

《嵌入式系统》

（第五讲）

厦门大学信息学院软件工程系 曾文华

2024年10月8日

- 第1章：嵌入式系统概述
- 第2章：ARM处理器和指令集
- 第3章：嵌入式Linux操作系统
- 第4章：嵌入式软件编程技术
- 第5章：开发环境和调试技术
- 第6章：Boot Loader技术
- 第7章：ARM Linux内核
- 第8章：文件系统
- 第9章：设备驱动程序设计基础
- 第10章：字符设备和驱动程序设计
- 第11章：块设备和驱动程序设计
- 第12章：网络设备驱动程序开发
- 第13章：嵌入式GUI及应用程序设计
- 第14章：Android操作系统（增加）



第5章 开发环境和调试技术

- 5.1 交叉开发模式概述
- 5.2 宿主机环境
- 5.3 目标板环境
- 5.4 交叉编译工具链
- 5.5 本地调试 (gdb)
- 5.6 远程调试 (gdb + gdbserver)
- 5.7 内核调试 (gdb + kgdb)
- 5.8 网络调试

5.1 交叉开发模式概述

- 交叉开发模式：宿主机（PC机：VMware下的Ubuntu）-目标板（FS3399M4实验箱：超级终端Xshell）
- GNU软件：
 - ① **Shell**：Shell基本上是一个命令解释器，类似于DOS下的command
 - ② **glibc**：glibc是GNU发布的libc库，即c运行库
 - ③ **GCC**：GCC（GNU Compiler Collection，GNU编译器套件）是由GNU开发的编程语言编译器
 - ④ **gdb**：UNIX及UNIX-like下的调试工具
 - ⑤ **vim**：vim是一个类似于vi的著名的功能强大、高度可定制的文本编辑器，在vi的基础上改进和增加了很多特性
 - ⑥ **Emacs**：Emacs是著名的集成开发环境和文本编辑器
- 宿主机与目标板的连接方式：
 - ① 串口（Debug_1， Debug_2）
 - ② 以太网接口（RJ45）
 - ③ USB接口
 - ④ JTAG接口（Joint Test Action Group）

MCU开发板



MPU开发板



实验箱上的JTAG接口

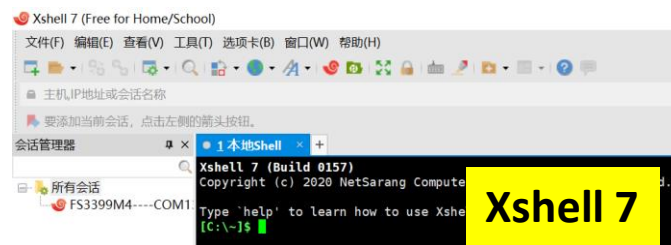
5.2 宿主机环境

5.2.1 串口终端
5.2.2 BOOTP
5.2.3 TFTP
5.2.4 交叉编译

• 5.2.1 串口终端

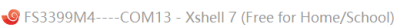
– Windows下的超级终端

- 超级终端是Windows自带的一个串口调试工具，其使用较为简单，被广泛使用在串口设备的初级调试上，如Xshell 7。



– Linux下的minicom

- minicom是一个串口通信工具，就像Windows下的超级终端。可用来与串口设备通信，如调试交换机和Modem等。它的Debian软件包的名称就叫minicom，用`apt-get install minicom`命令即可下载安装。如果宿主机是纯Linux环境，则需要使用minicom作为串口终端。



文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)

主机,IP地址或会话名称

要添加当前会话，点击左侧的箭头按钮。

1 FS3399M4---COM13

```
[ 28.871370] [rc.local] OK ] Started Getty on tty1.
[770]: [chmod: cannot access '/dev/cd_motor': No such file or directory OK
] Reached target Login Prompts.[ 28.884744]
rc.local[770]: [chmod: cannot access '/dev/stepper': No such file or directory OK
] Started Set console scheme.[ 28.897261]
rc.local[770]: [chmod: cannot access '/dev/servo': No such file or directory
[ 28.907873] rc.local[770]: [chmod: cannot access '/dev/rfid_module': No such file or directory
[ 28.916390] rc.local[770]: [chmod: cannot access '/dev/relay': No such file
[ 28.924357] rc.local[770]: [chmod: cannot access '/dev/zlg72xx': No such file
[ 28.932508] rc.local[770]: [chmod: cannot access '/dev/ttyUSB*': No such file
[ 28.940387] rc.local[770]: /home/linux/gt5Applications/mygt.sh: line 30: /s
关于 Xshell
```

```
Ubuntu 16.04.7 LTS localhost.localdomain ttyFI00
```

```
localhost login: linux
```

Password:

Last login: Thu Feb 11 16:28:45 UTC 2016 on ttyFIQ0

```
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.179 aarch64)
```

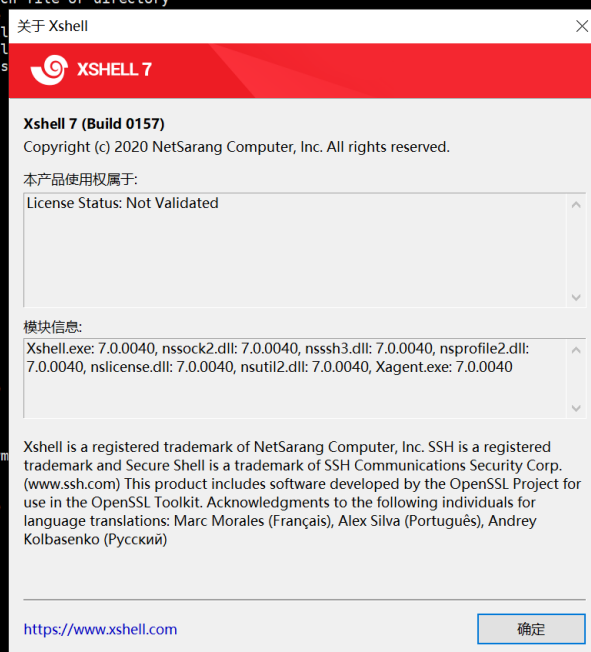
```
* Documentation: https://help.ubuntu.com
* Management:   https://landscape.canonical.com
* Support:      https://ubuntu.com/advantage
```

```
50 packages can be updated.
30 of these updates are security updates.
To see these additional updates run: apt list --upgradable
```

```
linux@localhost:~$ ls
Desktop      can_2_r      keys_buzzer      servo_driver.co
Documents    can_2_s      keys_buzzer_pwm  stepper_driver.co
Downloads    canread      keys_dcmotor      test.txt
FS3399_test canwrite     keys_gpio_leds    zlg72xx_adc
Music        dcmotor_driver.co keys_relay        zlg72xx_buzzer
Pictures     fairsight_keys.co keys_servo         zlg72xx_buzzer_pwm
Public       gpio_leds_driver.co keys_stepper       zlg72xx_clock
Templates    ir_buzzer     lm75_driver.co    zlg72xx_dcmotor
Video_MP4    ir_buzzer_pwm qt5applications   zlg72xx_driver.co
Videos       ir_dcmotor    relay_driver.co   zlg72xx_gpio_leds
adc_driver.co ir_gpio_leds   rs485_1           zlg72xx_lm75
buzzer_driver.co ir_relay       rs485_2_r         zlg72xx_relay
buzzer_pwm_driver.co ir_servo       rs485_2_s         zlg72xx_servo
can 1        ir_stepper     rs485_test        zlg72xx_stepper
```

[illegible]

Xshell-7.0.0108p.exe





PuTTY Configuration

PUTTY.EXE



Category:

- Session
 - Logging
- Terminal
 - Keyboard
 - Bell
 - Features
- Window
 - Appearance

COM21 - PuTTY

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)

Port

22

Connection type:

☐ Raw

☐ Telnet

☐ Rlogin

☒ SSH

☐ Serial

```
linux@localhost:~$ ls
Desktop          can_2_r          keys_buzzer      servo_driver.ko
Documents        can_2_s          keys_buzzer_pwm  stepper_driver.ko
Downloads        canread          keys_dcmotor     test.txt
FS3399_test      canwrite         keys_gpio_leds   zlg72xx_adc
Music            dcmotor_driver.ko keys_relay        zlg72xx_buzzer
Pictures         farsight_keys.ko keys_servo        zlg72xx_buzzer_pwm
Public           gpio_leds_driver.ko keys_stepper      zlg72xx_clock
Templates        ir_buzzer        lm75_driver.ko   zlg72xx_dcmotor
Video_MP4        ir_buzzer_pwm    qt5applications  zlg72xx_driver.ko
Videos           ir_dcmotor        relay_driver.ko   zlg72xx_gpio_leds
adc_driver.ko    ir_gpio_leds     rs485_1          zlg72xx_lm75
buzzer_driver.ko ir_relay          rs485_2_r        zlg72xx_relay
buzzer_pwm_driver.ko ir_servo          rs485_2_s        zlg72xx_servo
can_1            ir_stepper        rs485_test       zlg72xx_stepper
linux@localhost:~$
linux@localhost:~$
linux@localhost:~$
linux@localhost:~$
```

About

Open

Cancel

USB-SERIAL CH340

TerminalSessionsViewX serverToolsGamesSettingsMacrosHelp

SessionServersToolsGamesSessionsView

SplitMultiExecTunnelingPackagesSettingsHelp

Quick connect...

User sessions

192.168.137.2 (HwHIAIUser)

USB-SERIAL CH340

2 USB-SERIAL CH340

```
[759]: See 'systemctl status rc-local.service' for details.
chmod: cannot access '/sys/bus/iio/devices/iio:device0/in_accel_x_raw': No such file or directory
[ 28.722232] Starting Hold until boot process finishes up...rc.local
[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device0/in_accel_y_raw': No such file or directory
[ OK ] Started Hold until boot process finishes up.[ 28.738671] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device0/in_accel_z_raw': No such file or directory

[ 28.755135] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device0/sampling_frequency': No such file or directory[
OK [ 28.766797] ] rc.localStarted Serial Getty on ttyFIQ0.[759]:
chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_x_raw': No such file or directory
Starting Set console
rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_y_raw': No such file or directory
OK [ 28.797807] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_z_raw': No such file or directory
OK [ 28.812879] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_x_raw': No such file or directory
OK [ 28.825333] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_y_raw': No such file or directory
[ 28.837964] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_z_raw': No such file or directory
[ 28.846149] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_x_raw': No such file or directory
[ 28.854334] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_y_raw': No such file or directory
[ 28.862034] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_z_raw': No such file or directory
[ 28.870600] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_x_raw': No such file or directory
[ 28.878571] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_y_raw': No such file or directory
[ 28.886716] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_z_raw': No such file or directory
[ 28.894853] rc.local[759]: chmod: cannot access '/sys/bus/iio/devices/iio:device2/in_anglvel_x_raw': No such file or directory

Ubuntu 16.04.7 LTS

localhost login: li
Password:
Last login: Thu Feb 23 12:00:00 UTC 2018
Welcome to Ubuntu 16.04.7 LTS

 * Documentation:
 * Management:
 * Support:

50 packages can be updated.
30 of these updates are security updates.
To see these additional updates, type: apt-get update

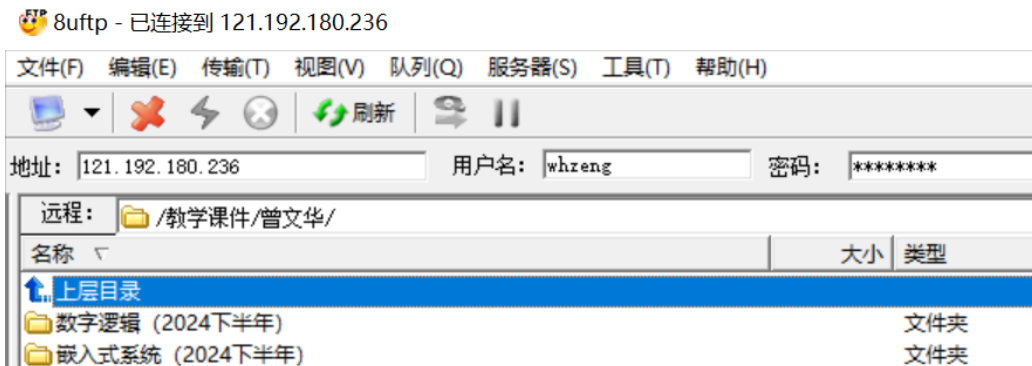
linux@localhost:~$
Desktop
Documents
Downloads
FS3399_test
Music
Pictures
Public
Templates
Video_MP4
Videos
adc_driver.ko
buzzer_driver.ko
buzzer_pwm_driver.ko
can_1
canread
canwrite
dcmotor_driver.ko
farsight_keys.ko
gpio_leds_driver.ko
ir_buzzer
ir_buzzer_pwm
ir_dcmotor
ir_gpio_leds
ir_relay
ir_servo
ir_stepper
keys_dcmotor
keys_gpio_leds
keys_relay
keys_servo
keys_stepper
lm75_driver.ko
qt5applications
relay_driver.ko
rs485_1
rs485_2_r
rs485_2_s
rs485_test
test.txt
zlg72xx_adc
zlg72xx_buzzer
zlg72xx_buzzer_pwm
zlg72xx_clock
zlg72xx_dcmotor
zlg72xx_driver.ko
zlg72xx_gpio_leds
zlg72xx_lm75
zlg72xx_relay
zlg72xx_servo
zlg72xx_stepper
```


• 5.2.2 BOOTP

- **BOOTP**（**Bootstrap Protocol**, **引导程序协议**）是一种引导协议，基于**IP/UDP**协议，也称自举协议，是**DHCP**协议的前身。**BOOTP**用于无盘工作站的局域网中，可以让无盘工作站从一个中心服务器上获得**IP**地址。通过**BOOTP**协议可以为局域网中的无盘工作站分配动态**IP**地址，这样就不需要管理员去为每个用户去设置静态**IP**地址。
- **BOOTP**的一般工作流程就是**BOOTP客户端**（目标板，实验箱）和**BOOTP服务器**（宿主机，PC机，Ubuntu）**之间的交互**，其流程如下：
 - ① 由**BOOTP**启动代码来启动**BOOTP**客户端，这个时候**BOOTP**客户端还没有**IP**地址。
 - ② **BOOTP**客户端使用广播形式的**IP**地址**255.255.255.255**向网络中发出**IP**地址查询要求。
 - ③ 运行**BOOTP**协议的服务器接收到这个请求，会根据请求中提供的**MAC**地址找到**BOOTP**客户端，并发送一个含有**IP**地址、服务器**IP**地址、网关等信息的回应帧。
 - ④ **BOOTP**客户端会根据该回应帧来获得自己的**IP**地址并通过专用文件服务器（如**TFTP**服务器）下载启动镜像文件，模拟成磁盘来完成启动。

• 5.2.3 TFTP

- TFTP（Trivial File Transfer Protocol，**简单文件传输协议**）是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。
- TFTP是**简化了的FTP**，TFTP没有用户权限管理的功能。
- FTP：File Transfer Protocol，**文件传输协议**



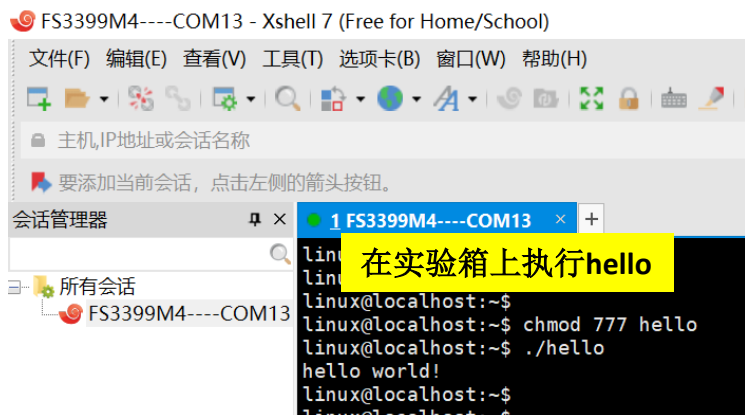
• 5.2.4 交叉编译

– **交叉编译**：在x86架构的**宿主机**（PC机，Ubuntu）上编译生成适用于ARM架构（FS3399M4**实验箱**）的ELF格式的可执行代码。

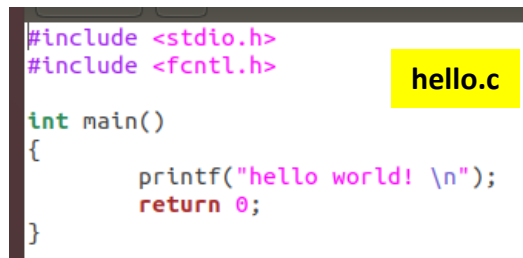
- 交叉编译：**aarch64-linux-gnu-gcc hello.c -o hello**

- 本地编译：**gcc hello.c -o hello_pc**

- **ELF**：Executable and Linkable Format，可执行与可链接格式，是一种用于二进制文件、可执行文件、目标代码、共享库和核心转储格式文件。



```
FS3399M4----COM13 - Xshell 7 (Free for Home/School)
文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)
主机, IP地址或会话名称
要添加当前会话, 点击左侧的箭头按钮。
会话管理器
1 FS3399M4----COM13 x +
lin
lin
在实验箱上执行hello
linux@localhost:~$
linux@localhost:~$ chmod 777 hello
linux@localhost:~$ ./hello
hello world!
linux@localhost:~$
```



```
#include <stdio.h>
#include <fcntl.h>

int main()
{
    printf("hello world! \n");
    return 0;
}
```



```
linux@linux-pc:~/workdir/fs3399/application/hello$
linux@linux-pc:~/workdir/fs3399/application/hello$ aarch64-linux-gnu-gcc hello.c -o hello
linux@linux-pc:~/workdir/fs3399/application/hello$
linux@linux-pc:~/workdir/fs3399/application/hello$ gcc hello.c -o hello_pc
linux@linux-pc:~/workdir/fs3399/application/hello$ ls
hello hello.c hello_pc
linux@linux-pc:~/workdir/fs3399/application/hello$
linux@linux-pc:~/workdir/fs3399/application/hello$ ./hello
hello world!
linux@linux-pc:~/workdir/fs3399/application/hello$
linux@linux-pc:~/workdir/fs3399/application/hello$ ./hello_pc
hello world!
linux@linux-pc:~/workdir/fs3399/application/hello$
linux@linux-pc:~/workdir/fs3399/application/hello$
在Ubuntu上执行hello_pc
```

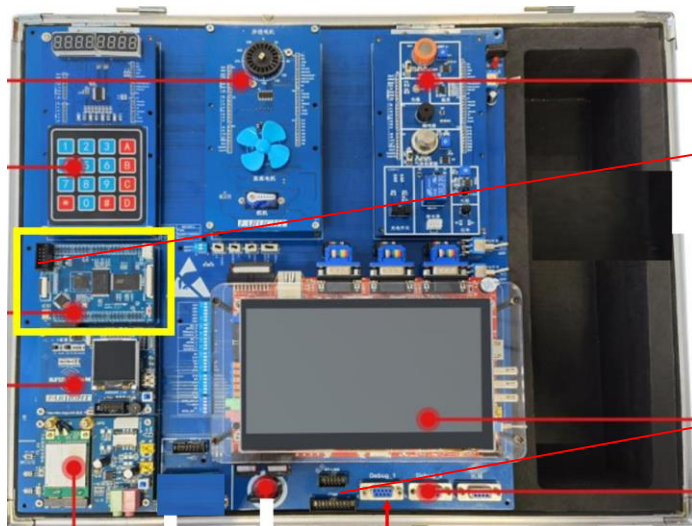
5.3 目标板环境

5.3.1 JTAG接口简介

5.3.2 Boot Loader简介

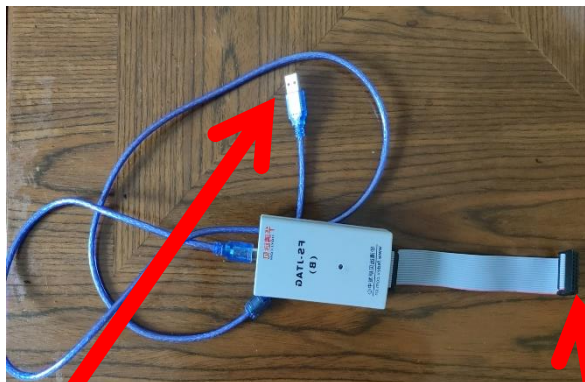
• 5.3.1 JTAG接口简介

- **JTAG**（Joint Test Action Group，联合测试工作组）是一种国际标准测试协议（IEEE 1149.1兼容），主要用于芯片内部测试。现在多数的高级器件都支持JTAG协议，如DSP、FPGA器件等。标准的JTAG接口是5线：**TMS**、**TCK**、**TDI**、**TDO**、**nTRST**，分别为模式选择、时钟、数据输入、数据输出线、系统复位。



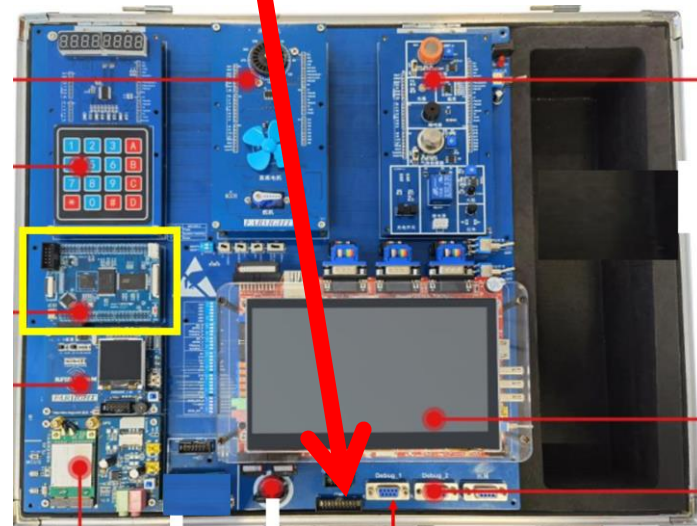
实验箱上的JTAG接口

实验箱MPU的JTAG 20

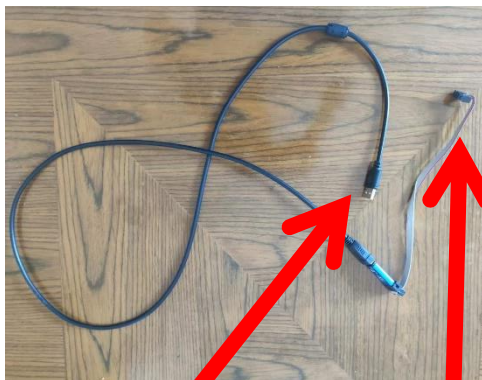


USB接口

20芯扁线

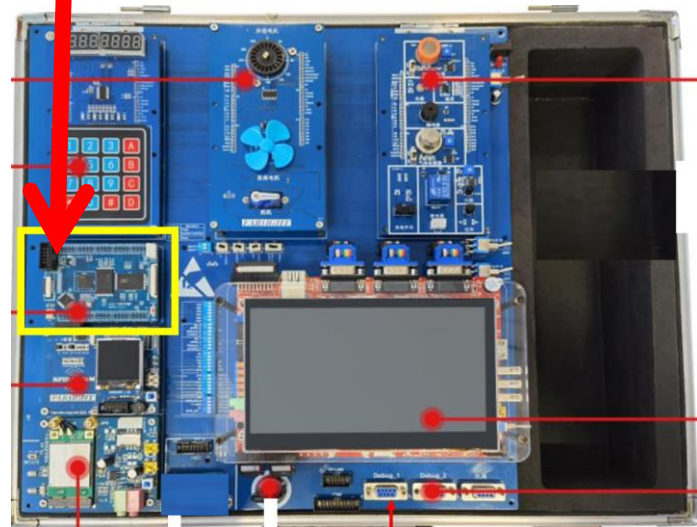


实验箱MCU的JTAG 10

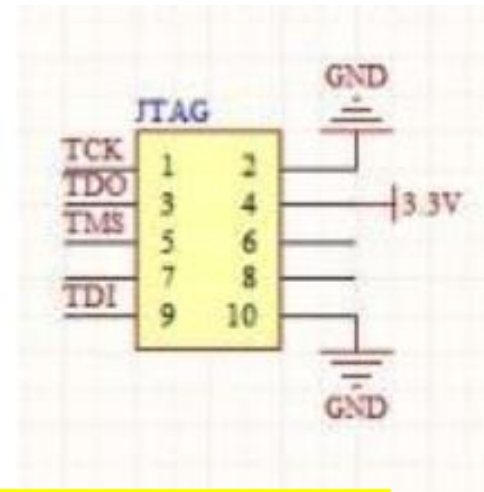
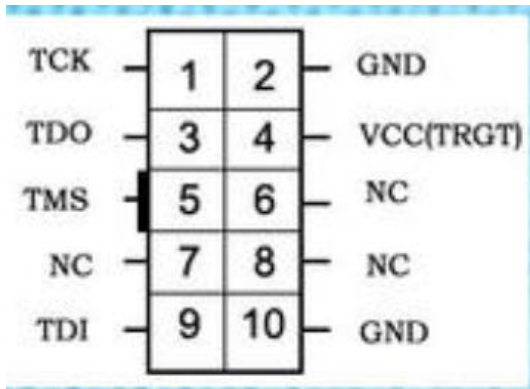


USB接口

10芯扁线



JTAG 10



TCK——测试时钟输入

TDI——测试数据输入，数据通过TDI输入JTAG口

TDO——测试数据输出，数据通过TDO从JTAG口输出

TMS——测试模式选择，TMS用来设置JTAG口处于某种特定的测试模式

NC——未用的管脚

VCC(TRGT)——电源（+5V）

GND——地线

• 5.3.2 Boot Loader简介

- 嵌入式Linux系统启动后，先执行**Bootloader**，进行硬件和内存的初始化工作，然后加载Linux内核和根文件系统映像，完成Linux系统的启动。
- **Bootloader**：**引导加载程序**，是嵌入式目标板（实验箱）加电后运行的第一段软件代码；是在操作系统内核运行之前用来初始化硬件设备、建立内存空间的映射图的小程序。

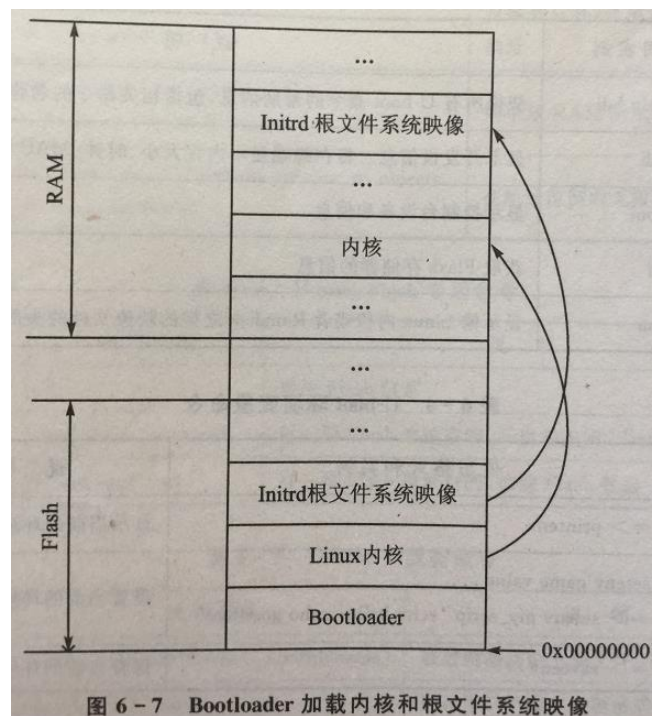


图 6 - 7 Bootloader 加载内核和根文件系统映像

Bootloader加载Linux内核和根文件系统映像

一 常见的Bootloader:

- **u-boot**: Universal Boot Loader, 是遵循GPL条款的开放源码项目, u-boot的作用是系统引导。
- **vivi**: 韩国Mizi公司开发的Bootloader引导程序。

FS3399M4实验箱开机后Xshell上显示的内容

```

linux@localhost:~$
linux@localhost:~$
linux@localhost:~$
linux@localhost:~$ 鑄滾ÿ 0.0.0.0 @NN^Nt 壘
U-Boot 2017.09 (Aug 31 2023 - 02:03:09 +0000)

Model: Rockchip RK3399 Evaluation Board
PreSerial: 2
DRAM: 2 GiB
Sysmem: init
Relocation Offset is: 7dbdf000
Using default environment

[debug] should_load_env()
[debug] env_load()
dwmmc@fe320000: 1, sdhci@fe330000: 0
Bootdev(atags): mmc 0
MMC0: HS400, 150Mhz
PartType: EFI
[debug]get env lcdtype:<NULL>
get part misc fail -1
boot mode: None
init_resource_list: Load resource from boot second pos
Load FDT from boot part
*** Warning use default panel ***
**** fs3399 linux M4 mipi070 **** u-boot ****

```

The screenshot shows a terminal window titled "ARM-Xshell-2.0". The address bar indicates the font is "Courier New" and size is "16". The terminal output shows the following sequence:

```
vivi>  
VIVI version 0.1.4 (root@BC) (gcc version 2.95.2 20000516 (release) [Rebel.com]) #0.1.4 四 5月 26 09:44:53  
ST 2005  
MMU table base address = 0x33DFC000  
Succeed memory mapping.  
NAND device: Manufacture ID: 0xec, Chip ID: 0x76 (Samsung K9D1208V0M)  
Found saved vivi parameters.  
Press Return to start the LINUX now, any other key for vivi  
type "help" for help.  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>  
vivi>
```

A large yellow rectangular box is overlaid on the right side of the terminal output, containing the Chinese text: "采用vivi引导程序的实验箱开机后Xshell上显示的内容"

采用vivi引导程序的实验箱 开机后Xshell上显示的内容

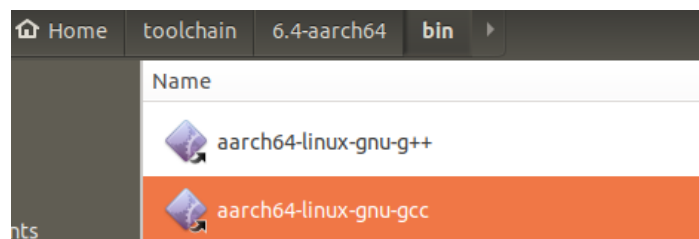
vivi

5.4 交叉编译工具链

5.4.1 交叉编译的构建

5.4.2 相关工具

- **本地编译**：在Ubuntu上，编译生成PC平台运行的程序。
 - `gcc hello.c -o hello_pc`
- **交叉编译**：在Ubuntu上，编译生成目标机（ARM，实验箱）平台运行的程序。
 - `aarch64-linux-gnu-gcc hello.c -o hello`
- **交叉编译工具链**：是一个由编译器、链接器、解释器组成的集成开发环境。



• 5.4.1 交叉编译的构建

- 交叉编译工具链是一个由标准库、编译器、链接器、汇编器、调试器组成的集成开发环境。
- ARM平台的交叉编译工具：**aarch64-linux-gnu-gcc**
- 制作交叉编译工具链的方法：
 - ① 从头编译
 - ② 脚本编译
 - ③ 下载使用

• 5.4.2 相关工具

– 1、glibc（标准库）

- glibc是GNU发布的libc库，即**C运行库**（GNU C Library）
- glibc是Linux系统中最底层的API

– 2、gcc（编译器）

- gcc（GNU Compiler Collection，**GNU编译器套件**），是由GNU开发的编程语言编译器
- gcc编译过程的4个阶段：
 - ① 预处理：生成“*.i”文件
 - ② 汇编：用as命令，编译源文件，生成汇编文件（“*.s”文件）
 - ③ 编译：用cc命令，生成目标文件（“*.o”文件）
 - ④ 链接：用ld命令，生成可执行文件

① 预处理 (-E) :

- `gcc -E -o hello.i hello.c`
- 生成 “**hello.i**” 文件

② 汇编 (-S) :

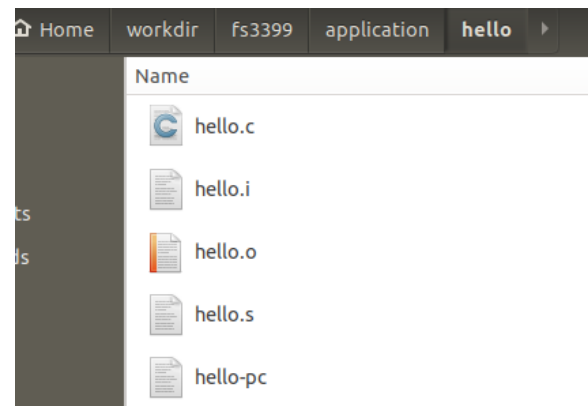
- `gcc -S -o hello.s hello.i`
- 生成 “**hello.s**” 文件（汇编文件，即汇编语言程序）

③ 编译 (-c) :

- `gcc -c -o hello.o hello.s`
- 生成 “**hello.o**” 文件（目标文件）

④ 链接 (-o) :

- `gcc -o hello-pc hello.o`
- 生成 “**hello-pc**” 文件（可执行文件）



– 执行可执行文件:

- `./hello-pc`
 - hello world!

```
linux@linux-pc:~/workdir/fs3399/application/hello$ ./hello-pc
hello world!
linux@linux-pc:~/workdir/fs3399/application/hello$
```

hello.i

```
hello.i x
# 1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 374 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 385 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 386 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 375 "/usr/include/features.h" 2 3 4
# 398 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
# 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
# 399 "/usr/include/features.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4

# 1 "/usr/lib/gcc/x86_64-linux-gnu/4.8/include/stddef.h" 1 3 4
# 212 "/usr/lib/gcc/x86_64-linux-gnu/4.8/include/stddef.h" 3 4
typedef long unsigned int size_t;
# 34 "/usr/include/stdio.h" 2 3 4

# 1 "/usr/include/x86_64-linux-gnu/bits/types.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 28 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4

typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;

typedef signed char __int8_t;
typedef unsigned char __uint8_t;
typedef signed short int __int16_t;
```

预处理文件

hello.s

```
hello.i x hello.s x
.file "hello.c"
.section .rodata
.LC0:
.string "hello world! "
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
movl $.LC0, %edi
call puts
movl $0, %eax
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4"
.section .note.GNU-stack,"",@progbits
```

汇编语言程序

- gcc常用编译选项:

- ✓ **-E**: 只进行预处理, 不编译

第一步: 预处理

- ✓ **-S**: 只汇编, 不编译

第二步: 汇编

- ✓ **-c**: 只编译、汇编, 不链接

第三步: 编译, “-c”表示编译

- ✓ **-g**: 包含调试信息

- ✓ **-I**: 指定include包含文件的搜索目录

- ✓ **-o**: 输出成指定文件名

第四步: 链接, “-o”表示输出

- ✓ **-v**: 详细输出编译过程中所采用的每一个选项

- ✓ **-C**: 预处理时保留注释信息

- ✓ **-ggdb**: 在可执行文件中包含可供GDB使用的调试信息

- ✓ **-fverbose-asm**: 在编译成汇编语言时, 把C变量的名称作为汇编语言中的注释

- ✓ **-save-temps**: 自动输出预处理文件、汇编文件、对象文件, 编译正常进行

- ✓ **-fsyntax-only**: 只测试源文件语法是否正确, 不会进行任何编译操作

- ✓ **-ffreestanding**: 编译成独立程序, 而非宿主程序

gcc 使用入门教程

- 例1：一个源文件（hello.c）

- ✓ `gcc -g -Wall -o hello-pc hello.c`
- ✓ `./hello-pc`
 - ✓ hello world!

```
linux@linux-pc:~/workdir/fs3399/application/hello$  
linux@linux-pc:~/workdir/fs3399/application/hello$ gcc -g -Wall -o hello-pc hello.c  
linux@linux-pc:~/workdir/fs3399/application/hello$  
linux@linux-pc:~/workdir/fs3399/application/hello$  
linux@linux-pc:~/workdir/fs3399/application/hello$ ./hello-pc  
hello world!  
linux@linux-pc:~/workdir/fs3399/application/hello$
```

- ✓ `-g`: 表示在生成的目标文件中带调试信息。
- ✓ `-Wall`: 选项 `-Wall` 开启编译器几乎所有常用的警告——强烈建议你始终使用该选项。
- ✓ `-o`: 机器码的文件名是通过 `-o` 选项指定的。该选项通常作为命令行中的最后一个参数。如果被省略，输出文件默认为“a.out”。
- ✓ `./`: 路径 `./` 指代当前目录，因此 `./hello-pc` 载入并执行当前目录下的可执行文件“hello-pc”。

- 例2：多个源文件（test_main.c，test_sum.c）

- ✓ `gcc -g -Wall -o test-pc test_main.c test_sum.c`

同学们可以在自己电脑的
Ubuntu环境下测试一下！

test_main.c

```
#include <stdio.h>
#include <stdlib.h>

extern int sum(int value);

struct inout {
    int value;
    int result;
};

int main(int argc, char * argv[])
{
    struct inout * io = (struct inout * ) malloc(sizeof(struct inout));

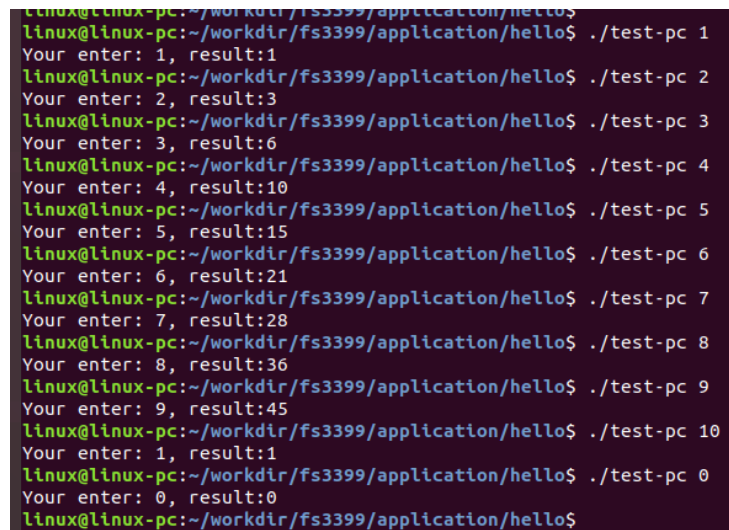
    if (NULL == io) {
        printf("Malloc failed!\n");
        return -1;
    }

    if (argc != 2) {
        printf("Wrong parameter!\n");
        return -1;
    }

    io -> value = *argv[1] - '0';
    io -> result = sum(io -> value);

    printf("Your enter: %d, result:%d\n", io -> value, io -> result);

    return 0;
}
```



A terminal window showing the execution of the test_main.c program. The user runs ./test-pc 1 through ./test-pc 10. The program outputs the result of the sum function for each input. For inputs 1 through 9, the result is the input value multiplied by 10. For input 0, the result is 0. The program also handles invalid input (./test-pc 10) by printing an error message and returning -1.

```
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 1
Your enter: 1, result:1
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 2
Your enter: 2, result:3
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 3
Your enter: 3, result:6
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 4
Your enter: 4, result:10
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 5
Your enter: 5, result:15
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 6
Your enter: 6, result:21
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 7
Your enter: 7, result:28
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 8
Your enter: 8, result:36
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 9
Your enter: 9, result:45
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 10
Your enter: 1, result:1
linux@linux-pc:~/workdir/fs3399/application/hello$ ./test-pc 0
Your enter: 0, result:0
linux@linux-pc:~/workdir/fs3399/application/hello$
```

test_sum.c

```
int sum(int value)
{
    int result = 0;
    int i = 0;

    for (i = 0; i < value; i++)
        result += (i + 1);

    return result;
}
```

– 3、binutils（链接器+汇编器）

- binutils是**与gcc配套的工具集**，binutils工具集中的部分工具除了被gcc在后台使用为我们创建程序文件之外，其他则有助于方便开发和调试。
- binutils**主要**包括：
 - ① **addr2line**：**指令地址翻译器**，用于得到程序指令地址所对应的函数，以及函数所在的源文件名和行号。
 - ② **ar**：**静态库生成器**，用于创建和修改档案文件，以及从档案文件中抽取文件。静态库（.a文件）就是一种档案文件，需要用它生成和管理。
 - ③ **as**：**汇编编译器**，用于将汇编代码转换为目标文件。
 - ④ **ld**：**链接器**。
 - ⑤ **nm**：**符号显示器**，用于列出程序文件中的符号及符号在内存中（开始）地址，符号包含C程序中的函数名和变量名。
 - ⑥ **objdump**：**信息查看器**，能显示程序文件的相关信息和对程序文件进行反汇编。
 - ⑦ **objcopy**：**段剪辑器**，可以用来从程序文件中拷贝出我们所指定的段；在将引导加载器烧至闪存中时，有时需要通过从程序中抽取段的方式生成烧写文件，这时objcopy工具就能派上用场。
 - ⑧ **ranlib**：**库索引生成器**，用于生成一个档案文件的内容索引，以加快对档案文件的查找速度；将该工具运用与静态库能提高库参与链接的效率。
 - ⑨ **readelf**：**显示有关ELF二进制文件的信息**，通过readelf -h *.exe进行查看。

5.5 本地调试（gdb）

- **gdb**: **GNU Debugger**, GNU调试器
- 本地调试:
 - 调试ARM可执行文件（实验箱上运行的可执行文件）
 - 在“串口超级终端Xshell 7”上运行: **gdb**
 - 此时“本地”是指实验箱
 - 调试x86可执行文件（Ubuntu上运行的可执行文件）
 - 在Ubuntu的“终端”上运行: **gdb**
 - 此时“本地”是指PC机（Ubuntu）

• 1、调试ARM可执行文件:

– 生成带调试信息（-g）的可执行文件，在Ubuntu的“终端”上执行:

- aarch64-linux-gnu-gcc -g -Wall -o test test_main.c test_sum.c

- 将生成“test”可执行文件

– 将“test”可执行文件上传到实验箱中

– 在Xshell 7中运行gdb调试工具:

- 方法一: gdb test

- 方法二: gdb

```
linux@localhost:~$  
linux@localhost:~$ gdb test-pc  
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1  
Copyright (C) 2016 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "aarch64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word"...  
Reading symbols from test-pc...done.  
(gdb) q  
linux@localhost:~$
```

gdb test

q

```
linux@localhost:~$  
linux@localhost:~$ gdb  
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1  
Copyright (C) 2016 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "aarch64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word"...  
(gdb) file test-pc  
Reading symbols from test-pc...done.  
(gdb) q  
linux@localhost:~$
```

gdb

file test

q

gdb的常用命令

命令 命令缩写 说明

list	l	显示多行源代码
break	b	设置断点,程序运行到断点的位置会停下来
info	i	描述程序的状态
run	r	开始运行程序
display	disp	跟踪查看某个变量,每次停下来都显示它的值
step	s	执行下一条语句,如果该语句为函数调用,则进入函数执行其中的第一条语句
next	n	执行下一条语句,如果该语句为函数调用,不会进入函数内部执行(即不会一步步地调试函数内部语句)
print	p	打印内部变量值
continue	c	继续程序的运行,直到遇到下一个断点
set var name=v		设置变量的值
start	st	开始执行程序,在main函数的第一条语句前面停下来
file		装入需要调试的程序
kill	k	终止正在调试的程序
watch		监视变量值的变化
backtrace	bt	查看函数调用信息(堆栈)
frame	f	查看栈帧
quit	q	退出GDB环境


```
root@localhost:/home/linux# gdb test
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "aarch64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test...done.
(gdb) r 5
Starting program: /home/linux/test 5
Your enter: 5, result:15
[Inferior 1 (process 1593) exited normally]
(gdb) q
root@localhost:/home/linux#
```

• 2、调试x86可执行文件：

– 在Ubuntu的“终端”下执行（-g表示包含调试信息）：

- `gcc -g -Wall -o test-pc test_main.c test_sum.c`
- 将生成“**test-pc**”可执行文件
- 方法一：**gdb test-pc**
- 方法二：**gdb**

gdb test-pc

```
linux@linux-pc:~/workdir/fs3399/application/hello$  
linux@linux-pc:~/workdir/fs3399/application/hello$ gdb test-pc  
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git  
Copyright (C) 2018 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "x86_64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word"...  
Reading symbols from test-pc...done.  
(gdb) █
```

gdb

```
linux@linux-pc:~/workdir/fs3399/application/hello$  
linux@linux-pc:~/workdir/fs3399/application/hello$ gdb  
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git  
Copyright (C) 2018 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "x86_64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word"...  
(gdb) file test-pc  
Reading symbols from test-pc...done.  
(gdb) █
```

file test-pc

请同学们在自己电脑的Ubuntu环境下做一下！

查看程序命令：list (l)

Reading symbols from test-pc...done.

(gdb) list

list

```
1      #include <stdio.h>
2      #include <stdlib.h>
```

```
3
4      extern int sum(int value);
```

```
5
6      struct inout {
7          int value;
8          int result;
9      };
```

10

(gdb) l

l

```
11      int main(int argc, char * argv[])
```

```
12      {
13          struct inout * io = (struct inout * ) malloc(s
```

```
14
15          if (NULL == io) {
16              printf("Malloc failed!\n");
17              return -1;
18          }
```

```
19
20          if (argc != 2) {
```

(gdb)

```
#include <stdio.h>
#include <stdlib.h>
```

```
extern int sum(int value);
```

```
struct inout {
    int value;
    int result;
};
```

```
int main(int argc, char * argv[])
{
    struct inout * io = (struct inout * ) malloc(sizeof(struct inout));
```

```
    if (NULL == io) {
        printf("Malloc failed!\n");
        return -1;
    }
```

```
    if (argc != 2) {
        printf("Wrong parameter!\n");
        return -1;
    }
```

```
    io -> value = *argv[1] - '0';
    io -> result = sum(io -> value);
```

```
    printf("Your enter: %d, result:%d\n", io -> value, io -> result);
```

```
    return 0;
}
```

请同学们在自己电脑的Ubuntu环境下做一下！

设置断点命令：break (b)

```
20         if (a
(gdb) break 20
Breakpoint 1 at 0x711: file test_main.c, line 20.
(gdb) b 21
Breakpoint 2 at 0x717: file test_main.c, line 21.
(gdb) info
```

b 21

查看断点命令：info break

```
Command Name Address
(gdb) info breakpoints
Num      Type      Disp Enb Address          What
1        breakpoint keep y  0x0000000000000711 in main at test_main.c:20
2        breakpoint keep y  0x0000000000000717 in main at test_main.c:21
(gdb)
```

```
#include <stdio.h>
#include <stdlib.h>

extern int sum(int value);

struct inout {
    int value;
    int result;
};

int main(int argc, char * argv[])
{
    struct inout * io = (struct inout *) malloc(sizeof(struct inout));

    if (NULL == io) {
        printf("Malloc failed!\n");
        return -1;
    }

    if (argc != 2) {
        printf("Wrong parameter!\n");
        return -1;
    }

    io->value = *argv[1] - '0';
    io->result = sum(io->value);

    printf("Your enter: %d, result:%d\n", io->value, io->result);

    return 0;
}
```

请同学们在自己电脑的Ubuntu环境下做一下！

连续运行命令：run (r)

run 5

```
(gdb)
(gdb) run 5
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/linux/workdir/fs3399/application/hello/test-pc 5

Breakpoint 1, main (argc=2, argv=0x7fffffffdee8) at test_main.c:20
20      if (argc != 2) {
(gdb) c
Continuing.
Your enter: 5, result:15
[Inferior 1 (process 4628) exited normally]
(gdb) r 5
Starting program: /home/linux/workdir/fs3399/application/hello/test-pc 5

Breakpoint 1, main (argc=2, argv=0x7fffffffdee8) at test_main.c:20
20      if (argc != 2) {
(gdb) c
Continuing.
Your enter: 5, result:15
[Inferior 1 (process 4629) exited normally]
(gdb)
```

继续运行命令：c

继续运行命令：c

```
#include <stdio.h>
#include <stdlib.h>

extern int sum(int value);

struct inout {
    int value;
    int result;
};

int main(int argc, char * argv[])
{
    struct inout * io = (struct inout *) malloc(sizeof(struct inout));

    if (NULL == io) {
        printf("Malloc failed!\n");
        return -1;
    }

    if (argc != 2) {
        printf("Wrong parameter!\n");
        return -1;
    }

    io->value = *argv[1] - '0';
    io->result = sum(io->value);

    printf("Your enter: %d, result:%d\n", io->value, io->result);

    return 0;
}
```

请同学们在自己电脑的Ubuntu环境下做一下！

单步执行命令：next (n)

```
type 'b' to set a breakpoint.
Reading symbols from ./application/hello/test-pc...done.
(gdb) b 11
Breakpoint 1 at 0x0e9: file test_main.c line 11.
(gdb) run 5
Starting program: /home/linux/workdir/application/hello/test-pc 5

Breakpoint 1, main (argc=2, argv=0x7ffffffdee8) at test_main.c:13
13      struct inout * io = (struct inout * ) malloc(sizeof(struct inout));
(gdb) next
15      if (NULL == io) {
(gdb) next
20      if (argc != 2) {
(gdb) n
25      io -> value = *argv[1] - '0';
(gdb) n
26      io -> result = sum(io -> value);
(gdb) n
28      printf("Your enter: %d, result:%d\n", io -> value, io -> result);
(gdb) n
Your enter: 5, result:15
30      return 0;
(gdb)
```

```
#include <stdio.h>
#include <stdlib.h>

extern int sum(int value);

struct inout {
    int value;
    int result;
};

int main(int argc, char * argv[])
{
    struct inout * io = (struct inout * ) malloc(sizeof(struct inout));

    if (NULL == io) {
        printf("Malloc failed!\n");
        return -1;
    }

    if (argc != 2) {
        printf("Wrong parameter!\n");
        return -1;
    }

    io -> value = *argv[1] - '0';
    io -> result = sum(io -> value);

    printf("Your enter: %d, result:%d\n", io -> value, io -> result);

    return 0;
}
```

5.6 远程调试（gdb + gdbserver）

- 远程调试：
 - 用于调试ARM程序，即在Ubuntu（宿主机）上调试运行在实验箱（目标机）上的程序
 - 在实验箱的“Xshell 7超级终端”上运行：gdbserver
 - 在Ubuntu的“终端”上运行：arm-linux-gdb

5.7 内核调试（gdb + kgdb）

- 使用**printk**函数：调试应用程序时，可以使用**printf**函数显示有关信息。调试Linux内核时，则是使用**printk**函数显示有关信息。
- 使用**kgdb**内核调试工具：
 - 在目标机（实验箱，Xshell 7）上运行**kgdb**
 - 在宿主机（PC机，Ubuntu）上运行**gdb**

5.8 网络调试

- 如果嵌入式系统（目标机，实验箱）上有网络通信程序，则需要网络调试工具。
- 在传统的网络分析和调试技术中，**嗅探器**（Sniffer）是最常见也是最重要的一种技术。
- **tcpdump**是一款功能强大、截取灵活的开源嗅探器工具。
- 除了tcpdump外，还有arp、ping、route、netstat等网络调试与诊断工具。

小结

- 嵌入式系统交叉开发环境：
 - 宿主机环境（PC电脑，Vmware + Ubuntu）
 - 目标板环境（实验箱，Xshell 7）
 - 交叉编译环境（x86环境编译ARM程序）
- 嵌入式系统调试技术：
 - gdb：本地调试
 - gdb + gdbserver：远程调试
 - gdb + kgdb：内核调试
 - tcpdump：网络调试（网络通信程序调试）

进一步探索

- 了解GNU工具的具体使用方法。

Thanks