Lab 0: RV64内核调试

1. 实验目的

安装虚拟机及Docker，通过在QEMU模拟器上运行Linux来熟悉如何从源代码开始将内核运行在QEMU模拟器上，学习使用GDB跟QEMU对代码进行联合调试，为后续实验打下基础。

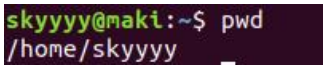
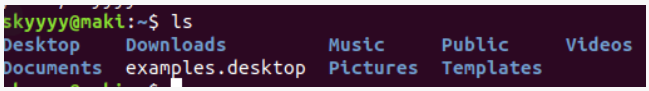
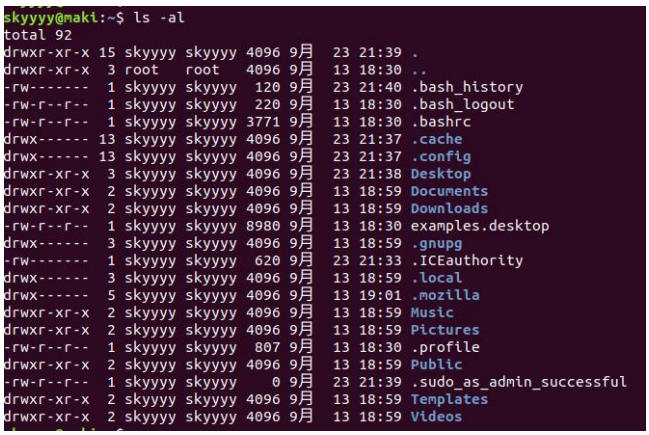
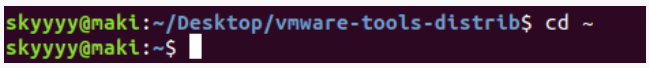
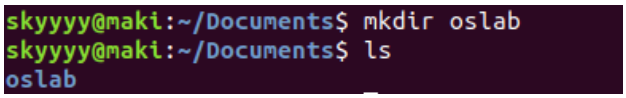
2. 实验内容及要求

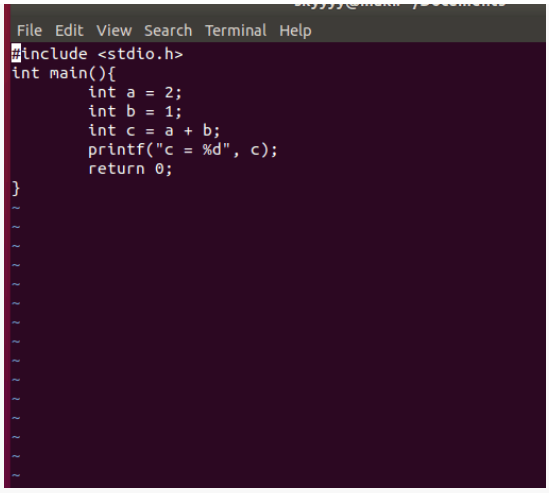
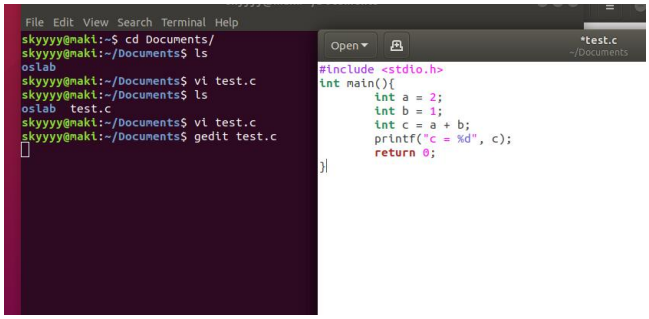
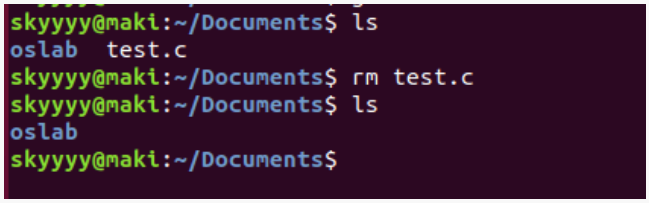
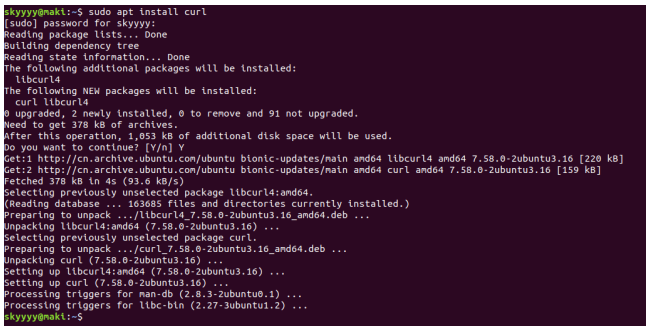
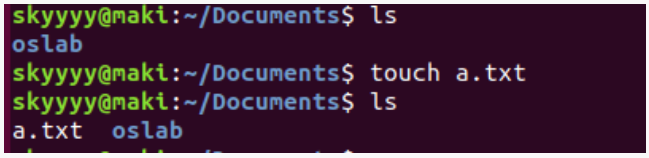
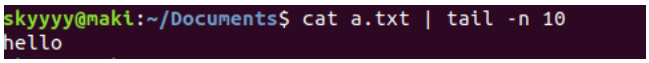
- 安装虚拟机软件、Ubuntu镜像，自行学习Linux基础命令。
- 安装Docker，下载并导入Docker镜像，创建并运行容器。
- 编译内核并用 gdb + QEMU 调试，在内核初始化过程中设置断点，对内核的启动过程进行跟踪，并尝试使用gdb的各项命令。

3. 操作方法和实验步骤

3.1 通过虚拟机安装Linux系统

终端命令

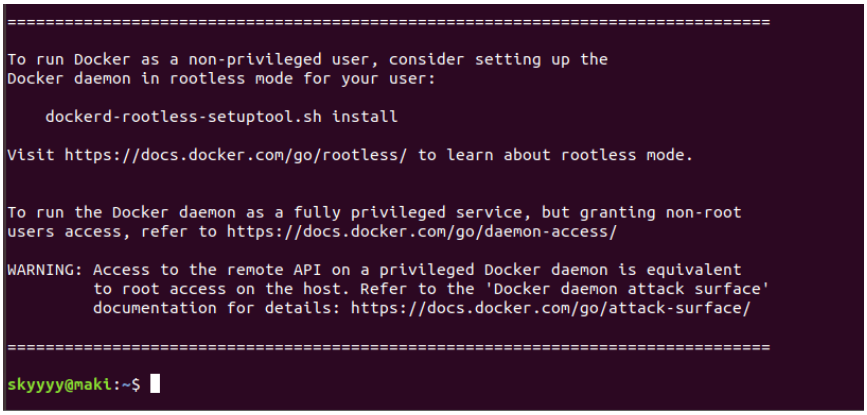
命令	作用	截图
pwd	打印出当前工作目录的名称	
ls	打印出当前工作目录的所有内容	
ls -al	使用长格式打印包括隐藏文件的所有文件及目录信息	
cd ~	切换至当前用户目录	
mkdir oslab	在当前工作目录下创建名为oslab的目录	

vi test.c	用vi编辑器创建并进入名为test.c的文件(vi编辑器命令在此不列)	
gedit test.c	用gedit编辑器打开test.c文件	
rm test.c	删除test.c文件	
sudo apt install curl	连接服务器安装curl	
touch a.txt	在当前工作目录创建a.txt文件	
cat a.txt | tail -n 10	查看a.txt尾部10行的内容	

3.2 安装Docker环境并创建容器

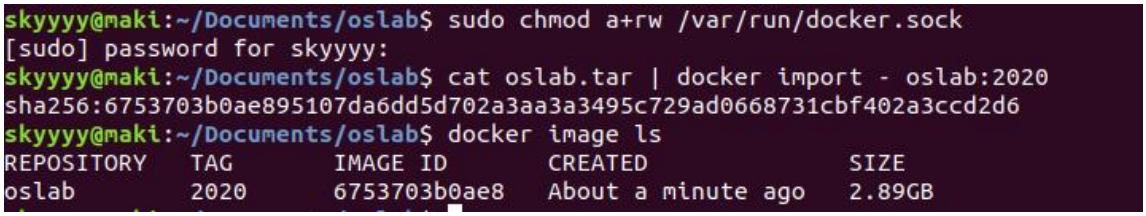
1. **安装docker**

```
### 使用官方安装脚本自动安装docker
$ 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   ### 注销后重新登陆生效
```



2. **下载并导入docker镜像**

```
### 首先进入oslab.tar所在的文件夹，然后使用该命令导入docker镜像
$ cat oslab.tar | docker import - oslab:2020
### 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 at
unix:///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 image ls
```



3. **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 -it oslab:2020 /bin/bash     ### -i:交互式操作 -t:终端
root@6a50e217856f:/#                      ### 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id
root@6a50e217856f:/# exit (或者CTRL+D)     ### 从容器中退出 此时运行docker ps，运行容器的列表为
空
```

```
skyyyy@maki:~/Documents/oslab$ docker run -it oslab:2020 /bin/bash
root@6a50e217856f:/#
root@6a50e217856f:/# exit
exit
```

```
### 查看当前运行的容器
$ docker ps
### 查看所有存在的容器
$ docker ps -a
```

```
skyyyy@maki:~/Documents/oslab$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
skyyyy@maki:~/Documents/oslab$ docker ps -a
CONTAINER ID   IMAGE        COMMAND       CREATED         STATUS                     PORTS     NAMES
6a50e217856f   oslab:2020   "/bin/bash"   3 minutes ago   Exited (0) 3 minutes ago           objective_dirac
```

```
### 启动处于停止状态的容器
$ docker start 6a50      ### 6a50 为容器id的前四位，id开头的几位便可标识一个容器
```

```
skyyyy@maki:~/Documents/oslab$ docker start 6a50
6a50
skyyyy@maki:~/Documents/oslab$ docker ps
CONTAINER ID   IMAGE        COMMAND       CREATED         STATUS          PORTS     NAMES
6a50e217856f   oslab:2020   "/bin/bash"   5 minutes ago   Up 10 seconds             objective_dirac
```

```
### 进入已经运行的容器 oslab的密码为2020
$ docker exec -it -u oslab -w /home/oslab 36 /bin/bash
$ docker exec -it 6a50 /bin/bash
```

```
skyyyy@maki:~/Documents/oslab$ docker exec -it -u oslab -w /home/oslab 6a /bin/bash
oslab@6a50e217856f:~$
```

```
skyyyy@maki:~$ docker exec -it 6a50 /bin/bash
root@6a50e217856f:/#
root@6a50e217856f:/# pwd
/
root@6a50e217856f:/# cd home/oslab
root@6a50e217856f:/home/oslab# ls
lab0
root@6a50e217856f:/home/oslab# cd lab0
root@6a50e217856f:/home/oslab/lab0# export TOP=`pwd`
root@6a50e217856f:/home/oslab/lab0# mkdir -p build/linux
root@6a50e217856f:/home/oslab/lab0# make -C linux O=$TOP/build/linux \
>            CROSS_COMPILE=riscv64-unknown-linux-gnu- \
>            ARCH=riscv CONFIG_DEBUG_INFO=y \
>            defconfig all -j$(nproc)
```

- `docker run -it oslab:2020 /bin/bash`

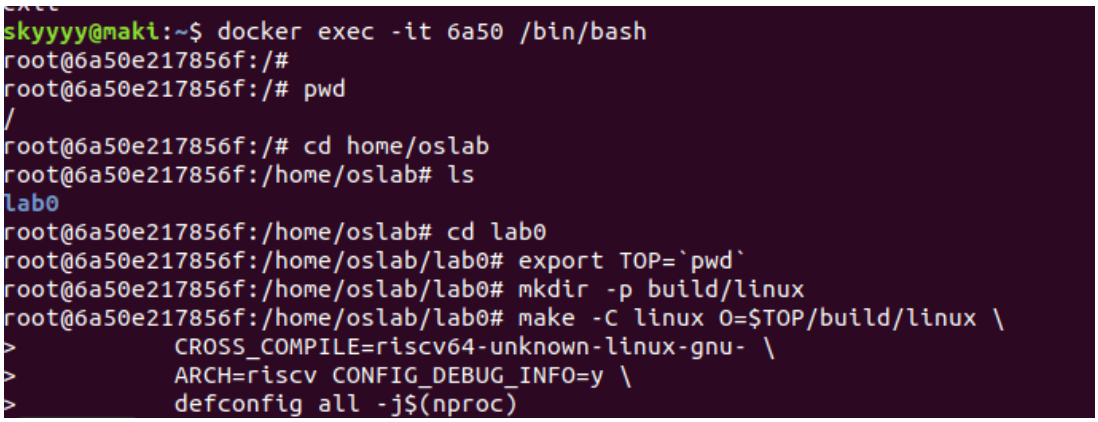
  `docker run`指创建一个新的容器并运行一个命令，`-it`指以交互模式运行容器并为容器分配一个伪输入终端，`oslab:2020`为使用的镜像，`/bin/bash`指在容器内执行`/bin/bash`命令

- `docker exec -it -u oslab -w /home/oslab 36 /bin/bash`

`docker exec`指在运行的容器中执行命令，`-it`指以交互模式运行容器，`-u`为登陆的用户名,`oslab -w /home/oslab`指定工作目录为/home/oslab，`36`为容器id前2位，`/bin/bash`指在容器内执行`/bin/bash`命令

3.3 编译linux内核

```
# pwd
/home/oslab
# cd lab0
# export TOP=`pwd`
# 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)
```



```
  CC      drivers/gpu/drm/drm_syncobj.o
  CC      drivers/gpu/drm/drm_lease.o
  CC      drivers/gpu/drm/drm_writeback.o
  CC      drivers/gpu/drm/drm_client.o
  AR      drivers/of/built-in.a
  CC      drivers/gpu/drm/drm_client_modeset.o
  CC      drivers/gpu/drm/drm_atomic_uapi.o
  CC      drivers/gpu/drm/drm_hdcp.o
  CC      drivers/gpu/drm/drm_managed.o
  CC      drivers/gpu/drm/drm_gem_shmem_helper.o
  CC      drivers/gpu/drm/drm_panel.o
  CC      drivers/mmc/core/queue.o
  CC      drivers/gpu/drm/drm_of.o
  CC      drivers/gpu/drm/drm_pci.o
  CC      drivers/gpu/drm/drm_debugfs.o
  CC      drivers/gpu/drm/drm_debugfs_crc.o
  CC      drivers/gpu/drm/drm_panel_orientation_quirks.o
  AR      fs/built-in.a
  AR      drivers/mmc/core/built-in.a
  AR      drivers/mmc/built-in.a
  AR      drivers/gpu/drm/built-in.a
  AR      drivers/gpu/built-in.a
  AR      drivers/built-in.a
  GEN     .version
  CHK     include/generated/compile.h
  LD      vmlinux.o
  MODPOST vmlinux.symvers
  MODINFO modules.builtin.modinfo
  GEN     modules.builtin
  LD      .tmp_vmlinux.kallsyms1
  KSYM    .tmp_vmlinux.kallsyms1.o
  LD      .tmp_vmlinux.kallsyms2
  KSYM    .tmp_vmlinux.kallsyms2.o
  LD      vmlinux
  SYSMAP  System.map
  MODPOST Module.symvers
  OBJCOPY arch/riscv/boot/Image
  CC [M]  fs/nfs/flexfilelayout/nfs_layout_flexfiles.mod.o
  GZIP    arch/riscv/boot/Image.gz
  LD [M]  fs/nfs/flexfilelayout/nfs_layout_flexfiles.ko
  Kernel: arch/riscv/boot/Image.gz is ready
make[1]: Leaving directory '/home/oslab/lab0/build/linux'
make: Leaving directory '/home/oslab/lab0/linux'
root@6a50e217856f:/home/oslab/lab0#
```

3.4 使用QEMU运行内核

```
# 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
```

```
root@6a50e217856f:/home/oslab/lab0# 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

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)
```

```
Starting network: udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending select for 10.0.2.15
udhcpc: lease of 10.0.2.15 obtained, lease time 86400
deleting routers
adding dns 10.0.2.3
OK

Welcome to Buildroot
buildroot login: 2020
Password:
Login incorrect
buildroot login:
buildroot login: root
#
```

退出qemu

`ctrl+a` `x`

```
#
# QEMU: Terminated
root@6a50e217856f:/home/oslab/lab0#
```

3.5 使用gdb调试内核

Terminal 1

```
# 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 -S -s
```

```
root@6a50e217856f:/home/oslab/lab0# 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 -S -s
```

Terminal 2

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

```
skyyyy@maki:~$ docker exec -it 6a50 /bin/bash
root@6a50e217856f:/# cd home/oslab/lab0
root@6a50e217856f:/home/oslab/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)
```

执行gdb命令

```
(gdb) target remote localhost:1234
```

- 含义：target remote 命令表示远程调试，而1234是默认的用于调试连接的端口号。
- 执行结果：

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

```
(gdb) b start_kernel
(gdb) b *0x80000000
(gdb) b *0x80200000
(gdb) info breakpoints
(gdb) delete 2
(gdb) info breakpoints
```

- 含义：

`b start_kernel` 指设置断点使程序在调用start_kernel函数时断住

`b *0x80000000` 指在地址0x80000000处设置断点

`b *0x80200000` 指在地址0x80200000处设置断点

`info breakpoints`指打印所有已设置断点和相关信息

`delete 2`指删除标号为2的断点

- 执行结果：

```
(gdb) b start_kernel
Breakpoint 1 at 0xffffffe000001714: file /home/oslab/lab0/linux/init/main.c, line 837.
(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   0xffffffe000001714 in start_kernel
                                                   at /home/oslab/lab0/linux/init/main.c:837
2       breakpoint     keep y   0x0000000080000000
3       breakpoint     keep y   0x0000000080200000
(gdb) delete 2
(gdb) delete 2
No breakpoint number 2.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0xffffffe000001714 in start_kernel
                                                   at /home/oslab/lab0/linux/init/main.c:837
3       breakpoint     keep y   0x0000000080200000
(gdb)
```

```
(gdb) continue
(gdb) delete 3
(gdb) continue
(gdb) step
(gdb) s
(gdb) (不做输入，直接回车)
(gdb) next
(gdb) n
(gdb) (不做输入，直接回车)
```

- 含义：

`continue`指从断点处继续运行程序

`delete 3`指删除标号为3的断点

`step/s`指单步进入函数内部并执行完

`next/n`指单步执行下一条语句

- 执行结果：

```
(gdb) continue
Continuing.

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

Breakpoint 1, start_kernel () at /home/oslab/lab0/linux/init/main.c:837
837             set_task_stack_end_magic(&init_task);
(gdb) step
set_task_stack_end_magic (tsk=<optimized out>)
    at /home/oslab/lab0/linux/kernel/fork.c:863
863             *stackend = STACK_END_MAGIC;    /* for overflow detection */
(gdb) s
start_kernel () at /home/oslab/lab0/linux/init/main.c:838
838             smp_setup_processor_id();
(gdb)
smp_setup_processor_id () at /home/oslab/lab0/linux/arch/riscv/kernel/smp.c:38
38              cpuid_to_hartid_map(0) = boot_cpu_hartid;
(gdb) next
start_kernel () at /home/oslab/lab0/linux/init/main.c:841
841             cgroup_init_early();
(gdb) n
843             local_irq_disable();
(gdb)
844             early_boot_irqs_disabled = true;
(gdb)
```

```
(gdb) disassemble
(gdb) nexti
(gdb) n
(gdb) stepi
(gdb) s
```

- 含义：

`disassemble`指反汇编pc附近的函数，即start_kernel

`nexti`指单步执行一条机器指令

`stepi`指单步进入一条机器指令

- 执行结果：

```
(gdb) disassemble
Dump of assembler code for function start_kernel:
   0xffffffe000001714 <+0>:     addi    sp,sp,-80
   0xffffffe000001716 <+2>:     sd      ra,72(sp)
   0xffffffe000001718 <+4>:     sd      s0,64(sp)
   0xffffffe00000171a <+6>:     sd      s1,56(sp)
   0xffffffe00000171c <+8>:     addi    s0,sp,80
   0xffffffe00000171e <+10>:    sd      s2,48(sp)
   0xffffffe000001720 <+12>:    sd      s3,40(sp)
   0xffffffe000001722 <+14>:    sd      s4,32(sp)
   0xffffffe000001724 <+16>:    sd      s5,24(sp)
   0xffffffe000001726 <+18>:    sd      s6,16(sp)
   0xffffffe000001728 <+20>:    auipc   a0,0x100a
   0xffffffe00000172c <+24>:    addi    a0,a0,1560 # 0xffffffe00100bd40 <init_task>
   0xffffffe000001730 <+28>:    auipc   ra,0x205
   0xffffffe000001734 <+32>:    jalr    92(ra) # 0xffffffe00020678c <set_task_stack_end_magic>
   0xffffffe000001738 <+36>:    jal     ra,0xffffffe000003730 <smp_setup_processor_id>
   0xffffffe00000173c <+40>:    jal     ra,0xffffffe000008d4e <cgroup_init_early>
   0xffffffe000001740 <+44>:    csrci   sstatus,2
=> 0xffffffe000001744 <+48>:    li      a5,1
   0xffffffe000001746 <+50>:    auipc   a4,0x106f
   0xffffffe00000174a <+54>:    sb      a5,-1786(a4) # 0xffffffe00107004c <early_boot_irqs_disabled>
   0xffffffe00000174e <+58>:    jal     ra,0xffffffe000004606 <boot_cpu_init>
   0xffffffe000001752 <+62>:    auipc   a1,0x9ff
   0xffffffe000001756 <+66>:    addi    a1,a1,-1682 # 0xffffffe000a000c0 <linux_banner>
   0xffffffe00000175a <+70>:    auipc   a0,0xb3d
   0xffffffe00000175e <+74>:    addi    a0,a0,-850 # 0xffffffe000b3e408
   0xffffffe000001762 <+78>:    auipc   ra,0x245
   0xffffffe000001766 <+82>:    jalr    -510(ra) # 0xffffffe000246564 <printk>
   0xffffffe00000176a <+86>:    addi    a0,s0,-72
   0xffffffe00000176e <+90>:    auipc   s4,0x106f
   0xffffffe000001772 <+94>:    addi    s4,s4,-1798 # 0xffffffe001070068 <initrd_end>
   0xffffffe000001776 <+98>:    jal     ra,0xffffffe0000033b4 <setup_arch>
   0xffffffe00000177a <+102>:   ld      s3,0(s4)
   0xffffffe00000177e <+106>:   auipc   s2,0x106f
   0xffffffe000001782 <+110>:   addi    s2,s2,-1806 # 0xffffffe001070070 <initrd_start>
   0xffffffe000001786 <+114>:   beqz    s3,0xffffffe0000017d4 <start_kernel+192>
   0xffffffe00000178a <+118>:   addi    s1,s3,-12
   0xffffffe00000178e <+122>:   li      a2,12
   0xffffffe000001790 <+124>:   auipc   a1,0xb3d
   0xffffffe000001794 <+128>:   addi    a1,a1,-896 # 0xffffffe000b3e410
   0xffffffe000001798 <+132>:   mv      a0,s1
   0xffffffe00000179a <+134>:   auipc   ra,0x47f
   0xffffffe00000179e <+138>:   jalr    2030(ra) # 0xffffffe000480f88 <memcmp>
--Type <RET> for more, q to quit, c to continue without paging--
```

```
Quit
(gdb) nexti
0xffffffe000001746      844             early_boot_irqs_disabled = true;
(gdb) n
850             boot_cpu_init();
(gdb) stepi
boot_cpu_init () at /home/oslab/lab0/linux/arch/riscv/include/asm/current.h:31
31              return riscv_current_is_tp;
(gdb) s
2492            set_cpu_online(cpu, true);
(gdb)
```

- `nexti/stepi`和`next/step`区别在于前者作用于机器指令层面，后者作用于源码层面；
- `next`和`step`区别在于前者需要单步执行下一条指令，后者单步进入下一条指令

```
(gdb) continue
#由于此时无断点，continue将一直执行下去，为了退出gdb，可以：①在gdb中按住ctrl+c退出当前正在运行的gdb命令，然后再用quit退出②在qemu中先按ctrl+a，松开后再按x先退出qemu，然后在gdb中quit退出。
(gdb) quit
```

- 含义：

`quit`退出gdb

- 执行结果：

```
2492            set_cpu_online(cpu, true);
(gdb) continue
Continuing.
^C
Program received signal SIGINT, Interrupt.
arch_cpu_idle () at /home/oslab/lab0/linux/arch/riscv/kernel/process.c:33
33              local_irq_enable();
(gdb) quit
A debugging session is active.

        Inferior 1 [process 1] will be detached.

Quit anyway? (y or n) y
Detaching from program: /home/oslab/lab0/build/linux/vmlinux, process 1
Ending remote debugging.
[Inferior 1 (process 1) detached]
root@6a50e217856f:/home/oslab/lab0#
```

4. 讨论和心得

注意容器挂载