## Lab2\_通过QEMU调试ARMv8的Linux内核

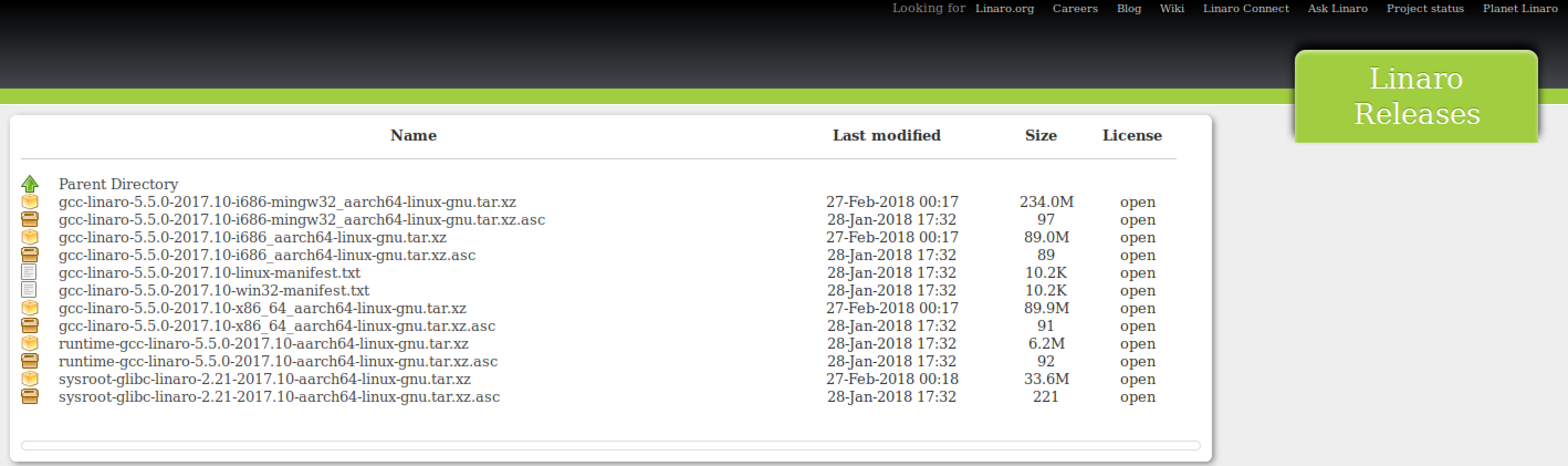
熟悉如何使用QEMU调试ARMv8的 Linux内核。

### 基本实验步骤说明：

由于书上只讲了如何在qemu里如何运行aarch64系统，没有讲述如何使用gdb来单步调试。这部分内容，将在这里做一些补充说明。

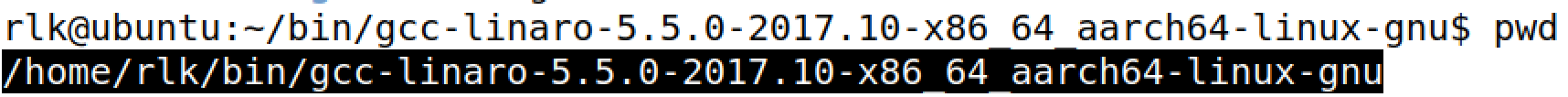
由于ubuntu 18.04里面的aarch64工具链已经没有了gdb，因为都使用gdb-multiarch这个工具，但是gdb-multiarch在单步调试aarch64的时候有问题，所以我们需要单独下载一个aarch64-gdb的工具。

到linaro官网上下载。



|  |
| --- |
| #wget http://releases.linaro.org/components/toolchain/binaries/5.5-2017.10/aarch64-linux-gnu/gcc-linaro-5.5.0-2017.10-x86\_64\_aarch64-linux-gnu.tar.xz |

然后解压到home目录下面，我建议大家在home目录下面建立bin文件夹，解压这里。比如我的路径如下：

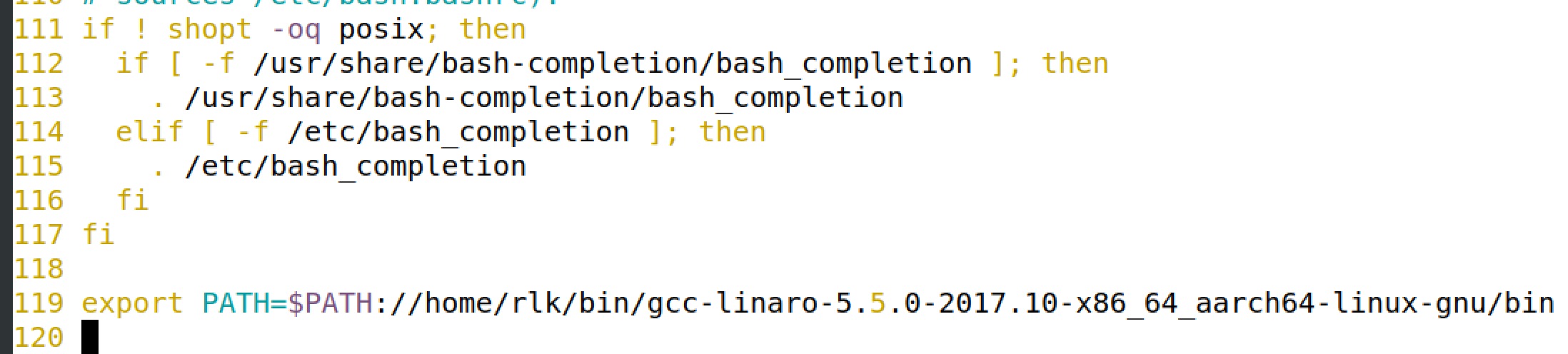


然后设置系统PATH路径。

|  |
| --- |
| # vim ~/.bashrc |

添加如下一行语句。

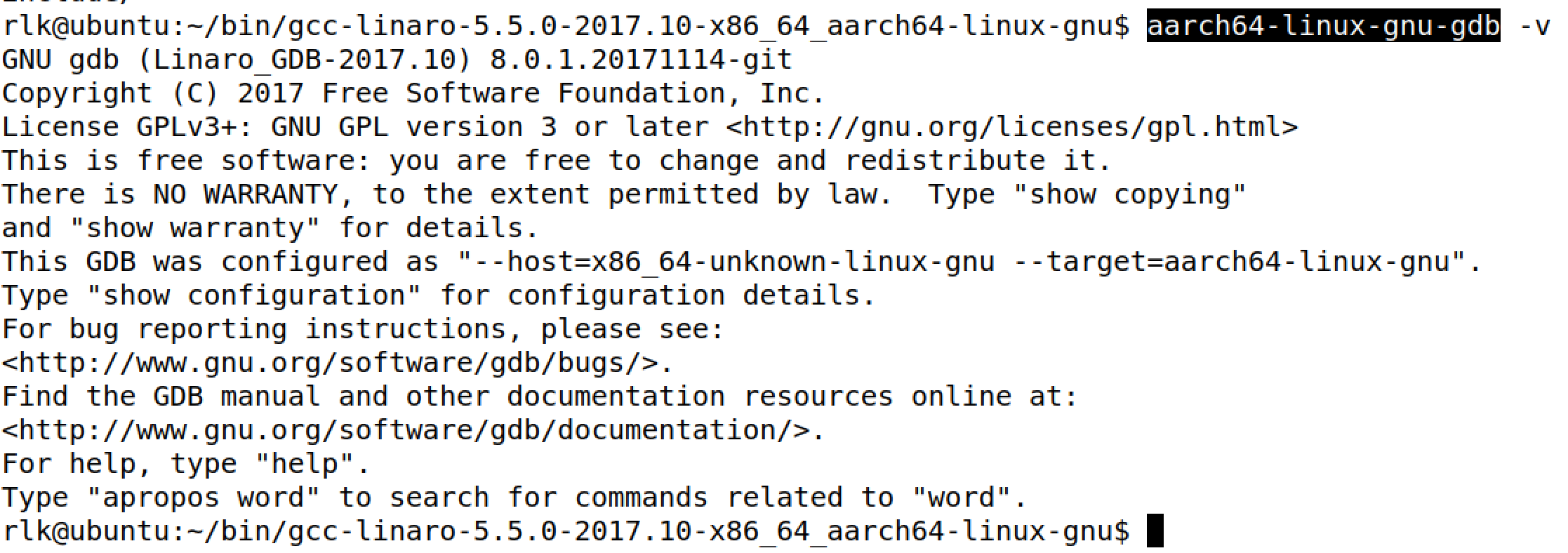
|  |
| --- |
| export PATH=$PATH:/home/rlk/bin/gcc-linaro-5.5.0-2017.10-x86\_64\_aarch64-linux-gnu/bin |



保存之后，运行如下命令。

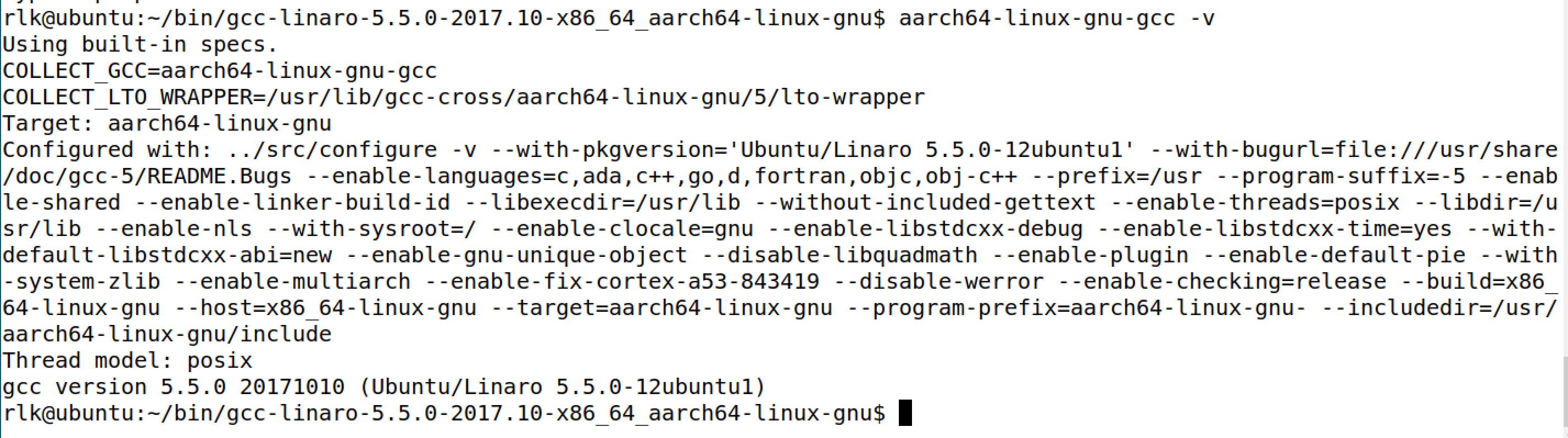
|  |
| --- |
| # source ~/.bashrc |

然后查看aarch64-linux-gnu-gdb命令是否安装正确。



下面来进行单步调试aarch64的内核。

首先保证aarch64-linux-gnu-gcc的版本是5.x的。



下面开始编译内核，首先设置console节点。

$ cd runninglinuxkernel\_4.0

$ cd \_install\_arm64

$ mkdir dev

$ sudo mknod console c 5 1

然后开始编译内核。

$ export ARCH=arm64

$ export CROSS\_COMPILE= aarch64-linux-gnu-

$ make defconfig

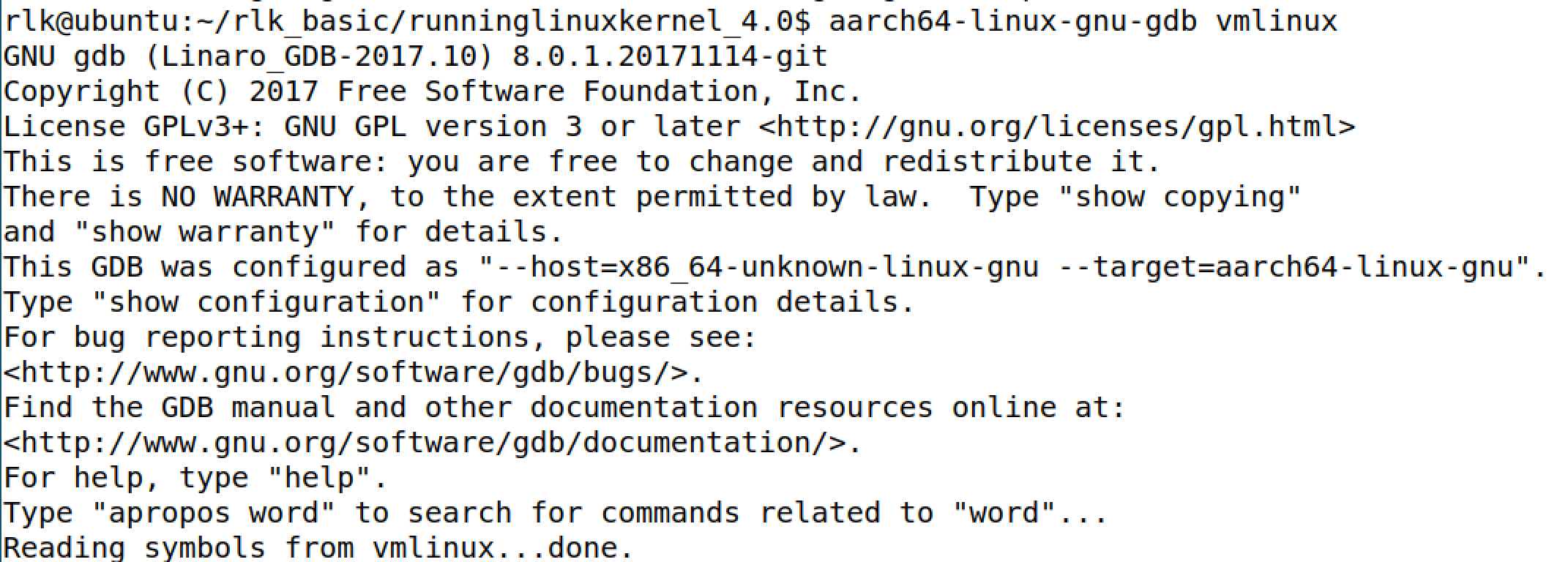
Make -j8

下面开始调试内核。

|  |
| --- |
| #qemu-system-aarch64 -machine virt -cpu cortex-a57 -machine type=virt -nographic -m 100 -kernel arch/arm64/boot/Image --append "rdinit=/ linuxrc console=ttyAMA0" -S -s |

在另一个终端：

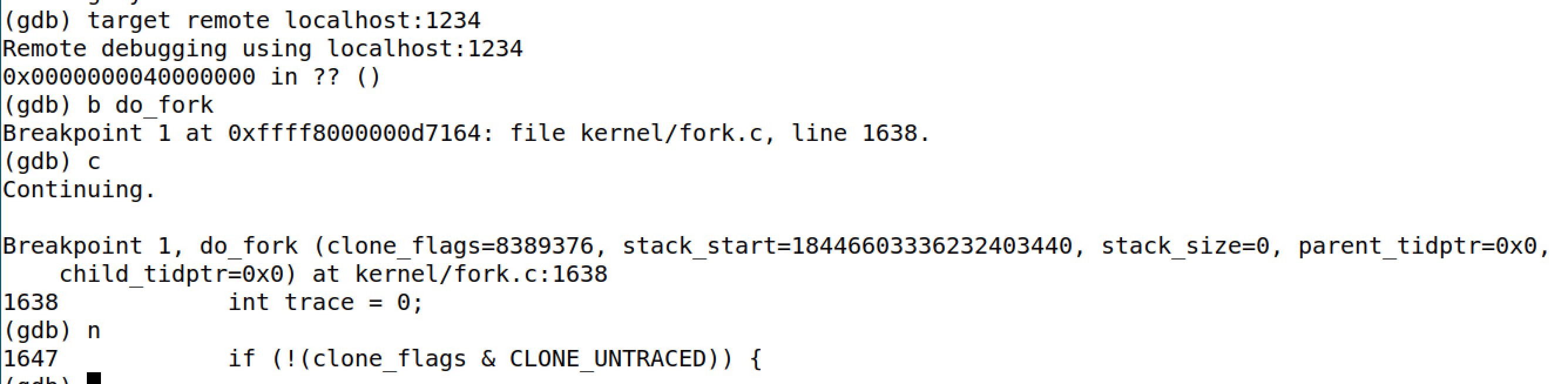
|  |
| --- |
| # cd runninglinuxkernel\_4.0  # aarch64-linux-gnu-gdb vmlinux |



在gdb命令行里输入：

|  |
| --- |
| (gdb) target remote localhost:1234  Remote debugging using localhost:1234  0x0000000040000000 in ?? ()  (gdb) b do\_fork  Breakpoint 1 at 0xffff8000000d7164: file kernel/fork.c, line 1638.  (gdb) c |

然后就会看到停在断点里。



接下来就可以单步调试aarch64内核啦！

请关注奔跑吧Linux社区微信公众号，里面有更多更有料更好玩的Linux干货：



奔跑吧Linux社区的旗舰篇视频节目，帮您打通Linux和职场任督二脉！





笨叔最新力作，全程约5小时高清，140多页ppt，8大实验，基于x86\_64的Centos 7.6和arm64，提供全套实验素材和环境。全面介绍kdump+crash在死机黑屏方面的实战应用，全部案例源自线上云服务器和嵌入式产品开发实际案例！

不能为客户和老板解决死机黑屏问题的都不是合格的linux运维或者研发攻城狮！





订阅视频请到淘宝店：<https://shop115683645.taobao.com>

