浙江大学实验报告

课程名称： 操作系统
实验项目名称：Rinux环境搭建和内核编译
学生姓名： 学号：何瑞桓 3190101928
电子邮件地址：3190101928@zju.edu.cn
实验日期： 2021年9月23日

一、实验内容

1.1 搭建docker

打开新建的Ubuntu虚拟机环境，输入ctrl+alt+t打开命令行终端，输入以下指令完成docker安装

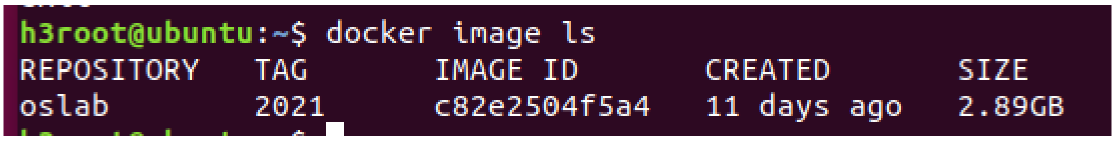
```
### 安装docker
$ sudo apt-get install curl
$ curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
### 将用户加入docker组，免sudo
$ sudo usermod -aG docker $USER
```

在学在浙大下载课程提供的oslab.tar文件，并放入主目录。执行以下指令将该docker镜像导入

```
`$ cat oslab.tar | docker import - oslab:2021`
```

输入以下指令查看当前存在的docker容器

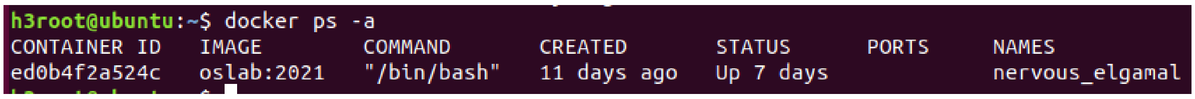
```
$ docker image ls
```



```
h3root@ubuntu:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
oslab         2021      c82e2504f5a4   11 days ago    2.89GB
```

输入以下指令查看所有存在的容器

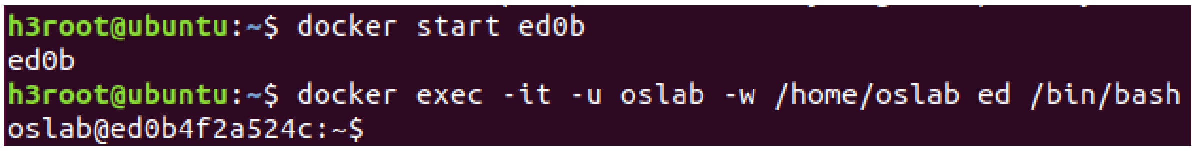
```
$ docker ps -a
```



```
h3root@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE        COMMAND       CREATED       STATUS     PORTS     NAMES
ed0b4f2a524c   oslab:2021   "/bin/bash"   11 days ago   Up 7 days            nervous_elgamal
```

开启并进入容器

```
docker start ed0b ### ed0b 为容器id的前四位
docker exec -it -u oslab -w /home/oslab ed /bin/bash ### ed 为容器id的前两位
```

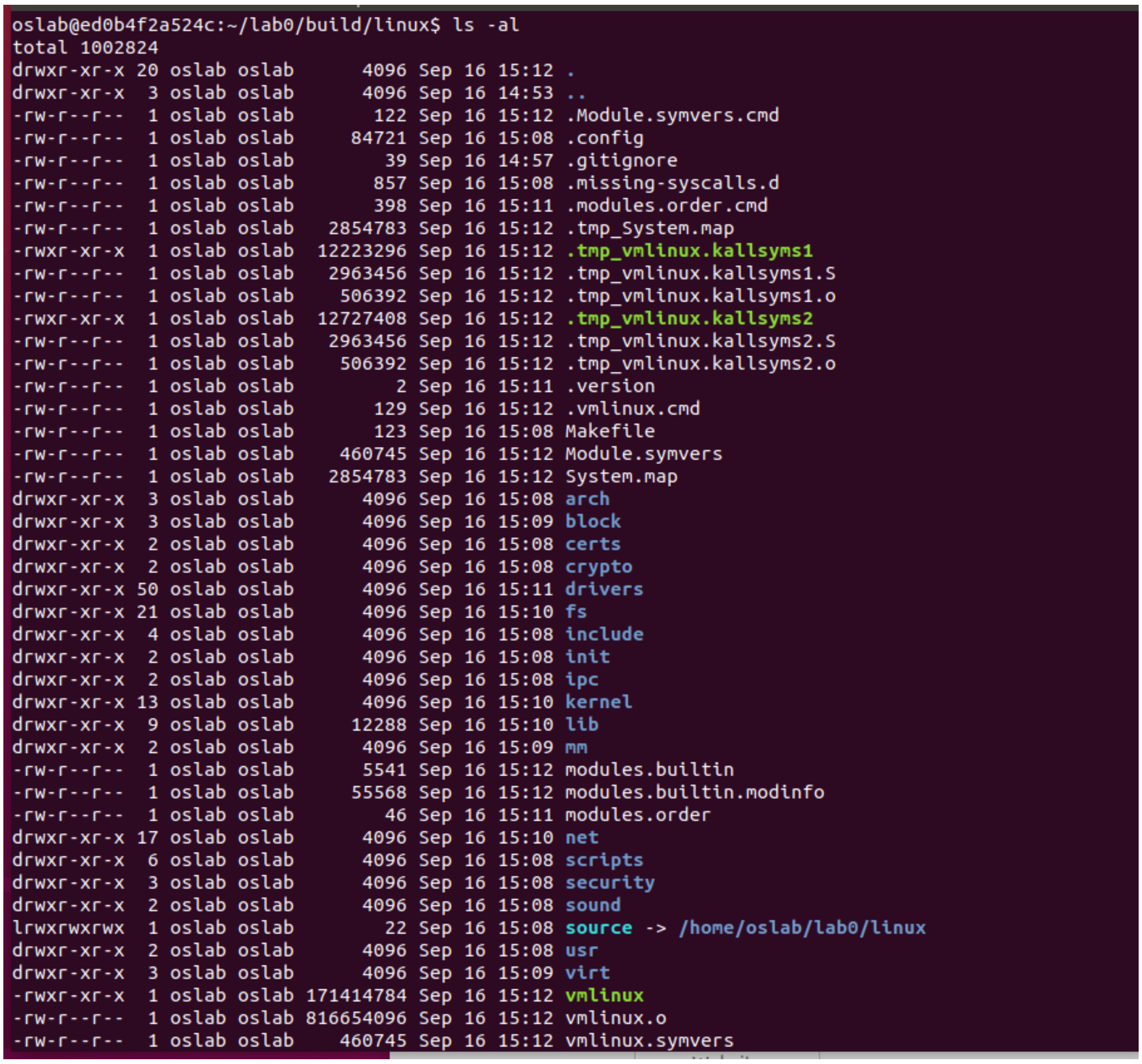


```
h3root@ubuntu:~$ docker start ed0b
ed0b
h3root@ubuntu:~$ docker exec -it -u oslab -w /home/oslab ed /bin/bash
oslab@ed0b4f2a524c:~$
```

1.2 编译Linux内核

```
### 进入实验目录并设置环境变量
# pwd
/home/oslab
# cd lab0
# export TOP=`pwd`
# export RISCV=/opt/riscv
# export PATH=$PATH:$RISCV/bin
# mkdir -p build/linux
# make -C linux O=$TOP/build/linux CROSS_COMPILE=riscv64-unknown-linux-gnu-
ARCH=riscv CONFIG_DEBUG_INFO=y defconfig all -j$(nproc)
```

编译产生文件如下

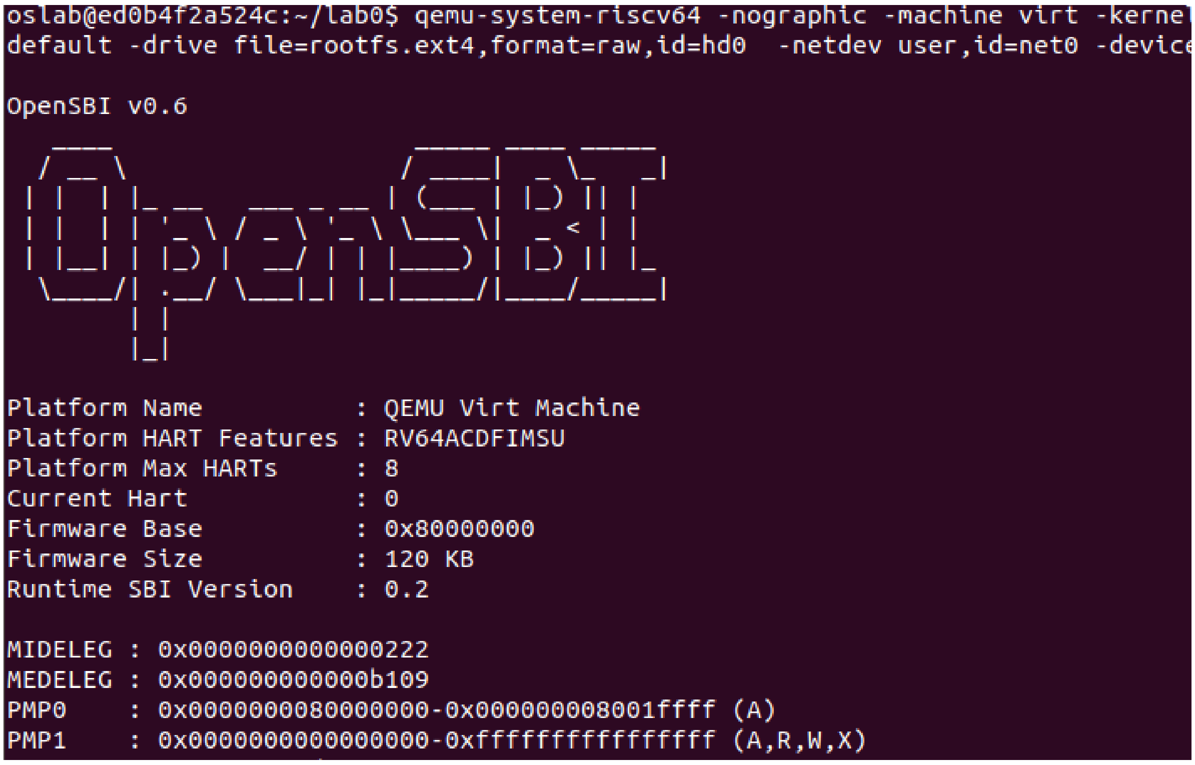


```
oslab@ed0b4f2a524c:~/lab0/build/linux$ ls -al
total 1002824
drwxr-xr-x 20 oslab oslab      4096 Sep 16 15:12 .
drwxr-xr-x  3 oslab oslab      4096 Sep 16 14:53 ..
-rw-r--r--  1 oslab oslab       122 Sep 16 15:12 .Module.symvers.cmd
-rw-r--r--  1 oslab oslab     84721 Sep 16 15:08 .config
-rw-r--r--  1 oslab oslab        39 Sep 16 14:57 .gitignore
-rw-r--r--  1 oslab oslab       857 Sep 16 15:08 .missing-syscalls.d
-rw-r--r--  1 oslab oslab       398 Sep 16 15:11 .modules.order.cmd
-rw-r--r--  1 oslab oslab   2854783 Sep 16 15:12 .tmp_System.map
-rwxr-xr-x  1 oslab oslab  12223296 Sep 16 15:12 .tmp_vmlinux.kallsyms1
-rw-r--r--  1 oslab oslab   2963456 Sep 16 15:12 .tmp_vmlinux.kallsyms1.S
-rw-r--r--  1 oslab oslab    506392 Sep 16 15:12 .tmp_vmlinux.kallsyms1.o
-rwxr-xr-x  1 oslab oslab  12727408 Sep 16 15:12 .tmp_vmlinux.kallsyms2
-rw-r--r--  1 oslab oslab   2963456 Sep 16 15:12 .tmp_vmlinux.kallsyms2.S
-rw-r--r--  1 oslab oslab    506392 Sep 16 15:12 .tmp_vmlinux.kallsyms2.o
-rw-r--r--  1 oslab oslab         2 Sep 16 15:11 .version
-rw-r--r--  1 oslab oslab       129 Sep 16 15:12 .vmlinux.cmd
-rw-r--r--  1 oslab oslab       123 Sep 16 15:08 Makefile
-rw-r--r--  1 oslab oslab    460745 Sep 16 15:12 Module.symvers
-rw-r--r--  1 oslab oslab   2854783 Sep 16 15:12 System.map
drwxr-xr-x  3 oslab oslab      4096 Sep 16 15:08 arch
drwxr-xr-x  3 oslab oslab      4096 Sep 16 15:09 block
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 certs
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 crypto
drwxr-xr-x 50 oslab oslab      4096 Sep 16 15:11 drivers
drwxr-xr-x 21 oslab oslab      4096 Sep 16 15:10 fs
drwxr-xr-x  4 oslab oslab      4096 Sep 16 15:08 include
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 init
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 ipc
drwxr-xr-x 13 oslab oslab      4096 Sep 16 15:10 kernel
drwxr-xr-x  9 oslab oslab     12288 Sep 16 15:10 lib
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:09 mm
-rw-r--r--  1 oslab oslab      5541 Sep 16 15:12 modules.builtin
-rw-r--r--  1 oslab oslab     55568 Sep 16 15:12 modules.builtin.modinfo
-rw-r--r--  1 oslab oslab        46 Sep 16 15:11 modules.order
drwxr-xr-x 17 oslab oslab      4096 Sep 16 15:10 net
drwxr-xr-x  6 oslab oslab      4096 Sep 16 15:08 scripts
drwxr-xr-x  3 oslab oslab      4096 Sep 16 15:08 security
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 sound
lrwxrwxrwx  1 oslab oslab        22 Sep 16 15:08 source -> /home/oslab/lab0/linux
drwxr-xr-x  2 oslab oslab      4096 Sep 16 15:08 usr
drwxr-xr-x  3 oslab oslab      4096 Sep 16 15:09 virt
-rwxr-xr-x  1 oslab oslab 171414784 Sep 16 15:12 vmlinux
-rw-r--r--  1 oslab oslab 816654096 Sep 16 15:12 vmlinux.o
-rw-r--r--  1 oslab oslab    460745 Sep 16 15:12 vmlinux.symvers
```

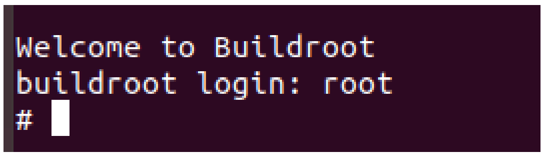
1.3 使用QEMU运行内核

输入以下命令运行该Linux内核

```
### 用户名root，没有密码
# qemu-system-riscv64 -nographic -machine virt -kernel
build/linux/arch/riscv/boot/Image \
-device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0"
\
-bios default -drive file=rootfs.ext4,format=raw,id=hd0 \
-netdev user,id=net0 -device virtio-net-device,netdev=net0
```



```
oslab@ed0b4f2a524c:~/lab0$ qemu-system-riscv64 -nographic -machine virt -kerne
default -drive file=rootfs.ext4,format=raw,id=hd0  -netdev user,id=net0 -device

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)
```



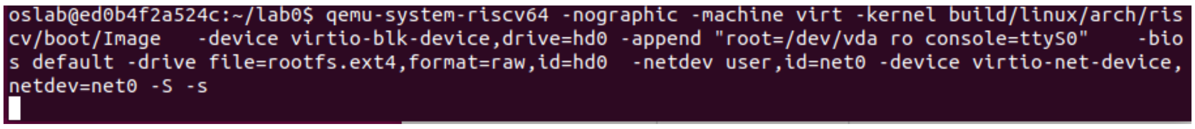
```
Welcome to Buildroot
buildroot login: root
#
```

成功进入了界面并登陆了root账户

退出QEMU模拟器的方法为：使用`ctrl`+`a`(macOS下为`control+a`)，松开后再按下`x`键即可退出QEMU

1.4 使用 GDB 对内核进行调试

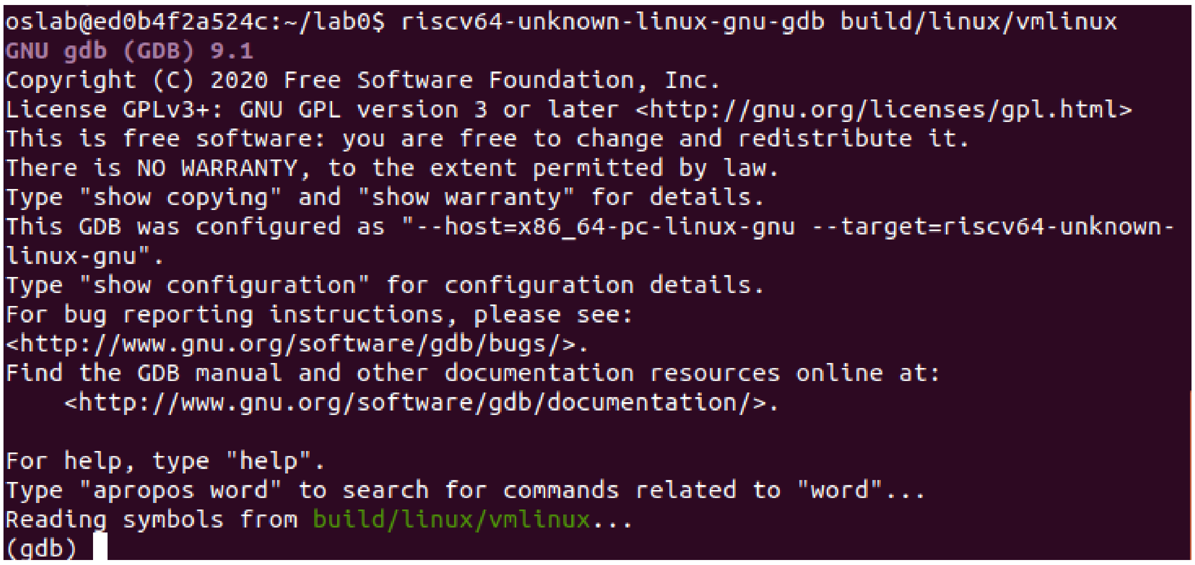
在通过QEMU运行的指令后加-S -s进入调试模式



```
oslab@ed0b4f2a524c:~/lab0$ qemu-system-riscv64 -nographic -machine virt -kernel build/linux/arch/ris
cv/boot/Image   -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0"    -bio
s default -drive file=rootfs.ext4,format=raw,id=hd0  -netdev user,id=net0 -device virtio-net-device,
netdev=net0 -S -s
```

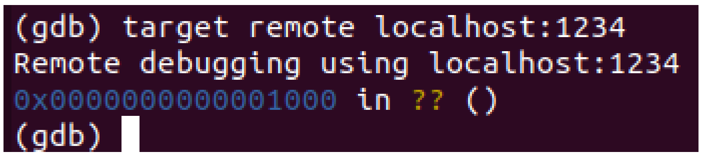
随后打开另一个终端，进入该docker容器，并运行GDB

```
# riscv64-unknown-linux-gnu-gdb build/linux/vmlinux
```



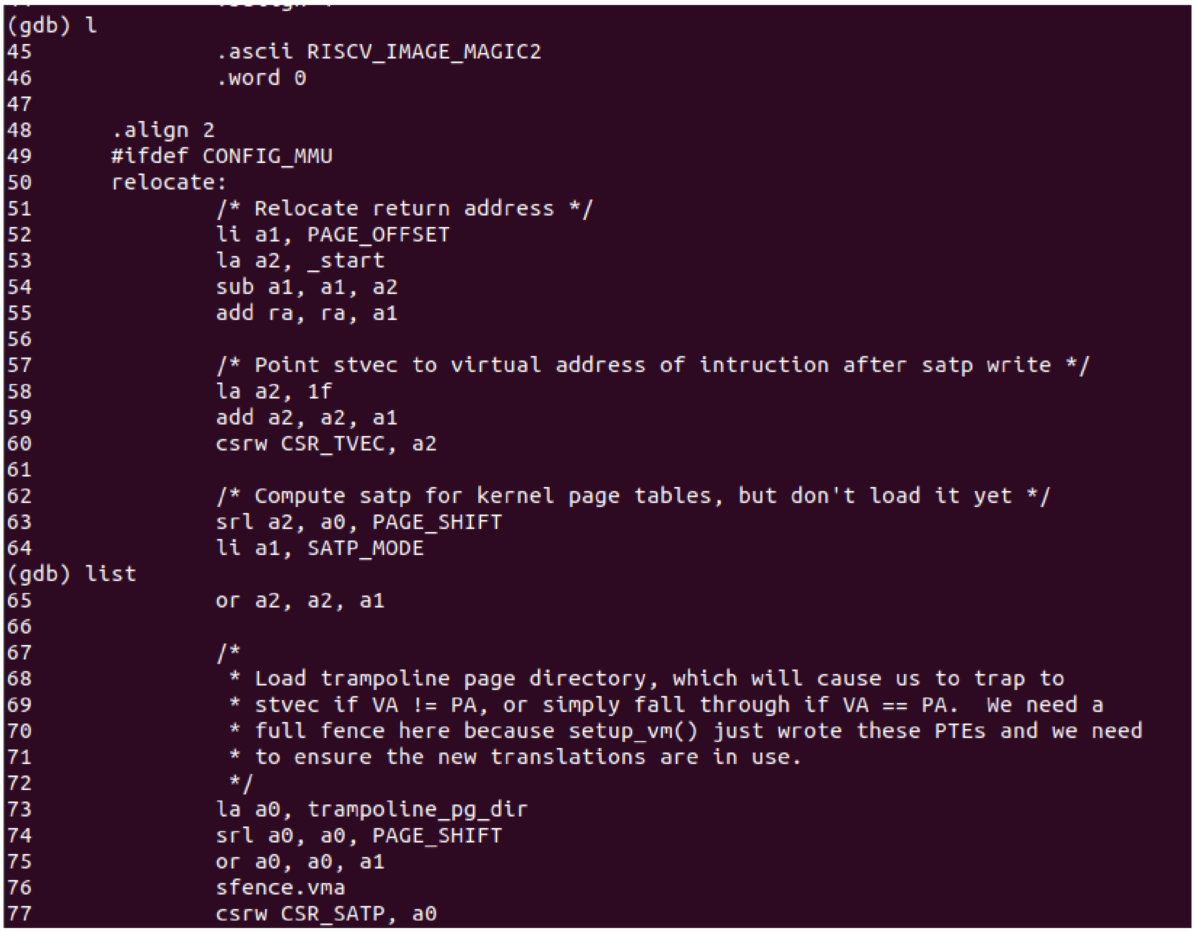
```
oslab@ed0b4f2a524c:~/lab0$ riscv64-unknown-linux-gnu-gdb build/linux/vmlinux
GNU gdb (GDB) 9.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=riscv64-unknown-
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 build/linux/vmlinux...
(gdb) 
```

连接QEMU



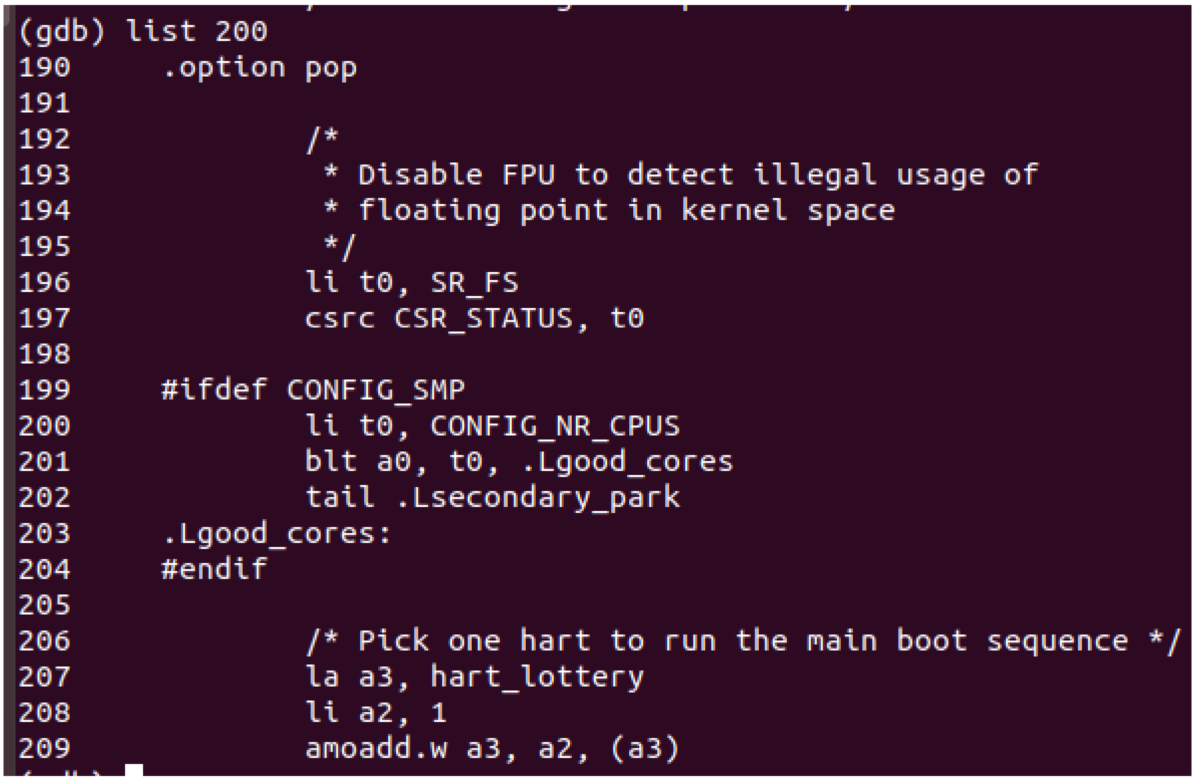
```
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x0000000000001000 in ?? ()
(gdb) 
```

输入 list 或 l 可以查看原代码

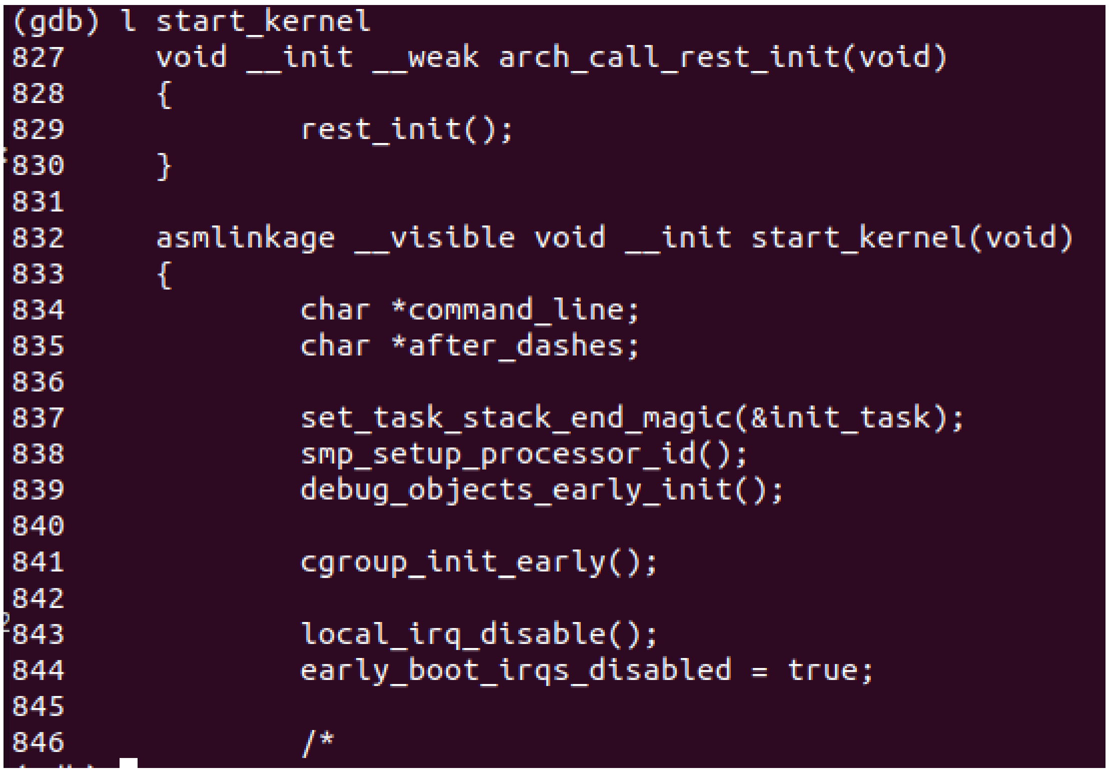


```
(gdb) l
45              .ascii RISCV_IMAGE_MAGIC2
46              .word 0
47
48      .align 2
49      #ifdef CONFIG_MMU
50      relocate:
51              /* Relocate return address */
52              li a1, PAGE_OFFSET
53              la a2, _start
54              sub a1, a1, a2
55              add ra, ra, a1
56
57              /* Point stvec to virtual address of intruction after satp write */
58              la a2, 1f
59              add a2, a2, a1
60              csrw CSR_TVEC, a2
61
62              /* Compute satp for kernel page tables, but don't load it yet */
63              srl a2, a0, PAGE_SHIFT
64              li a1, SATP_MODE
(gdb) list
65              or a2, a2, a1
66
67              /*
68               * Load trampoline page directory, which will cause us to trap to
69               * stvec if VA != PA, or simply fall through if VA == PA.  We need a
70               * full fence here because setup_vm() just wrote these PTEs and we need
71               * to ensure the new translations are in use.
72               */
73              la a0, trampoline_pg_dir
74              srl a0, a0, PAGE_SHIFT
75              or a0, a0, a1
76              sfence.vma
77              csrw CSR_SATP, a0
```

后接数字或函数名可以跳到对应位置



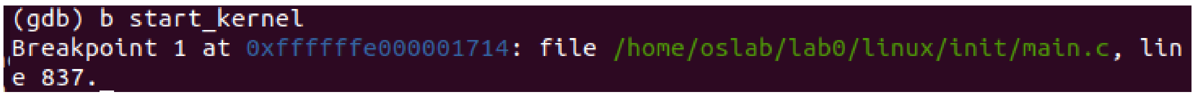
```
(gdb) list 200
190     .option pop
191
192             /*
193              * Disable FPU to detect illegal usage of
194              * floating point in kernel space
195              */
196             li t0, SR_FS
197             csrc CSR_STATUS, t0
198
199     #ifdef CONFIG_SMP
200             li t0, CONFIG_NR_CPUS
201             blt a0, t0, .Lgood_cores
202             tail .Lsecondary_park
203     .Lgood_cores:
204     #endif
205
206             /* Pick one hart to run the main boot sequence */
207             la a3, hart_lottery
208             li a2, 1
209             amoadd.w a3, a2, (a3)
```



```
(gdb) l start_kernel
827     void __init __weak arch_call_rest_init(void)
828     {
829             rest_init();
830     }
831
832     asmlinkage __visible void __init start_kernel(void)
833     {
834             char *command_line;
835             char *after_dashes;
836
837             set_task_stack_end_magic(&init_task);
838             smp_setup_processor_id();
839             debug_objects_early_init();
840
841             cgroup_init_early();
842
843             local_irq_disable();
844             early_boot_irqs_disabled = true;
845
846             /*
```

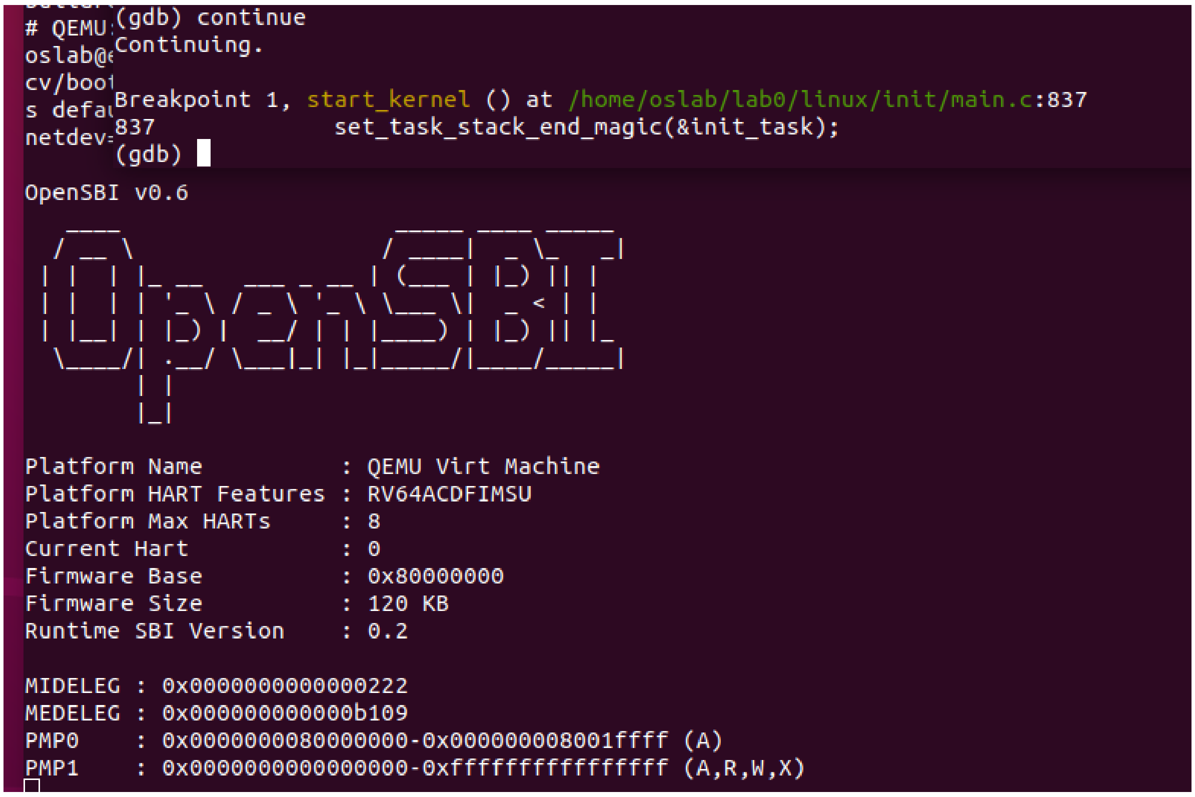
输入 run/r 可以运行正在进行调试的程序

输入 breakpoint/b 后接函数名或行数可以在对应位置设置断点



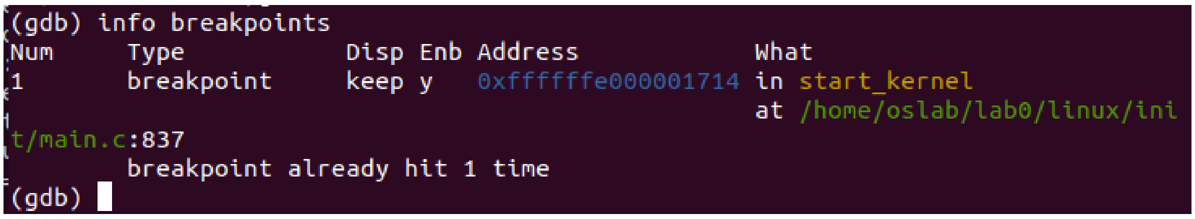
```
(gdb) b start_kernel
Breakpoint 1 at 0xffffffe000001714: file /home/oslab/lab0/linux/init/main.c, lin
e 837.
```

输入 continue 后运行到该程序的下一个断点



```
# QEMU (gdb) continue
oslab@e Continuing.
cv/boot
s defau Breakpoint 1, start_kernel () at /home/oslab/lab0/linux/init/main.c:837
netdev= 837             set_task_stack_end_magic(&init_task);
        (gdb) 

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)
```

info 后接各类信息名称可查看相关信息



```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0xffffffe000001714 in start_kernel
                                                   at /home/oslab/lab0/linux/ini
t/main.c:837
        breakpoint already hit 1 time
(gdb) 
```

print 后接变量名可以查看当前该变量值

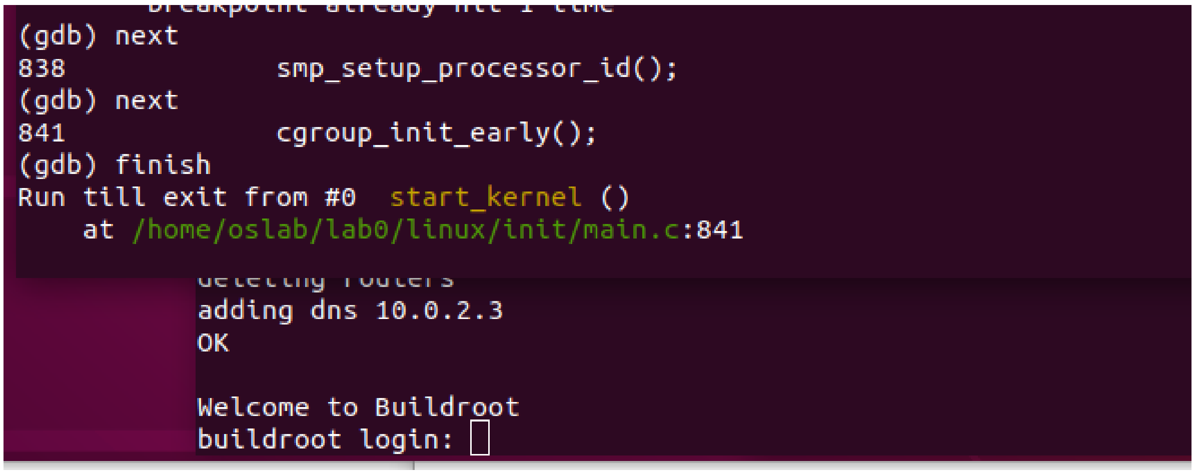
在单步调试中:

next / n 不进入的单步执行

step 进入的单步执行

finish 如果已经进入了某函数，而想退出该函数返回到它的调用函数中，可使用命令finish

until 结束当前循环



```
(gdb) next
838             smp_setup_processor_id();
(gdb) next
841             cgroup_init_early();
(gdb) finish
Run till exit from #0  start_kernel ()
    at /home/oslab/lab0/linux/init/main.c:841
```

```
adding dns 10.0.2.3
OK

Welcome to Buildroot
buildroot login: 
```

kill 异常终止当前 gdb 控制下的程序

quit 退出 gdb

二、讨论、心得

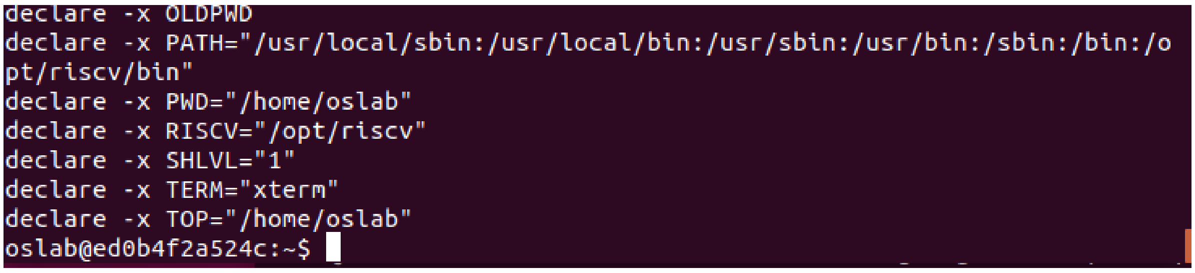
本来觉得每次进入docker都要进行export修改环境变量过于麻烦，想要对环境变量进行永久修改，但修改docker下的 /etc/profile 并没有产生效果，后检查发现若在profile或bashrc文件中按顺序写入

```
export RISCV=/opt/riscv
export PATH=$PATH:$RISCV/bin
```

事实上第二个RISCV并不会被读作/opt/riscv，所以修改该文件时应该写作

```
export RISCV=/opt/riscv
export PATH=$PATH:/opt/riscv/bin
```

重启docker后，输入export查看环境变量，可以看到已经修改完成了。



```
declare -x OLDPWD
declare -x PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/o
pt/riscv/bin"
declare -x PWD="/home/oslab"
declare -x RISCV="/opt/riscv"
declare -x SHLVL="1"
declare -x TERM="xterm"
declare -x TOP="/home/oslab"
oslab@ed0b4f2a524c:~$
```

注：对环境变量的永久更改需要在root权限下进行，在运行docker的指令中将-u 后的 oslab 改为 0 即可以 root 账户运行docker。

本次实验总体来说还比较简单，按实验指导做就可以完成，docker的使用可以查阅相关文档和网上的教程进行。并没有遇到较大的阻碍和困难。