Geniatech XPI-905X3

Android Software Development Instruction

目　录

No.1、Preparing the development environment..... 2
A、Install ubuntu system on VMware Workstation..... 2
B、installing JDK1.8 on Ubuntu 64-bit system..... 15
C、Install some necessary software packages and GNU tool chain..... 17
No.2、SDK Catalogue..... 19
No.3、How to build Code..... 25
1、GPIO..... 27
2、I2C..... 30
3、UART..... 31
4、SPI..... 32

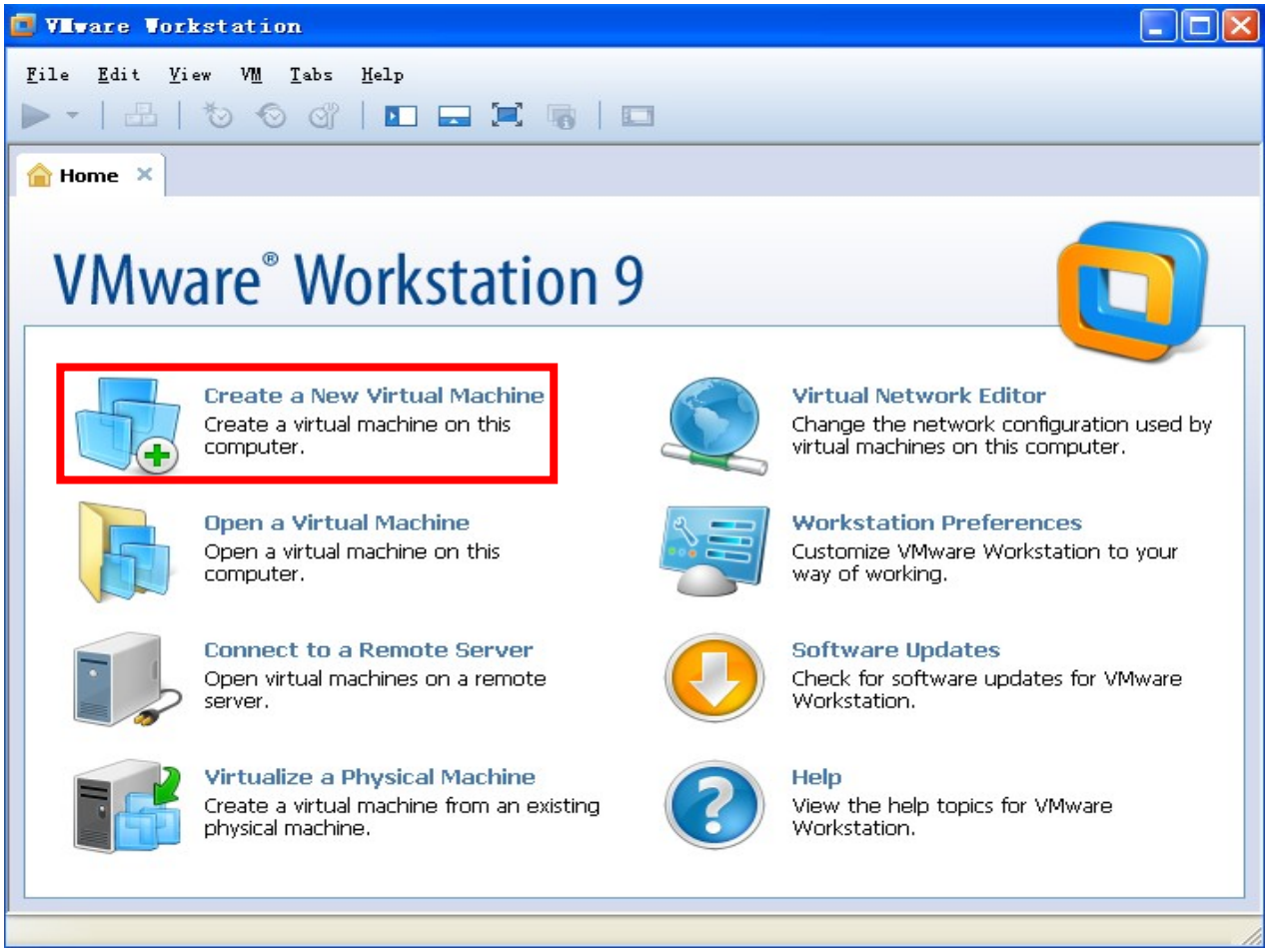
No.1、Preparing the development environment

Firstly you must install a 64-bit ubuntu operation system. You'd better not use a virtual machine but a real 64-bit ubuntu operating system PC if you can. Here we use a virtual machine which shows how to set up the enironment to compile the source code. The VMware Workstation version is 9.0.0 build-812388.

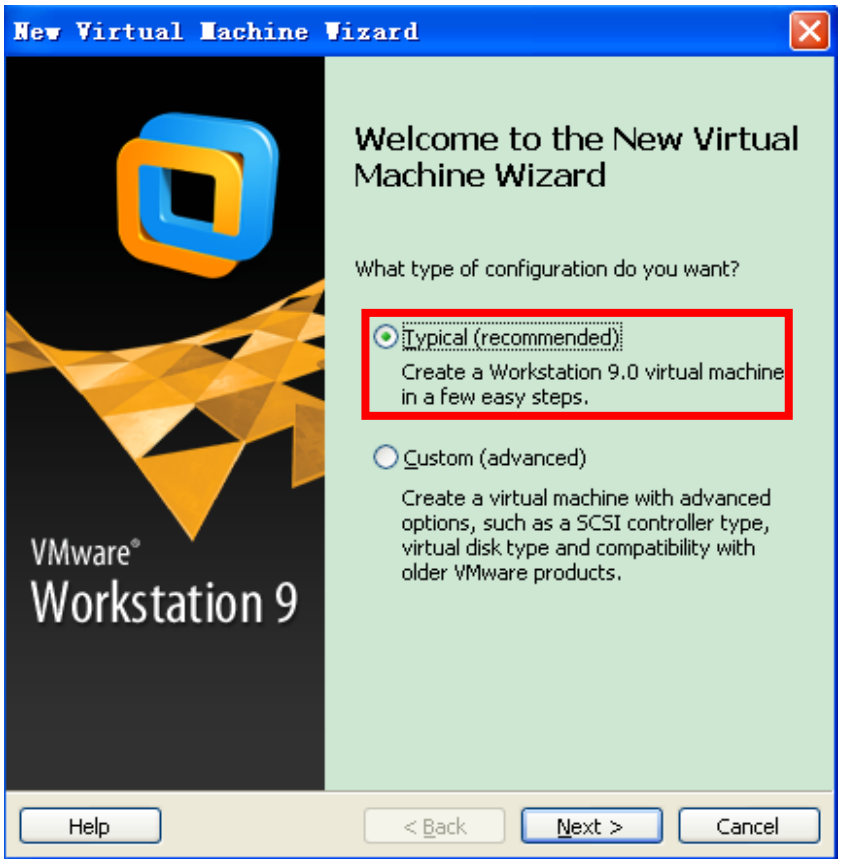
A、Install ubuntu system on VMware Workstation.

Here we use the ubuntu-14.04.5-desktop-amd64.iso image file as demonstration to install ubuntu virtual system. Please refer to following instructions:

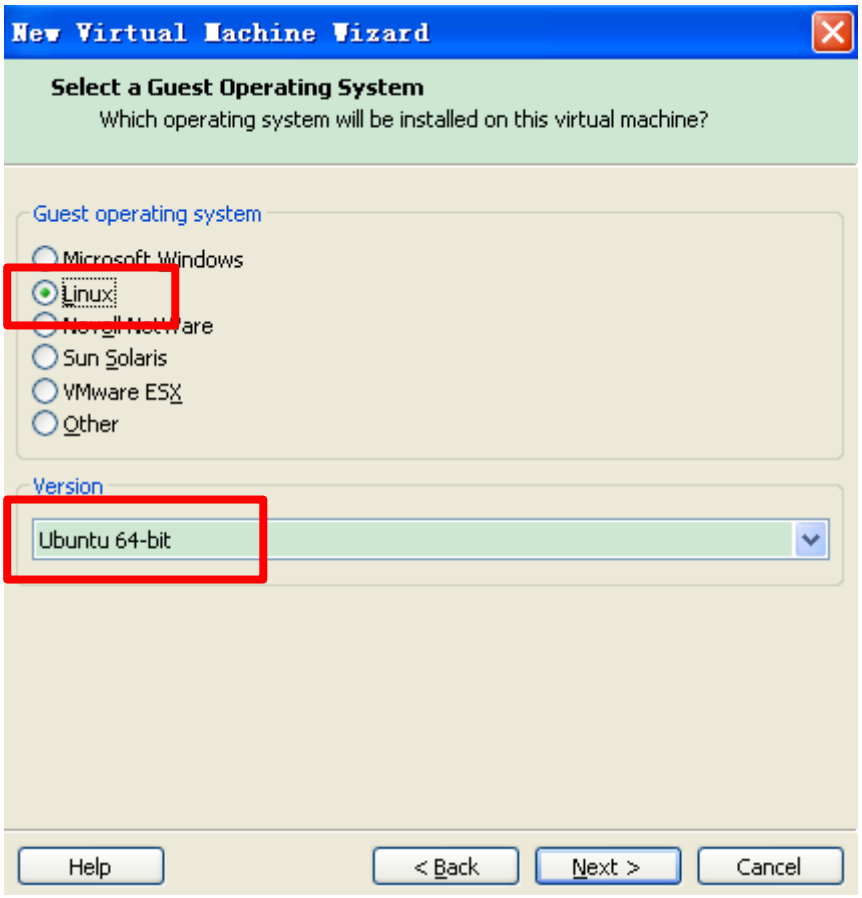
1. Open the VmwareWorkstation, click on the graph of the "Create a New Virtual Machine", and



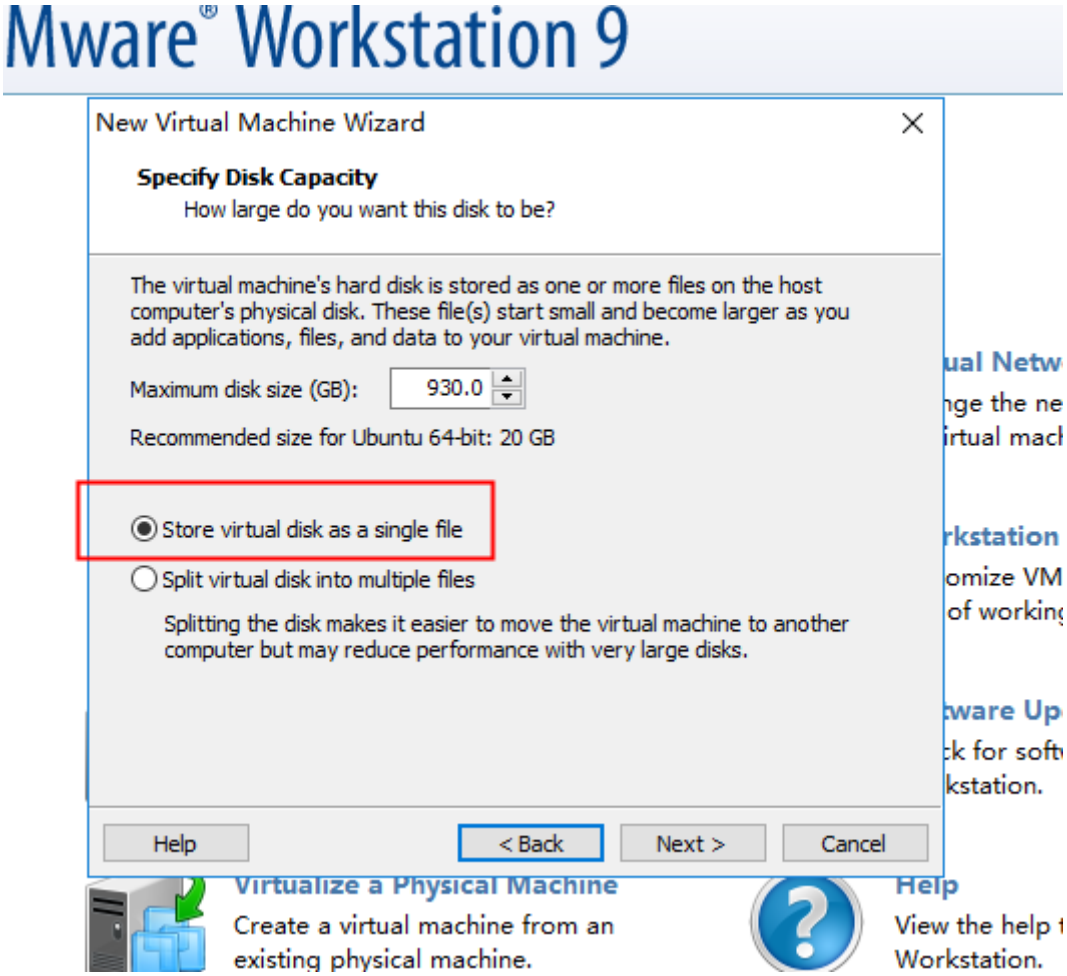
open the "New Virtual Machine Wizard".



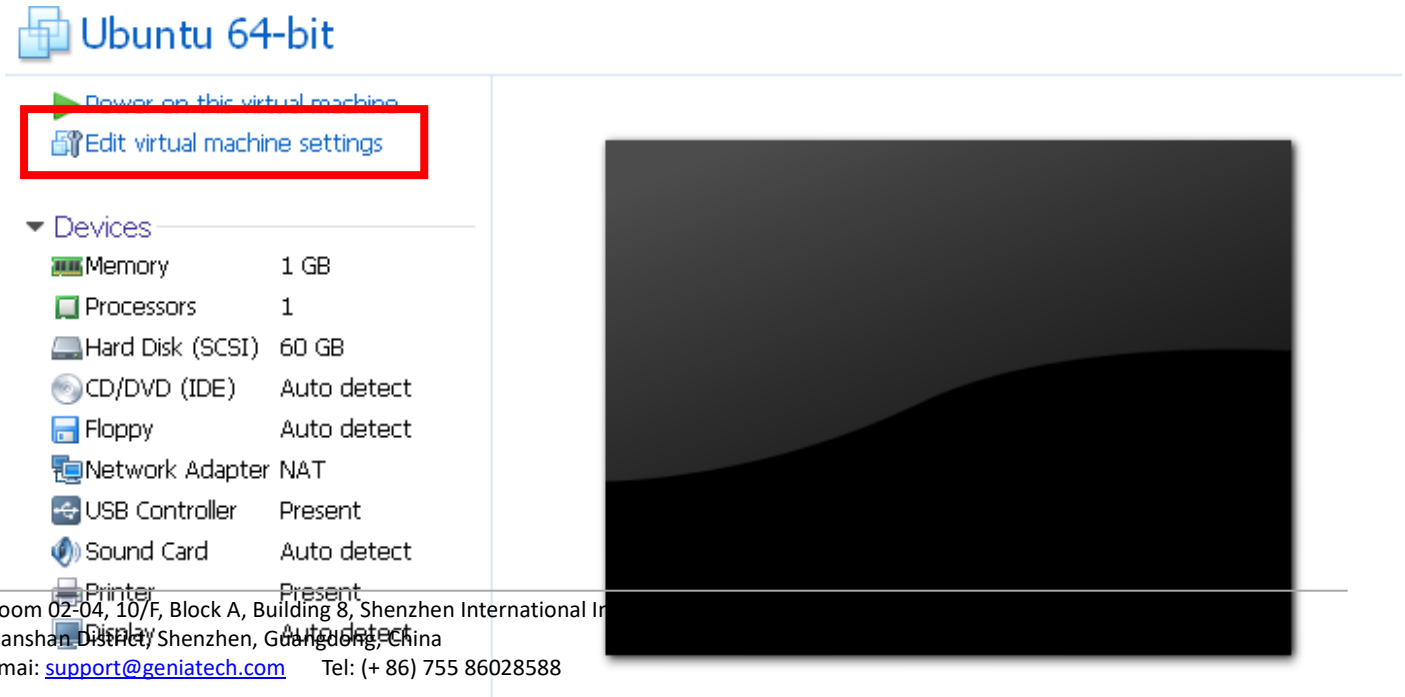
2. Default select "Typical" configuration and click on “Next” button, and then default select the “I will install the operating system later” item.
3. Click on the “Next” button to select a guest operating system, as in the following figure.



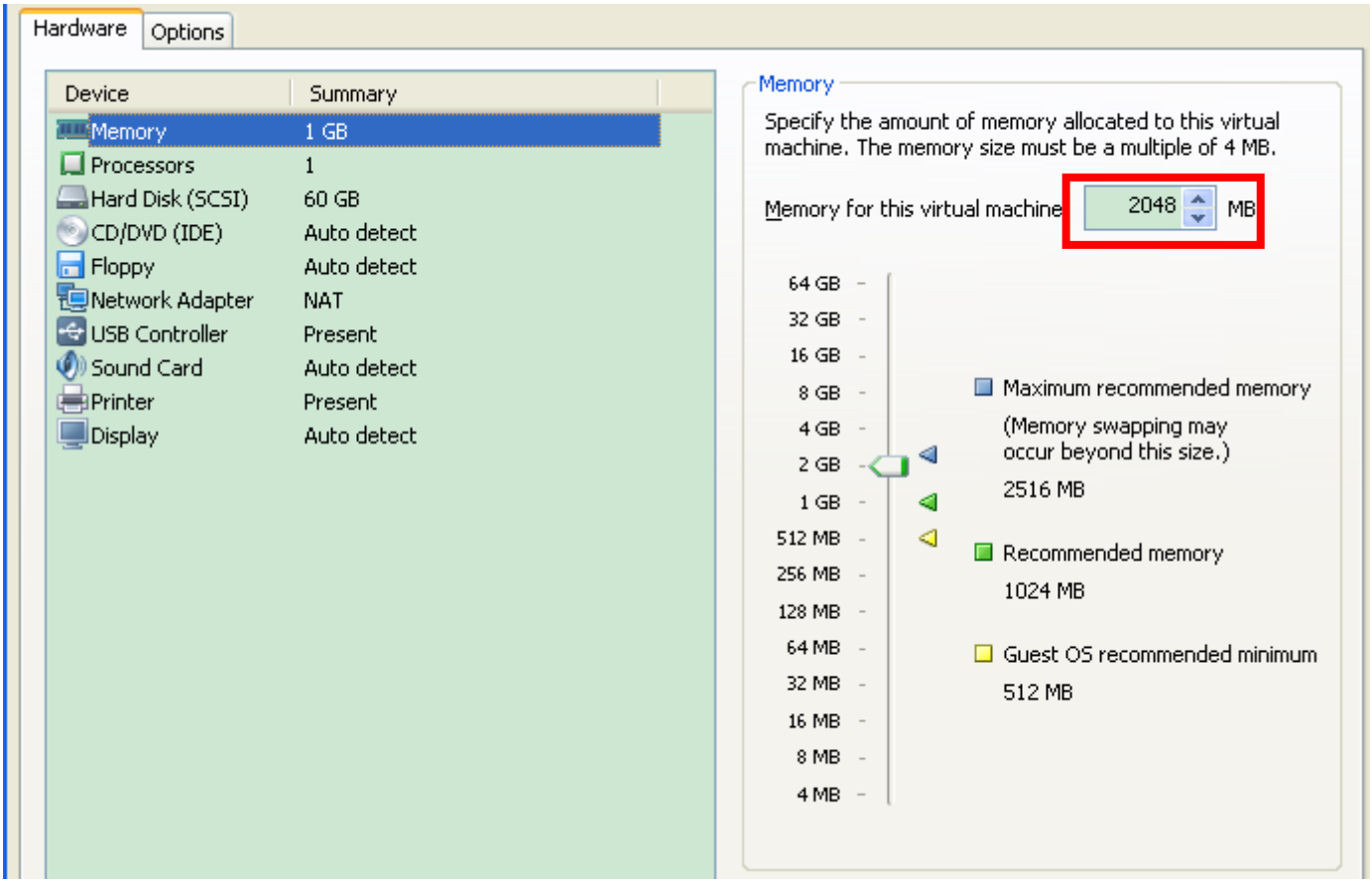
4. Click on the "Next" button, set the virtual machine name and path as you like.
5. Click on the "Next" button, specify disk capacity, in general 60G is enough. Here select "Store virtual disk as a single file" item.



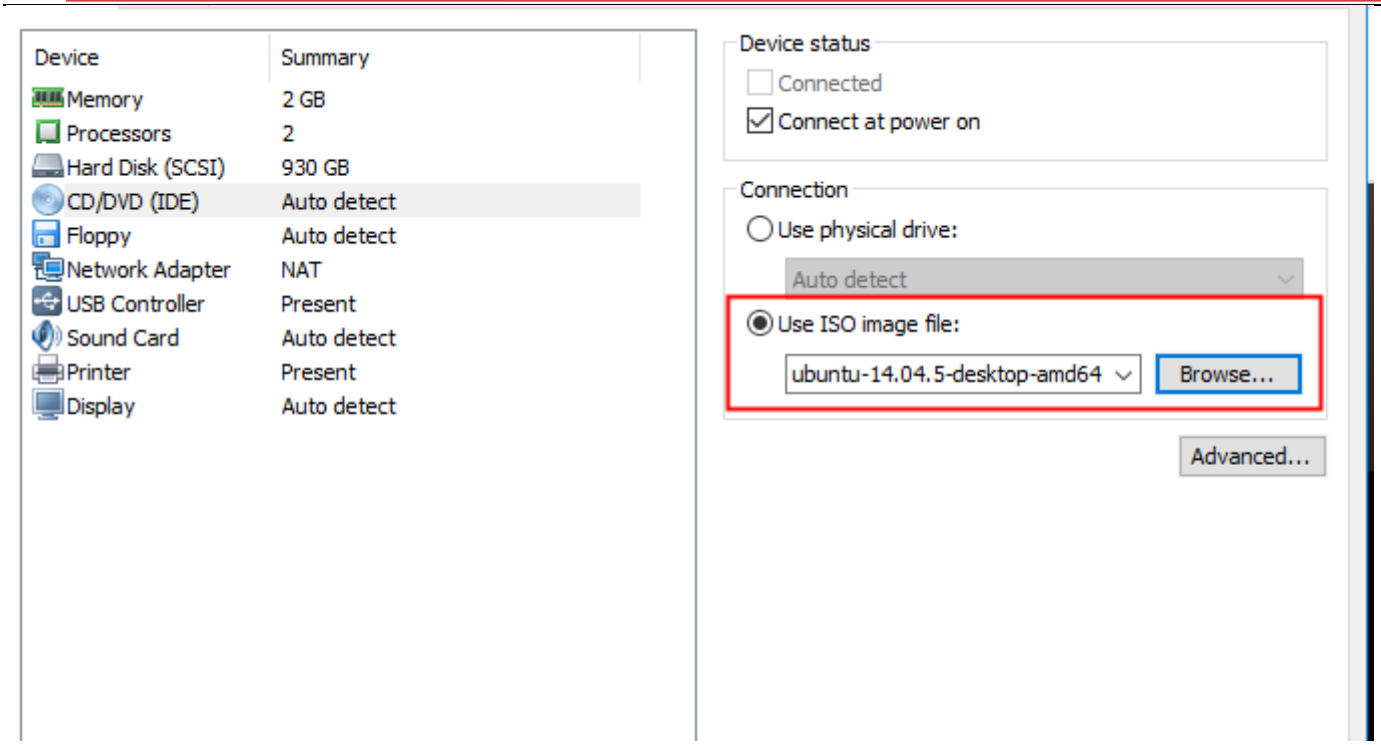
6. Click on the “Next” button, confirm setup information, after that click on the “finish” button to complete the configuration.
7. Now click on the “Edit virtual machine settings”, choose “Harware” tab, and set the memery to 2GB or more.

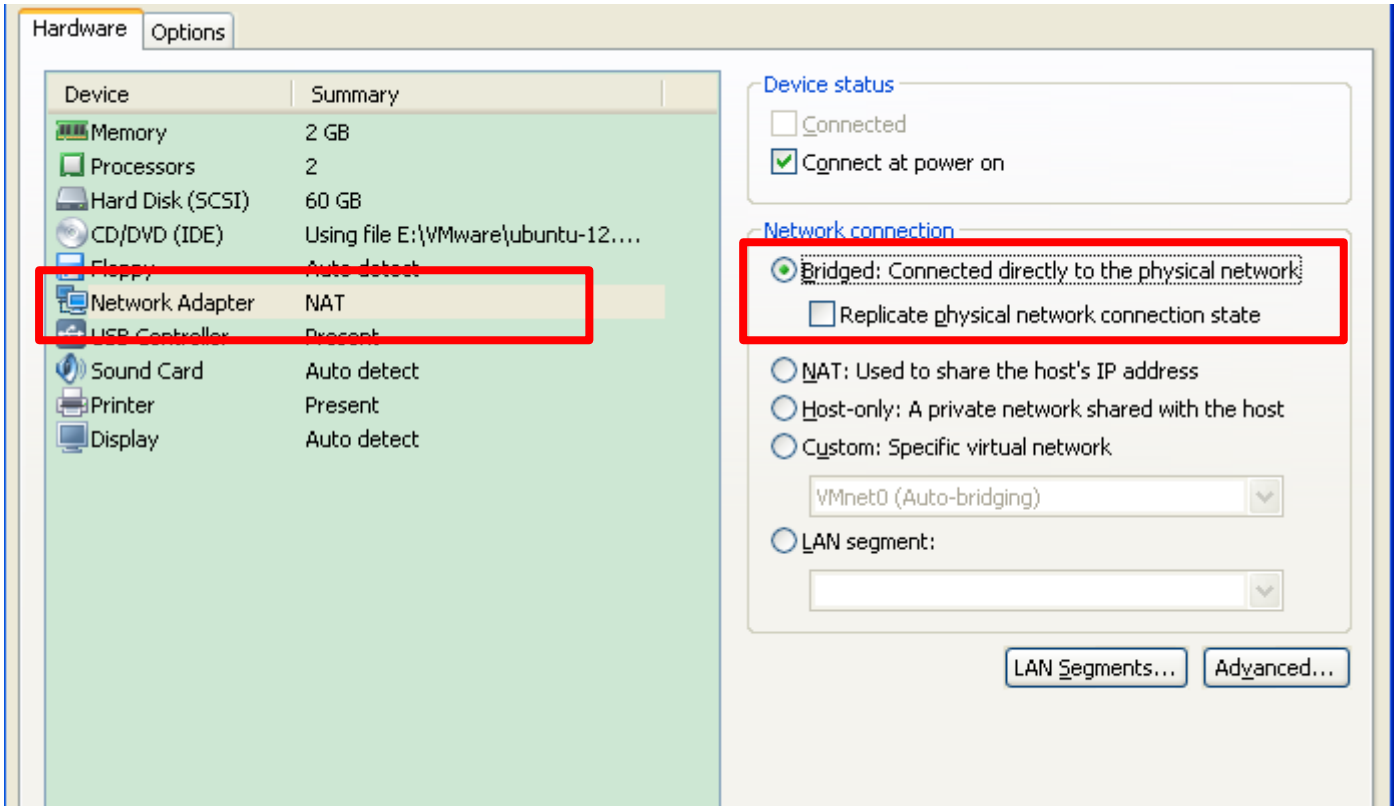


You'd better set the processor to 2.



8. select “CD/DVD(IDE)” item on the hardware tab, afterwards choose the path to the ubuntu-14.04.5-desktop-amd64.iso image file in the right side bar, and then select “Network Adapter” item to choose bridged network connection for the convenience of virtual machines and host access, just as followings. Click on the “ok” button to complete this configuration.

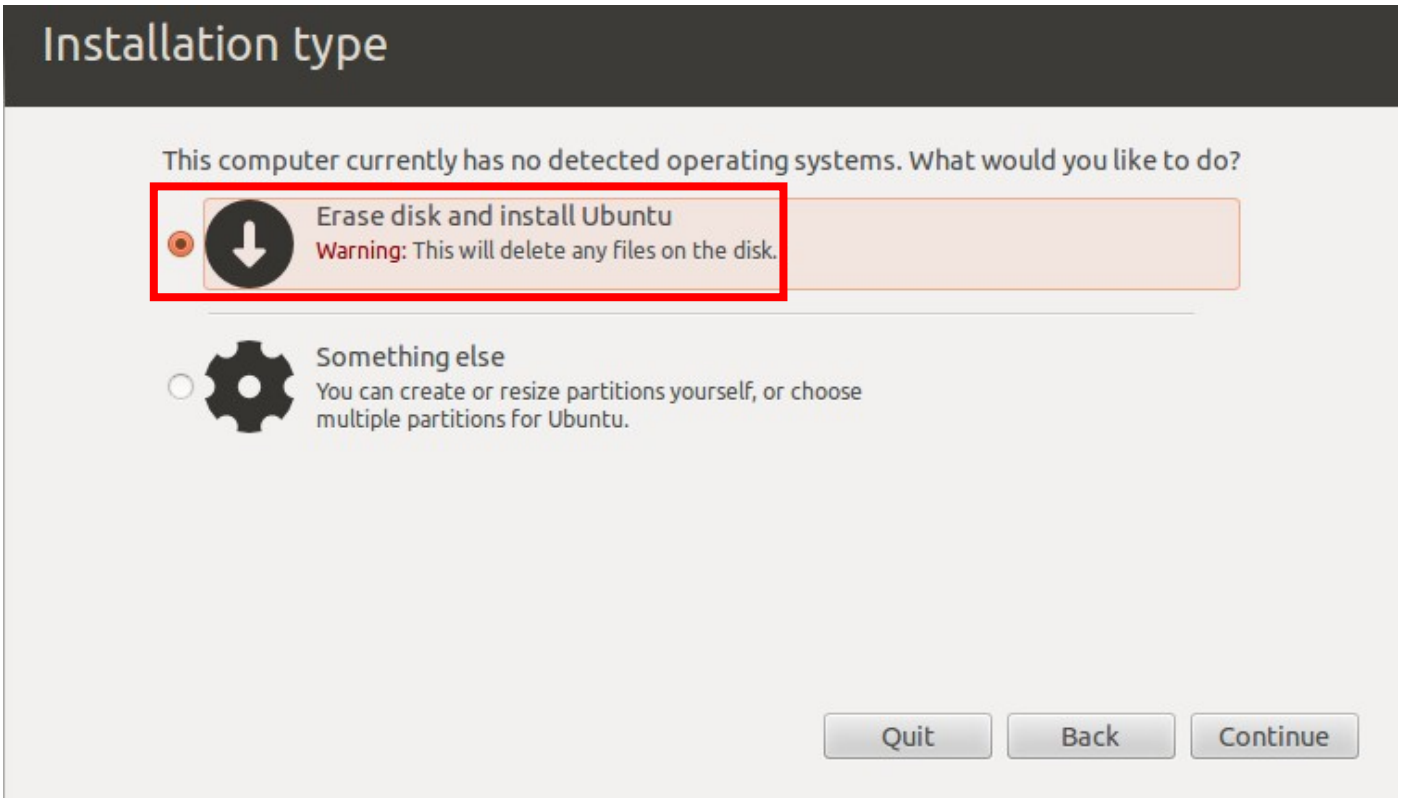

Device
Summary
Memory
2 GB
Processors
2
Hard Disk (SCSI)
930 GB
CD/DVD (IDE)
Auto detect
Floppy
Auto detect
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Device status
Connected
Connect at power on
Connection
Use physical drive:
Auto detect
Use ISO image file:
ubuntu-14.04.5-desktop-amd64
Browse...
Advanced...



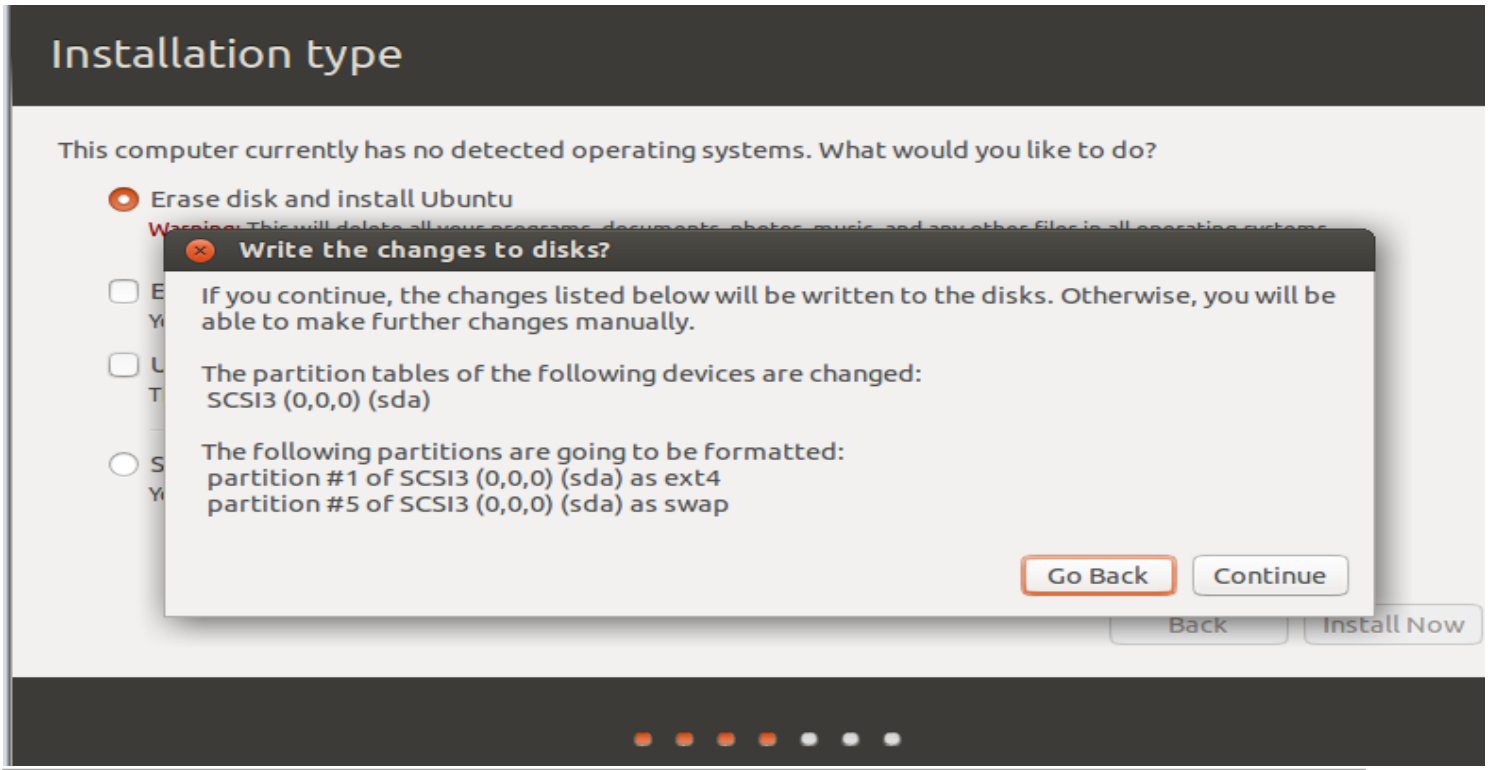
9. Click on “Power on this virtual machine” to start to install ubuntu operating system. Defaultselect English

language and press “Intall Ubuntu” to install the ubuntu operating system.

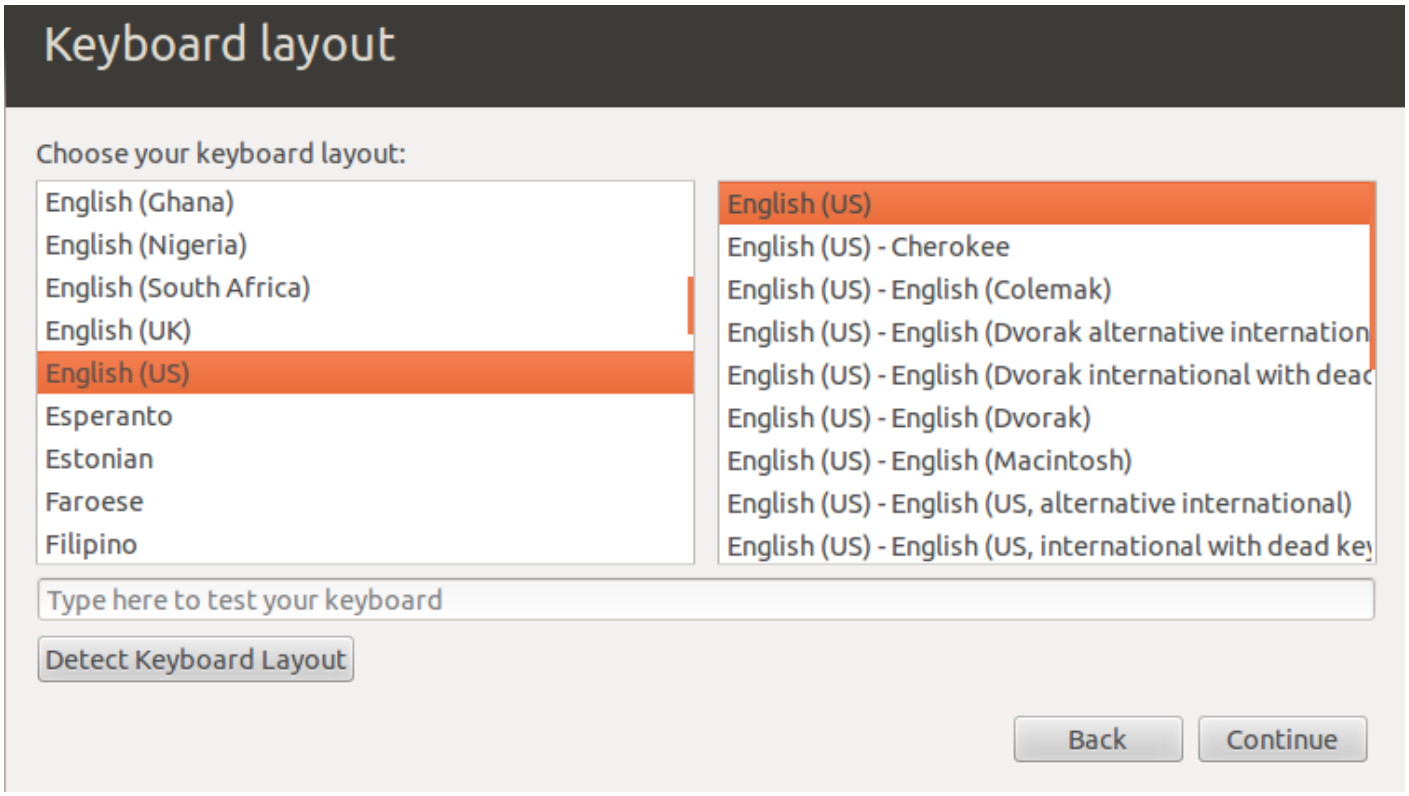
10. Click on "Continue" button to prepare to install Ubuntu. And prepare disk space, default select "Erase disk and install Ubuntu" item, and then click on the "Continue" button. Select the default driver and click on "Install



Now" button to start to install the Ubuntu.



11. Choose a positon where you are, and click on “Continue” button. Select the default keyboard layout, just like below figures.



12. After that set your user name and password, click on the “Continue” button, and then pop up the installing interface. Please wait a moment for this process.

13. After installation, click on “Restart Now” button to restart into the Ubuntu login screen with the user name and password you set before.

B、installing JDK1.8 on Ubuntu 64-bit system

1. At first, you can press“Ctrl+Alt+T” to open a Terminal on Ubuntu virtual machine.
2. Open a Terminal on Ubuntu virtual machine, and then search the jdk version.

   $ apt-cache search jdk

```
jolie@ubuntu:~$ apt-cache search jdk
default-jdk - Standard Java or Java compatible Development Kit
default-jdk-doc - Standard Java or Java compatible Development Kit (documentation)
default-jre - Standard Java or Java compatible Runtime
default-jre-headless - Standard Java or Java compatible Runtime (headless)
gcj-4.8-jdk - GCJ and Classpath development tools for Java(TM)
gcj-jdk - gcj and Classpath development tools for Java(TM)
gcj-native-helper - Standard helper tools for creating gcj native packages
icedtea-7-jre-jamvm - Alternative JVM for OpenJDK, using JamVM
icedtea-7-plugin - web browser plugin based on OpenJDK and IcedTea to execute Java applets
libcommons-collections3-java - Apache Commons Collections - Extended Collections API for Java
libcommons-lang-java - Extension of the java.lang package
libcommons-lang-java-doc - Documentation for an extension of the java.lang package
mauve - free test suite for the Java Class libraries
openjdk-7-dbg - Java runtime based on OpenJDK (debugging symbols)
openjdk-7-demo - Java runtime based on OpenJDK (demos and examples)
openjdk-7-doc - OpenJDK Development Kit (JDK) documentation
openjdk-7-jdk - OpenJDK Development Kit (JDK)
openjdk-7-source - OpenJDK Development Kit (JDK) source files
java-package - Utility for creating Java Debian packages
libcolt-java - scalable scientific and technical computing in Java
libcolt-java-doc - scalable scientific and technical computing in Java (doc)
fakeroot-ng - Gives a fake root environment
freemind - Java Program for creating and viewing Mindmaps
icedtea-6-jre-cacao - Alternative JVM for OpenJDK, using Cacao
icedtea-6-jre-jamvm - Alternative JVM for OpenJDK, using JamVM
icedtea-6-plugin - web browser plugin based on OpenJDK and IcedTea to execute Java applets
japitools - Java API compatibility testing tools
java3ds-fileloader - Java3D 3DS File Loader
jtreg - Regression Test Harness for the OpenJDK platform
jvm-7-avian-jre - lightweight virtual machine using the OpenJDK class library
```

3. Install the JDK 1.8 by the following command .

   $ sudo apt-get install openjdk-8-jdk

If this installation fails and report the following error, you will need to manually download JDK 8 to install

E: Unable to locate package openjdk-8-jdk

The steps:

1. Download openjdk-8-jdk by links to http://www.geniatech.net/down-eng/tools/openjdk-1.8.tar.bz2;
2. Copy openjdk-1.8.tar.bz2 into your virtual machine
3. Unzip openjdk-1.8.tar.bz2;

   $ tar -jxvf openjdk-1.8.tar.bz2
4. Configure environment variables by vi editor;

   $ sudo vi /etc/profile

   At the end of the file, add the following:

```
#Java Env
export JAVA_HOME=/usr/openjdk-1.8.0-internal
export CLASSPATH=.:$.JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bi
NOTE:JAVA_HOME=/usr/openjdk-1.8.0-internal ------is the path where you put openjdk-1.8.0-internal
```

5. When you're done editing, save and exit, and then type the following instructions to refresh the environment configuration to take effect

   $ source /etc/profile

```
fxn@fxn-virtual-machine:/$ sudo vi /etc/profile
fxn@fxn-virtual-machine:/$
fxn@fxn-virtual-machine:/$
fxn@fxn-virtual-machine:/$
fxn@fxn-virtual-machine:/$ source /etc/profile
fxn@fxn-virtual-machine:/$
```

4. Finally execute the following command to see the Java version that you just installed.

   $ java -version

```
fxn@fxn-virtual-machine:~$
fxn@fxn-virtual-machine:~$ java -version
openjdk version "1.8.0-internal"
OpenJDK Runtime Environment (build 1.8.0-internal-root_2016_06_30_10_55-b00)
OpenJDK 64-Bit Server VM (build 25.0-b70, mixed mode)
fxn@fxn-virtual-machine:~$
```

C、Install some necessary software packages and GNU tool chain.

1. Install some necessary software packages which will be used during compilation later. You can enter the following commands in the Terminal. During this process, enter "Y" to continue.

   $ sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl libc6-dev libncurses5-dev x11proto-core-dev libx11-dev libreadline6-dev libgl1-mesa-glx libgl1-mesa-dev g++-multilib mingw32

```
root@jolie-ubt:/home/jp# apt-get install git-core gnupg flex bison  gperf build-e
ssential zip curl libc6-dev libncurses5-dev x11proto-core-dev libx11-dev libreadl
ine6-dev libgl1-mesa-glx libgl1-mesa-dev g++-multilib mingw32 tofrodos python-mar
kdown libxml2-utils xsltproc zlib1g-dev
```

   tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev lib32z1 libxml2-utils lzop

2. Download the 64-bit GNU tool chain for kernel from the following URL.

http://www.geniatech.net/down-eng/tools/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf.tar.bz2 .

You need to unzip it to /opt/ ,an execute the following commands to make sure that you put it in the right place.

$ls -al /opt/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc-6.3.1

```
feng@feng-ThinkPad-Edge-E431:~$ ls -al /opt/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc-6.3.1
-rwxr-xr-x 1 11827 9000 907504 2月  16  2017 /opt/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc-6.3.1
feng@feng-ThinkPad-Edge-E431:~$
```

3. Change to the directory where the file you just downloaded, remove the .zip of the file name directly, don't need to unzip. And then unpack the aarch64-linux-gnu.tar.xz file. And then configure the environment variables as below command.

   $ xz -d aarch64-linux-gnu.tar.xz

   $tar -xvf aarch64-linux-gnu.tar

   $ export PATH=/opt/aarch64-linux-gnu/bin:$PATH

4. If it prompts that it cannot find the libGL.so.1 library when you build the source code, then you can execute the following command to create a soft link.

   If you install the 64-bit ubuntu server system, please use this command:

$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so If you install the 64-bit ubuntu system, please use this command:

$ sudo ln -s /usr/lib/x86_64-linux-gnu/mesa/libGL.so.1 /usr/lib/x86_64-linux-gnu/libGL.so

No.2、SDK Catalogue

Android root directory：

Android source code root directory	Description
art	ART running environment
bionic	System C library
bootable	Contains the code for the recovery program.
bootloader	uboot
build	Android Makefile
common	Linux kernel
cts	Android compatibility testing suite standard
dalvik	Dalvik machine
developers	Developer directory
development	Appliaction development relation
device	Device related configuration
external	Open source module related files
frameworks	Application frameworks, Android system kernel is edited by Jave and C++
hardware	Hardware abstract level code
kernel	Android kernel
libcore	Kernel library files
libnativehelper	Dynamic library is JNI library's base.
packages	Application package
pdk	Plug Development Kit abbreviation, local development suite.

platform_testing	Platform testing
prebuilts	Pre-edit some resource in X86 and ARM
sdk	SDK and simulated machine.
system	bottom file system library ,application and components.
test	test
toolchain	Tool chain files
tools	Tool files
vendor	Manufacturer customization code
out	Compile output directory

1.Uboot directory ：

bootloader/uboot-repo/

BL2: Boot Loader 2 which is the first external software loaded and executed by the SoC

BL30: SCP firmware

BL31: ARM Trusted Firmware

BL32: TEE or Secure OS

BL33:Loader responsible to load kernel into DDR to execute. In amlogic reference software, BL33 will be U-boot

2.Linux kernel directory structure：

common/

├── arch

　　├── arm64/boot/dts/amlogic //hardware DTS directory structure

├── block

├── certs

├── crypto

├── Documentation

├── drivers

```
        ├── amlogic //Amlogic SOC various module drivers
├── firmware
├── fs
├── include
├── init
├── ipc
├── kernel
├── lib
├── mm
├── net
├── samples
├── scripts
├── security
├── sound
        ├── soc/amlogic //Amlogic audio driver
├── tools
├── usr
└── virt
```

3. Hardware directory structure

hardware/amlogic/

```
├── audio //Amlogic audio HAL
├── bluetooth
├── bootctrl
├── camera //Amlogic camera HAL
├── consumerir
├── gatekeeper
├── gralloc
├── hdmi_cec
```

├── hwcomposer
├── keymaster
├── LibAudio //audio player and decode library
├── lights
├── media
├── media_modules //decoder driver
├── memtrack
├── power
├── screen_source
├── tb_modules
├── thermal
├── tv_input
└── wifi

4. Vendor directory structure

vendor/amlogic/common/

├── aml_mp_sdk //tsplayer interface code
├── apps //Amlogic APK source code
    ├── AppInstaller
    ├── DLNA
    ├── DroidLiveTv
    ├── DroidTvSettings
    ├── FileBrowser
    ├── IrBlaster
    ├── MboxLauncher2
    ├── Miracast
    ├── NativeImagePlayer
    ├── OTAUpgrade2
    ├── PPPoE
    ├── RemoteIME

```
        ├── SubTitle
        ├── TvInput
        ├── VendorOverlay
        └── VideoPlayer
├── arm_isp
├── broadcom
├── external
        ├── busybox
        ├── DTVKit //DTVKit package
        ├── dvb
        ├── exfat //exfat open source package
        ├── ffmpeg //ffmpeg open source package
        ├── icu
        ├── libaudioeffect
        ├── libtiff
        ├── libzvbi
        ├── ntfs-3g //ntfs open source package
        ├── pppoe
        ├── recovery //Amlogic recovery extension
├── frameworks
        ├── av
                ├── libaudioeffect
                ├── Libstagefright
                ├── mediaextconfig
        ├── core //Amlogic java API support package
        ├── pppoe
        ├──  services
                ├── data_integrity_guard
                ├── dv_config
```

├── fbc_tool
├── hdmicec //CEC extension
├── Imageplayer
├── miracast_hdcp2
├── remotecontrol
├── screencontrol //record and catch screen
├── subtiltleserver
├── systemcontrol // HDMI hotplug and HDCP certification
├── zygote_proxy ├── gms
├── gpu
├── gpu-lib
├── hdcp //hdcp provision key to get program
├── interfaces
├── ir_tools
├── LibTsPlayer
├── npu
├── prebuilt
├── accelerateboot
├── exoplayer
├── LeanbackCustomizer
libkeystonecorrection
├── libmedia
├── libmediadrm //playready Widevine library
├── libstagefrighthw //OMX library
├── livetv
├── multiwifi //multi wifi automatic identification library
└── videofirm //decoder library
├── provision //provision to write key module
├── scripts

├── system

├── tdk //SOC secureos package

├── tools //tools for compiling

├── tv //TV input service

No.3、How to build Code

1.You can find corresponding Android launch choice and kernel config for the reference hardware by doing the following:

(1) Bootloader

cd bootloader/uboot-repo

./mk g12a_u212_v1 --systemroot

Copy the uboot to platform dir:

cp build/u-boot.bin ../../device/amlogic/franklin/bootloader.img;

cp build/u-boot.bin.usb.bl2 ../../device/amlogic/franklin/upgrade/u-boot.bin.usb.bl2;

cp build/u-boot.bin.usb.tpl ../../device/amlogic/franklin/upgrade/u-boot.bin.usb.tpl;

cp build/u-boot.bin.sd.bin ../../device/amlogic/franklin/upgrade/u-boot.bin.sd.bin;

(bootloader/uboot-repo/build/u-boot.bin for EMMC, bootloader/uboot-repo/build/u-boot.bin.sd.bin for making bootcard)

DRM Version:

./mk g12a_u212_v1 --bl32 ../../vendor/amlogic/common/tdk/secureos/g12a/bl32.img --systemroot

(2) Android

. build/envsetup.sh

lunch franklin-userdebug

make otapackage

2. How to Upgrade

There are 4 ways for update.

Upgrade with SD card

Follow instructions below *ONLY IF* your device is already running a version of Openlinux based Android 8.0 reference firmware release.

4 steps:

1) Copy recovery.img, update zip file (e.g. ampere-ota-20190131.zip) to SD card. 2) Copy the factory_update_param.aml to SD card;

3) Power off.

4) Press VOL- and POWER key at same time, then it will upgrade the code. □

Upgrade with USB burn tool □

Update with OTA

Update with fastboot

Below is the way of burning each partitions:

Partition name	make command	imgae name	burning ways
all partitions	make otapackage	aml_upgrade_package.img frankline-ota-*.zip	by OTA upgrade by USB burning
boot	make bootimage	boot.img	In uboot cmdline: U disk: usb_update boot boot.img SD card: dc_update boot boot.img
logo	make logoimg	logo.img	usb_update logo logo.img sdc_update logo logo.img
recovery	make recoveryimage	recovery.img	usb_update recovery recovery.img sdc_update recovery recovery.img
system	make systemimage	system.img	usb_update system system.img sdc_update system system.img
dtb	make dtbimage	dt.img	usb_update _aml_dtb dt.img
uboot	./mk	uboot.bin	usb_update bootloader u-boot.bin

	g12a_u212_v1 --systemroot		sdc_update bootloader u-boot.bin
vendor	make vendorimage	vendor.img	usb_update vendor vendor.img sdc_update vendor vendor.img
odm	make odm_image	odm.img	usb_update odm odm.img sdc_update odm odm.img

1、GPIO

GPIO mapping

GPIOs should be mapped to a specific device in DTS before it can be used. Here is an example below:

```
foo_device {
    compatible = "acme,foo";
     ...
    led-gpios = <&gpio 15 GPIO_ACTIVE_HIGH>, /* red */
        <&gpio 16 GPIO_ACTIVE_HIGH>, /* green */
        <&gpio 17 GPIO_ACTIVE_HIGH>; /* blue */

    power-gpios = <&gpio 1 GPIO_ACTIVE_LOW>;
};
```

GPIO can be defined in node in two ways: xxx-gpio and xxx-gpios, and it has three data fields, explained further below:

&gpio: refer to gpio controller node.

GPIO_ACTIVE_LOW: flags of this GPIO. For more details, please refer to Reference.

Note: &gpio can also be gpio_ao when using AO domain GPIO, see also chapter Overview.

Get GPIO resource

In driver code, use following APIs to get GPIO resources defined in DTS. The returned value of those APIs is a pointer to gpio_desc structure. More API usages can be found in Reference.

```
#include <linux/gpio/consumer.h>

struct gpio_desc *gpiod_get(struct device *dev, const char *con_id,
                            enum gpiod_flags flags)

struct gpio_desc *gpiod_get_index(struct device *dev, const char *con_id,
                                  unsigned int idx, enum gpiod_flags flags)

struct gpio_desc *gpiod_get_optional(struct device *dev, const char *con_id,
                                     enum gpiod_flags flags)

struct gpio_desc *gpiod_get_index_optional(struct device *dev, const char *con_id,
                           unsigned int index, enum gpiod_flags flags)

struct gpio_descs *gpiod_get_array(struct device *dev, const char *con_id,
                                   enum gpiod_flags flags)

struct gpio_descs *gpiod_get_array_optional(struct device *dev, const char *con_id, enum gpiod_flags
                                            flags)
```

Example:

```
struct gpio_desc *red;
struct gpio_desc *green;
struct gpio_desc *power;

red = gpiod_get_index(dev, "led", 0, GPIOD_OUT_HIGH);
green = gpiod_get_index(dev, "led", 1, GPIOD_OUT_HIGH);
power = gpiod_get(dev, "power", GPIOD_OUT_HIGH);
```


Room 02-04, 10/F, Block A, Building 8, Shenzhen International Innovation Valley, Dashi Road, Nanshan District, Shenzhen, Guangdong, China
Emai: support@geniatech.com Tel: (+ 86) 755 86028588

GPIO direction can be set as input or output. For output, GPIO can be set as high level or low level.APIs are listed below:

```
/*direction*/
int gpiod_get_direction(const struct gpio_desc *desc)
int gpiod_direction_input(struct gpio_desc *desc)
int gpiod_direction_output(struct gpio_desc *desc, int value)

int gpiod_get_value(const struct gpio_desc *desc);
void gpiod_set_value(struct gpio_desc *desc, int value);

int gpiod_get_raw_value(const struct gpio_desc *desc)
void gpiod_set_raw_value(struct gpio_desc *desc, int value)
int gpiod_get_raw_value_cansleep(const struct gpio_desc *desc)
void gpiod_set_raw_value_cansleep(struct gpio_desc *desc, int value)
int gpiod_direction_output_raw(struct gpio_desc *desc, int value)
```

Example:

```
/* set red gpio high */

gpiod_direction_output(red, 1);
```

Put GPIO

When a GPIO is not used anymore, there are several APIs which can be used to release the GPIO resources.

```
void gpiod_put(struct gpio_desc *desc)

void gpiod_put_array(struct gpio_descs *descs)

void devm_gpiod_put(struct device *dev, struct gpio_desc *desc)

void devm_gpiod_put_array(struct device *dev, struct gpio_descs *descs)
```

2. Sysfs for GPIO

GPIO can be accessed in user space via sysfs. It takes two steps to access a GPIO:

1)Export GPIO

2)Set/Get GPIO direction and set GPIO output value or get GPIO input value

To export a GPIO via sysfs, for kernel 4.9, the GPIO ID needs to be got first. On Amlogic platform, a GPIO ID is equal to BASE + OFFSET, where BASE and OFFSET are integers. BASE is dependent on

GPIO controller which can be found under /sys/class/gpio. Example for board S905X2:

```
ls sys/class/gpio/

export gpiochip410 gpiochip496 unexport
```

There are two GPIO controllers above, their BASEes are 410 and 496 respectively.

```
cat sys/class/gpio/gpiochip410/label

cat sys/class/gpio/gpiochip496/label
```

Check the output on the command line, if the label is "aobus-bank", it is an AO domain GPIO controller, and the GPIO names are like GPIOAO_[N], otherwise it is an EE Domain GPIO controller.

Choose the correct base number for a GPIO in this step. Once BASE is settled, OFFSET can be found in the following file:

/include/dt-bindings/gpio/meson-g12a-gpio.h

2、I2C

I2C bus DTS setting，as like S905X2：

common/arch/arm64/boot/dts/amlogic/mesong12a.dtsi

```
i2c0: i2c@1f000 {
        compatible = "amlogic,meson-g12a-i2c";
        status = "disabled";
        reg = <0x0 0x1f000 0x0 0x20>;
        interrupts = <GIC_SPI 21 IRQ_TYPE_EDGE_RISING>, <GIC_SPI 91
IRQ_TYPE_EDGE_RISING>;
        #address-cells = <1>;
        #size-cells = <0>; clocks = <&clkc CLKID_I2C>;
        clock-names = "clk_i2c"; };

i2c1: i2c@1e000 {
        compatible = "amlogic,meson-g12a-i2c";
        status = "disabled";
         reg = <0x0 0x1e000 0x0 0x20>;
        interrupts = <GIC_SPI 214 IRQ_TYPE_EDGE_RISING>, <GIC_SPI 92
IRQ_TYPE_EDGE_RISING>;
        #address-cells = <1>;
```

Supports 5 I2C controler ， I2C bus configure：

interrupts = <GIC_SPI 21 IRQ_TYPE_EDGE_RISING>; // Interrupt related Settings

clocks = <&clkc CLKID_I2C>; //I2C clock、

status = "disabled"; //I2C initial switching state of the bus

S905x2 producr I2C Configure and bus devices

common/arch/arm64/boot/dts/amlogic/g12a_s905x2_u212.dts

```
aliases {
        serial0 = &uart_AO;
        serial1 = &uart_A;
        serial2 = &uart_B;
        serial3 = &uart_C;
        serial4 = &uart_AO_B;
        tsensor0 = &p_tsensor;
        tsensor1 = &d_tsensor;
        i2c0 = &i2c0;
        i2c1 = &i2c1;
        i2c2 = &i2c2;
        i2c3 = &i2c3;
        i2c4 = &i2c_AO;
 };
```

- clock-frequency

frequency of bus clock in Hz.

- i2c-bus

For I2C adapters that have child nodes that are a mixture of both I2C devices and non-I2C devices, the 'i2c-bus' subnode can be used for populating I2C devices. If the 'i2c-bus' subnode is present, only subnodes of this will be considered as I2C slaves. The properties, '#address-cells' and '#size-cells' must be defined under this subnode if present.

3、UART

The Amlogic Meson SoC UART Serial Interface is present on a large range of SoCs, and can be present either in the "Always-On" power domain or the "Everything-Else" power domain.

The particularity of the "Always-On" Serial Interface is that the hardware is active since power-on and

does not need any clock gating and is usable as very early serial console.

required:

- compatible
- reg
- interrupts
- clocks

```
Example: UART controller node that consumes the clock and reset generated
   by the clock controller
uart_AO: serial@4c0 {
            compatible = "amlogic,meson-uart";
            reg = <0x4c0 0x14>;
            interrupts = <0 90 1>;
            clocks = <&clkc_AO CLKID_AO_UART1>;
            resets = <&clkc_AO RESET_AO_UART1>;
            status = "disabled";
};
```

4、SPI

SPI busses can be described with a node for the SPI master device and a set of child nodes for each SPI slave on the bus. For this discussion, it is assumed that the system's SPI controller is in SPI master mode. This binding does not describe SPI controllers in slave mode.

The SPI master node requires the following properties:

- #address-cells - number of cells required to define a chip select address on the SPI bus.
- #size-cells - should be zero.
- compatible - name of SPI bus controller following generic names recommended practice.

No other properties are required in the SPI bus node. It is assumed that a driver for an SPI bus device will understand that it is an SPI bus. However, the binding does not attempt to define the specific method for assigning chip select numbers. Since SPI chip select configuration is

flexible and non-standardized, it is left out of this binding with the assumption that board specific platform code will be used to manage chip selects. Individual drivers can define additional properties to support describing the chip select layout.

SPI example for an MPC5200 SPI bus:

```
spi@f00 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,mpc5200b-spi","fsl,mpc5200-spi";
        reg = <0xf00 0x20>;
        interrupts = <2 13 0 2 14 0>;
        interrupt-parent = <&mpc5200_pic>;

        ethernet-switch@0 {
            compatible = "micrel,ks8995m";
            spi-max-frequency = <1000000>;
            reg = <0>;
        };

codec@1 {
        compatible = "ti,tlv320aic26";
        spi-max-frequency = <100000>;
        reg = <1>;
        };
};
```