

# 浙江大学

## 本科实验报告

课程名称：操作系统

姓 名：夏尤楷

学 院：计算机科学与技术学院

系：计算机科学与技术系

专 业：计算机科学与技术

学 号：3210104331

指导教师：夏莹杰

2023 年 10 月 6 日

# 浙江大学操作系统实验报告

实验名称: GDB & QEMU 调试 64 位 RISC-V LINUX

电子邮件地址: 3210104331@zju.edu.cn 手机: 15058004449

实验地点: 玉泉曹光彪西 503 实验日期: 2023 年 10 月 6 日

## 一、实验目的和要求

1. 使用交叉编译工具, 完成 Linux 内核代码编译;
2. 使用 QEMU 运行内核;
3. 熟悉 GDB 和 QEMU 联合调试。

## 二、实验过程

本实验使用 Ubuntu 22.04.2 LTS Windows Subsystem for Linux 2 进行。

### 1. 搭建实验环境

首先输入以下命令, 输入后如下图:

```
1. $ sudo apt install gcc-riscv64-linux-gnu
2. $ sudo apt install autoconf automake autotools-dev curl libmpc-dev libm
   pfr-dev libgmp-dev \
   gawk build-essential bison flex texinfo gperf libtool patchutils bc \
   zlib1g-dev libexpat-dev git
```

```
tangkeke@DESKTOP-100KJCA: $ sudo apt install gcc-riscv64-linux-gnu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-riscv64-linux-gnu binutils-x86-64-linux-gnu cpp-11-riscv64-linux-gnu
  cpp-riscv64-linux-gnu gcc-11-cross-base-ports gcc-11-riscv64-linux-gnu gcc-11-riscv64-linux-gnu-base
  gcc-12-cross-base-ports libasan6-riscv64-cross libatomic1-riscv64-cross libbinutils libc6-dev-riscv64-cross
  libc6-riscv64-cross libctf0 libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross libgomp1-riscv64-cross
  linux-libc-dev-riscv64-cross
Suggested packages:
  binutils-doc gcc-11-locales cpp-doc gcc-11-doc gdb-riscv64-linux-gnu gcc-doc
The following NEW packages will be installed:
  binutils-riscv64-linux-gnu cpp-11-riscv64-linux-gnu cpp-riscv64-linux-gnu gcc-11-cross-base-ports
  gcc-11-riscv64-linux-gnu gcc-11-riscv64-linux-gnu-base gcc-12-cross-base-ports gcc-riscv64-linux-gnu
  libasan6-riscv64-cross libatomic1-riscv64-cross libc6-dev-riscv64-cross libc6-riscv64-cross
  libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross libgomp1-riscv64-cross linux-libc-dev-riscv64-cross
The following packages will be upgraded:
  binutils binutils-common binutils-x86-64-linux-gnu libbinutils libctf0
5 upgraded, 16 newly installed, 0 to remove and 103 not upgraded.
Need to get 32.8 MB/38.2 MB of archives.
After this operation, 112 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

(第二个命令输入后的情况忘了截图了)

再输入如下命令, 安装用于启动 riscv64 平台上的内核的模拟器 qemu。

```
1. $ sudo apt install qemu-system-misc
```

输入后需要等待片刻才能安装完成。

最后输入如下命令, 安装 gdb 来对在 qemu 上运行的 Linux 内核进行调试, 如图:

```
1. $ sudo apt install gdb-multiarch
```

```
tangeke@DESKTOP-I00KJCA:~$ sudo apt install gdb-multiarch
[sudo] password for tangeke:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gdb-multiarch
0 upgraded, 1 newly installed, 0 to remove and 103 not upgraded.
Need to get 4589 kB of archives.
After this operation, 18.2 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 gdb-multiarch amd64 12.1-0ubuntu1~22.04 [4589 kB]
Ign:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 gdb-multiarch amd64 12.1-0ubuntu1~22.04
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 gdb-multiarch amd64 12.1-0ubuntu1~22.04 [4589 kB]
Fetched 58.5 kB in 5min 28s (178 B/s)
Selecting previously unselected package gdb-multiarch.
(Reading database ... 38636 files and directories currently installed.)
Preparing to unpack .../gdb-multiarch_12.1-0ubuntu1~22.04_amd64.deb ...
Unpacking gdb-multiarch (12.1-0ubuntu1~22.04) ...
Setting up gdb-multiarch (12.1-0ubuntu1~22.04) ...
```

## 2. 获取 Linux 源码和已经编译好的文件系统

从 <https://www.kernel.org> 下载最新的 Linux 源码(实验期间的最新源码为 6.6-rc4 版本)的压缩包并解压, 如图:

```
tangeke@DESKTOP-I00KJCA:~/os$ ls
linux-6.6-rc4  linux-6.6-rc4.tar.gz
```

并输入以下命令, 使用 git 工具 clone 实验仓库, 以获得准备好的根文件系统的镜像, 如图:

```
1. $ git clone https://gitee.com/zju_xiayingjie/os23fall-stu.git
2. $ cd os23fall-stu/src/lab0
3. $ ls
4. rootfs.img # 已经构建完成的根文件系统的镜像
```

```
tangeke@DESKTOP-I00KJCA:~/os$ git clone https://gitee.com/zju_xiayingjie/os23fall-stu.git
Cloning into 'os23fall-stu'...
remote: Enumerating objects: 147, done.
remote: Counting objects: 100% (147/147), done.
remote: Compressing objects: 100% (121/121), done.
remote: Total 147 (delta 33), reused 97 (delta 6), pack-reused 0
Receiving objects: 100% (147/147), 1.94 MiB | 1.18 MiB/s, done.
Resolving deltas: 100% (33/33), done.
tangeke@DESKTOP-I00KJCA:~/os$ cd os23fall-stu/src/lab0/
tangeke@DESKTOP-I00KJCA:~/os/os23fall-stu/src/lab0$ ls
rootfs.img
```

## 3. 编译 Linux 内核

输入以下命令:

```
1. $ cd path/to/linux
2. $ make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
# 使用默认配置
3. $ make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- -j$(nproc) # 编译
```

前两条命令执行后效果如下图：

```
tangkeke@DESKTOP-100KJCA: ~/os$ cd linux-6.6-rc4/
tangkeke@DESKTOP-100KJCA: ~/os/linux-6.6-rc4$ make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
YACC scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/conf
*** Default configuration is based on 'defconfig'
configuration written to .config
```

第三条命令执行需要等待较长时间。

#### 4. 使用 QEMU 运行内核

输入以下命令：

```
1. qemu-system-riscv64 -nographic -machine virt -kernel linux-6.6-rc4/arch/riscv/boot/Image \
2. -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
3. -bios default -drive file=os23fall-stu/src/lab0/rootfs.img,format=raw,id=hd0
```

执行后，运行内核效果如下图：

```
tangkeke@DESKTOP-100KJCA: ~/os$ qemu-system-riscv64 -nographic -machine virt -kernel linux-6.6-rc4/arch/riscv/boot/Image \
> -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
> -bios default -drive file=os23fall-stu/src/lab0/rootfs.img,format=raw,id=hd0
OpenSBI v0.9
OpenSBI
```

使用 Ctrl+A，松开后再按下 X 键即可退出 QEMU，如图：

```
[ 0.448500] CLK: Disabling unused clocks
[ 0.482703] EXT4-fs (vda): mounted filesystem c3e9bbca-ec22-47f9-a368-187b21172fc1 ro with ordered data mode. Quota mode: disabled.
[ 0.483523] VFS: Mounted root (ext4 filesystem) readonly on device 254:0.
[ 0.485837] devtmpfs: mounted
[ 0.515069] Freeing unused kernel image (initmem) memory: 2204K
[ 0.515778] Run /sbin/init as init process
Please press Enter to activate this console.
/ # QEMU: Terminated
tangkeke@DESKTOP-100KJCA: ~/os$
```

#### 5. 使用 GDB 对内核进行调试

开启两个终端，在终端一中输入以下命令，使用 QEMU 启动 Linux：

```
1. qemu-system-riscv64 -nographic -machine virt -kernel linux-6.6-rc4/arch/riscv/boot/Image \
2. -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
```

```
3. -bios default -drive file=os23fall-stu/src/lab0/rootfs.img,format=raw,id=hd0 -S -s
```

在终端二中依次输入如下命令，使用 GDB 与 QEMU 远程通信（使用 tcp::1234 端口）进行调试：

```
1. gdb-multiarch linux-6.6-rc4/vmlinux
2. (gdb) target remote :1234 # 连接 qemu
3. (gdb) b start_kernel # 设置断点
4. (gdb) continue # 继续执行
5. (gdb) quit # 退出 gdb
```

终端一在调试期间显示如下：

```
tangkeke@DESKTOP-I00KJCA: ~/os
Platform Name       : riscv-virtio,gemu
Platform Features   : timer,mfdeleg
Platform HART Count : 1
Firmware Base       : 0x80000000
Firmware Size       : 100 KB
Runtime SBI Version : 0.2

Domain0 Name        : root
Domain0 Boot HART    : 0
Domain0 HARTs        : 0*
Domain0 Region00     : 0x0000000080000000-0x000000008001ffff ()
Domain0 Region01     : 0x0000000000000000-0xffffffffffffff (R,W,X)
Domain0 Next Address : 0x0000000080200000
Domain0 Next Arg1    : 0x0000000087000000
Domain0 Next Mode     : S-mode
Domain0 SysReset     : yes

Boot HART ID        : 0
Boot HART Domain    : root
Boot HART ISA        : rv64imafdcsv
Boot HART Features   : scounteren,mcounteren,time
Boot HART PMP Count  : 16
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits : 54
Boot HART MHPM Count : 0
Boot HART MHPM Count : 0
Boot HART MIDELEG    : 0x0000000000000022
Boot HART MEDELEG    : 0x0000000000000b109
[ 0.000000] Linux version 6.6.0-rc4 (tangkeke@DESKTOP-I00KJCA) (riscv64-linux-gnu-gcc (Ubuntu 11.4.0-lubuntul~22.04) 11.4.0,
GNU ld (GNU Binutils for Ubuntu) 2.38) #1 SMP Fri Oct 6 01:06:29 CST 2023
```

终端二在依次输入命令后显示如下：

```
tangkeke@DESKTOP-I00KJCA: ~/os
tangkeke@DESKTOP-I00KJCA: ~/os$ gdb-multiarch linux-6.6-rc4/vmlinux
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 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:
<https://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 linux-6.6-rc4/vmlinux...
(No debugging symbols found in linux-6.6-rc4/vmlinux)
(gdb) target remote :1234
Remote debugging using :1234
0x0000000000001000 in ?? ()
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff80a006ac
(gdb) continue
Continuing.

Breakpoint 1, 0xffffffff80a006ac in start_kernel ()
(gdb) quit
A debugging session is active.

    Inferior 1 [process 1] will be detached.

Quit anyway? (y or n) y
Detaching from program: /home/tangkeke/os/linux-6.6-rc4/vmlinux, process 1
Ending remote debugging.
[Inferior 1 (process 1) detached]
```

尝试使用 GDB 的各项命令，观察 GDB 所在终端窗口变化：

使用 “layout asm”，窗口上部显示出内核的汇编代码：

```
B> 0xffffffff80a006ac <start_kernel>      addi    sp, sp, -96
0xffffffff80a006ae <start_kernel+2>      sd      ra, 88(sp)
0xffffffff80a006b0 <start_kernel+4>      sd      s0, 80(sp)
0xffffffff80a006b2 <start_kernel+6>      sd      s1, 72(sp)
0xffffffff80a006b4 <start_kernel+8>      addi    s0, sp, 96
0xffffffff80a006b6 <start_kernel+10>     sd      s2, 64(sp)
0xffffffff80a006b8 <start_kernel+12>     sd      s3, 56(sp)
0xffffffff80a006ba <start_kernel+14>     sd      s4, 48(sp)
0xffffffff80a006bc <start_kernel+16>     sd      s5, 40(sp)
0xffffffff80a006be <start_kernel+18>     sd      s6, 32(sp)
0xffffffff80a006c0 <start_kernel+20>     sd      s7, 24(sp)
0xffffffff80a006c2 <start_kernel+22>     auipc   a0, 0xa0d
0xffffffff80a006c6 <start_kernel+26>     addi    a0, a0, 1022
0xffffffff80a006ca <start_kernel+30>     auipc   ra, 0xff60d
0xffffffff80a006ce <start_kernel+34>     jalr    -8(ra)
0xffffffff80a006d2 <start_kernel+38>     auipc   ra, 0x4
0xffffffff80a006d6 <start_kernel+42>     jalr    -114(ra)
0xffffffff80a006da <start_kernel+46>     auipc   ra, 0xe
0xffffffff80a006de <start_kernel+50>     jalr    -1786(ra)
0xffffffff80a006e2 <start_kernel+54>     csrwi   sstatus, 2
0xffffffff80a006e6 <start_kernel+58>     li      a5, 1
```

输入 “i r ra”，显示寄存器 ra 内存储的值：

```
(gdb) i r ra
ra                0xffffffff80001158      0xffffffff80001158 <_start_kernel+136>
```

输入 “backtrace”，查看函数的调用的栈帧和层级关系：

```
(gdb) backtrace
#0  0xffffffff80a006ac in start_kernel ()
#1  0xffffffff80001158 in _start_kernel ()
```

## 三、讨论和心得

这次实验让我对 QEMU、GDB 等工具有了初步的了解。以往对代码的调试更多地是在 IDE 这样高集成的、拥有较友好的图形界面的环境下进行，而这次使用 GDB 的调试是通过在只有纯文字的终端输入命令来进行，这样子可能麻烦了些，操作起来没那么自然了，但操作更触及程序根本，占用内存更小了。这样的调试是值得学习的。

同时，这次实验使我对 Linux 命令的使用更加熟练了。

## 四、思考题

1. 输入 `riscv64-linux-gnu-gcc <source.c> -o <file>`。
2. 输入 `riscv64-linux-gnu-objdump <file> -d`。
3. 在 GDB 所在的窗口终端
  - (1) 输入 `layout asm`。
  - (2) 输入 `b * 0x80000000`。
  - (3) 输入 `i b`。
  - (4) 输入 `b * 0x80200000`。
  - (5) 输入 `clear * 0x80000000`。
  - (6) 输入 `continue`。
  - (7) 输入 `next`。

(8) 输入 quit 后，在 QEMU 所在终端依次按下 ctrl+A 和 x。

4.输入 make clean。

5.Image 是 vmlinux 被 objcopy 处理后的产物。vmlinux 是 elf 格式的，可用于定位内核问题，但不能直接引导 Linux 系统启动；Image 是二进制的，可直接引导 Linux 系统启动。

## 五、附录

无。