-raspberrypi4-64.wic.bz2

Then to write into SD card, we change the extension .wic => .img and write or

sudo dd bs=”1024” if=core-image-minimal-raspberrypi4-64.wic of=/dev/sd’X’ status=progress conv=fsync “

****

Finally, we go to config.txt in boot partition and comment all command.Text

Description automatically generated with medium confidence

## Ch2root.sh

**Ch2root.sh:** The techincal to sync the process with current linux of PC to install some essential packet to the new OS. In here, I use ubuntu 22.04.

###################################### UBUNTU 22.04 ##################################

#--1. Install QEMU

sudo apt install lib32ncurses6 qemu-user-static git pv libncurses5-dev -y

#--2. Download the Ubuntu root filesystem

mkdir ubuntu22

cd ubuntu22

wget -c <https://cdimage.ubuntu.com/ubuntu-base/releases/22.04/release/ubuntu-base-22.04.1-base-armhf.tar.gz>

sudo -s

mkdir rootfs

tar -xvf ubuntu-base-22.04.1-base-armhf.tar.gz -C rootfs

#--3. Copy qemu-user-static and firmwares

cp /usr/bin/qemu-arm-static rootfs/usr/bin/

#--4. Modify etc/apt/sources.list to un-comment all the repositories except the ones starting with deb-src

sed -i 's%^# deb %deb %' rootfs/etc/apt/sources.list

#--5. chroot

wget <https://github.com/bigdolphin/chroot/raw/main/ch2root.sh>

chmod a+x ch2root.sh

./ch2root.sh -m rootfs/

#--6. Update the repositories

chmod 1777 /tmp

# resolve the nameserver for new OS

echo "nameserver 8.8.8.8" | tee /etc/resolv.conf > /dev/null

apt update

#--7. Install minimal packages

# Extra packet for ubuntu 22.04 --------#

apt-get install linux-generic-hwe-22.04

#---------------------------------------#

apt install -y language-pack-en-base sudo bash-completion dialog vim nano lsof udev lsb-base htop psmisc locate rsyslog

apt install -y bison flex bc build-essential cmake automake autoconf cmake-curses-gui pkg-config yasm tmux git

apt install -y net-tools dkms r8168-dkms ethtool iputils-ping alsa-utils unzip net-tools netbase ifupdown network-manager ntp usbutils ssh whois

apt install -y apt-utils subversion graphviz espeak i2c-tools evtest sox onboard device-tree-compiler alsamixergui

#--8. Exit chroot and unmount proc, sys, dev, dev/pts

exit

./ch2root.sh -u rootfs/

#--9. Copy rootfs to sd

# Please make sure that unmounted rootfs before rsync

rsync -aP rootfs/ img\_mount/

sync

#--10. Edit password ###

cd img\_mount/etc

sudo nano passwd

#--11 unmount ----

umount img\_mount/

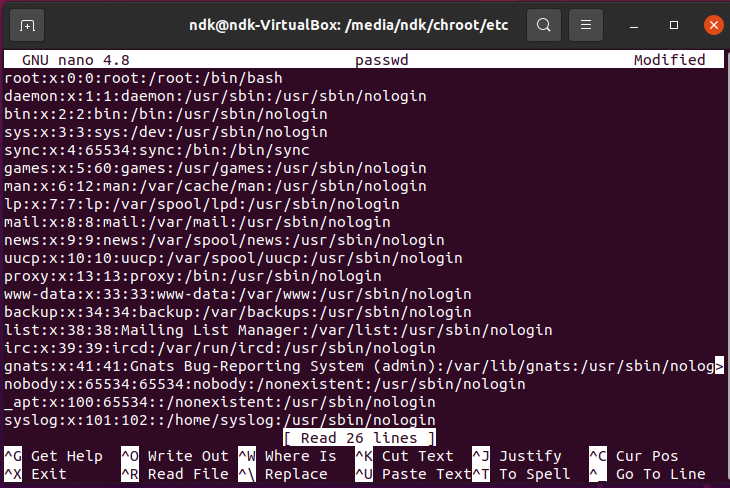


Fig 15. In etc/passwd

Delete only ‘x’ in root => No password required in login

# Week 7-8: Build OS for Rockchip RK3588

## Export Configuration

Export OS config, build ubuntu for Rockchop RK3588.

To export the config of the kernel

cat /proc/config.gz | gunzip > running.config

Here is 2 reference for building an OS for rockchip RK3588, we use Firefly\_Linux\_SDK

mkdir ~/proj/rk3588\_sdk/

cd ~/proj/rk3588\_sdk/

Method 1:

repo init --no-clone-bundle --repo-url https://gitlab.com/firefly-linux/git-repo.git -u https://gitlab.com/firefly-linux/manifests.git -b master -m rk3588\_linux\_release.xml

(in case of this doesn’t work, the can change to method 2)

Download Firefly\_Linux\_SDK sub-volume compressed package: [Linux SDK](https://en.t-firefly.com/doc/download/142.html)

md5sum rk3588\_linux\_release\_20230114\_v1.0.6c\_0\*

After confirming that it is correct, you can unzip:

mkdir -p ~/proj/rk3588\_sdk

cd ~/proj/rk3588\_sdk

cat path/to/rk3588\_linux\_release\_20230114\_v1.0.6c\_0\* | tar -xv

# export data

.repo/repo/repo sync -l

.repo/repo/repo start firefly --all

In the device/rockchip/rk3588/ directory, there are configuration files (xxxx.mk) for different board types, which are used to manage the compilation configuration of each project of the SDK. The relevant configuration introduction:

*# Target arch*

export RK\_ARCH=arm64

*# Uboot defconfig*

export RK\_UBOOT\_DEFCONFIG=xxxx\_defconfig

*# Kernel defconfig*

export RK\_KERNEL\_DEFCONFIG=xxxx\_defconfig

*# Kernel defconfig fragment*

export RK\_KERNEL\_DEFCONFIG\_FRAGMENT=xxxx.config

*# Kernel dts*

export RK\_KERNEL\_DTS=roc-rk3588s-pc.dts

*# parameter for GPT table*

export RK\_PARAMETER=parameter-xxxx.txt

*# rootfs image path*

export RK\_ROOTFS\_IMG=ubuntu\_rootfs/rootfs.img

The parameter.txt file contains the partition information of the firmware. Take parameter-ubuntu-fit.txt as an example:

path: device/rockchip/rk3588/parameter-ubuntu-fit.txt

FIRMWARE\_VER: 1.0

MACHINE\_MODEL: RK3588

MACHINE\_ID: 007

MANUFACTURER: RK3588

MAGIC: 0x5041524B

ATAG: 0x00200800

MACHINE: 0xffffffff

CHECK\_MASK: 0x80

PWR\_HLD: 0,0,A,0,1

TYPE: GPT

CMDLINE: mtdparts=rk29xxnand:0x00002000@0x00004000(uboot),0x00002000@0x00006000(misc),0x00020000@0x00008000(boot:bootable),0x00040000@0x00028000(recovery),0x00010000@0x00068000(backup),0x00c00000@0x00078000(rootfs),0x00040000@0x00c78000(oem),-@0x00cb8000(userdata:grow)

uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9

To set up the environment:

sudo apt-get install repo git ssh make gcc libssl-dev liblz4-tool expect g++ patchelf chrpath gawk texinfo chrpath diffstat binfmt-support qemu-user-static live-build bison flex fakeroot cmake gcc-multilib g++-multilib unzip device-tree-compiler ncurses-dev

Then type ./build.sh

## Yocto SDK

Inside the Firefly\_Linux\_SDK, Yocto is also included in

*# Go to yocto SDK*

cd build/conf

*# Select the configuration file and find the configuration file corresponding to the board*

ln -fs rk3588/(device\_target).conf local.conf

source oe-init-build-env

bitbake core-image-minimal

# Reference

[1. Overview — Firefly Wiki (t-firefly.com)](https://wiki.t-firefly.com/en/Firefly-Linux-Guide/manual_yocto.html#get-sdk)

[GNU make](https://www.gnu.org/software/make/manual/make.html)

[Cùng nhau học Linux kernel | # \*\*𝗨-𝗯𝗼𝗼𝘁/Kernel Boot Process\*\* | Facebook](https://www.facebook.com/groups/259967441230713/posts/649202618973858/)

[u-boot – Trung Kien's Blog (wordpress.com)](https://kienltb.wordpress.com/category/u-boot/)

[Concept of Booting: What is Booting Process? Type of Booting & Examples (toppr.com)](https://www.toppr.com/guides/computer-science/computer-fundamentals/classification-of-computers/concept-of-booting/)

[7.3 Image Options | Building a Custom Linux Distribution | InformIT](https://www.informit.com/articles/article.aspx?p=2514911&seqNum=3)

lsmod

[Ubuntu 20.04 e1000e - Reset adapter unexpectedly FIX (github.com)](https://gist.github.com/amit177/63c86ee05110091f6fdda4c87a4209d0)

[How to install Ethernet driver on Ubuntu Server 20.04.4 LTS - Ask Ubuntu](https://askubuntu.com/questions/1402709/how-to-install-ethernet-driver-on-ubuntu-server-20-04-4-lts)

[1. Compile Linux firmware — Firefly Wiki (t-firefly.com)](https://wiki.t-firefly.com/en/ROC-RK3588S-PC/linux_compile.html#linux-sdk-configuration-introduction)