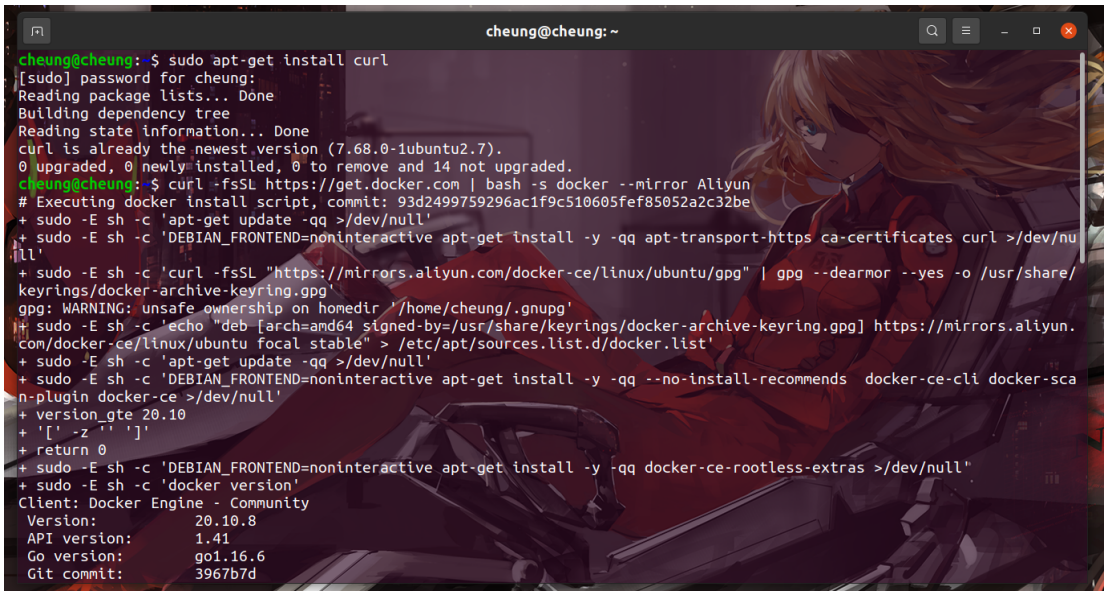
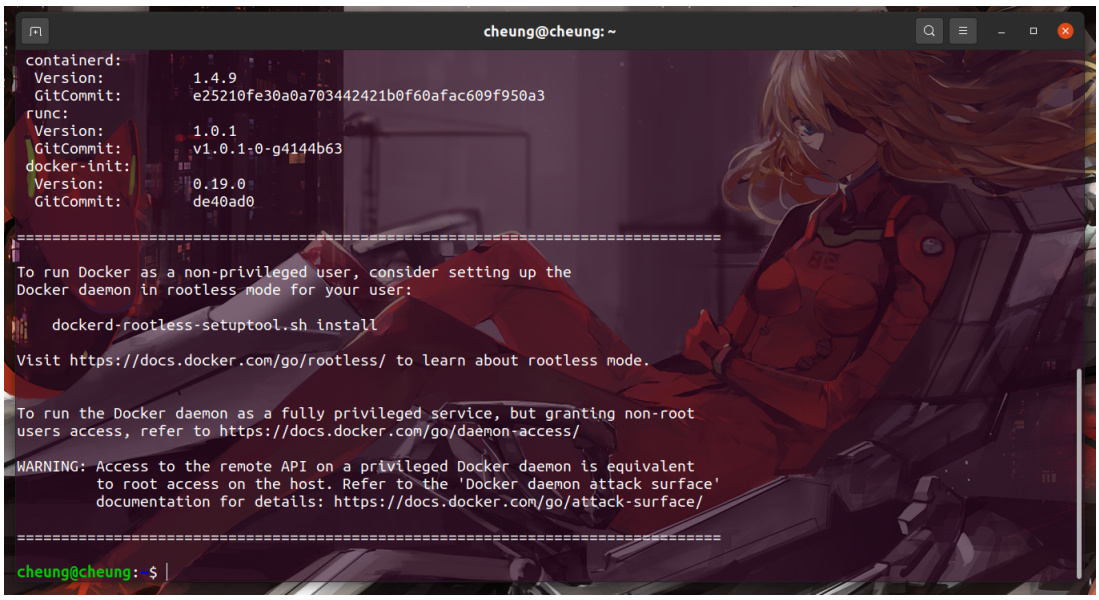
浙江大学实验报告

一、实验内容

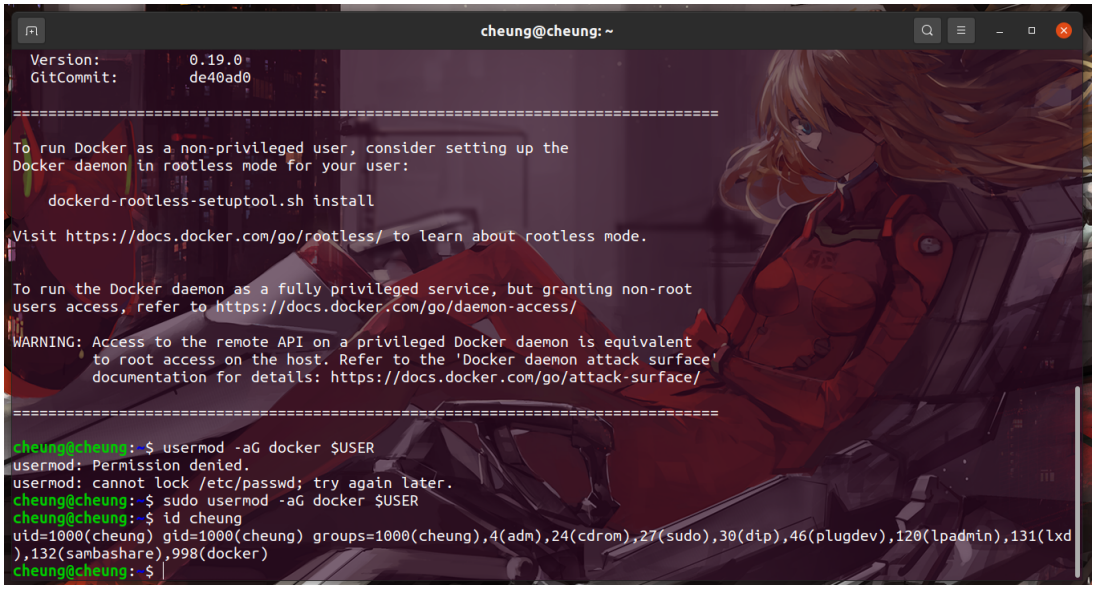
搭建docker环境

首先，根据实验指导，安装curl和docker。如下图，系统提示我已经安装过了curl，并且在安装docker时输出了安装所使用的命令、docker的版本信息等信息。最后，还输出了docker的使用注意事项。

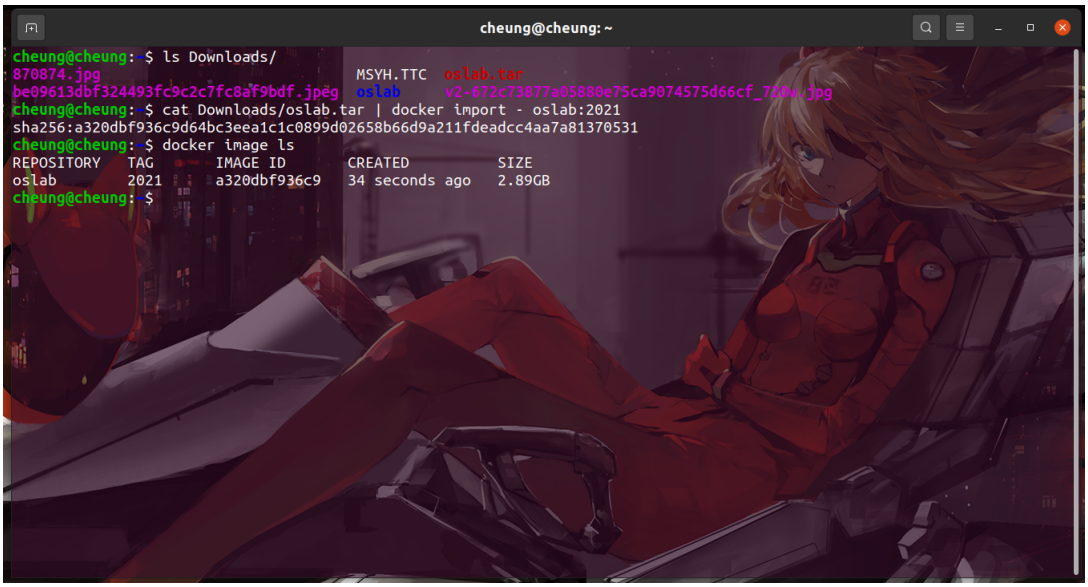




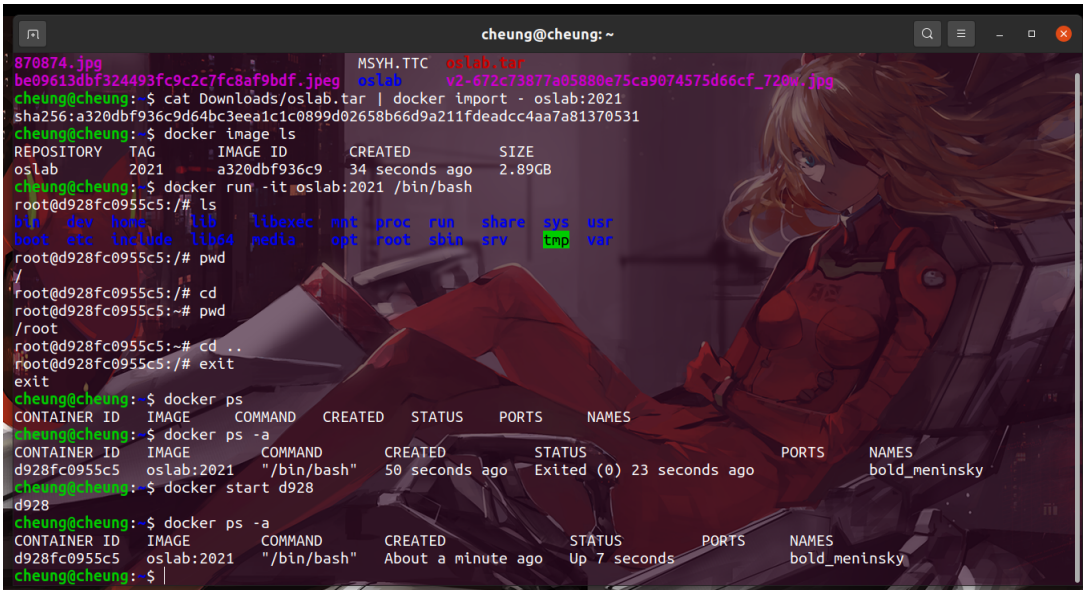
将当前用户`cheung`加入docker的用户组，这里在实践中发现需要给予管理者权限。加入用户组后，利用`id`命令查看当前用户的信息，可以看到当前用户所属的用户组中已经包括了docker。



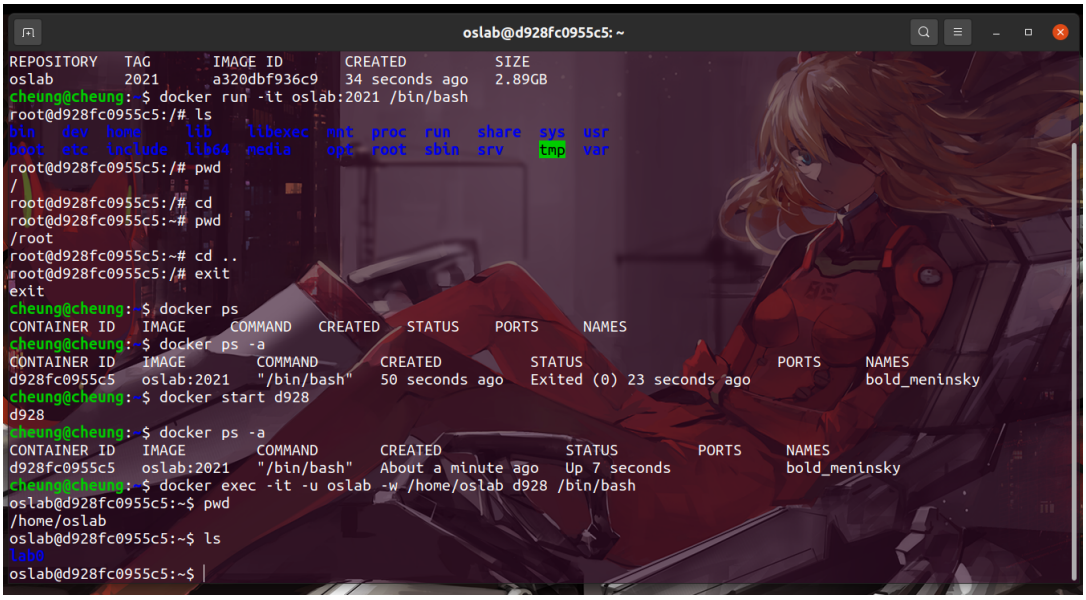
如下图，重启电脑后将老师下发的docker镜像导入。然后通过命令查看镜像，可以看到镜像已经被成功导入。



如下图，创建一个容器，在容器中简单执行若干条命令后退出，再查看所有容器的状态，启动容器后再次查看所有容器的状态。可以看到创建的容器的id为d928开头的字符串。在容器再次启动前，容器的状态为`Exited(0)`，启动后状态为`Up`。



如下图，进入正在运行的容器。登入时，我们以`oslab`用户的身份登入，可以看到登入后显示的用户名为`oslab`。简单运行两条命令，容器运行正常。



编译 linux 内核

如下图，根据实验指导执行一系列命令。在环境变量创建后通过`echo`命令检验其创建成功。在编译时，系统会输出许多信息。

如下图，第一次执行的时候，编译并未成功，报错为找不到编译器。经过检查，发现是我误解实验指导pdf文件的排版所致。命令中的一个空格恰好出现在了pdf文件中的空格位置，我输入时并未发现这个问题，导致命令找不到对应的gcc。

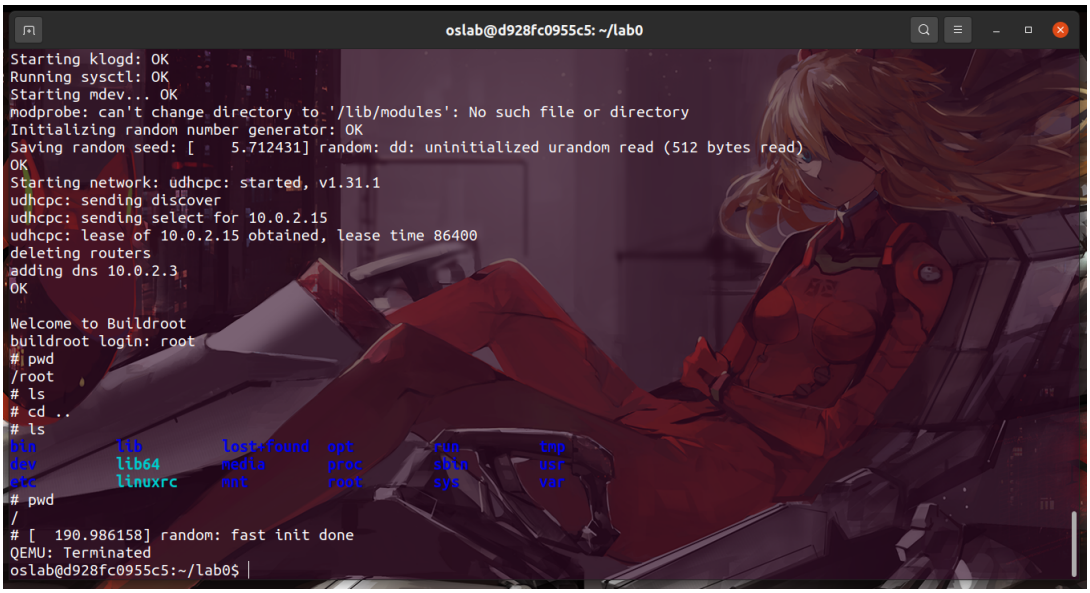
如下图，重新输入命令并注意ARCH前的空格。这一次，可以看到命令顺利执行，出现了大量输出信息。

等待命令执行完毕，进入`build/linux/arch/riscv/boot`目录下查看，可以看到出现了内核文件。编译linux内核成功。

使用QEMU运行内核

输入命令，使用QEMU运行内核。系统会输出许多初始化信息。

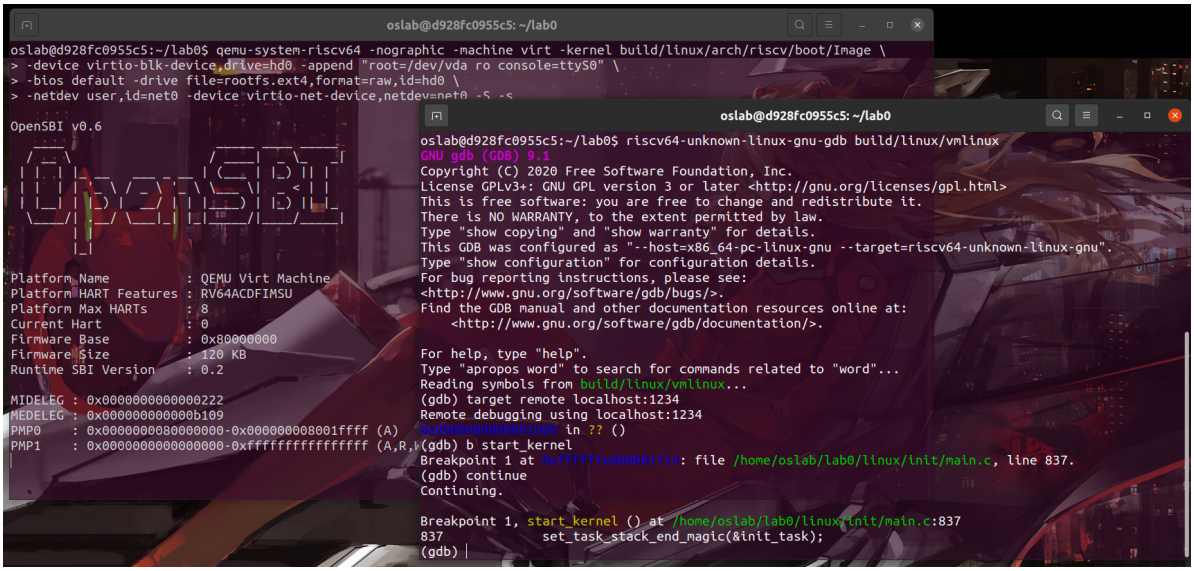
等待系统提示输入用户名，以`root`身份登入后，可以简单执行几条命令。由命令行的提示符可以看出，已经进入了QEMU模拟器。最后退出模拟器。



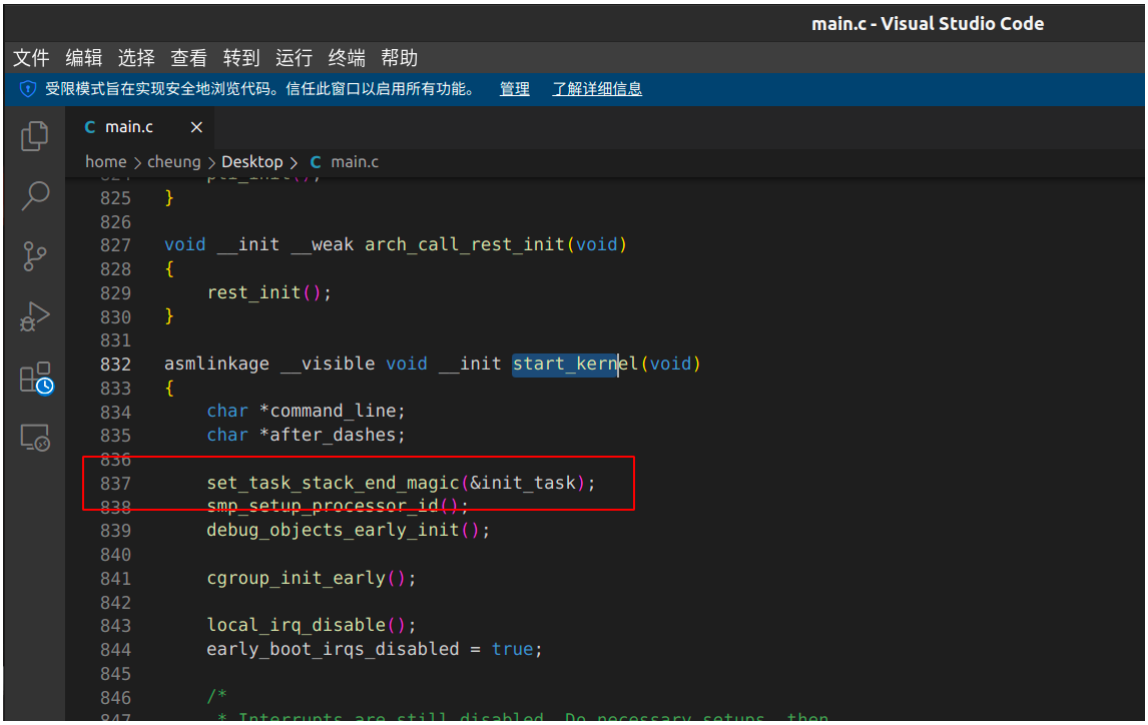
使用 gdb 对内核进行调试

在一个终端中进入模拟器，可以看到因为指定了`-S -s`参数，模拟器没有运行，没有任何输出。

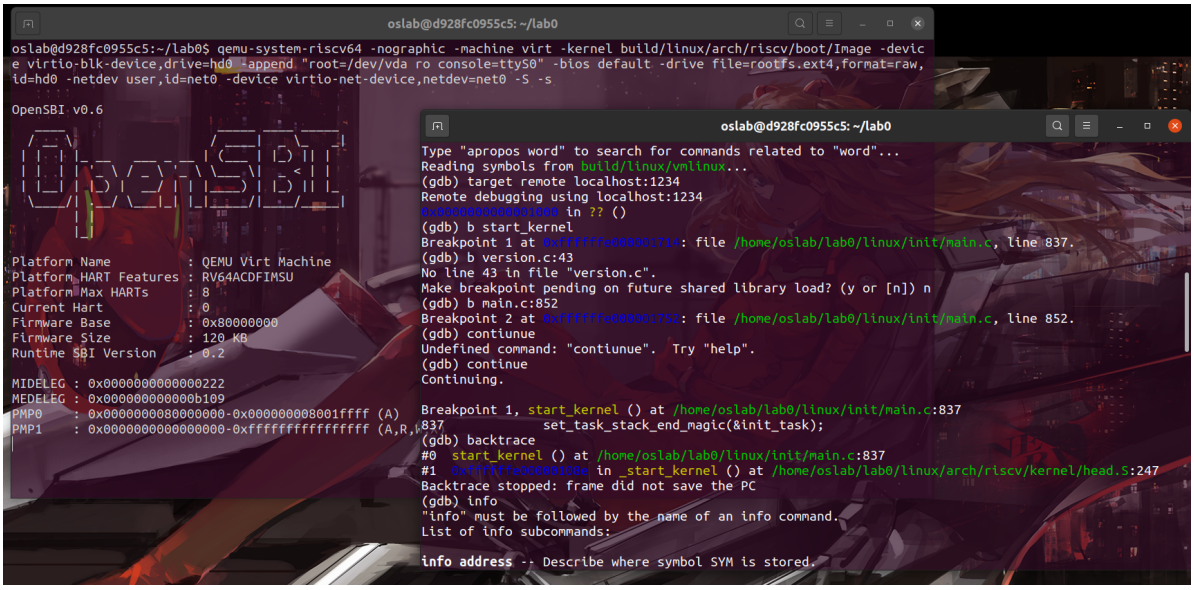
根据实验指导，在另一个终端进入容器。连接上QEMU后，设置断点并继续执行。运行至断点处，可以看到因为断点，模拟器在输出初始化信息的过程中停了下来。同时，gdb显示断点在`main.c`中的837行。



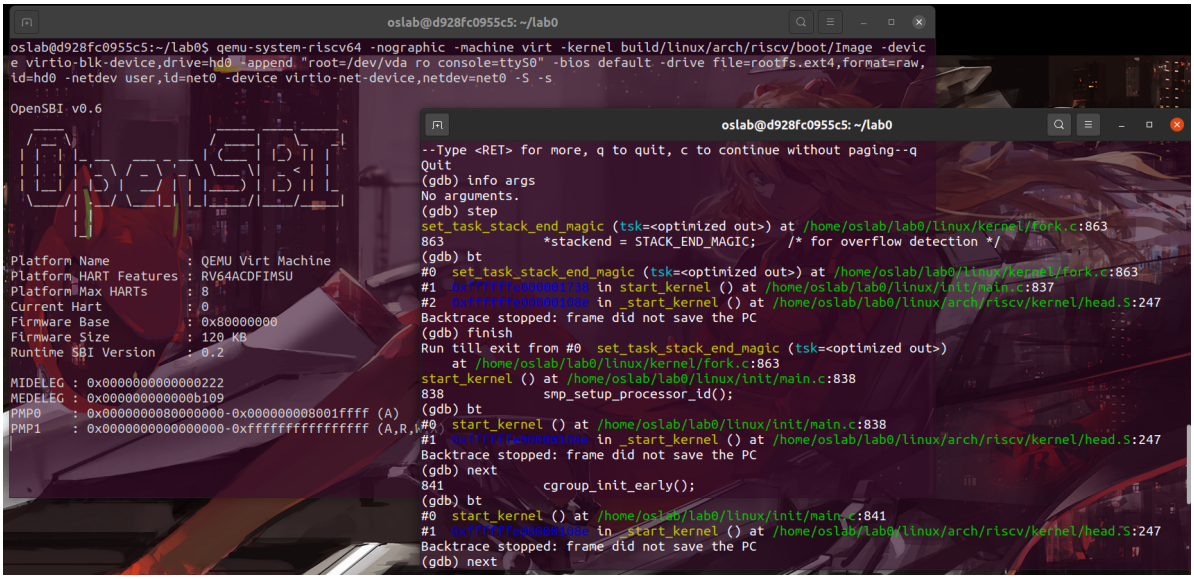
根据gdb给出的信息，打开源文件。可以看到断点设置在了该函数的第一个执行语句处。



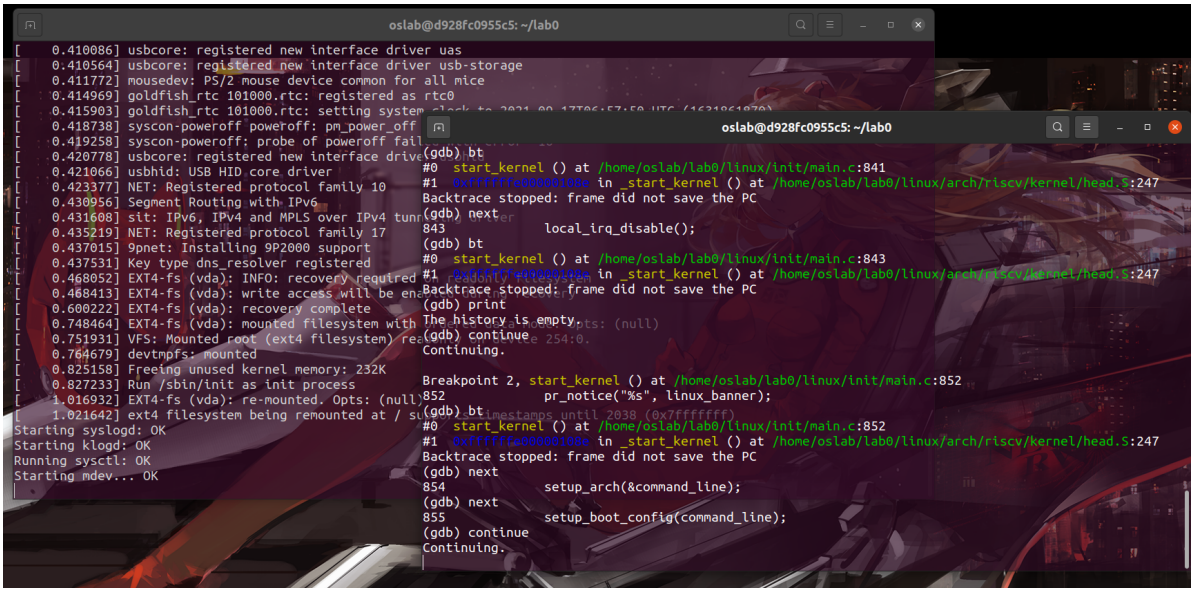
查看目录下的其他源文件，再选取一两个位置设置断点。除了实验指导中的断点，我还在`version.c`的43行和`main.c`的852行设置了断点。但是因为不甚了解源文件的结构，`version.c`断点处似乎并不会被执行，因此我没有让它在未来链接时设置断点，而只增加设置了第二个断点。内核初始化至第一个断点时，通过`backtrace`或者`b`命令，可以看到此处有两层函数调用。



在此处，通过`info args`查看该函数的参数，显示没有参数，这与上述源代码中的截图一致（该函数没有传入参数）。通过`step`或`s`命令进入函数，再次查看函数调用层级，可以看到比`start_kernel`中的多了一层。通过`finish`命令结束该函数的运行后，再次查看调用函数层级，可以看到又回到了两层。通过`next`或`n`命令单步执行，可以看到行数的变化。



通过`continue`命令一直执行，执行到第二个断点处停下。简单查看信息后，再次`continue`。此后不再有断点，可以看到模拟器正常执行至出现命令行提示符。



二、讨论心得

本次实验主要遇到两个问题，第一个问题是命令的空格问题。由于pdf的排版原因，编译命令与启动QEMU命令都有空格出现在回车处，一开始因为没有彻底理解命令每个参数的意义，我没有键入空格，导致错误。在了解命令的使用方法，以及各参数的意义后，我才加入空格，正确执行了命令。在这个过程中，实验指导中的链接很好地帮助了我理解命令的结构、docker以及qemu。

第二个问题是容器的环境问题。在进行调试时我发现，退出一个容器后再登入，容器会无法使用gcc等工具。经过查找资料与询问助教，我发现这是因为容器的环境变量又回到了初始状态。此前步骤中我手动添加的三个环境变量都被重置了，在我重新添加三个环境变量后，模拟器又可以正常运行。此外，我还检查了之前编译内核时产生的文件，发现文件没有被删除。似乎，重新进入容器时，**环境变量等环境会被重置**，但是**容器中的文件不会被重置**。

而要解决这个问题，助教给我的解决方案是将环境变量设置挂载到一个外部目录。一个我正在采用且不太直接的方案是，准备一个脚本文件`pre.sh`，内有每次进入容器都要执行的命令如下：

```
cd lab0
export TOP=`pwd`
export RISCV=/opt/riscv
export $PATH=$PATH:$RISCV/bin
```

进入容器后，首先执行`source pre.sh`命令，即可利用脚本将环境变量设置好。

与之类似的一个方法是直接在`.bashrc`文件内写好命令。