Xilinx Development Environment

Ultrascale+ Linux Application Development

ALINX

Version Record

Version	Date	Release By	Description
Rev1.0	2021-04-05	Rachel Zhou	First Release

We promise that this tutorial is not a permanent, consistent document. We will continue to revise and optimize the tutorial based on the feedback of the forum and the actual development experience.

Content

Part 1: Building a Minimalist Working Environment

Part 1.1: Introduction to the system environment

This tutorial uses QtCreator as a development tool under ubuntu to cross-compile applications. Therefore, under ubuntu, we need to install cross-compilation tools and various library files that need to be used. These can be completed using the sdk.sh installation package. For the origin of the installation package, please refer to the tutorial in the petalinux chapter, which is not introduced here.

Part 1.2: System and basic tool installation

1) Under the win10 system, install the virtual machine software VMware.

   Refer to the tool installation "vmware installation.pdf"

2) Use ubuntu-18.04.2-desktop-amd64.iso to create a linux system.

   Refer to the tool to install "ubuntu virtual machine creation.pdf"

3) Install the necessary libraries under ubuntu

```
sudo apt-get install build-essential
```

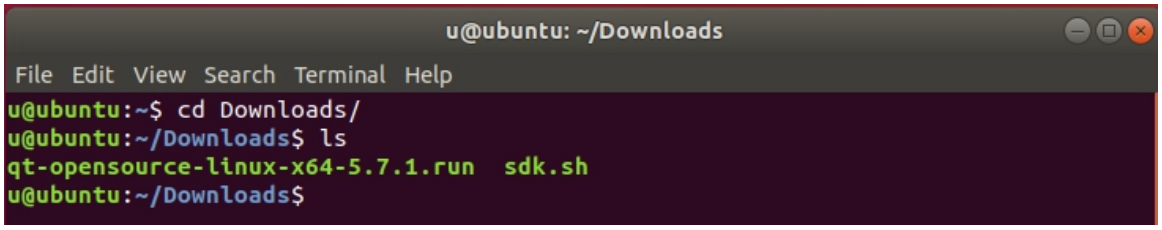
   Refer to the tool installation "ubuntu library installation.pdf"

4) Install QtCreator under ubuntu

   Refer to the tool installation "Qt installation under linux.pdf"

Part 1.3: Compilation environment installation

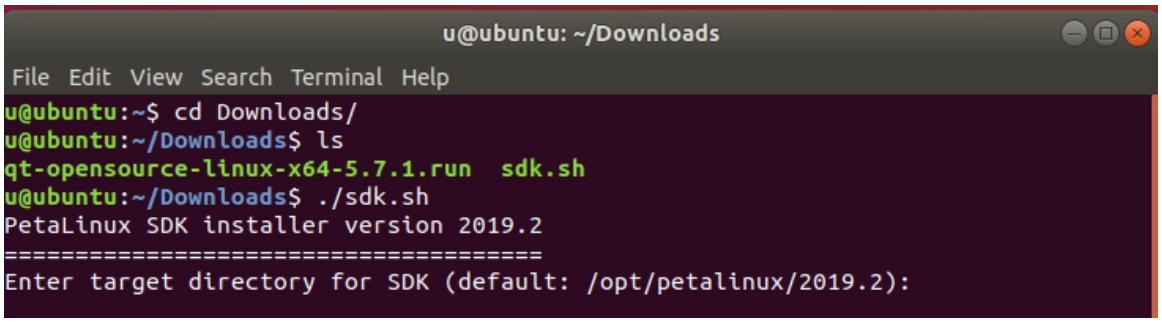
1) Copy sdk.sh to the ubuntu system and open the terminal



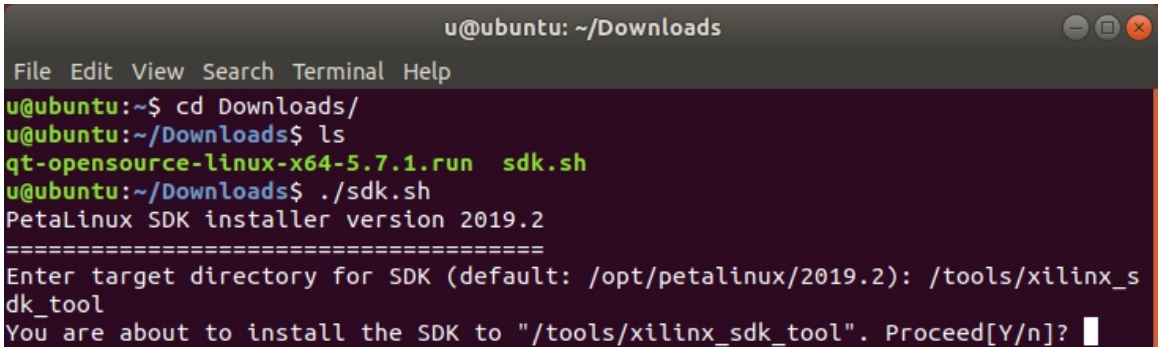
2) Create a tools directory in the root directory and modify permissions

```
sudo mkdir /tools
sudo chmod 777 –R /tools
```

3) Run sdk.sh



4) Enter the installation directory, where the installation directory is set to "/tools/xilinx_sdk_tool", and press Enter

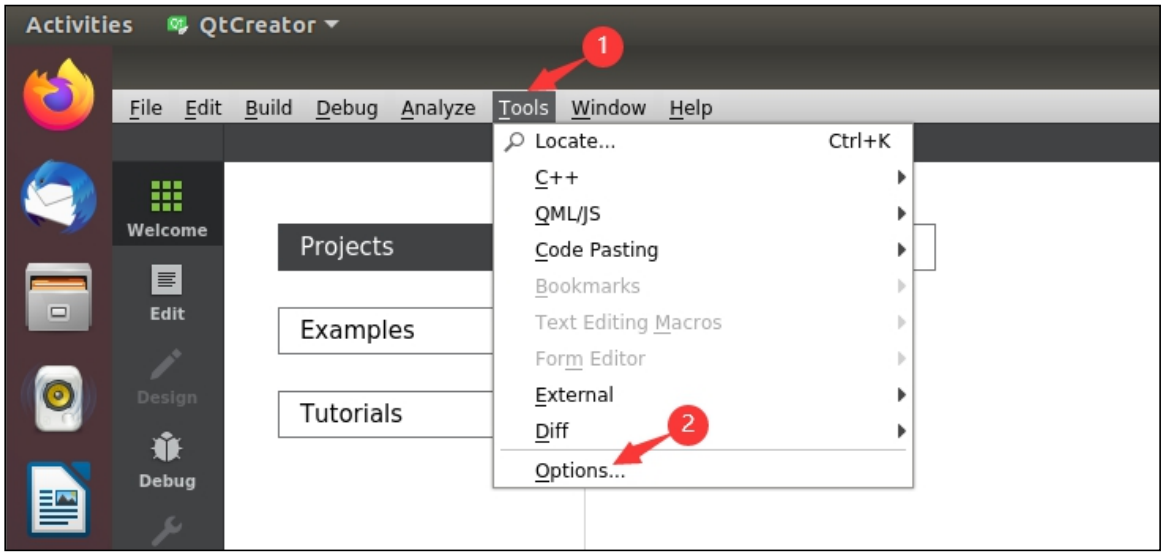


5) Enter "Y" to start the installation until the end

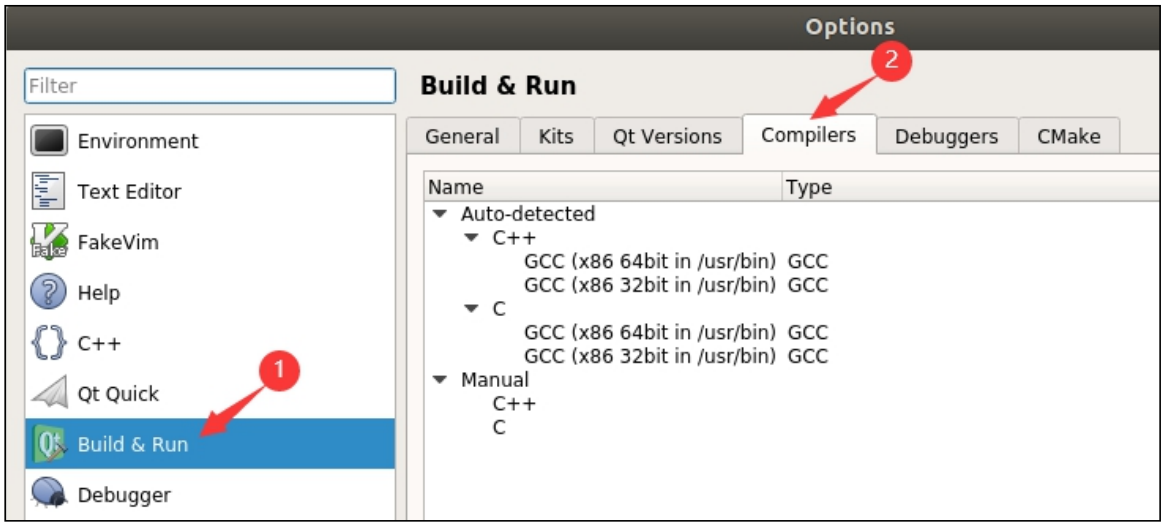
```
u@ubuntu: ~/Downloads
File  Edit  View  Search  Terminal  Help
u@ubuntu:~$ cd Downloads/
u@ubuntu:~/Downloads$ ls
qt-opensource-linux-x64-5.7.1.run  sdk.sh
u@ubuntu:~/Downloads$ ./sdk.sh
PetaLinux SDK installer version 2019.2
======================================
Enter target directory for SDK (default: /opt/petalinux/2019.2): /tools/xilinx_s
dk_tool
You are about to install the SDK to "/tools/xilinx_sdk_tool". Proceed[Y/n]? Y
Extracting SDK.......
```

Part 1.4: QtCreator configuration

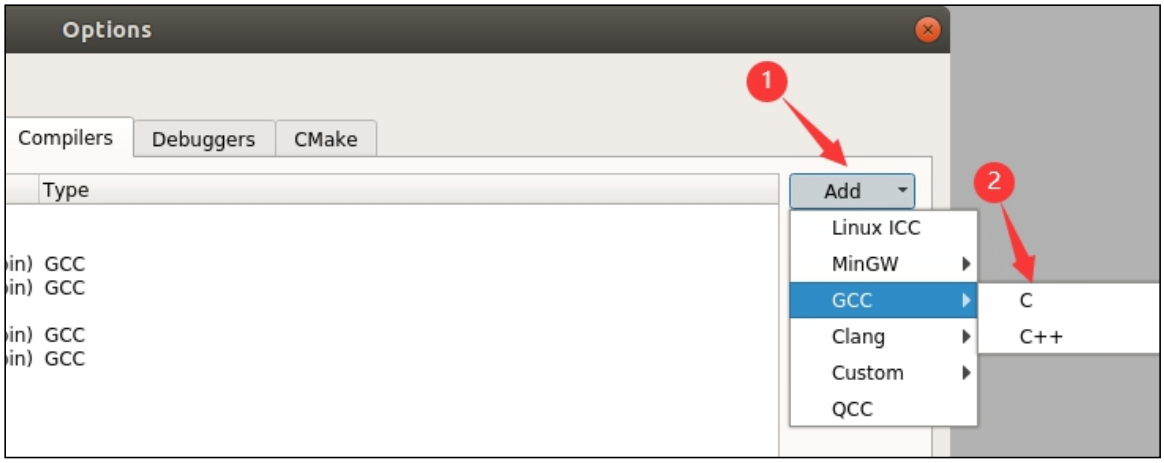
1) Open the QtCreator tool settings



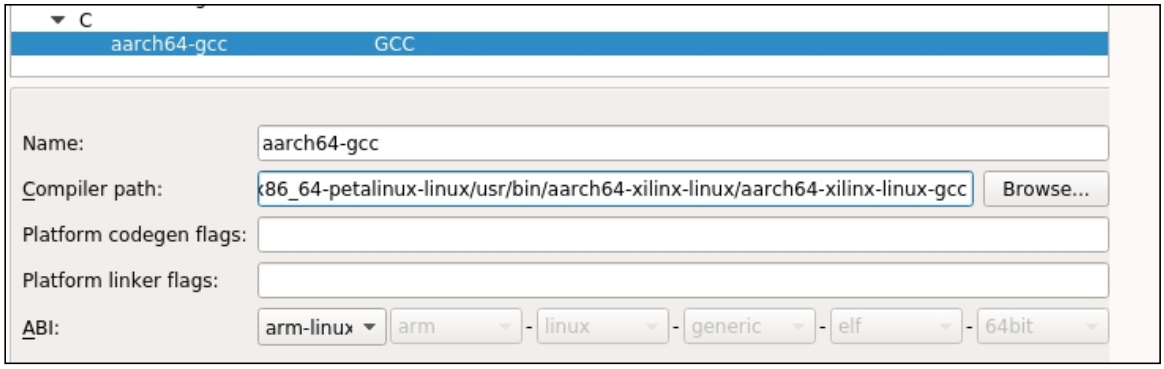
2) Open the compilation tool settings



3) Add C compilation tools



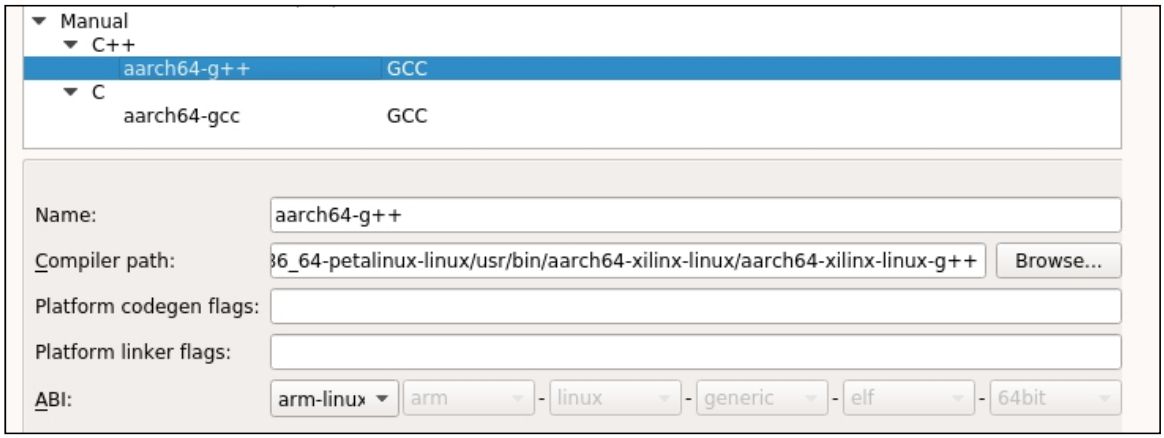
4) Set name and path



Where the path is

“ /tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-gcc”

5) Same as the above steps, add C++ compilation tools

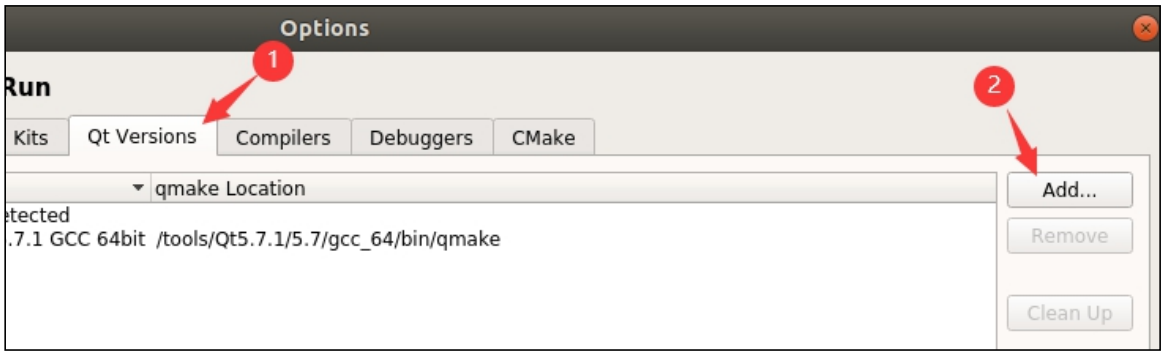


Where the path is

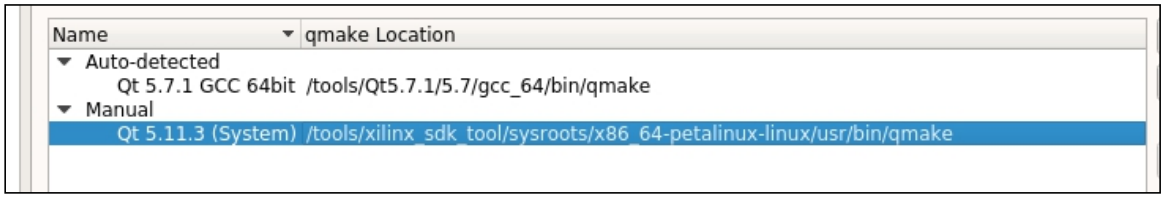
“ /tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-g++”

6) Click the button "Apply" to complete this setting

7) Switch to "Qt Versions" and click the add button



8) Select qmake

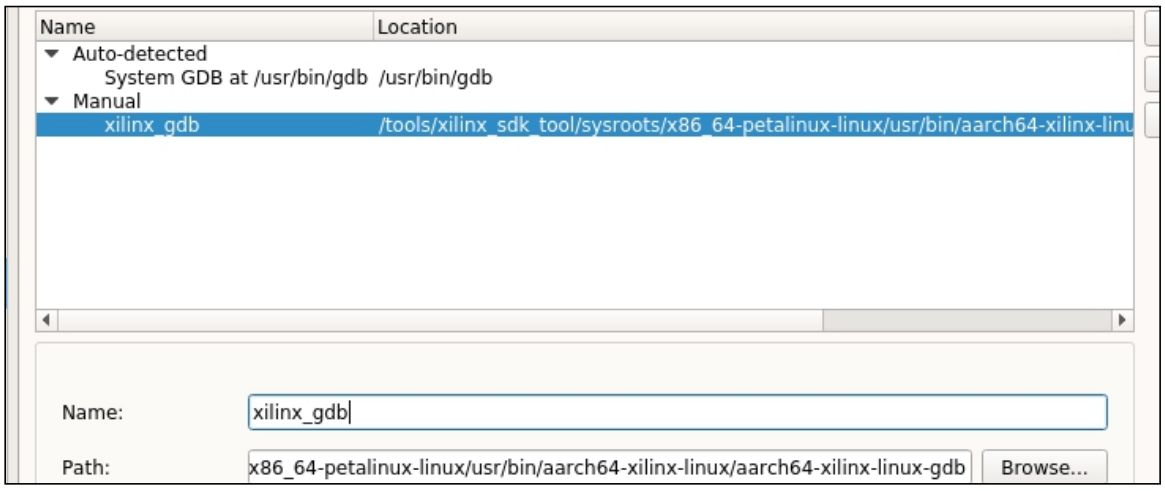


Where the path is

" /tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/qmake"

9) Click the button "Apply" to complete this setting

10) Switch to the "Debuggers" item and click the "Add" button

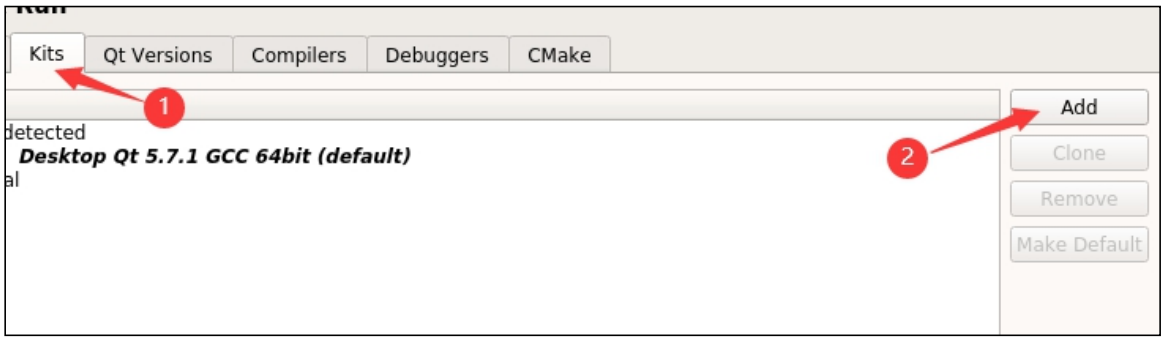


Add items to modify the content as shown in the figure, the path is

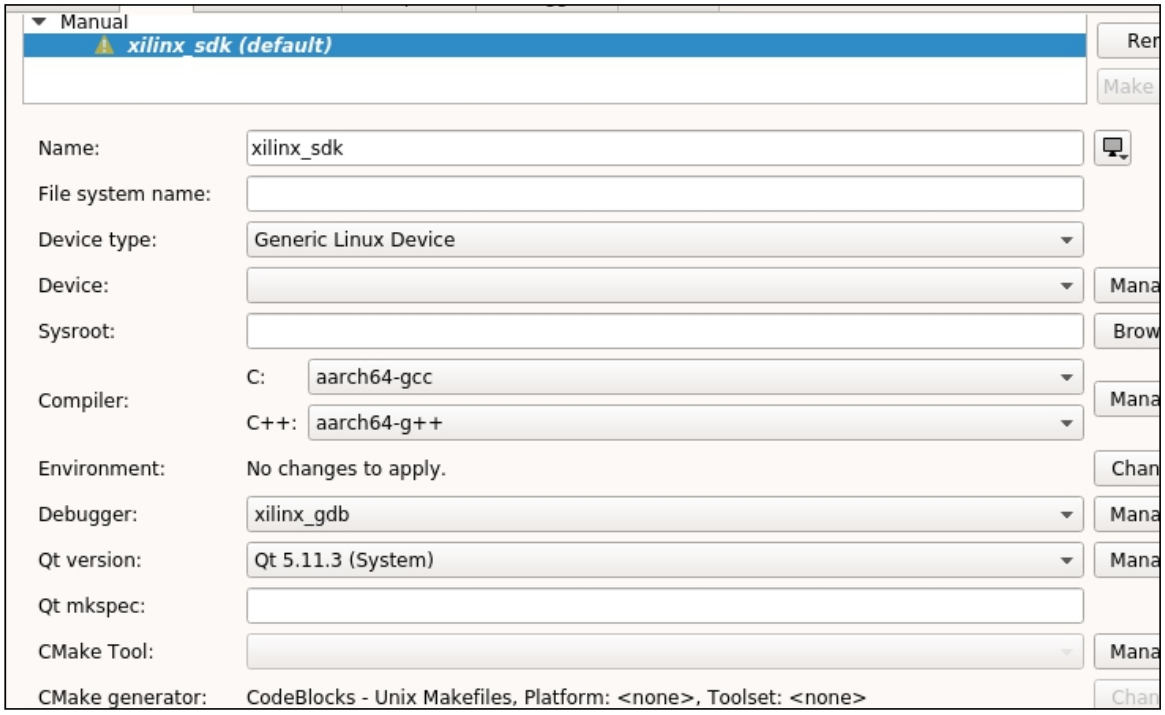
"/tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-gdb"

11) Click the button "Apply" to complete this setting

12)Switch to the "Kits" item and click the "Add" button



13)Modify the contents of the newly added items as shown in the figure



14)Click the button "OK" to complete the setting

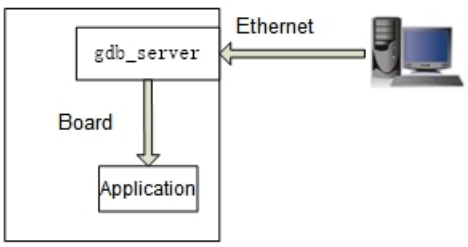
Part 2: Hello World with Remote Debugging

Part 2.1: GDB Introduction

GDB, the GNU Project debugger, allows you to see what is happening "internally" when a program is executed, or what a program is doing when it crashes. The main functions are as follows:

1) Start the program, specify any content that may affect its behavior, such as environment variables, operating parameters, etc.
2) Stop the program under specified conditions
3) Check what happened when the program stopped, such as register status, variable value, etc.
4) Manually modify various values in the running of the program, so that the program function can be verified more conveniently

The GNU debugging tools under Linux are gdb and gdbserver. Among them, gdb and gdbserver can complete remote debugging of applications under Linux on the target board. gdbserver is a small application program that runs on the target board and can monitor the operation of the debugged process, and can communicate with gdb on the host computer through the network. Developers can input commands through gdb of the host computer to control the operation of the process on the target board and view the contents of memory and registers



Part 2.2: QT Creator Introduction

Qt Creator is a cross-platform integrated development environment (IDE) that allows developers to create applications across desktop, mobile and embedded platforms.

The Qt library we usually say is the library we call when we develop desktop and other applications, and it will be linked into our running program. And Qt Creator is a tool that helps us develop applications. Therefore, the two can be regarded as completely different things.

Part 2.3: Experiment Goal

1) Get to know the Qt Creator project
2) QtCreator configures GDB connection board
3) Move the executable target program to the board
4) Pass parameters to the running program

Part 2.4: QT Creator Project

The project file of Qt Creator is xxx.pro. "QT=" in the file specifies the QT library components that need to be used. If it is empty, it means that the QT library is not called.

Part 2.5: Get the IP Address of the FPGA Board

1) Development board, equipped with sd card of linux system and powered on
2) View ip address

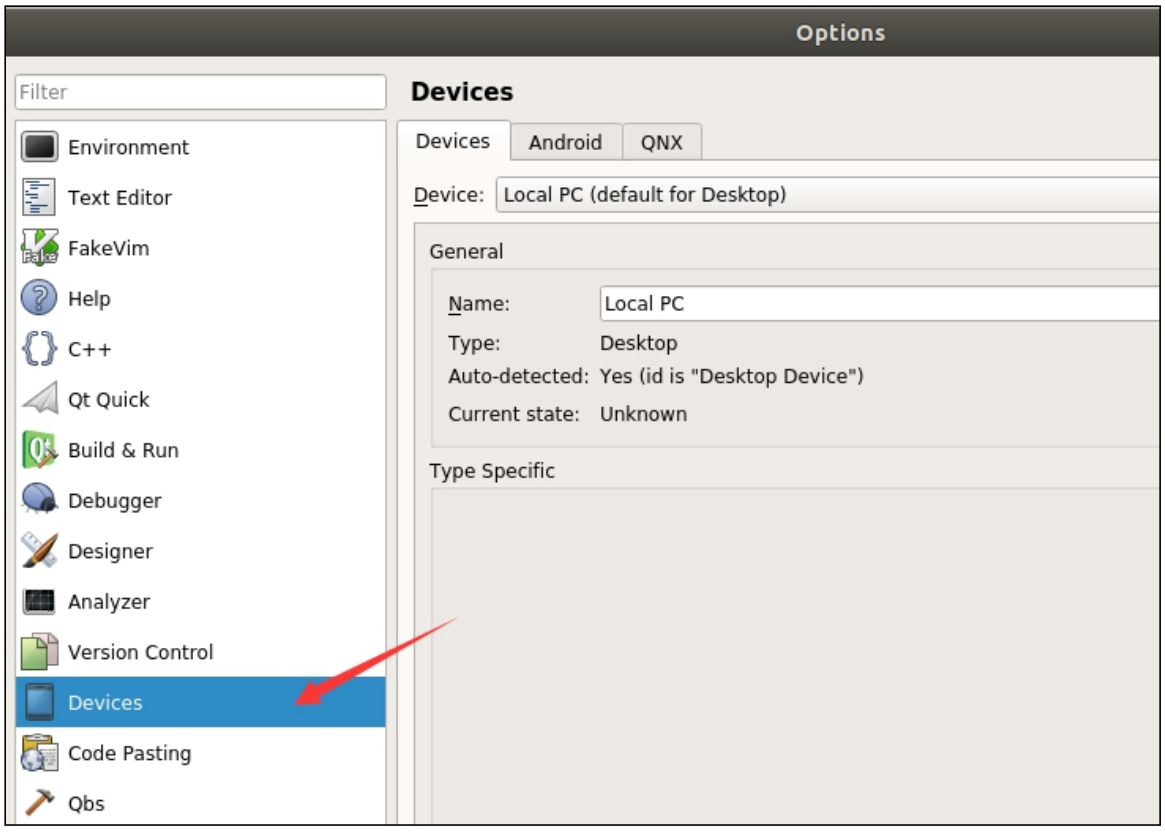
The ip address of the board can be checked through the "ipconfig" command. DHCP is enabled by default in the system. If the local DHCP service is not supported, the query result is as follows:

```
root@petalinux:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:01
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:29
```

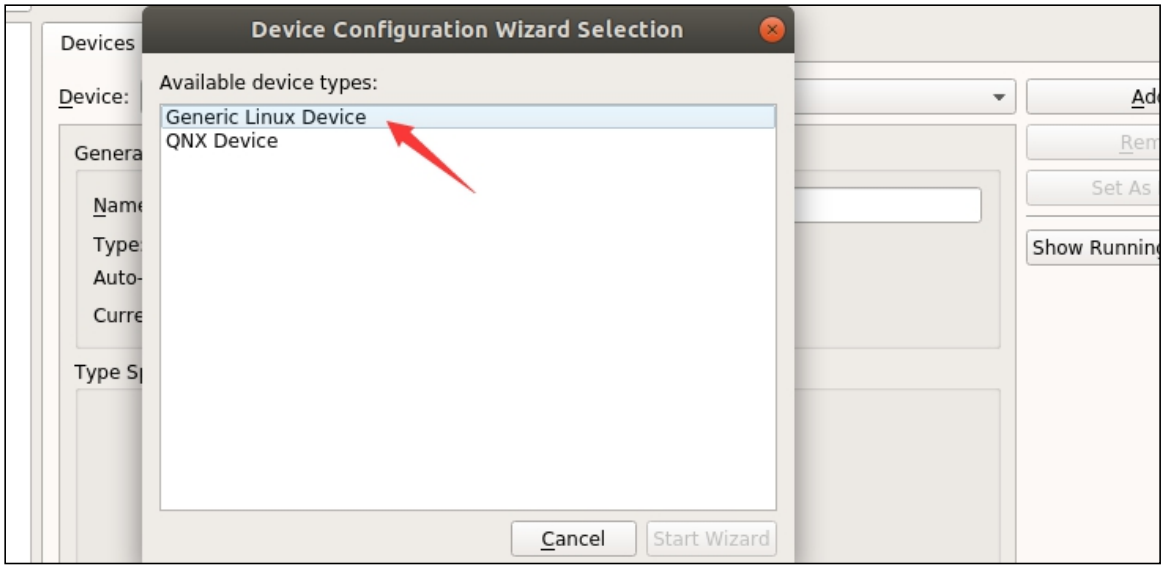
At this time, you need to manually assign an ip address, you can use the command "ipconfig eth0 192.168.1.45 netmask 255.255.255.0". Here we set the ip address of the board as: 192.168.1.45

Part 2.6: QtCreator Configures GDB Connection board

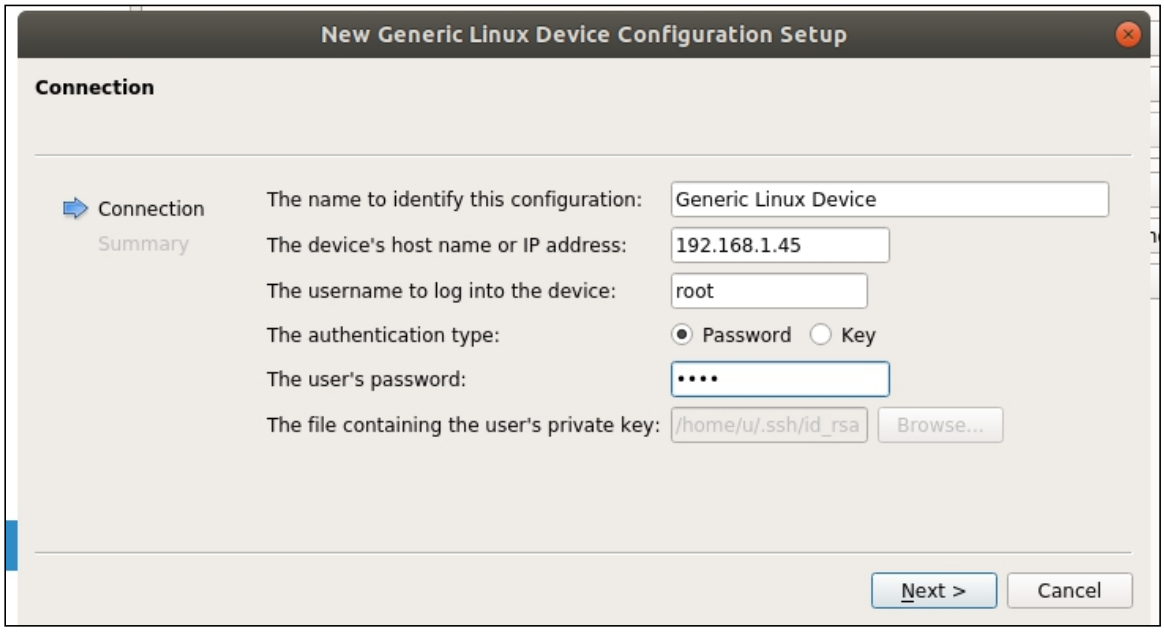
1) Open the tool settings of QtCreator



2) Click the "Add..." button and select "Generic Linux Device"

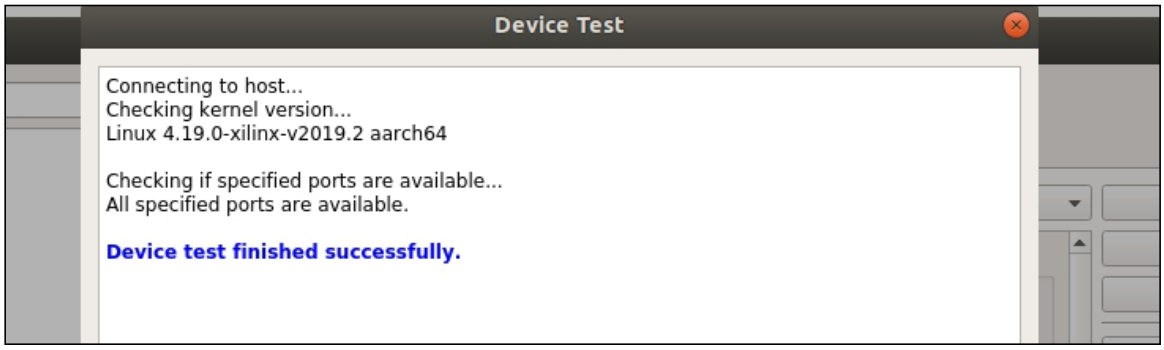


3) The parameter settings are as follows



The password is root by default and needs to be filled in

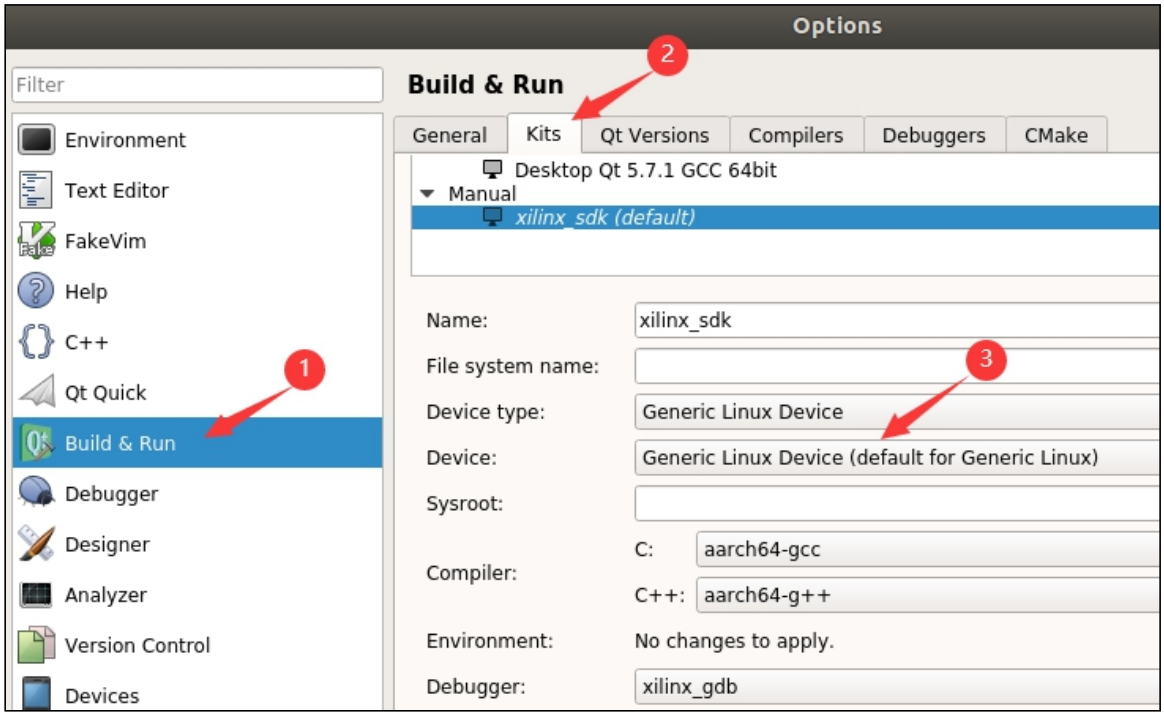
4) Click "Next" and "Finish", in the test that appears later, the interface after success is as follows



If it is not successful, you need to check the network connection

5) Go back to the "Device" setting item, click "Apply" to complete this setting

6) Switch settings and configuration as follows



7) Click "OK" to complete the setting

Part 2.7: Run Hello World

1) Open the helloworld project

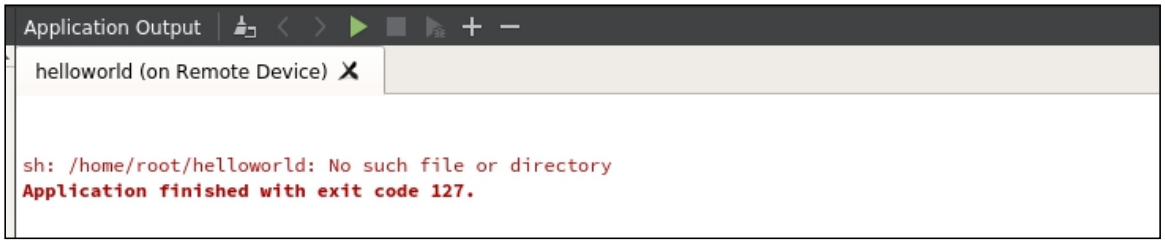
2) Compile the application

3) Click the run button of QtCreator

We can see that the running print information on the right indicates that the program has been run and exited.

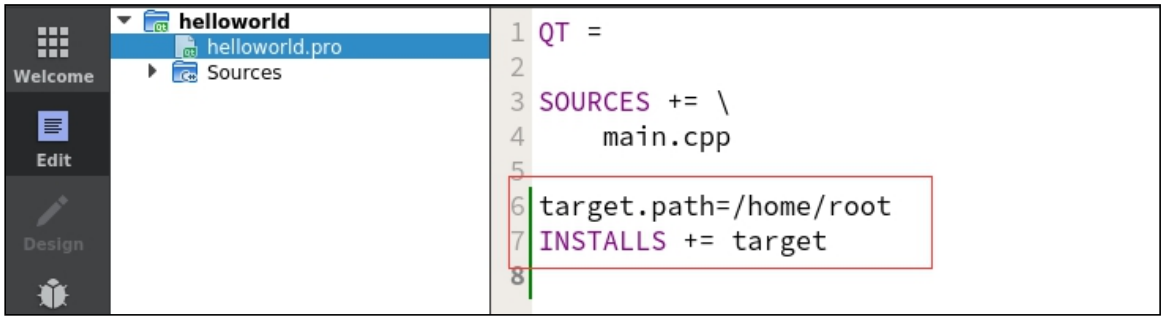
① If the following error is prompted, you need to check whether the board is powered on and enter the linux system, and whether the network is normal

② If the following error is prompted, you need to clear the compilation result and recompile



4) At this time, we check the user directory "/home/root" of the board and find that there is an extra file helloworld

5) We open the pro file of the project

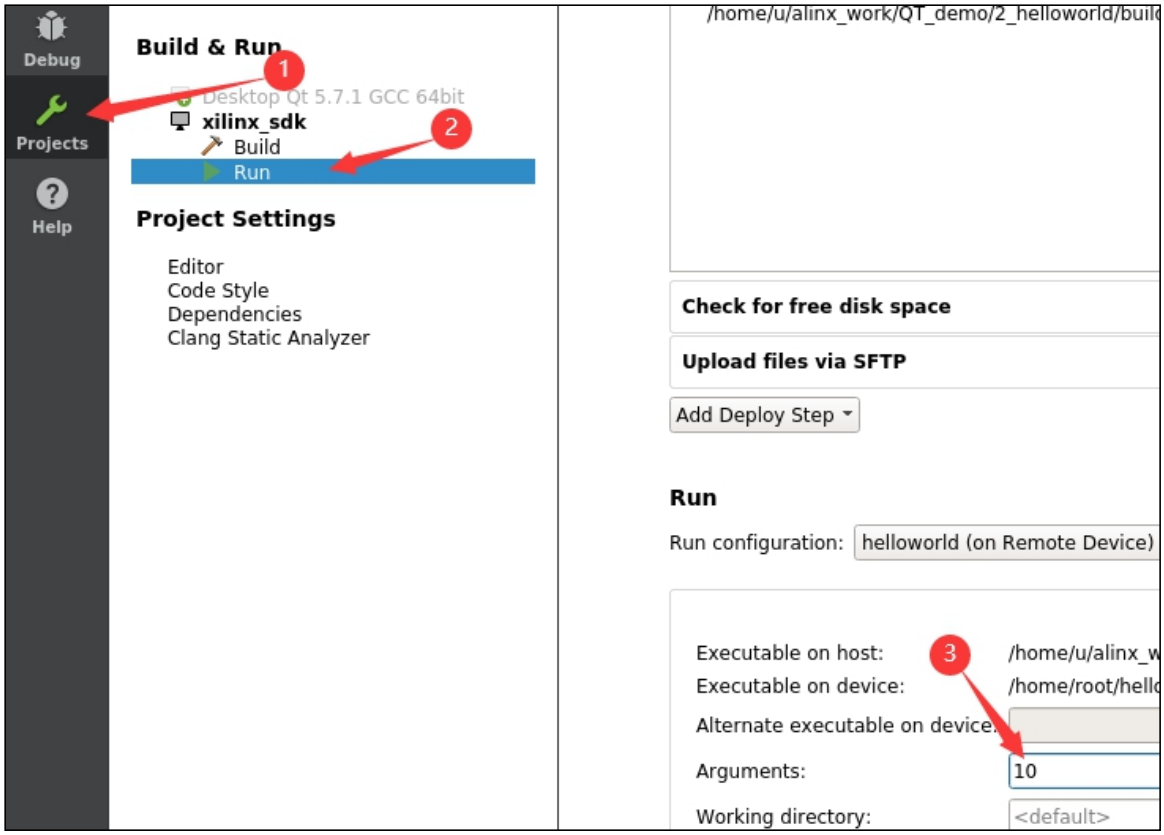


6) The red box is the directory where the program is automatically loaded when running, which explains why the file helloworld appears in the "/home/root" directory of the development board system

Part 2.8: Operation Parameter Setting

Sometimes, when we run the program, we need to add some operating parameters. Below we introduce how to add operating parameters to the running program during remote debugging.

1) Open the following setting items

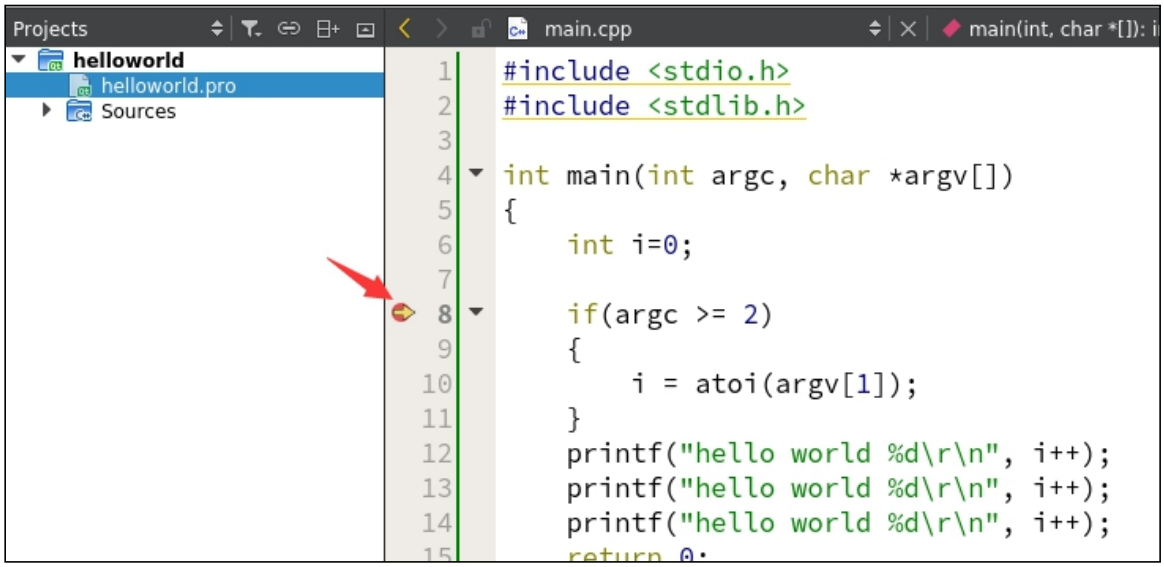


The parameter here is set to 10

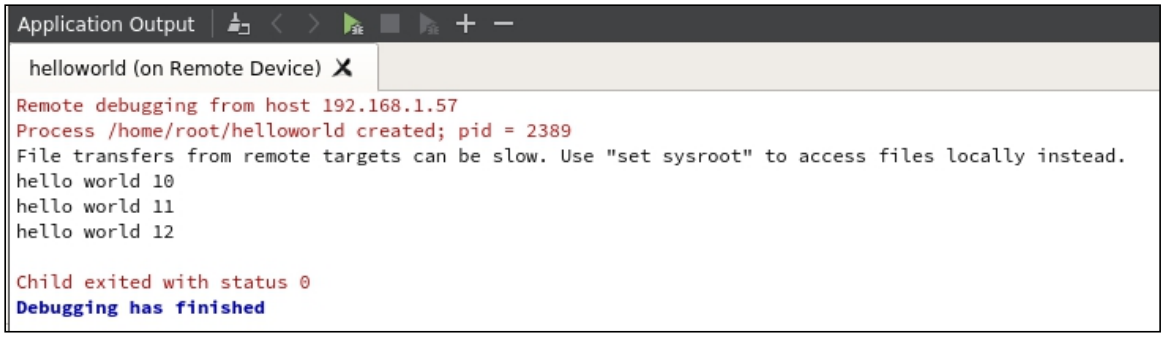
2) Set a breakpoint at the if position of the main function

3) Click the debug button or shortcut key F5

4) We see that the program has stopped at the breakpoint we set



5) Press F5 to continue running, we see the print information as follows



We can see that the starting number after "hello world" has changed to 10, indicating that the parameters we set have taken effect

Part 3: OpenCV Edge Detection

Part 3.1: OpenCV Introduction

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The establishment of OpenCV provides a common infrastructure for computer vision applications and accelerates the application of machine perception in commercial products. As a BSD-licensed product, OpenCV makes it easy for companies to use and modify the code.

The library has more than 2500 optimization algorithms, including a complete set of classic and most advanced computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, recognize objects, classify human behavior in videos, track camera movement, track moving objects, extract 3D models of objects, generate 3D point clouds from stereo cameras, and stitch together images to generate High-resolution images of the entire scene, find similar images from the image library, remove red eyes from images taken with flash, track eye movements, identify the scene and create markers to cover augmented reality, etc. OpenCV has more than 47,000 user communities and an estimated download volume of more than 18 million. It is widely used in companies, research groups and government agencies.

Many implementations of Xilinx's reVision are based on the opencv interface, and the hardware implementation is xfopencv. This will help us realize the hardware (FPGA) acceleration of the corresponding algorithm

Part 3.2: Edge Detection Introduction

Edge detection is a basic problem in image processing and computer vision. The purpose of edge detection is to identify points with obvious brightness changes in digital images. Significant changes in image attributes usually reflect important events and changes in attributes. These include discontinuities in depth, discontinuities in surface orientation, changes in material properties, and changes in scene lighting. Edge detection is a research field in image processing and computer vision, especially in feature extraction.

Image edge information is mainly concentrated in high frequency bands. Generally speaking, image sharpening or edge detection is essentially high-frequency filtering. We know that the differential operation is to find the rate of change of the signal, which has the effect of strengthening the high frequency components. In the spatial calculation, the sharpening of the image is the calculation of the differential. Due to the discrete signal of the digital image, the differential operation becomes the calculation of the difference or gradient. There are a variety of edge detection (gradient) operators in image processing. Commonly used include ordinary first-order difference, Robert operator (cross-difference), Sobel operator, etc., which are based on finding gradient strength. Laplace operator (second-order difference) is based on zero-crossing detection. By calculating the gradient and setting the threshold, the edge image is obtained.

The general steps of edge detection are:

1) **Filtering:** The edge detection algorithm is mainly based on the first and second derivatives of image intensity, but the derivatives are

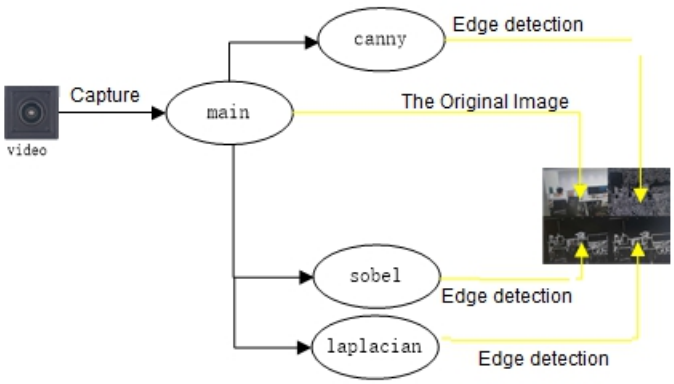
very sensitive to noise, so filters are needed to improve the performance of noise-related edge detectors

2) **Enhancement:** The basis of edge enhancement is to determine the change value of the neighborhood intensity of each point of the image. The enhancement algorithm can highlight the points with significant changes in the intensity value of the gray point neighborhood
3) **Detection:** There are many points in the neighborhood that have large gradient values, but in a specific application, these points are not the edge points to be found, and need to be selected

Part 3.3: Experiment Goal

1) Use opencv to capture usb camera images
2) Edge detection of canny, sobel, and laplacian algorithms
3) Use opencv to display images
4) Responding to opencv window closing operation

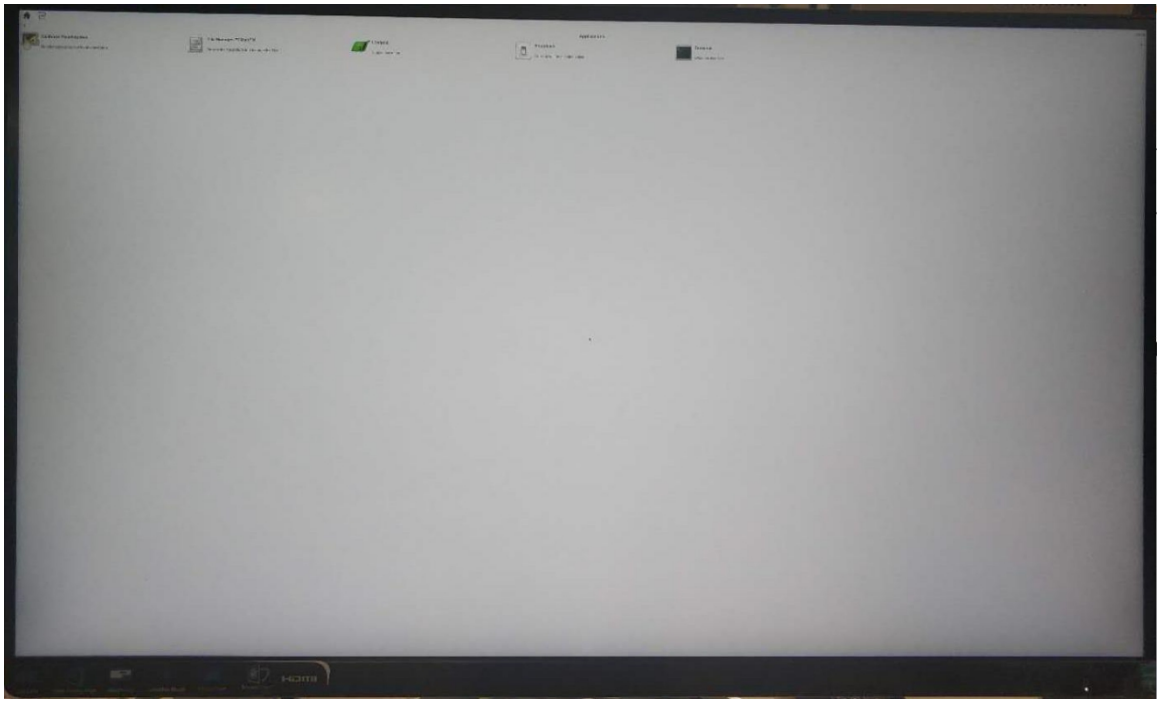
Part 3.4: Software Flow Chart



Part 3.5: Operation Preparation

1) dp interface connected to the display
2) Plug in the usb camera
3) After the development board is powered on, wait for the system

startup to complete, the dp monitor can correctly display the boot desktop

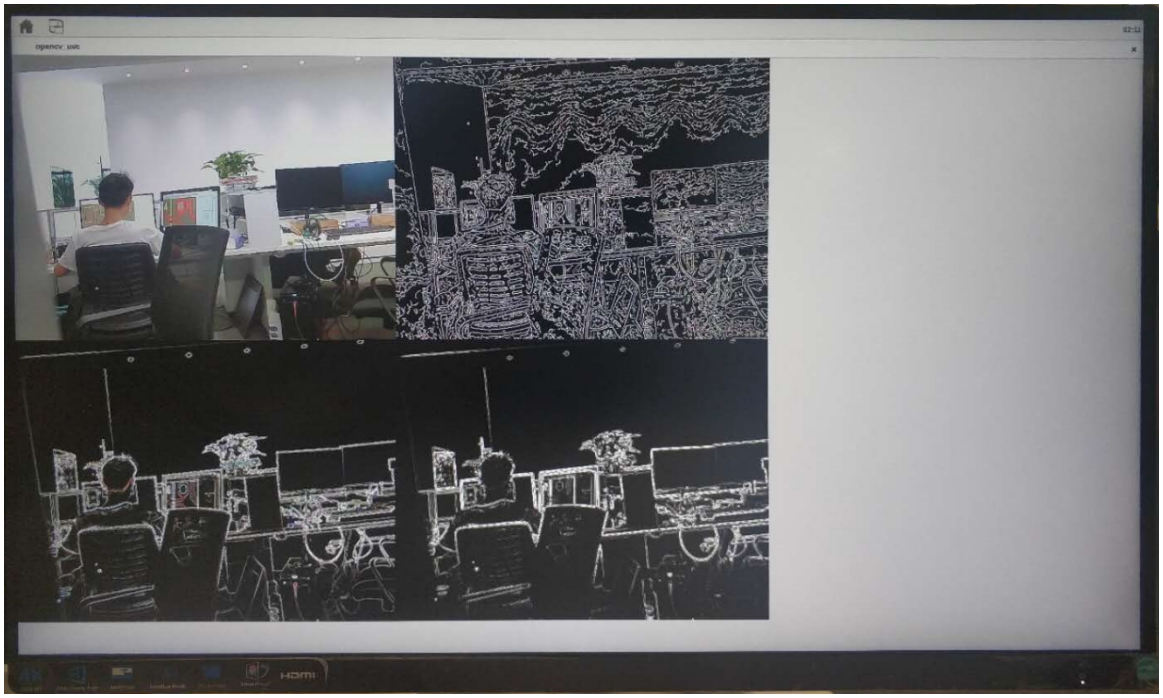


4) In the terminal, you can view the usb camera device

```
root@petalinux:~# ls /dev/video*
/dev/video0  /dev/video1
root@petalinux:~#
```

Part 3.6: Program Running

You can refer to the example in the previous chapter, open the 3_opencv_edgedete project, and use QtCreator to run the application directly. The running effect is as follows



Part 3.7: Code analysis

1) Set environment variables

Enter the following command in the terminal, the effect is equivalent to the call setenv in the code

```
export DISPLAY=:0.0
```

2) Start the desktop

```
/etc/init.d/xserver-nodm_xx start
```

3) Adjust monitor resolution

If the above experiment uses a 4K monitor, so the icons on the boot desktop are very small. In order to better display the effect, in the application, through the system call, set the display output resolution to 1080P. The command is as follows:

```
xrandr --output DP-1 --mode 1920x1080
```

4) Display output management

The system uses output power management by default, that is, the display goes blank after a period of time. At this time, we can use the command to cancel the function, the command is as follows:

```
xset s 0 0
xset dpms 0 0 0
```

5) Multi-threaded operation

This experiment uses three different algorithms to complete edge detection. The three algorithms are assigned to three thread tasks, which can make full use of ARM's quad-core A53

6) usb camera capture

The camera resolution is set to 640x480, and the camera device that is turned on by default is: /dev/video0, because the algorithm calculation here is a bit slow, and the displayed image will be a little delayed.

7) canny edge detection

A multi-level edge detection algorithm developed by John F. Canny in 1986 is considered by many to be the optimal algorithm for edge detection. The three main evaluation criteria for optimal edge detection are:

- Low error rate: Identify as many actual edges as possible while reducing false alarms caused by noise as much as possible.
- High positioning: The marked edge should be as close as possible to the actual edge in the image.
- Minimal response: The edges in the image can only be identified once.

cv:: canny function parameters are as follows:

InputArray	image	Input image
OutputArray	edges	Output edge image
double	threshold1	Threshold 1
double	Threshold2	Threshold 2
int	apertureSize=3	Sobel operator size
bool	L2gradient=false	Whether to use a more accurate way to calculate the image gradient

8) sobel edge detection

This is a more commonly used edge detection method. The function of the Soble operator integrates Gaussian smoothing and differential derivation. It is also called the first-order differential operator. The derivation operator is used to obtain the derivation in both horizontal and vertical directions. , What is obtained is a gradient image of the image in the X direction and the Y direction. Disadvantages: Sensitive and easily affected. Gaussian blur (smoothing) is used to reduce noise, and the edge positioning accuracy is not high enough.

cv:: sobel function parameters are as follows:

InputArray	src	Input image
OutputArray	dst	Output edge image
int	ddepth	Output image depth
int	dx	Order of derivative x.
int	dy	Order of derivative y
int	ksize = 3	Sobel kernel size
double	scale = 1	Optional scale factor for calculated derivative value
double	delta = 0	Optional increment value to add to the result before storing the result in dst
Int	borderType=BORDER_DEFAULT	Pixel extrapolation

9) Laplacian edge detection

The Laplacce operator is a second-order differential operator in the n-dimensional Euclidean space. It is more appropriate when you only care about the position of the edge without considering the grayscale difference of the surrounding pixels. The response of Laplace operator to isolated pixels is stronger than the response to edges or lines, so it is only suitable for noise-free images. In the presence of noise, low-pass filtering is required before edge detection using the Laplacian operator. Therefore, the usual segmentation algorithm is to combine the Laplacian operator and the smoothing operator to generate a new template. Laplace operator, like Sobel operator, belongs to spatial sharpening filtering operation.

cv:: Laplacce function parameters are as follows:

InputArray	src,	Input image
OutputArray	dst	Output edge image
int	ddepth	Output image depth
int	ksize = 1	Used to calculate the size of the second derivative kernel
double	scale = 1	Optional scale factor used to calculate Laplacce
double	delta = 0	Optional increment value to add to the result before storing the result in dst
Int	borderType=BORDER_DEFAULT	Pixel extrapolation

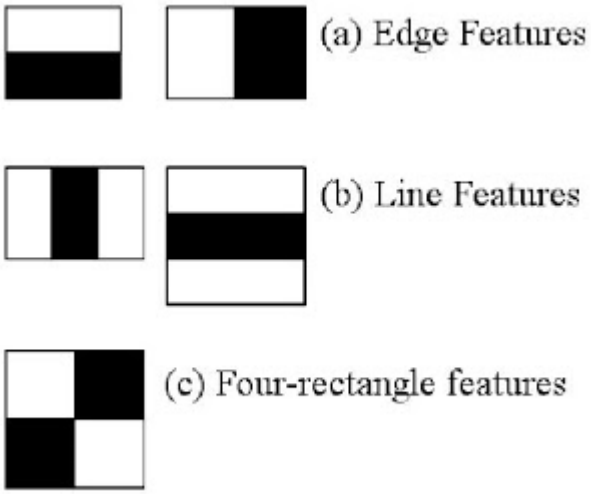
Part 4: OpenCV+Qt Face Detection

Part 4.1: OpenCV's Cascade Classifier

The classifier is a device that judges whether a certain thing belongs to a certain classification. Cascade classifier: It can be understood as connecting N single-class classifiers in series. If a thing can belong to all the classifiers connected in this series, the final result is judged as yes, and if one item does not match, it is judged as no. For example, a human face has many attributes. We make each attribute into a classifier. If a model meets all the attributes of a human face defined by us, then we believe that this model is a human face. So what do these attributes refer to? For example, a human face needs to have two eyebrows, two eyes, a nose, a mouth, a roughly U-shaped chin or outline, and so on.

The target detection of the cascade classifier based on Haar features is an effective target detection method proposed by Paul Viola and Michael Jones in the paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. This is a method based on machine learning, in which a cascade function is trained from many positive and negative images. Then, it is used to detect objects in other images.

We will perform face detection here. The algorithm first requires a large number of positive images (face images) and negative images (no face images) to train the classifier. Then we need to extract features from it.



Aar features are divided into three categories: edge features, linear features, central features and diagonal features, which are combined into feature templates. There are white and black rectangles in the feature template, and the feature value of the template is defined as the sum of white rectangle pixels and minus black rectangle pixels. The Haar eigenvalue reflects the gray level changes of the image. For example, some features of the face can be simply described by rectangular features, such as: the eyes are darker than the cheeks, the sides of the nose are darker than the bridges of the nose, and the mouth is darker than the surroundings. However, the rectangular feature is only sensitive to some simple graphic structures, such as edges and line segments, so it can only describe structures with specific directions (horizontal, vertical, diagonal).

By changing the size and position of the feature template, a large number of features can be listed in the image sub-window. The feature template in the above figure is called the "feature prototype"; the feature that the feature prototype is expanded in the image sub-window (translation and expansion) is called the "rectangular feature"; the value of the rectangular feature is called the "feature value".

The rectangle feature can be located at any position of the image, and the size can also be changed arbitrarily, so the rectangle feature value is a function of the three factors of the rectangle template type, the rectangle position and the rectangle size. Therefore, the change of category, size and position makes a small detection window contain a lot of rectangular features. For example, the number of rectangular features can reach 160,000 in a detection window with a size of 24*24 pixels. So there are two problems to be solved: (1) How to calculate so many features quickly? (2) Which rectangular features are the most effective for classifying the classifier?

Part 4.2: Integral Graph

In order to quickly calculate features, an integral map is introduced here, no matter how large the image is, it will reduce the calculation of a given pixel to operations involving only four pixels. Integral map is a fast algorithm that can find the sum of pixels in all areas of the image by traversing the image only once, which greatly improves the efficiency of image feature value calculation.

The main idea of the integral graph is to store the sum of pixels in the rectangular area formed by the image from the starting point to each point as an element of an array in the memory. When the sum of pixels in a certain area is to be calculated, the elements of the array can be directly indexed. Recalculate the pixel sum of this area, thereby speeding up the calculation (this has a corresponding name, called the dynamic programming algorithm). The integral graph can use the same time (constant time) to calculate different features under multiple scales, thus greatly improving the detection speed.

The integral graph is a matrix representation method that can describe global information. The way the integral map is constructed

is that the value ii(i,j) at position (i,j) is the sum of all pixels in the upper left corner of the original image (i,j):

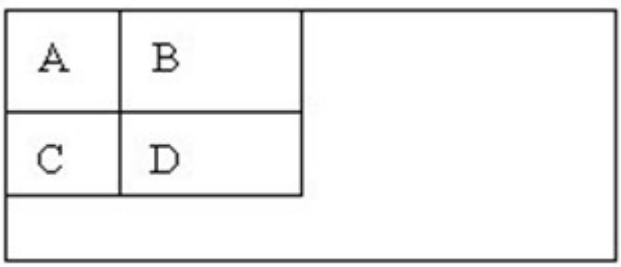
$$\mathrm{ii}(i,j) = \sum_{k \le i \; l \le j} \mathrm{f}(k,l)$$

Integral graph construction algorithm

1) Use s(i,j) to represent the cumulative sum in the row direction, and initialize s(i,-1)=0
2) Use ii(i,j) to represent an integral image, and initialize ii(-1,i)=0
3) Scan the image line by line, recursively calculate the cumulative sum s(i,j) of each pixel (i,j) in the row direction and the value of the integral image ii(i,j)

   s(i,j)=s(i,j-1)+f(i,j)

   ii(i,j)=ii(i-1,j)+s(i,j)
4) Scan the image once, and when it reaches the pixel at the bottom right corner of the image, the integral image ii is constructed



After the integral map is constructed, the cumulative sum of pixels in any matrix area in the image can be obtained by simple calculations as shown in the figure.

Assuming that the four vertices of D are α, β, γ, and δ, the sum of the pixels of D can be expressed as

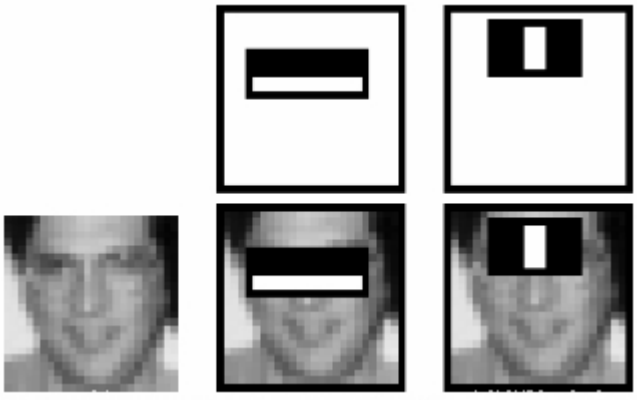
Dsum = ii(α)+ii(β)-(ii(γ)+ii(δ));

The Haar-like eigenvalue is nothing more than the difference between the sum of the two matrix pixels, which can also be

completed in a constant time. Therefore, the calculation of the characteristic value of the rectangular feature is only related to the integral graph of the end point of the characteristic rectangle, so no matter how the scale of the characteristic rectangle is transformed, the time consumed for the calculation of the characteristic value is constant. In this way, as long as the image is traversed once, the characteristic values of all sub-windows can be obtained.

Part 4.3: Select Feature

Of all these features we calculate, most are irrelevant. For example, consider the picture below. The top row shows two good features. The first feature selected seems to focus on the attribute that the eye area is usually darker than the nose and cheek areas. The second feature selected depends on the properties of the eyes that are darker than the bridge of the nose. But the same window applied to the cheeks or any other place has no effect. So how do we choose the best function from more than 160,000 functions? This can be achieved by the AdaBoost algorithm.



We apply each feature to all training images. For each feature, it will find the best threshold, and divide the face into two categories: positive and negative. Obviously, there will be correct or incorrect classifications. We choose the features with the smallest error rate,

which means that they are the features that classify human faces and non-face images most accurately.(The process is not as simple as this. Each image has an equal weight at the beginning. After each classification, the weight of the misclassified image will increase. Then do the same process. Calculate the new error rate, there will be new Weight. The process will continue until the required accuracy or error rate is reached or the required number of features is found)

Part 4.4: Qt Introduction

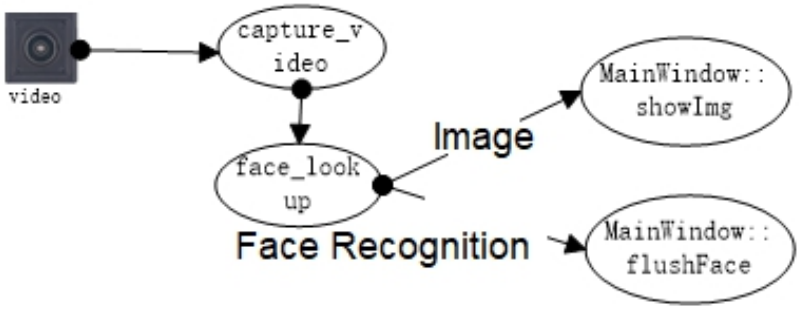
Qt is a cross-platform C++ development library, mainly used to develop Graphical User Interface (GUI) programs, of course, you can also develop Command User Interface (CUI) programs without an interface. Qt also has bindings for scripting languages such as Python, Ruby, and Perl, which means that scripting languages can be used to develop programs based on Qt. Qt supports many operating systems, such as general operating systems Windows, Linux, Unix, smart phone systems Android, iOS, WinPhone, embedded systems QNX, VxWorks, and so on. Although Qt is often used as a GUI library to develop graphical interface applications, this is not all of Qt; in addition to drawing beautiful interfaces (including controls, layout, and interaction), Qt also contains many other functions, such as multithreading , Access to the database, image processing, audio and video processing, network communication, file operations, etc., these Qt have been integrated.

Although Qt also supports mobile phone operating systems, since Android itself already has Java and Kotlin, and iOS itself already has Objective-C and Swift, Qt's market share on the mobile terminal can almost be ignored.

Part 4.5: Experiment Goal

1) Call Qt library to display video image and face block diagram
2) The use of the CascadeClassifier class of the opencv library

Part 4.6: Software Flow Chart

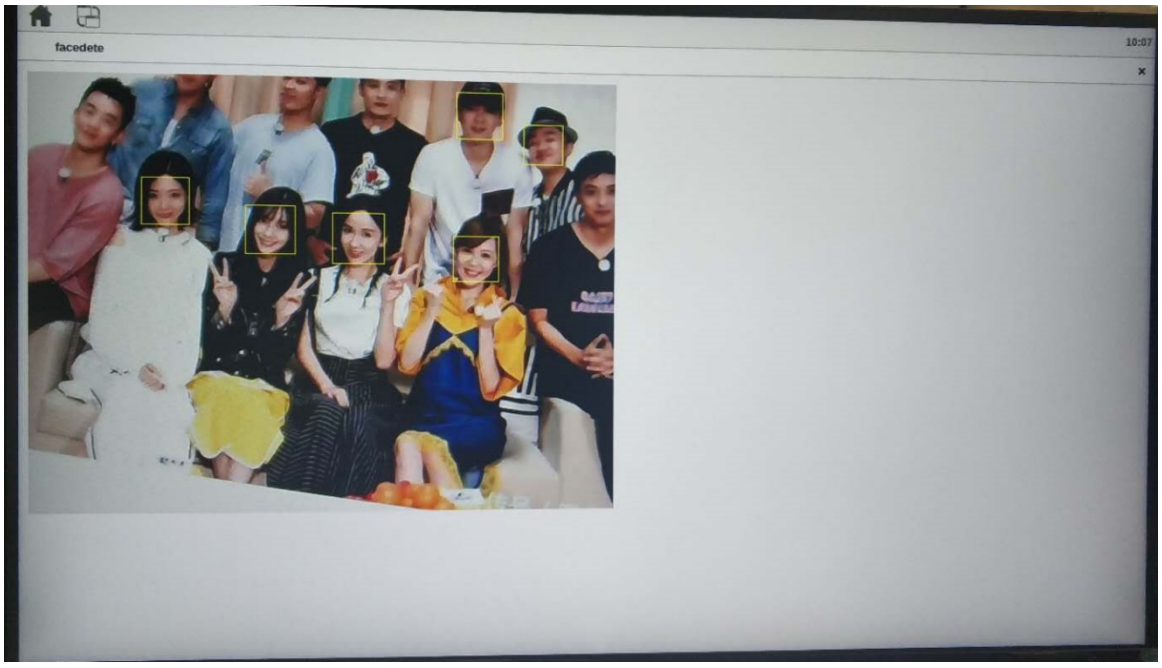


Part 4.7: Operation Preparation

1) Prepare according to the previous chapter
2) Copy the haar_train folder to the /home/root directory of the development board

Part 4.8: Program Running

Run the application, the effect is as follows



Part 4.9: Code analysis

1) Face detection is mainly face_cascade.detectMultiScale in the face_lookup.cpp file During initialization, call face_cascade.load to load the trained model haarcascade_frontalface_alt.xml.
2) In MainWindow::flushFace, the face block diagram will be displayed based on the above detection results
3) Because of the training model, the detection here requires that the face cannot be inclined or sideways.
4) .pro file

   Because the Q display interface is called, in the facedete.pro file, several Qt components of core gui widgets are added after "QT="
5) Qt's signals and slots

   The program binds the signal of face_lookup and MainWindow to the slot, and MainWindow handles the picture that needs to be displayed and marks the position of the face
6) Same as the above routine, the display effect is relatively lagging.

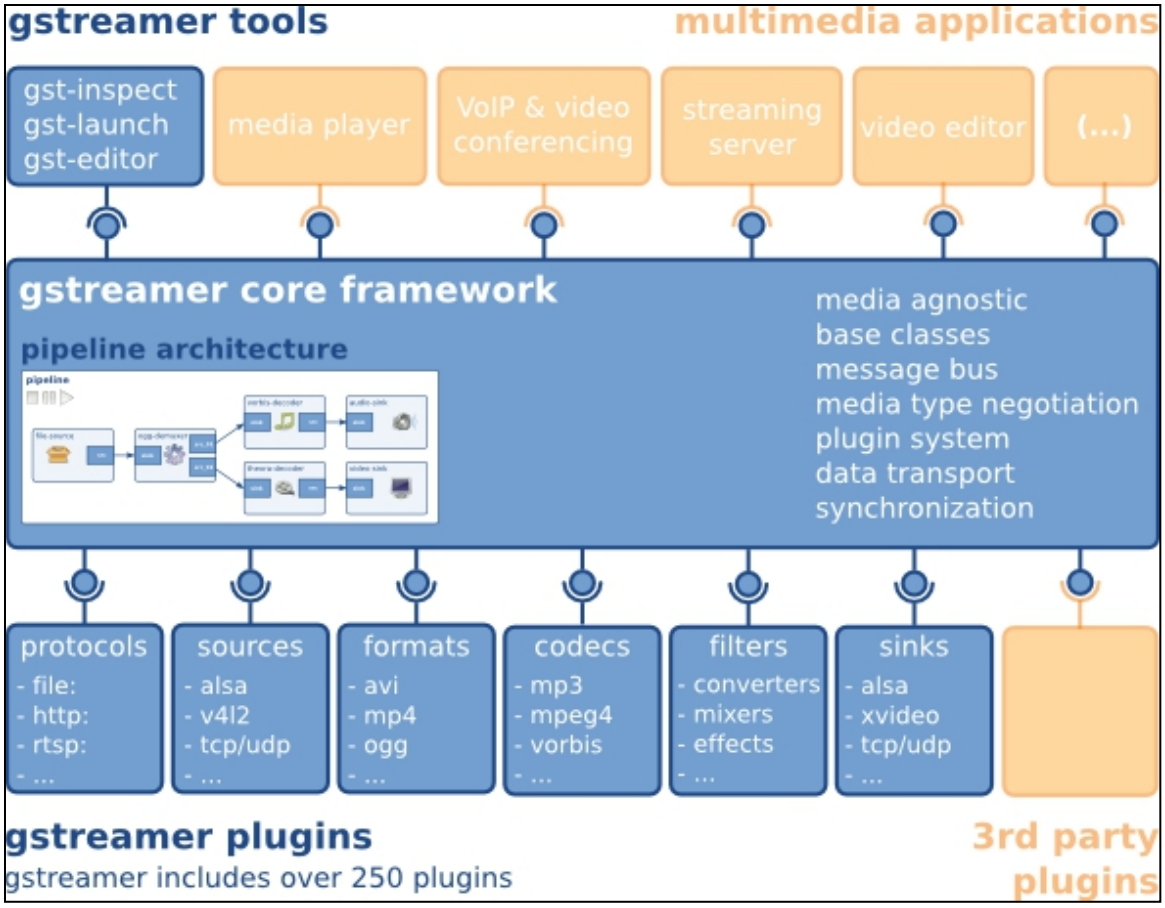
Part 5: GStreamer's Camera Display

Part 5.1: GStreamer Introduction

GStreamer is a framework for creating streaming media applications. The basic design ideas come from the Oregon Graduate School's ideas about video pipelines, and also draw lessons from DirectShow's design ideas. Its characteristics are as follows:

1) Clear structure and powerful function
2) Object-oriented programming ideas
3) Flexible and expandable functions
4) High performance
5) Separation of core library and plug-in

The architecture of GStreamer is as follows. It supports applications ranging from simple Ogg/Vorbis playback and audio/video streaming to complex audio (mixing) and video (non-linear editing) processing. Applications can transparently utilize advanced codecs and filtering techniques.



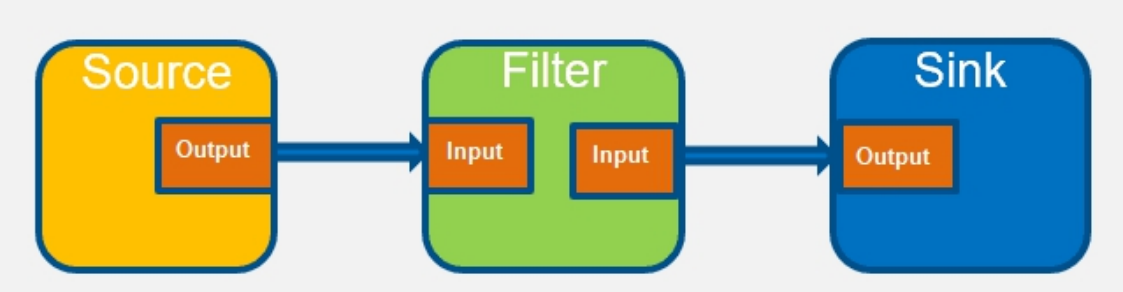
Part 5.2: Basic concepts of GStreamer

1) Element

Elements are the most basic components that make up the pipeline. Several Elements can be connected together to create a pipeline to complete a special task. For example, media playback or video recording. Elements are classified according to function:

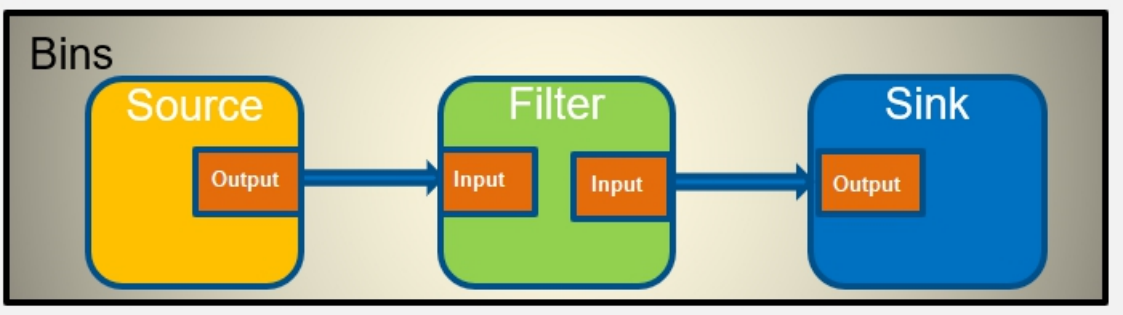
- Source elements
- Sink elements
- Filter elements

The application block diagram is as follows:



2) bin and Pipeline

Bins is a container that can hold elements. Pipelines are a special subcategory of Bins. Pipelines can operate all elements contained within itself. As shown below



3) Pad

Pads are equivalent to the interface of a component, and each component is connected through this interface, so that the data stream can be transmitted in these components. Pads category:

- always
- sometimes
- on request

4) Cap

Caps describes the format of the data stream that can pass through the pad or currently pass through the pad. You can view the media types supported by the component through the gst-inspect-1.0 command.

5) Bus

Each pipeline contains a bus by default, and the application does not need to create another bus. The application program sets up a

message processor on the bus. When the main loop is running, the bus will poll the message processor for new messages. When the messages are collected, the bus will call the corresponding callback function to complete the task

6) Buffer

The buffer contains the data stream in the created pipe. Usually a source component will create a new buffer, and at the same time the component will pass the data in the buffer to the next component. When using GStreamer's underlying structure to create a media pipeline, you don't need to process the buffers yourself, and the component will automatically process these buffers. The buffer zone is mainly composed of the following components:

- Pointer to a block of memory
- The size of the memory
- Buffer timestamp
- A reference count indicates the number of components used in the buffer. When there is no component to reference, this reference will be used to destroy the buffer

7) Events

Events include control information in the pipeline, such as search information and flow termination signals.

Part 5.3: Experiment Goal

gstreamer tool usage

gstreamer programming

Part 5.4: Gstreamer Common Tools

1) gst-inspect-1.0

Without parameters, it will list all available elements, that is, all

the elements you can use, with a file name, it will use this file as a plug-in of GStreamer, try to open it, and then list all the internal elements, With an element of GStreamer, all the information of the element will be listed.

Print the list of plug-ins and their information, such as:

Print the list of supported plugins	gst-inspect-1.0 -a
Print plugin filesrc information	gst-inspect-1.0 filesrc

Pad Templates: This section will list all the types of Pads and their Caps. Through these you can confirm whether you can connect to a certain element

2) gst-discoverer

This tool can be used to view the Caps contained in the file, which is a package of the GstDiscoverer object. Accept a URI from the command line, and then print out all the information. This is useful when looking at how the media is encoded and reused, so that we can determine what element to put in the pipeline.

3) gst-launch-1.0

This tool can create a pipeline, initialize and then run. It allows you to quickly test the pipeline before you formally write the code to see if it works. If you need to view the Caps generated by an element in the pipeline, use the -v parameter in gst-launch, as follows

```
gst-launch-1.0 -v v4l2src device=/dev/video0
```

As you can see from the picture below, the default video format of the camera is YUY2 with a resolution of 2592x1944

```
root@petalinux:~# gst-launch-1.0 -v v4l2src device=/dev/video0
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
/GstPipeline:pipeline0/GstV4l2Src:v4l2src0.GstPad:src: caps = video/x-raw, format=(string)YUY2, wid
th=(int)2592, height=(int)1944, pixel-aspect-ratio=(fraction)1/1, interlace-mode=(string)progressiv
e, framerate=(fraction)25/1, colorimetry=(string)bt709
ERROR: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Intern[ 3714.942617] xhci-hcd xhci-
hcd.0.auto: ERROR unknown event type 37
al data stream error.
Additional debug info:
../../../../git/libs/gst/base/gstbasesrc.c(3055): gst_base_src_loop (): /GstPipeline:pipeline0/GstV
4l2Src:v4l2src0:
streaming stopped, reason not-linked (-1)
Execution ended after 0:00:00.435641850
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
root@petalinux:~#
```

Part 5.5: gst-launch-1.0 camera display

1) Confirm the video format supported by your usb camera

   Copy tools/uvc_fmt to the FPGA development board to run, you can scan the supported resolution

```
root@petalinux:~# ./uvc_fmt 0
query /dev/video0 format
        1.YUYV 4:2:2
                1920x1080@60
                640x480@60
                800x600@60
                1024x768@60
                1600x1200@25
                2048x1536@25
                2592x1944@25
root@petalinux:~#
```

2) Under the control terminal, enter the following command

```
gst-launch-1.0 v4l2src device=/dev/video0 !
video/x-raw,format=YUY2,width=1920,height=1080 ! kmssink
bus-id=fd4a0000.zynqmp-display fullscreen-overlay=1
```

   Note: The resolution and video format here need to be supported by your own usb camera, which can be set according to the resolution supported by your camera, and the corresponding resolution monitor also needs to be supported. When the video is displayed, the resolution of the monitor will be set to be consistent with the captured image

3) running result

   If the image can be displayed, we can see that the image will be very smooth, but the disadvantage is that it can only display the image on a single screen, and it can only display the resolution

supported by both the camera and the monitor.

Part 5.6: Programming Run Program

Through commands, we can quickly verify various scenarios we envisioned, but if we want more status monitoring or user control, this needs to be achieved through programming.

The routines in this chapter use programs to achieve the above command effects

1) The matters needing attention are the same as above, you need to modify the resolution and format supported by the usb camera
2) The program monitors and handles various status changes by adding a callback function sink_message for message events.
3) After running the program, you can see that the displayed image is the same as the command run. At this time, unplug the camera, the program prints the following information and the program exits.

```
get error: Could not read from resource.
play error
playing out
Application finished with exit code 0.
```

Part 5.7: Code Analysis

1) gst_parse_launch

This function creates a pipelin, the function parameters are as follows:

const gchar *	pipeline_description	Describe the pipeline
GError **	error	Error return

2) gst_bus_add_watch

Add the callback function of the bus to process the necessary messages

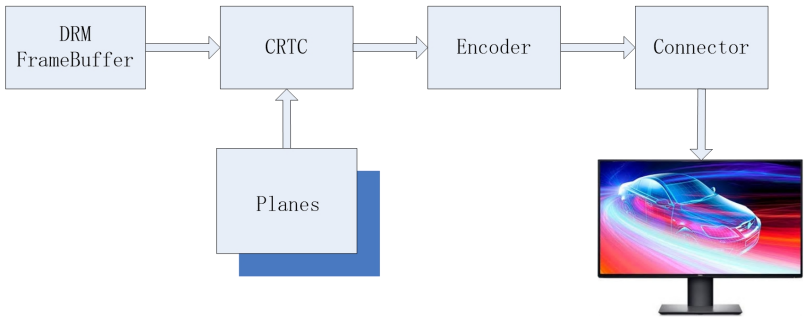
Part 6: Qt+DRM+Gstreamer Camera Display

Part 6.1: DRM Introduction

DRM, the full name of Direct Rendering Manager. It was created to solve the problem of collaborative use of Video Card resources by multiple programs. It provides a set of APIs to user space for output management.

DRM has now covered many functions previously handled by user space programs, such as frame buffer management and mode setting, memory shared objects and memory synchronization. Some of these extensions have specific names, such as graphical execution manager GEM or kernel mode settings KMS, which belong to the DRM subsystem.

DRM shows that it mainly involves the following 5 modules



1) DRM Framebuffer

From the developer's point of view, FrameBuffer is a piece of memory, and writing data in a specific format to the memory means outputting content to the screen. So FrameBuffer is a canvas. For example, for the FrameBuffer initialized to 16-bit color, the two bytes in the FrameBuffer represent a point on the screen, from top to bottom, from left to right, the screen position and the memory address have a

linear relationship in order.

2) CRTC

The task of CRTC is to read the image to be displayed from the Framebuffer and output it to the Encoder according to the corresponding format. Although CRTC literally means a cathode ray picture tube controller, CRT has long been eliminated in ordinary display equipment. The main functions of CRTC in DRI are as follows:

- Configure the resolution (kernel) suitable for the display and output the corresponding timing (hardware logic)
- Scan the framebuffer and send it to one or more display devices

3) Planes

With the continuous updating of software technology, the requirements for hardware performance are getting higher and higher. Under the premise of meeting the normal use of functions, the requirements for power consumption are becoming more and more demanding. Originally the GPU can handle all graphics tasks, but due to its high power consumption during runtime, the designers decided to hand over some simple tasks to the Display Controller for processing (such as compositing), and let the GPU focus on drawing (that is, rendering) This main task. Reduce the burden on the GPU, so as to achieve the purpose of reducing power consumption and improving performance. Thus, Plane (hardware layer unit) was born. Planes mainly superimpose and output multiple layers. Some hardware supports layer translation, scaling, cropping, alpha channel, etc.

4) Encoder

Encode a certain format of image signal (such as RGB, YUV, etc.) into the signal that the connector needs to output. Taking HDMI as an example, the frame/line synchronization/display content is

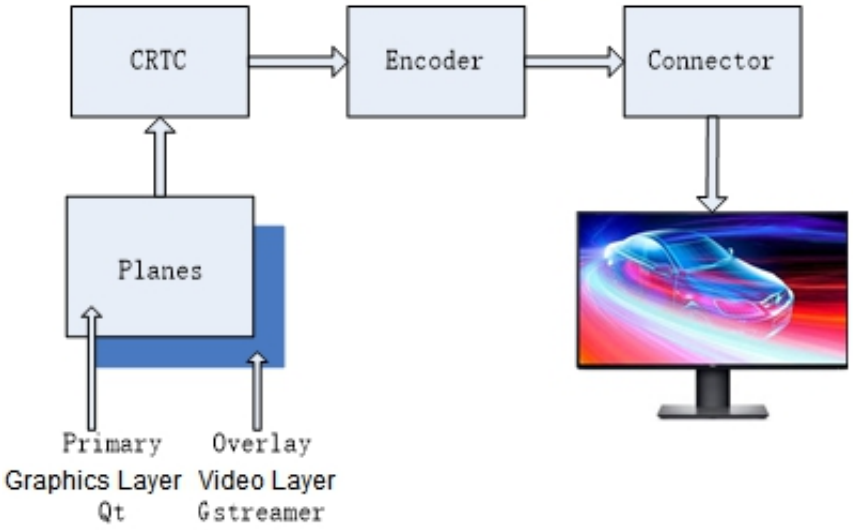
output through the serial bus of TMDS data, and then the parallel sequence is encoded into serial sequence according to the HDMI standard, which is the task of the Encoder.

5) Connector

Connector is actually the physical interface connected to the display. The common ones are VGA/HDMI/DVI/DP, etc. The parameters supported by the display can be read out through the connector

Part 6.2:Experiment Introduction

The DP output of MPSoc supports two layers, and the upper layer (Primary) supports the alpha channel. The experiment in this chapter uses Qt to output content to the Primary layer, and Gstreamer captures video data to the following layer (Overlay). Finally, these two layers are mixed and output by the hardware. The working block diagram is as follows:



Part 6.3: Experiment Goal

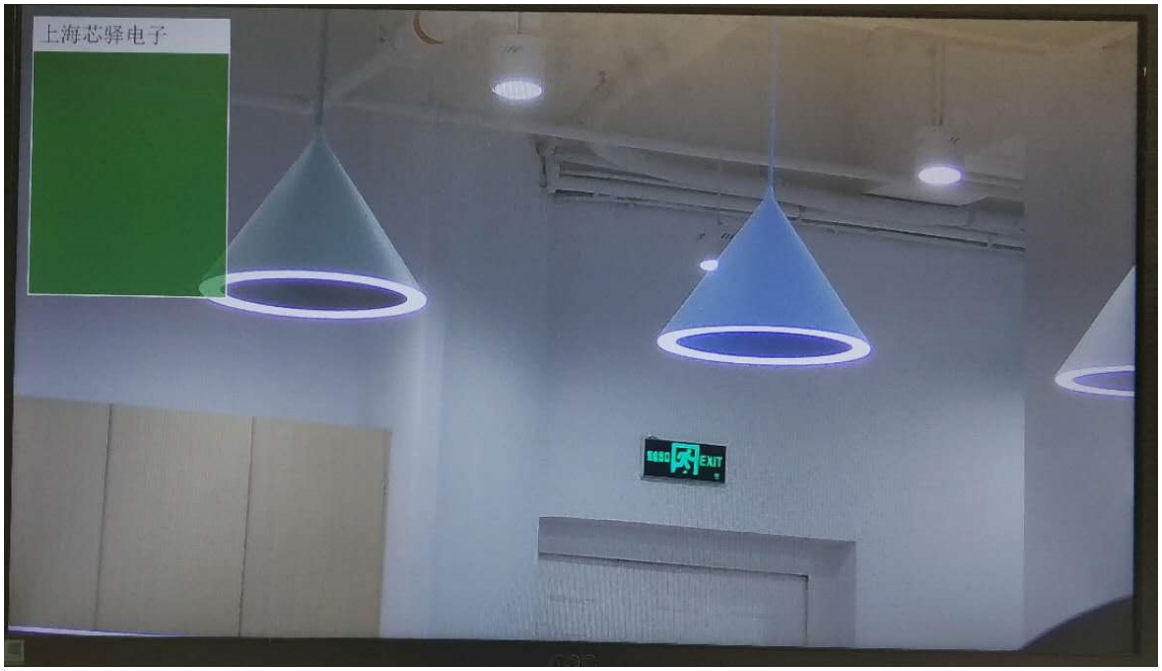
1) DRM output management

2) DRM layer management
3) Qt and GStreamer hierarchical display

Part 6.4: Run the program

1) It should be noted that this experiment requires a usb camera to support 1080 resolution (refers to the bare stream, such as the YUYV422 format above)
2) Experimental operation effect

   You can see that a Qt interface is superimposed on the video



Part 6.5: Code analysis

1) Check whether the display is connected

   In the main function, call init_drm to determine whether the initialization is successful, and return -2, which means there is no supported display format, indicating that the monitor is not connected
2) Set the monitor output resolution

   Set the output resolution and refresh rate through the function drmModeSetCrtc. There is no buffer set here, so the third

parameter is set to -1

3) Set the transparency properties of the graphics layer

The graphics layer supports global and single-pixel settings for alpha values. Here it is set to a single pixel, that is, g_alpha_en is set to 0

4) Qt display

Qt is displayed in the Primary layer by default

5) GStreamer display

The plane information can be obtained through drmModeGetPlaneResources, including the id number of the plane. The ID of the overlay layer obtained here is 35. As you can see from the code, the attributes of kmssink are set as follows:

```
kmssink sink-type=dp bus-id=fd4a0000.zynqmp-display fullscreen-overlay=false plane-id=35
```

Part 7: Linux Register Operation

Part 7.1: Linux Memory Map

Memory mapping is to map a memory area of user space to kernel space. After the mapping is successful, the user's modification of this memory area can be directly reflected in the kernel space. Similarly, the modification of this area by the kernel space can also directly reflect the user space. This is very efficient for transferring large amounts of data between kernel space and user space.

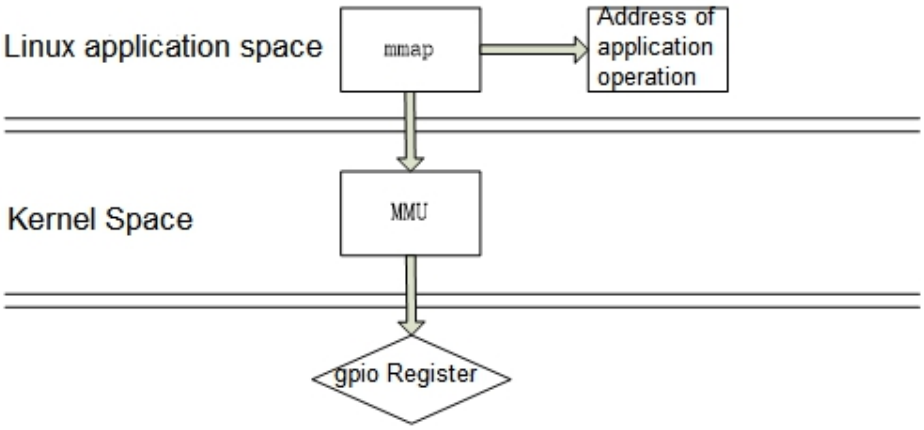
For example, to read a file from the hard disk into the memory, the data must be copied through the file system, and the data copy operation is realized by the file system and the hardware driver. In theory, the efficiency of copying data is the same. But accessing files on the hard disk through memory mapping is more efficient than calling the read and write systems. Why?

The reason is that read() is a system call in which data is copied. It first copies the content of the file from the hard disk to a buffer in the kernel space, and then copies the data to the user space. In this process, the data is actually copied twice. And mmap() is also a system call, there is no data copy in mmap(), and the real data copy is performed when the page fault is interrupted. Because mmap() directly maps the file to the user space, the interrupt handling function directly copies the file from the hard disk to the user space according to this mapping relationship, and only one data copy is performed. Therefore, the efficiency of memory mapping is higher than that of read/write.

Part 7.2: Experiment Introduction

Sometimes, the function of a certain peripheral has been verified under bare metal. If the peripheral does not use interrupts, at this time, we can use the method of mapping registers under linux to directly transplant the program under bare metal to linux. In this way, you do not need to develop a driver.

This example transplants the gpio example under vitis to drive ps_led, and the LED flashes once per second. For other operations, the process is similar.



Part 7.3: Run the program

After running the application register_opt, you can see that the LED light of PS flashes once per second

Part 7.4: Code analysis

1) Regarding the register address of gpio, here are all copied from the gpio routine of vitisi, and our development idea should be like this later, first use vitisi's bare metal program to verify. For many peripherals and FPGA-side ips, vitisi will help us generate good operation methods and operation addresses, so that we don't need to find the corresponding relationship.
2) When opening /dev/mem, use the option O_SYNC

After writing data to the outside, usually the data is written to the cache buffer. O_SYNC will ensure that data is written to the peripheral before returning. It should be noted that O_SYNC here only affects write operations

3) Call msync

If you need to write more data to the peripheral at one time, if you call O_SYNC at this time, it will seriously affect the performance of the system. At this time, if we do not use O_SYNC, but call msync after writing the data, this will improve Write performance.

4) Read operation consistency problem

If you need to read the data of the peripheral, because of the existence of the cache, the data fetched in the application is the data in the cache, not the latest state of the peripheral. At this time, the read may be an incorrect value.