**浙江大学实验报告**

课程名称： 操作系统分析及实验 实验类型： 综合型/设计性

实验项目名称： docker+GDB+QEMU调试64位RISC-V LINUX

学生姓名： 李云帆 专业： 计算机科学与技术 学号： 3200102555

电子邮件地址： 3200102555@zju.edu.cn 手机： 17300988837

实验日期： 2022 年 9 月 13 日

**一、实验目的**

搭建虚拟机、docker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。

**二、实验内容**

重新编译内核是一件比你想像的还要简单的事情，它甚至不需要你对内核有任何的了解，只要你具备一些基本的Linux操作系统的知识就可以进行。

本次实验，要求你在RedHat Fedora Core 5的Linux系统里，下载并重新编译其内核源代码；然后，配置GNU的启动引导工具grub，成功运行你刚刚编译成功的Linux内核。

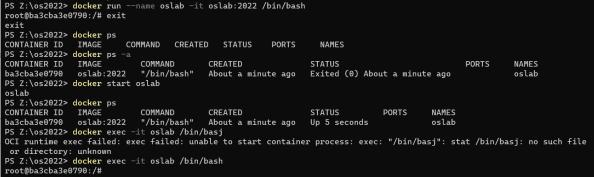
1. **主要仪器设备（必填）**windows11, intel i7-10750h
2. **操作方法和实验步骤**
3. **安装Docker环境并创建容器**
   1. **安装Docker并启动**参照 <https://docs.docker.com/get-docker/> 自行在本机安装 Docker 环境，安装完成后启动 Docker 软件。
   2. **下载并导入Docker镜像**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # 进入 oslab.tar 所在的文件夹  $ cd path/to/oslab # 替换为你下载文件的实际路径  # 导入docker镜像  $ docker import oslab.tar oslab:2022  # 查看docker镜像  $ docker image ls | | | | | |
| REPOSITORY oslab | TAG  2022 | IMAGE ID 9192b7dc0d06 | CREATED 47 seconds | ago | SIZE  2.89GB |



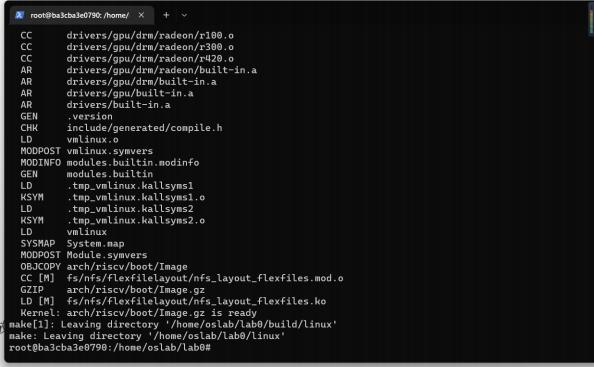
* 1. **从镜像创建一个容器并进入该容器**

|  |
| --- |
| # 从镜像创建一个容器  $ docker run --name oslab -it oslab:2022 /bin/bash  root@95efacf34d2c:/# # 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@95efacf34d2c:/# exit (或者CTRL+D) # 从容器中退出  # 查看当前运行的容器  $ docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  # 查看所有存在的容器  $ docker ps -a  CONTAINER ID IMAGE COMMAND CREATED STATUS  PORTS NAMES  95efacf34d2c oslab:2022 "/bin/bash" About a minute ago Exited (0) About a  minute ago oslab  # 启动处于停止状态的容器  $ docker start oslab  $ docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS  NAMES  95efacf34d2c oslab:2022 "/bin/bash" 2 minutes ago Up 26 seconds  oslab  # 从本地终端进入容器  $ docker exec -it oslab /bin/bash  root@95efacf34d2c:/# |



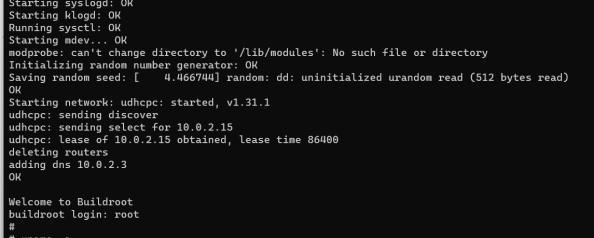
1. **编译Linux内核**

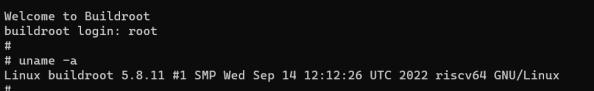
|  |
| --- |
| $ cd /home/oslab/lab0  $ ls  linux rootfs.ext4  $ mkdir -p build/linux  # 编译 Linux 内核  $ make -C linux \  O=/home/oslab/lab0/build/linux \  CROSS\_COMPILE=riscv64-unknown-linux-gnu- \  ARCH=riscv \  CONFIG\_DEBUG\_INFO=y \  defconfig \  all \  -j$(nproc) |



1. **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 |



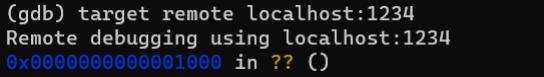
**4. 使用GDB调试内核**首先需要使用 QEMU 模拟运行内核。

|  |
| --- |
| $ 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 \  -S \  -s  # -S: 表示启动时暂停执行，这样我们可以在 GDB 连接后再开始执行  # -s: -gdb tcp::1234 的缩写，会开启一个 tcp 服务，端口为 1234，可以使用 GDB 连接并进行调 试 |

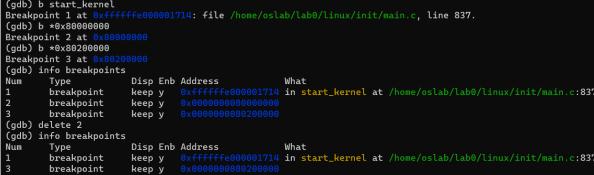
再打开一个终端，进入同一个 **Docker** 容器，并切换到 lab0 目录，使用 GDB 进行调试。

|  |
| --- |
| # 进入同一个 Docker 容器  $ docker exec -it oslab /bin/bash  # 切换到 lab0 目录  $ cd /home/oslab/lab0/  # 使用 GDB 进行调试  $ riscv64-unknown-linux-gnu-gdb build/linux/vmlinux |

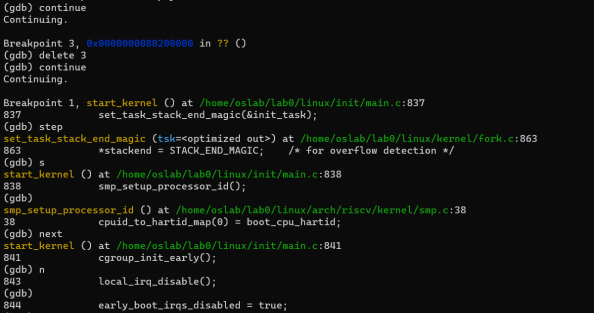
|  |
| --- |
| (gdb) target remote localhost:1234 |



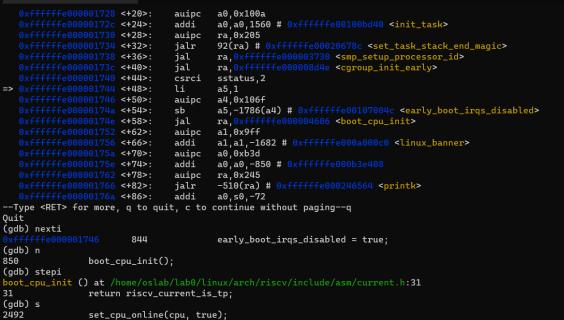
|  |
| --- |
| (gdb) b start\_kernel  (gdb) b \*0x80000000  (gdb) b \*0x80200000  (gdb) info breakpoints  (gdb) delete 2  (gdb) info breakpoints |



|  |
| --- |
| (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) |



|  |
| --- |
| (gdb) disassemble  (gdb) nexti  (gdb) n  (gdb) stepi  (gdb) s |

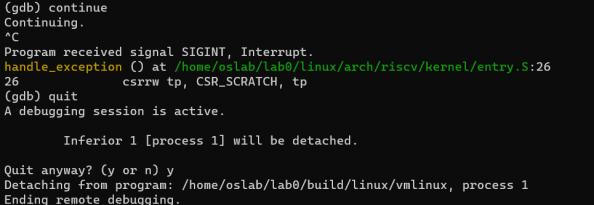






启

：



**五.讨论心得**

在安装docker的时候, 运行docker desktop出现Hradware assisted virtualization and data execution protection must enabled BIOS错误

由于之前用过虚拟机, 所以不会是没有虚拟化的原因, 问题应该在于hyper-v. 通过查询资料发现应该打开hyper-V然而不知道为什么我的windows里面没有这个选项

查阅资料发现该功能被隐藏了, 运行了以下代码即可

|  |
| --- |
| pushd "%~dp0"  dir /b %SystemRoot%\servicing\Packages\\*Hyper-V\*.mum >hyper-v.txt  for /f %%i in ('findstr /i . hyper-v.txt 2^>nul') do dism /online /norestart /add-package:"%SystemRoot%\servicing\Packages\%%i"  del hyper-v.txt  Dism /online /enable-feature /featurename:Microsoft-Hyper-V-All /LimitAccess /ALL |

然后用下面两行命令启用hyper-v并确保其守护进程自动运行就好了

|  |
| --- |
| dism.exe /Online /Enable-Feature:Microsoft-Hyper-V /All  bcdedit /set hypervisorlaunchtype auto |

重启后docker就可以正常运行了.