# 实验 0: Rinux 环境搭建和内核编译

## 0 实验简介

搭建实验虚拟机、docker运行环境。通过在QEMU上运行Linux来熟悉如何从源代码开始将内核运行在QEMU模拟器上，并且掌握使用gdb协同QEMU进行联合调试，为后续实验打下基础。

## 1 实验目的

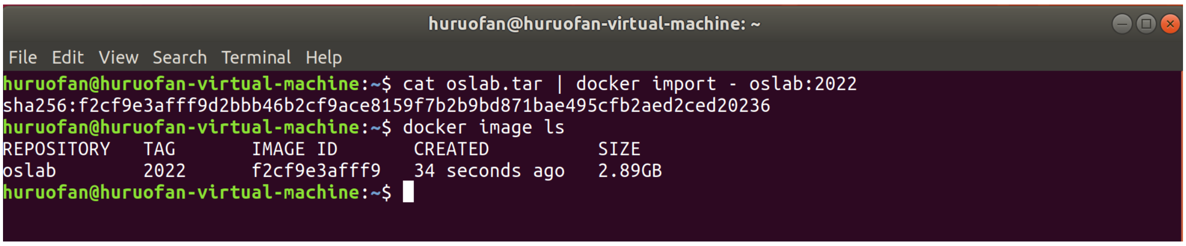
- 了解容器的使用
- 使用交叉编译工具, 完成Linux内核代码编译
- 使用QEMU运行内核
- 熟悉GDB和QEMU联合调试

## 2 操作方法和实验步骤

### 2.1 搭建 Docker环境

#### 下载并导入docker镜像

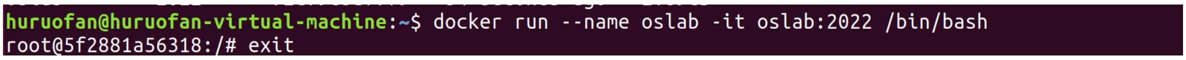
```
### 导入docker镜像
$ cat oslab.tar | docker import - oslab:2022
### 执行命令后若出现以下错误提示
### ERROR: Got permission denied while trying to connect to the Docker daemon
socket atunix:///var/run/docker.sock### 可以使用下面命令为该文件添加权限来解决
### $ sudo chmod a+rw /var/run/docker.sock
### 查看docker镜像
$ docker images
```


huruofan@huruofan-virtual-machine: ~
File Edit View Search Terminal Help

```
huruofan@huruofan-virtual-machine:~$ cat oslab.tar | docker import - oslab:2022
sha256:f2cf9e3afff9d2bbb46b2cf9ace8159f7b2b9bd871bae495cfb2aed2ced20236
huruofan@huruofan-virtual-machine:~$ docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
oslab        2022      f2cf9e3afff9   34 seconds ago   2.89GB
huruofan@huruofan-virtual-machine:~$
```

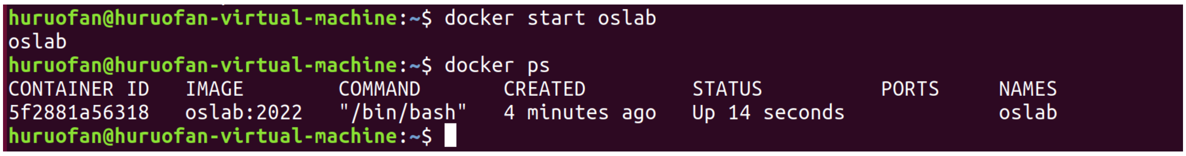
#### 从镜像中创建一个容器并进入该容器

```
### 从镜像创建一个容器
$ docker run --name oslab -it oslab:2022 /bin/bash # --name:容器名称 -i:交互式操作
-t:终端
### 提示符变为 '#' 表明成功进入容器后面的字符串根据容器而生成，为容器
### exit (or CTRL+D)
```



```
huruofan@huruofan-virtual-machine:~$ docker run --name oslab -it oslab:2022 /bin/bash
root@5f2881a56318:/# exit
```

```
### 启动处于停止状态的容器
$ docker start oslab
# oslab为容器名称
$ docker ps# 可看到容器已经启动
### 从终端连入 docker 容器
$ docker exec -it oslab /bin/bash
```



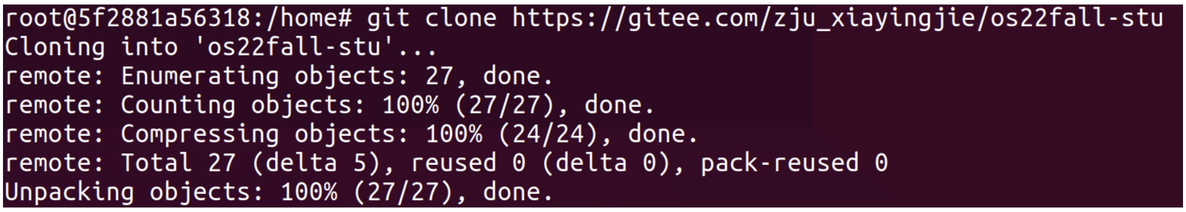
```
huruofan@huruofan-virtual-machine:~$ docker start oslab
oslab
huruofan@huruofan-virtual-machine:~$ docker ps
CONTAINER ID   IMAGE        COMMAND       CREATED         STATUS         PORTS     NAMES
5f2881a56318   oslab:2022   "/bin/bash"   4 minutes ago   Up 14 seconds            oslab
huruofan@huruofan-virtual-machine:~$ 
```

## 2.2 获取 Linux 源码和已经编译好的文件系统

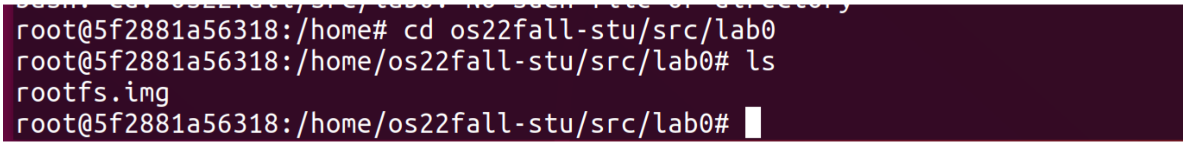
1. 进入home目录。
2. 使用git 工具 clone 本仓库。其中已经准备好了根文件系统的镜像。根文件系统为Linux Kenrel 提供了基础的文件服务，在启动

Linux Kernel 时是必要的。

```
# git clone https://gitee.com/zju_xiayingjie/os22fall-stu
# cd os22fall-stu/src/lab0
# ls
```

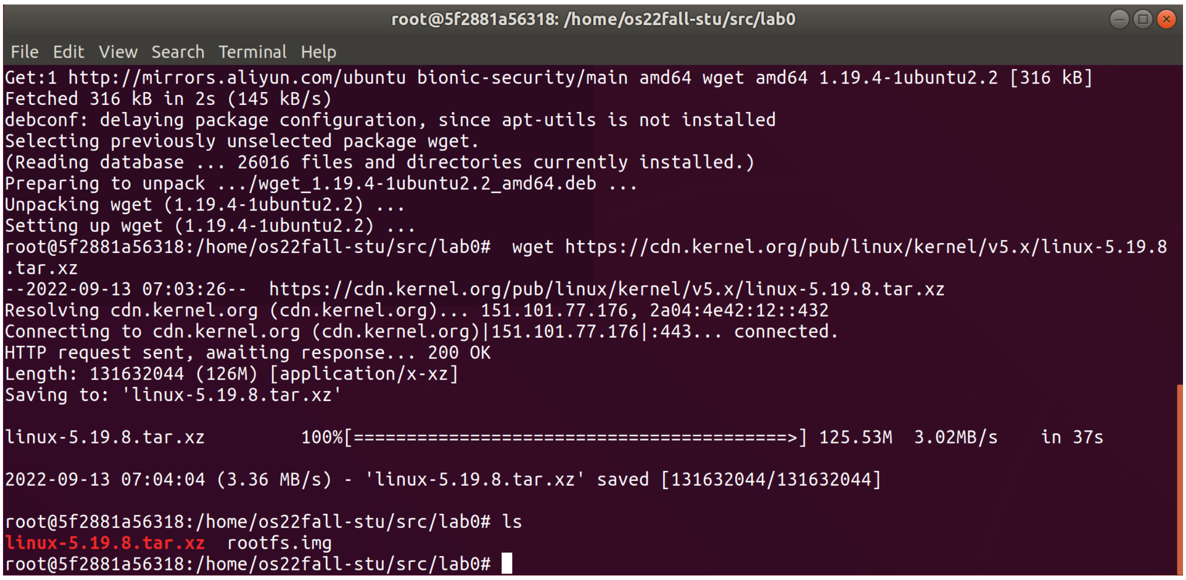


```
root@5f2881a56318:/home# git clone https://gitee.com/zju_xiayingjie/os22fall-stu
Cloning into 'os22fall-stu'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 27 (delta 5), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (27/27), done.
```



```
root@5f2881a56318:/home# cd os22fall-stu/src/lab0
root@5f2881a56318:/home/os22fall-stu/src/lab0# ls
rootfs.img
root@5f2881a56318:/home/os22fall-stu/src/lab0# 
```

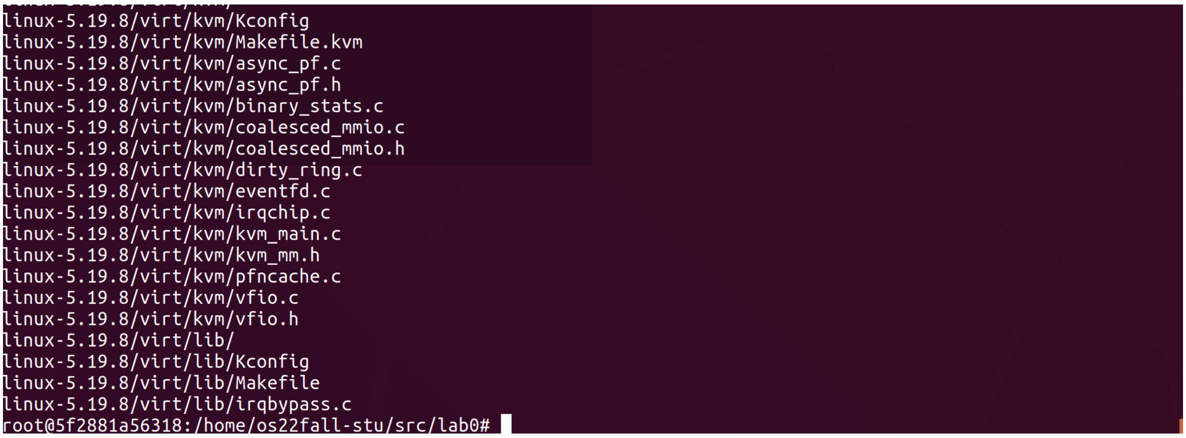
3. 在当前目录下，从[https://www.kernel.org](https://www.kernel.org)下载最新稳定版本的 Linux 源码。



root@5f2881a56318: /home/os22fall-stu/src/lab0
File Edit View Search Terminal Help

```
Get:1 http://mirrors.aliyun.com/ubuntu bionic-security/main amd64 wget amd64 1.19.4-1ubuntu2.2 [316 kB]
Fetched 316 kB in 2s (145 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package wget.
(Reading database ... 26016 files and directories currently installed.)
Preparing to unpack .../wget_1.19.4-1ubuntu2.2_amd64.deb ...
Unpacking wget (1.19.4-1ubuntu2.2) ...
Setting up wget (1.19.4-1ubuntu2.2) ...
root@5f2881a56318:/home/os22fall-stu/src/lab0#  wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8
.tar.xz
--2022-09-13 07:03:26--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.77.176, 2a04:4e42:12::432
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.77.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 131632044 (126M) [application/x-xz]
Saving to: 'linux-5.19.8.tar.xz'

linux-5.19.8.tar.xz         100%[=======================================>] 125.53M  3.02MB/s    in 37s

2022-09-13 07:04:04 (3.36 MB/s) - 'linux-5.19.8.tar.xz' saved [131632044/131632044]

root@5f2881a56318:/home/os22fall-stu/src/lab0# ls
linux-5.19.8.tar.xz  rootfs.img
root@5f2881a56318:/home/os22fall-stu/src/lab0# 
```

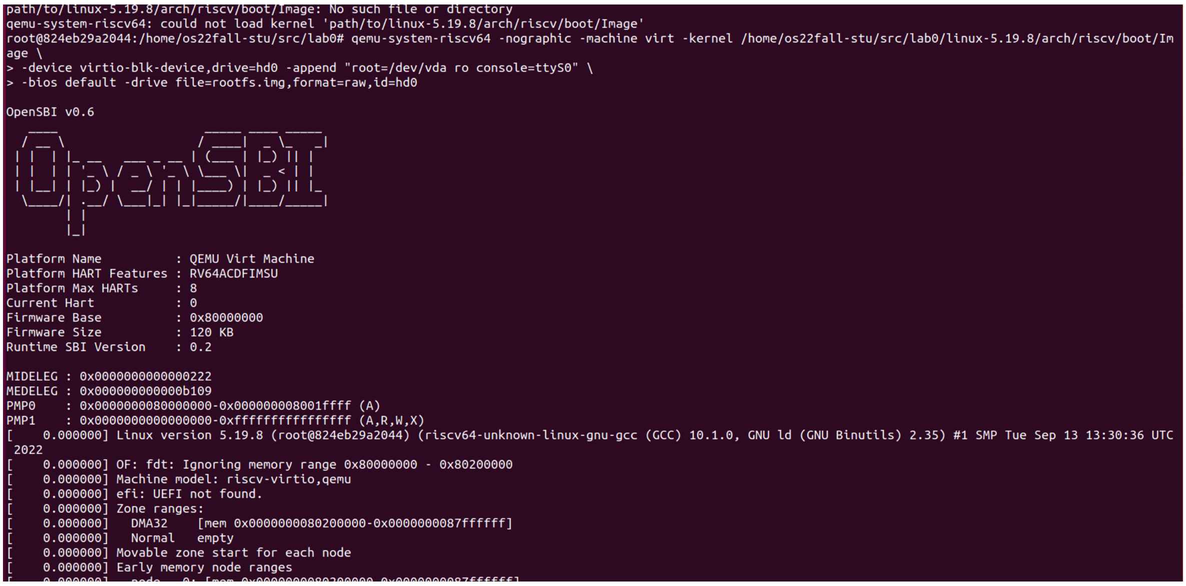
4. 使用解压缩Linux 源码包至 /home/os22fall-stu/src/lab0 目录下



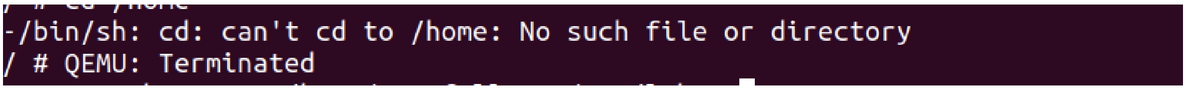
```
linux-5.19.8/virt/kvm/Kconfig
linux-5.19.8/virt/kvm/Makefile.kvm
linux-5.19.8/virt/kvm/async_pf.c
linux-5.19.8/virt/kvm/async_pf.h
linux-5.19.8/virt/kvm/binary_stats.c
linux-5.19.8/virt/kvm/coalesced_mmio.c
linux-5.19.8/virt/kvm/coalesced_mmio.h
linux-5.19.8/virt/kvm/dirty_ring.c
linux-5.19.8/virt/kvm/eventfd.c
linux-5.19.8/virt/kvm/irqchip.c
linux-5.19.8/virt/kvm/kvm_main.c
linux-5.19.8/virt/kvm/kvm_mm.h
linux-5.19.8/virt/kvm/pfncache.c
linux-5.19.8/virt/kvm/vfio.c
linux-5.19.8/virt/kvm/vfio.h
linux-5.19.8/virt/lib/
linux-5.19.8/virt/lib/Kconfig
linux-5.19.8/virt/lib/Makefile
linux-5.19.8/virt/lib/irqbypass.c
root@5f2881a56318:/home/os22fall-stu/src/lab0#
```

## 2.3 使用QEMU运行内核

```
# pwd
/home/os22fall-stu/src/lab0/
#qemu-system-riscv64 -nographic -machine virt -kernel ./linux-5.19.8/arch/riscv/boot/Image \
-device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
-bios default -drive file=rootfs.img,format=raw,id=hd0
```



```
path/to/linux-5.19.8/arch/riscv/boot/Image: No such file or directory
qemu-system-riscv64: could not load kernel 'path/to/linux-5.19.8/arch/riscv/boot/Image'
root@824eb29a2044:/home/os22fall-stu/src/lab0# qemu-system-riscv64 -nographic -machine virt -kernel /home/os22fall-stu/src/lab0/linux-5.19.8/arch/riscv/boot/Im
age \
> -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
> -bios default -drive file=rootfs.img,format=raw,id=hd0

OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
[    0.000000] Linux version 5.19.8 (root@824eb29a2044) (riscv64-unknown-linux-gnu-gcc (GCC) 10.1.0, GNU ld (GNU Binutils) 2.35) #1 SMP Tue Sep 13 13:30:36 UTC
 2022
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] Machine model: riscv-virtio,qemu
[    0.000000] efi: UEFI not found.
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x0000000087ffffff]
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
```

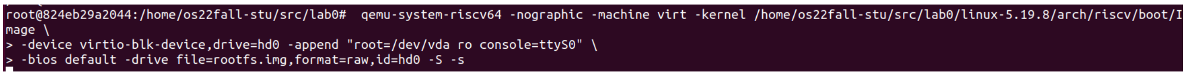


```
/ # cd /home
-/bin/sh: cd: can't cd to /home: No such file or directory
/ # QEMU: Terminated
```

## 2.4 使用gdb调试内核

对于终端1

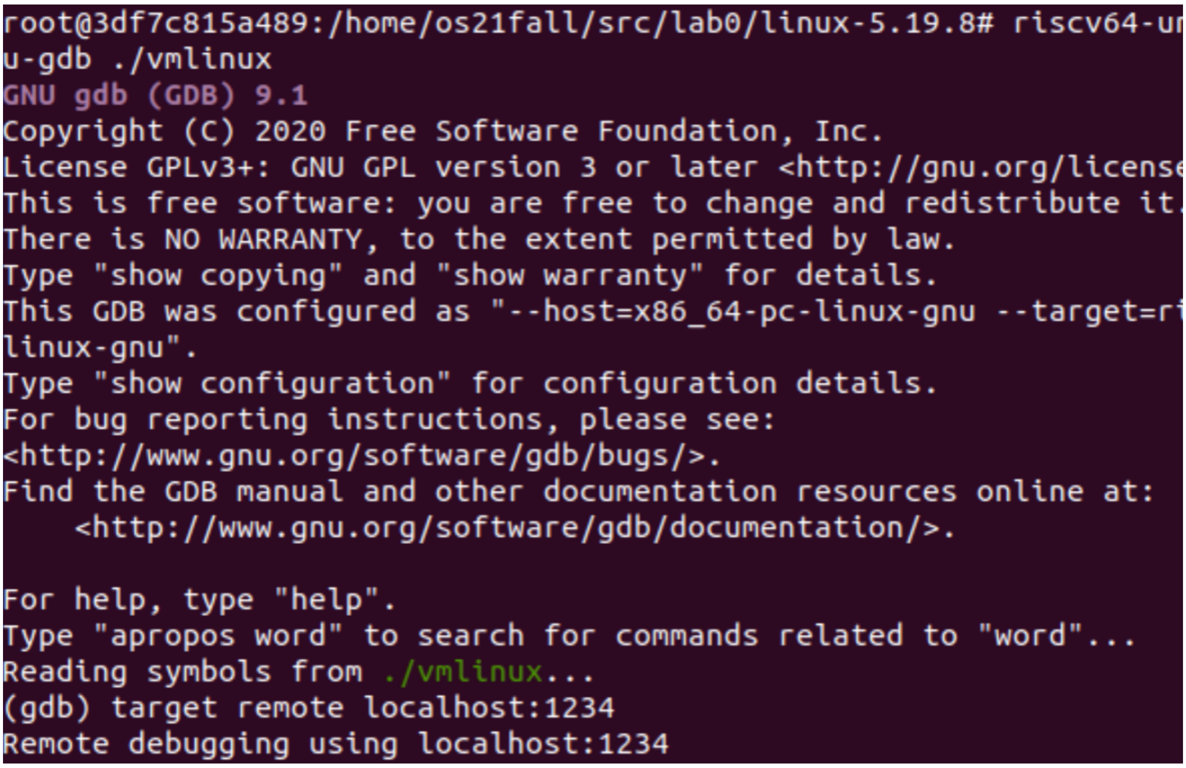
```
### Terminal 1
# pwd
/home/os22fall-stu/src/lab0/
# export RISCV=/opt/riscv
### 设置环境变量
# export PATH=$PATH:$RISCV/bin
# qemu-system-riscv64 -nographic -machine virt -kernel ./linux-
5.19.8/arch/riscv/boot/Image \
-device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
-bios default -drive file=rootfs.img,format=raw,id=hd0 -S -s
```



```
root@824eb29a2044:/home/os22fall-stu/src/lab0#  qemu-system-riscv64 -nographic -machine virt -kernel /home/os22fall-stu/src/lab0/linux-5.19.8/arch/riscv/boot/I
mage \
> -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
> -bios default -drive file=rootfs.img,format=raw,id=hd0 -S -s
```

对于终端2

```
### Terminal 2
# export RISCV=/opt/riscv
### 设置环境变量
# export PATH=$PATH:$RISCV/bin
# riscv64-unknown-linux-gnu-gdb ./linux-5.19.8/vmlinux
```



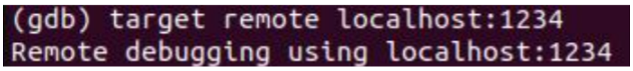
```
root@3df7c815a489:/home/os21fall/src/lab0/linux-5.19.8# riscv64-un
u-gdb ./vmlinux
GNU gdb (GDB) 9.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/license
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=ri
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 ./vmlinux...
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
```

当连接成功后，尝试以下的命令等：

```
(gdb) target remote localhost:1234 ### 连接 qemu
(gdb) b start_kernel ### 设置断点
(gdb) continue ### 继续执行
(gdb) quit ### 退出 gdb
```

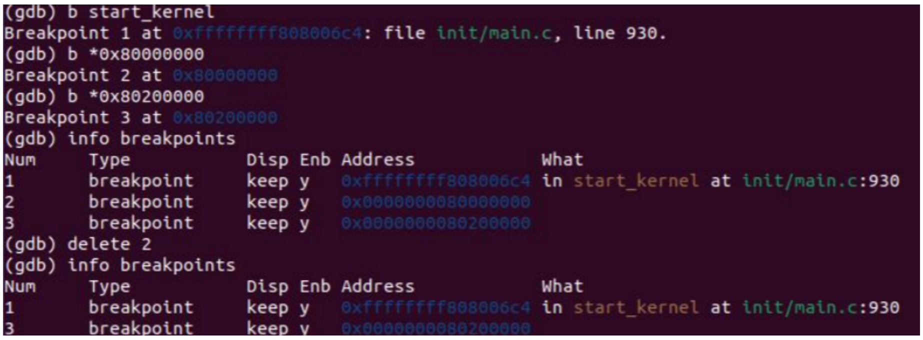
以下执行的指令为设置断点，显示断点，单步执行，跳过函数的单步执行

首先：远程连接



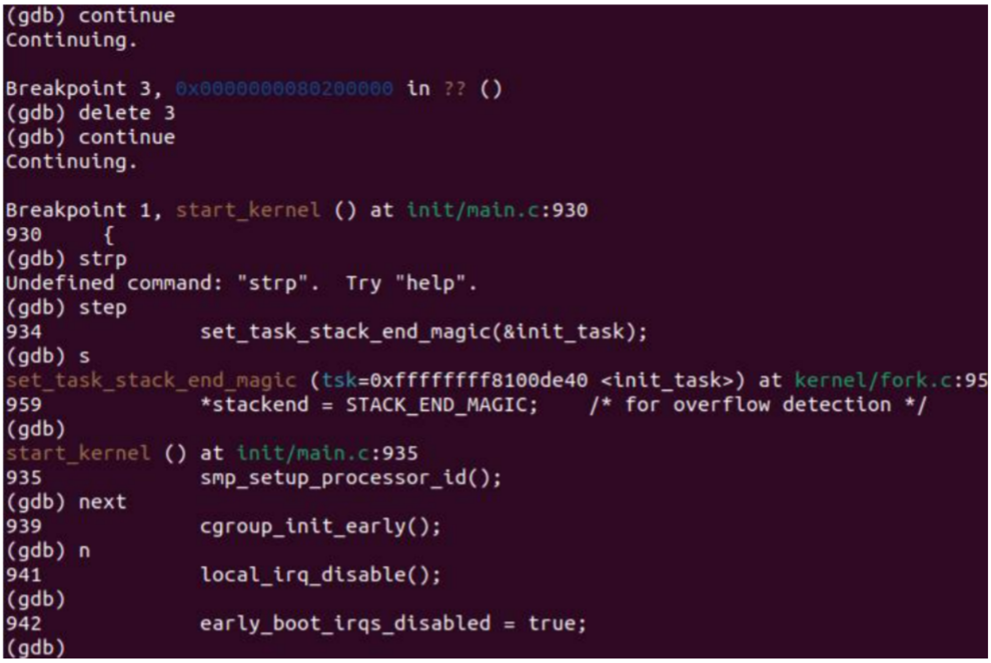
```
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
```

其次：设置断点，显示断点和删除断点



```
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff808006c4: file init/main.c, line 930.
(gdb) b *0x80000000
Breakpoint 2 at 0x80000000
(gdb) b *0x80200000
Breakpoint 3 at 0x80200000
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0xffffffff808006c4 in start_kernel at init/main.c:930
2       breakpoint     keep y   0x0000000080000000
3       breakpoint     keep y   0x0000000080200000
(gdb) delete 2
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0xffffffff808006c4 in start_kernel at init/main.c:930
3       breakpoint     keep y   0x0000000080200000
```

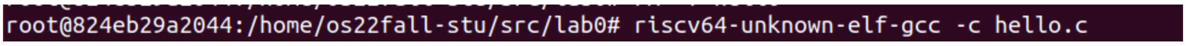
最后，运行以及单步调试



```
(gdb) continue
Continuing.

Breakpoint 3, 0x0000000080200000 in ?? ()
(gdb) delete 3
(gdb) continue
Continuing.

Breakpoint 1, start_kernel () at init/main.c:930
930     {
(gdb) strp
Undefined command: "strp".  Try "help".
(gdb) step
934             set_task_stack_end_magic(&init_task);
(gdb) s
set_task_stack_end_magic (tsk=0xffffffff8100de40 <init_task>) at kernel/fork.c:95
959             *stackend = STACK_END_MAGIC;    /* for overflow detection */
(gdb)
start_kernel () at init/main.c:935
935             smp_setup_processor_id();
(gdb) next
939             cgroup_init_early();
(gdb) n
941             local_irq_disable();
(gdb)
942             early_boot_irqs_disabled = true;
(gdb)
```

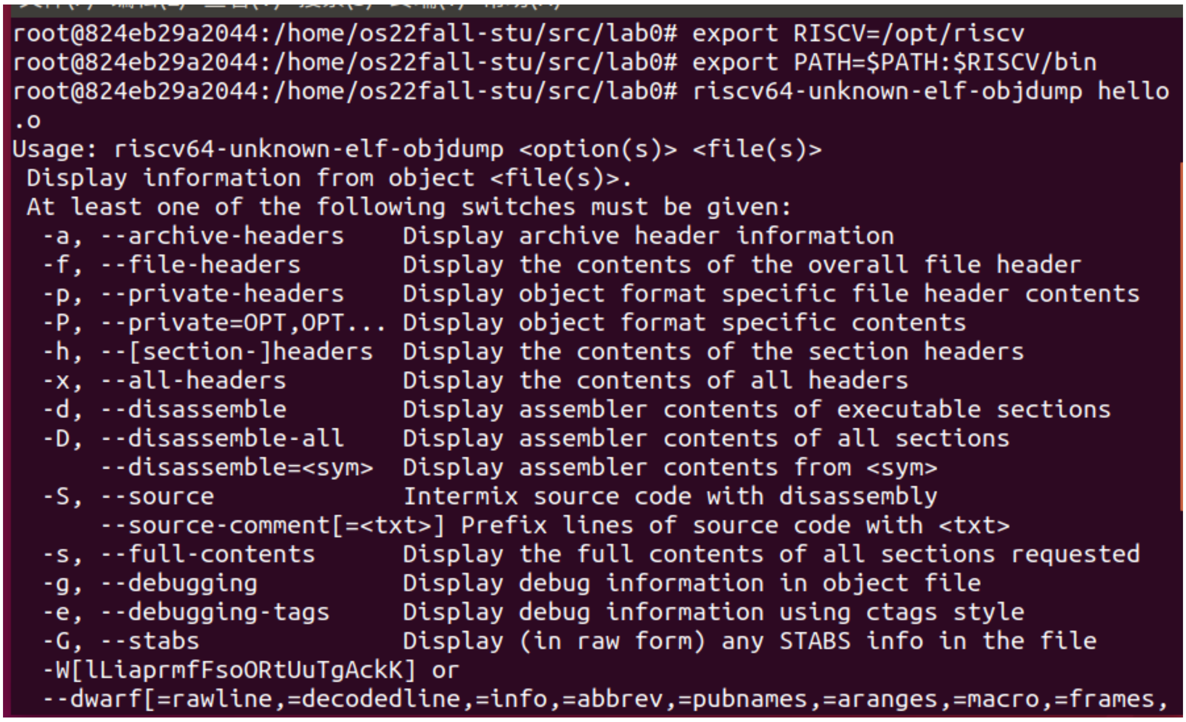
# 思考题

1. 使用 riscv64-unknown-elf-gcc 编译单个 .c 文件



```
root@824eb29a2044:/home/os22fall-stu/src/lab0# riscv64-unknown-elf-gcc -c hello.c
```

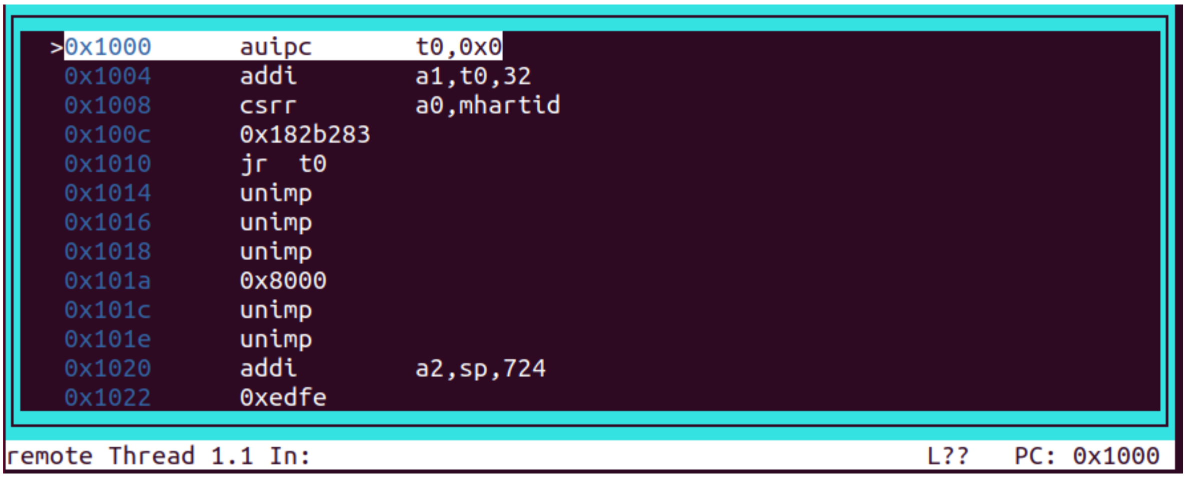
2. 使用 riscv64-unknown-elf-objdump 反汇编 1 中得到的编译产物



```
root@824eb29a2044:/home/os22fall-stu/src/lab0# export RISCV=/opt/riscv
root@824eb29a2044:/home/os22fall-stu/src/lab0# export PATH=$PATH:$RISCV/bin
root@824eb29a2044:/home/os22fall-stu/src/lab0# riscv64-unknown-elf-objdump hello
.o
Usage: riscv64-unknown-elf-objdump <option(s)> <file(s)>
 Display information from object <file(s)>.
 At least one of the following switches must be given:
  -a, --archive-headers    Display archive header information
  -f, --file-headers       Display the contents of the overall file header
  -p, --private-headers    Display object format specific file header contents
  -P, --private=OPT,OPT... Display object format specific contents
  -h, --[section-]headers  Display the contents of the section headers
  -x, --all-headers        Display the contents of all headers
  -d, --disassemble        Display assembler contents of executable sections
  -D, --disassemble-all    Display assembler contents of all sections
      --disassemble=<sym>  Display assembler contents from <sym>
  -S, --source             Intermix source code with disassembly
      --source-comment[=<txt>] Prefix lines of source code with <txt>
  -s, --full-contents      Display the full contents of all sections requested
  -g, --debugging          Display debug information in object file
  -e, --debugging-tags     Display debug information using ctags style
  -G, --stabs              Display (in raw form) any STABS info in the file
  -W[lLiaprmfFsoORtUuTgAckK] or
  --dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
```

3. 调试 Linux 时:

在 GDB 中查看汇编代码：使用layout asm



```
  >0x1000       auipc       t0,0x0
   0x1004       addi        a1,t0,32
   0x1008       csrr        a0,mhartid
   0x100c       0x182b283
   0x1010       jr  t0
   0x1014       unimp
   0x1016       unimp
   0x1018       unimp
   0x101a       0x8000
   0x101c       unimp
   0x101e       unimp
   0x1020       addi        a2,sp,724
   0x1022       0xedfe

remote Thread 1.1 In:                                          L??    PC: 0x1000
```

2.在 0x80000000 处下断点（如三中步骤）

3.查看所有已下的断点

4.在 0x80200000 处下断点

5.清除 0x80000000 处的断点

6.继续运行直到触发 0x80200000 处的断点

7.单步调试一次

8. 退出 QEMU

5.vmlinux 和 Image 的关系和区别是什么?

vmlinux是Linux内核编译出来的原始的内核文件，而Image是Linux内核编译时，处理vmlinux后生成的二进制内核映像

## 心得

在这次实验中的重点在于对需要使用的docker，gdb等有一个初步的了解，下面我来描述一下我碰到了什么困难。第一是实验中给出的命令参考，不能不加思考的照搬，这里面最明显的就是经常出现的path/to，当然这个由于每个人系统的不同肯定也会不尽相同，我就在使用gdb的时候对vmlinux吃了亏，另外，由于绝对路径有时过长，可以使用./的方式引出一个相对路径，这样表示的话会更加清楚；第二是对于使用gdb调试时，我们需要对qemu和gdb中都 配置好riscv的环境，在一次实验中我由于忘了这点造成了一些麻烦；第三是对于这些命令，基本我还处于一个慢慢学的状态，只能说对于基础的调试语句有了基本的认识，希望能在之后的实验中一边做一边熟练掌握这个gdb调试的工具；最后是在于makefile里需要加上额外参数的事情，由于我一开始没有仔细读文档，导致重新编译。