

# 浙江大学



课程名称	:	操作系统原理与实践
题 目	:	Lab 0: GDB & QEMU 调试 64 位 RISC-V LINUX
授课教师	:	申文博
助 教	:	王鹤翔、陈淦豪、许昊瑞
姓 名	:	潘潇然
学 号	:	3220106049
地 点	:	32舍367

# 一、实验过程与步骤

## 1. 实验环境搭建

- 首先 `sudo apt update` 以更新环境内的软件包列表
- 之后运行 `sudo apt install gcc-riscv64-linux-gnu` 和 `sudo apt install autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev git` 以安装编译内核所需要的交叉编译工具链和用于构建程序的软件包

```
root@PiXe1Ran9E:/mnt/d/Learning/OS/lab# sudo apt install gcc-riscv64-linux-gnu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
binutils binutils-common binutils-riscv64-linux-gnu binutils-x86-64-linux-gnu cpp-11-riscv64-linux-gnu
cpp-riscv64-linux-gnu gcc-11-cross-base-ports gcc-11-riscv64-linux-gnu gcc-11-riscv64-linux-gnu-base
gcc-12-cross-base-ports libasan6-riscv64-cross libatomic1-riscv64-cross libbinutils libc6-dev-riscv64-cross
libc6-riscv64-cross libcc1-0 libctf0 libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross libgomp1-riscv64-cross
libisl23 libmpc3 linux-libc-dev-riscv64-cross
Suggested packages:
binutils-doc gcc-11-locales cpp-doc gcc-11-doc make manpages-dev autoconf automake libtool flex bison
gdb-riscv64-gnu gcc-doc
The following NEW packages will be installed:
binutils-riscv64-linux-gnu cpp-11-riscv64-linux-gnu cpp-riscv64-linux-gnu gcc-11-cross-base-ports
gcc-11-riscv64-linux-gnu gcc-11-riscv64-linux-gnu-base gcc-12-cross-base-ports gcc-riscv64-linux-gnu
libasan6-riscv64-cross libatomic1-riscv64-cross libc6-dev-riscv64-cross libgcc-riscv64-cross libcc1-0
libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross libgomp1-riscv64-cross libisl23 libmpc3
linux-libc-dev-riscv64-cross
The following packages will be upgraded:
binutils binutils-common binutils-x86-64-linux-gnu libbinutils libctf0
5 upgraded, 19 newly installed, 0 to remove and 125 not upgraded.
Need to get 39.0 MB of archives.
After this operation, 115 MB of additional disk space will be used.
Do you want to continue? [Y/n] u
Preparing to unpack .../23-libc6-dev-riscv64-cross_2.35-0ubuntu3cross4_all.deb ...
Unpacking libc6-dev-riscv64-cross (2.35-0ubuntu3cross4) ...
Setting up gcc-12-cross-base-ports (12.3.0-1ubuntu1~22.04cross1) ...
Setting up binutils-common:amd64 (2.38-4ubuntu2.6) ...
Setting up binutils-riscv64-linux-gnu (2.38-4ubuntu2.6) ...
Setting up gcc-11-riscv64-linux-gnu-base:amd64 (11.4.0-1ubuntu1~22.04cross1) ...
Setting up libmpc3:amd64 (1.2.1-2build1) ...
Setting up gcc-11-cross-base-ports (11.4.0-1ubuntu1~22.04cross1) ...
Setting up linux-libc-dev-riscv64-cross (5.15.0-22.22cross4) ...
Setting up libc6-riscv64-cross (2.35-0ubuntu3cross4) ...
Setting up libgomp1-riscv64-cross (12.3.0-1ubuntu1~22.04cross1) ...
Setting up libbinutils:amd64 (2.38-4ubuntu2.6) ...
Setting up libisl23:amd64 (0.24-2build1) ...
Setting up libatomic1-riscv64-cross (12.3.0-1ubuntu1~22.04cross1) ...
Setting up libcc1-0:amd64 (12.3.0-1ubuntu1~22.04) ...
Setting up cpp-11-riscv64-linux-gnu (11.4.0-1ubuntu1~22.04cross1) ...
Setting up libctf0:amd64 (2.38-4ubuntu2.6) ...
Setting up libgcc-s1-riscv64-cross (12.3.0-1ubuntu1~22.04cross1) ...
Setting up libc6-dev-riscv64-cross (2.35-0ubuntu3cross4) ...
Setting up libasan6-riscv64-cross (11.4.0-1ubuntu1~22.04cross1) ...
Setting up cpp-riscv64-linux-gnu (4:11.2.0-1ubuntu1) ...
Setting up binutils-x86-64-linux-gnu (2.38-4ubuntu2.6) ...
Setting up binutils (2.38-4ubuntu2.6) ...
Setting up libgcc-11-dev-riscv64-cross (11.4.0-1ubuntu1~22.04cross1) ...
Setting up gcc-11-riscv64-linux-gnu (11.4.0-1ubuntu1~22.04cross1) ...
Setting up gcc-riscv64-linux-gnu (4:11.2.0-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/lib/wsl/lib/libcuda.so.1 is not a symbolic link
```

```
root@PiXe1Ran9E:/mnt/d/Learning/OS/lab# sudo apt install autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev \
gawk build-essential bison flex texinfo gperf libtool patchutils bc \
zlib1g-dev libexpat-dev git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libexpat1-dev' instead of 'libexpat-dev'.
gawk is already the newest version (1:5.1.0-1ubuntu0.1).
gawk set to manually installed.
The following additional packages will be installed:
bzzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11 gcc-11-base
libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomici libauthten-sasl-perl
libc-dev-bin libc-devtools libc6 libc6-dev libclone-perl libcrypt-dev libcurl4 libdata-dump-perl libdeflate0
libdpkg-perl libencode-locale-perl libexpat1 libfakeroot libfile-fcntrlock-perl libfile-listing-perl libfl-dev
libf12 libfont-afm-perl libfontconfig libfreetype6 libgcc-11-dev libgd3 libgmpxx4ldbl libgomp1 libhtml-form-perl
libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl
libhttp-daemon-perl libhttp-date-perl libhttp-message-perl libhttp-negotiate-perl libio-html-perl
libio-socket-ssl-perl libitm libjbig0 libjpeg-turbo8 libjpeg8 liblsan0 libltdl-dev libltdl7 liblwp-mediatypes-perl
liblwp-protocol-https-perl libmailt-smtp-ssl-perl libnet-ssleay-perl libnsl-dev
libquadmath0 libstdc++-11-dev libtext-unidecode-perl libtiff5 libtime-date-perl libtirpc-dev libtray-tiny-perl
libtsan0 libubsan1 liburi-perl libwebp7 libwww-perl libwww-robotrules-perl libxml-libxml-perl libxml-libxml-perl
libxml-namespacesupport-perl libxmlxml-parser-perl libxmlxml-sax-base-perl libxmlxml-sax-expat-perl libxmlxml-sax-perl libxmlxpm4
linux-libc-dev lto-disabled-list m4 make manpages-dev perl-openssl-defaults rpcsvc-proto tex-common
Suggested packages:
autoconf-archive gnu-standards autoconf-doc gettext bison-doc bzzip2-doc cpp-doc gcc-11-locales debian-keyring
flex-doc g++-multilib g++-11-multilib gcc-11-doc gcc-multilib gdb gcc-11-multiarch git-daemon-run
git-daemon-susvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn libdigest-hmac-perl
libgssapi-perl glIBC-doc bzip libgd-tools gmp-doc libgmp10-doc libtool-doc libcrypt-ssleay-perl libmpfr-doc
libstdc++-11-doc gfortran | fortran95-compiler gcj-jdk libsub-name-perl libbusiness-isbn-perl libauthten-ntlm-perl
libxml-sax-expatx-perl m4-doc make-doc debhelper texlive-base texlive-latex-base texlive-plain-generic
texlive-fonts-recommended
Recommended packages:
libnss-nis libnss-nisplus
The following NEW packages will be installed:
autoconf automake autotools-dev bc bison build-essential bzzip2 cpp cpp-11 dpkg-dev fakeroot flex fontconfig-config
fonts-dejavu-core g++ g++-11 gcc gcc-11-base gperf libalgorithm-diff-perl libalgorithm-diff-xs-perl
libalgorithm-merge-perl libasan6 libatomici libauthten-sasl-perl libc-dev-bin libc-devtools libc6-dev libclone-perl
libcrypt-dev libdata-dump-perl libdeflate0 libdpkg-perl libencode-locale-perl libexpat1 libfakeroot
libfile-fcntrlock-perl libfile-listing-perl libfl-dev libfreetype6 libgcc-11-dev libgd3 libgmpxx4ldbl libgomp1 libhtml-form-perl
libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
libhttp-message-perl libhttp-negotiate-perl libio-html-perl libio-socket-ssl-perl libitm libjbig0 libjpeg-turbo8
libjpeg8 liblsan0 libltdl7 liblwp-mediatypes-perl liblwp-protocol-https-perl libmailt-tools-perl
libmpc-dev libmpfr-dev libnet-htp-perl libnet-smtp-perl libnet-ssleay-perl libnsl-dev libquadmath0
libstdc++-11-dev libtext-unidecode-perl libtiff5 libtime-date-perl libtirpc-dev libtool libtray-tiny-perl libtsan0
libubsan1 liburi-perl libwebp7 libwww-perl libwww-robotrules-perl libxml-libxml-perl libxml-namespacesupport-perl
libxmlxml-parser-perl libxmlxml-sax-base-perl libxmlxml-sax-expat-perl libxmlxml-sax-perl libxmlxpm4 linux-libc-dev
lto-disabled-list m4 make manpages-dev patchutils perl-openssl-defaults rpcsvc-proto tex-common texinfo zlib1g-dev
The following packages will be upgraded:
update-perl-sax-parsers: Registering Perl SAX parser XML::LibXML::SAX::Parser with priority 50...
update-perl-sax-parsers: Registering Perl SAX parser XML::LibXML::SAX with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Setting up libwww-robotrules-perl (6.02-1) ...
Setting up libgcc-11-dev:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up gcc-11 (11.4.0-1ubuntu1~22.04) ...
Setting up cpp (4:11.2.0-1ubuntu1) ...
Setting up libhtml-parser-perl:amd64 (3.76-1build2) ...
Setting up libc6-dev:amd64 (2.35-0ubuntu3.8) ...
Setting up libtiff5:amd64 (4.3.0-6ubuntu0.10) ...
Setting up libfontconfig1:amd64 (2.13.1-4.2ubuntu5) ...
Setting up libio-socket-ssl-perl (2.074-2) ...
Setting up libhttp-message-perl (6.36-1) ...
Setting up libhtml-form-perl (6.07-1) ...
Setting up libhttp-negotiate-perl (6.01-1) ...
Setting up libhttp-cookies-perl (6.10-1) ...
Setting up libtool (2.4.6-15build2) ...
Setting up libhtml-tree-perl (5.07-2) ...
Setting up libhtml-format-perl (2.12-1.1) ...
Setting up gcc (4:11.2.0-1ubuntu1) ...
Setting up libexpat1-dev:amd64 (2.4.7-1ubuntu0.3) ...
Setting up libnet-smtp-ssl-perl (1.04-1) ...
Setting up libmailt-tools-perl (2.21-1) ...
Setting up libgd3:amd64 (2.3.0-2ubuntu2) ...
Setting up texinfo (6.8-4build1) ...
Setting up libstdc++-11-dev:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up zlib1g-dev:amd64 (1:1.2.11.dfsg-2ubuntu9.2) ...
Setting up libhttp-daemon-perl (6.13-1ubuntu0.1) ...
Setting up libc-devtools (2.35-0ubuntu3.8) ...
Setting up g++-11 (11.4.0-1ubuntu1~22.04) ...
Setting up g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.9ubuntu3) ...
Setting up liblwp-protocol-https-perl (6.10-1) ...
Setting up libwww-perl (6.61-1) ...
Setting up libxmlxml-parser-perl:amd64 (2.46-3build1) ...
Setting up libxmlxml-sax-expat-perl (0.51-1) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::SAX::Expat with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/lib/wsl/lib/libcuda.so.1 is not a symbolic link
Processing triggers for man-db (2.10.2-1) ...
```

- 运行 `sudo apt install qemu-system-misc` 以安装qemu

- 运行 `sudo apt install gdb-multiarch` 以安装gdb进行调试工作

```
root@PiXiRan9E:/mnt/d/Learning/OS/lab# sudo apt install gdb-multiarch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  gdb libbabeltrace1 libboost-regex1.74.0 libc6-dbg libdebuginfod-common libdebuginfod1 libipt2
  libsource-highlight-common libsource-highlight4v5
Suggested packages:
  gdb-doc gdbserver
The following NEW packages will be installed:
  gdb gdb-multiarch libbabeltrace1 libboost-regex1.74.0 libc6-dbg libdebuginfod-common libdebuginfod1 libipt2
  libsource-highlight-common libsource-highlight4v5
0 upgraded, 10 newly installed, 0 to remove and 120 not upgraded.
Need to get 23.3 MB of archives.
After this operation, 53.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libdebuginfod-common all 0.186-1build1 [7878 B]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libbabeltrace1 amd64 1.5.8-2build1 [160 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libdebuginfod1 amd64 0.186-1build1 [12.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/main amd64 libipt2 amd64 2.0.5-1 [46.4 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 libsource-highlight-common all 3.1.9-4.1build2 [64.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 libboost-regex1.74.0 amd64 1.74.0-14ubuntu3 [511 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/main amd64 libsource-highlight4v5 amd64 3.1.9-4.1build2 [207 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 gdb amd64 12.1-0ubuntu1~22.04.2 [3920 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 gdb-multiarch amd64 12.1-0ubuntu1~22.04.2 [4591 kB]
43% [9 gdb-multiarch 2844 kB/4591 kB 62%]                                         354 kB/s 43s
```

## 2. 获取 Linux 源码和已经编译好的文件系统

- 为了正常编译Linux内核，我们需要在linux内用wget获取最新Linux源码，同时不能放置在 /mnt 目录，此处放置在 /usr 目录，运行 `sudo wget https://git.kernel.org/torvalds/t/linux-6.11-rc7.tar.gz`

```
root@PiXiRan9E:/usr# sudo wget https://git.kernel.org/torvalds/t/linux-6.11-rc7.tar.gz
--2024-09-10 21:53:32-- https://git.kernel.org/torvalds/t/linux-6.11-rc7.tar.gz
Resolving git.kernel.org (git.kernel.org)... 145.40.73.55, 2604:1380:40e1:4800::1
Connecting to git.kernel.org (git.kernel.org)|145.40.73.55|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.11-rc7.tar.gz [following]
--2024-09-10 21:53:33-- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.11-rc7.tar.gz
Reusing existing connection to git.kernel.org:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'linux-6.11-rc7.tar.gz.1'

linux-6.11-rc7.tar.gz.1    [ =>                               ]  99.74K  80.4KB/s
```

- 之后解压文件 `sudo tar -xzvf linux-6.11-rc7.tar.gz`, 这里由于速度太快没截到图
  - `git clone`实验指导仓库, 之后观察到在 `lab0` 目录下有 `rootfs.img` 镜像

```
root@PiXe1Ran9E:/usr# git clone https://github.com/ZJU-SEC/os24fall-stu.git
Cloning into 'os24fall-stu'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 78 (delta 33), reused 66 (delta 21), pack-reused 0 (from 0)
Receiving objects: 100% (78/78), 1.97 MiB | 88.00 KiB/s, done.
Resolving deltas: 100% (33/33), done.
root@PiXe1Ran9E:/usr# cd os24fall-stu/src/lab0
root@PiXe1Ran9E:/usr/os24fall-stu/src/lab0# ls
rootfs.img
root@PiXe1Ran9E:/usr/os24fall-stu/src/lab0#
```

### 3. 编译Linux内核

- 使用默认配置(`defconfig`)

```
root@PiXe1Ran9E:/usr/linux-6.11-rc7# make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
HOSTCC  scripts/basic/fixdep
HOSTCC  scripts/kconfig/conf.o
HOSTCC  scripts/kconfig/confdata.o
HOSTCC  scripts/kconfig/expr.o
LEX     scripts/kconfig/lexer.lex.c
YACC    scripts/kconfig/parser.tab.[ch]
HOSTCC  scripts/kconfig/lexer.lex.o
HOSTCC  scripts/kconfig/menu.o
HOSTCC  scripts/kconfig/parser.tab.o
HOSTCC  scripts/kconfig/preprocess.o
HOSTCC  scripts/kconfig/symbol.o
HOSTCC  scripts/kconfig/util.o
HOSTLD  scripts/kconfig/conf
*** Default configuration is based on 'defconfig'
#
# configuration written to .config
#
```

- 为避免内存耗尽，同时避免过高并行度导致编译出错，因此使用8线程编译，`make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- -j8`

```
root@PiXe1Ran9E:/usr/linux-6.11-rc7# make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- -j8
WRAP    arch/riscv/include/generated/uapi/asm/errno.h
UPD     include/generated/uapi/linux/version.h
WRAP    arch/riscv/include/generated/uapi/asm/fcntl.h
WRAP    arch/riscv/include/generated/uapi/asm/ioctl.h
WRAP    arch/riscv/include/generated/uapi/asm/ioctls.h
WRAP    arch/riscv/include/generated/uapi/asm/ipcbuf.h
WRAP    arch/riscv/include/generated/uapi/asm/mman.h
WRAP    arch/riscv/include/generated/uapi/asm/msgbuf.h
WRAP    arch/riscv/include/generated/uapi/asm/param.h
WRAP    arch/riscv/include/generated/uapi/asm/poll.h
WRAP    arch/riscv/include/generated/uapi/asm posix_types.h
WRAP    arch/riscv/include/generated/uapi/asm/resource.h
HOSTCC  scripts/dtc/dtc.o
WRAP    arch/riscv/include/generated/uapi/asm/sembuf.h
```

## 4. 使用QEMU运行内核

- 运行下图命令以运行内核，注意修改镜像文件的路径

```
[ 0.435137] EXT4-fs (vda): mounted filesystem c3e9bbca-ec22-47f9-a368-187b21172fc1 ro with ordered data mode. Quota mode: disabled.
[ 0.436154] VFS: Mounted root (ext4 filesystem) readonly on device 254:0.
[ 0.438582] devtmpfs: mounted
[ 0.460797] Freeing unused kernel image (initmem) memory: 2256K
[ 0.461502] Run /sbin/init as init process
Please press Enter to activate this console.
```

## 5. 使用 GDB 对内核进行调试

- 这里需要打开两个terminal，一个使用QEMU启动Linux，另一个使用GDB与QEMU远程通信
- 首先打开一个terminal运行QEMU，注意与上一步不同的是，需要在命令最后加上`-s -s`，其中`-s`表示QEMU在启动后需要等待GDB连接，而不是立刻开始执行CPU，`-s`是`-gdb tcp::1234`的简写，告诉QEMU在tcp端口1234上启动一个GDB服务器。因此运行后不会立即产生任何输出

```
root@PiXe1Ran9E:/usr# qemu-system-riscv64 -nographic -machine virt -kernel /usr/linux-6.11-rc7/arch/riscv/boot/Image -device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" -bios default -drive file=/mnt/d/learning/os/lab/os24fall-stu/svc/lab0/rootfs.img,format=raw,id=hd0 -S -s
```

- 之后开启另一个terminal，先输入`gdb-multiarch /usr/linux-6.11-rc7/vmlinux`启动gdb，之后利用gdb进行连接qemu，设置断点等操作，之后继续执行，最后退出gdb

```
(gdb) target remote :1234
Remote debugging using :1234
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x00000000000001000 in ?? ()
(gdb) b start_kernel
No symbol table is loaded. Use the "file" command.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (start_kernel) pending.
(gdb) continue
Continuing.
quit
[Inferior 1 (process 1) exited normally]
(gdb) quit
root@PiXe1Ran9E:/mnt/d/Learning/OS/lab#
```

## 6. 接下来，对一个具体的C语言程序利用gdb进行调试，练习gdb各项基本指令

- 首先编写一个有嵌套调用的程序，这里简单以一个计算斐波那契数列的函数作为例子

```
#include <stdio.h>

int fn(int n) {
    if (n == 1 || n == 2) return 1;
    return fn(n - 1) + fn(n - 2);
}

int main() {
    int num, sum;

    printf("Input: ");
    scanf("%d", &num);

    sum = fn(num);

    printf("Sum: %d\n", sum);

    return 0;
}
```

- 之后运行`gcc -g test.c -o test`对此程序进行编译

- 运行 `gdb test` 进入调试
- 首先 `layout asm` 查看汇编代码

```

0x1189 <fn>    endbr64
0x118d <fn+4>   push  %rbp
0x118e <fn+5>   mov    %rsp,%rbp
0x1191 <fn+8>   push  %rbx
0x1192 <fn+9>   sub   $0x18,%rsp
0x1196 <fn+13>  mov    %edi,-0x14(%rbp)
0x1199 <fn+16>  cmpl  $0x1,-0x14(%rbp)
0x119d <fn+20>  je    0x11a5 <fn+28>
0x119f <fn+22>  cmpl  $0x2,-0x14(%rbp)
0x11a3 <fn+26>  jne   0x11ac <fn+35>
0x11a5 <fn+28>  mov    $0x1,%eax
0x11aa <fn+33>  jmp   0x11ca <fn+65>
0x11ac <fn+35>  mov    -0x14(%rbp),%eax
0x11af <fn+38>  sub   $0x1,%eax
0x11b2 <fn+41>  mov    %eax,%edi
0x11b4 <fn+43>  call   0x1189 <fn>
0x11b9 <fn+48>  mov    %eax,%ebx

```

exec No process In:  
(gdb)

- 之后 `start` 开始运行程序，将会停止在 `main` 函数开头

```

0x55555555551d0 <main>    endbr64
0x55555555551d4 <main+4>   push  %rbp
0x55555555551d5 <main+5>   mov    %rsp,%rbp
0x55555555551d8 <main+8>   sub   $0x10,%rsp
> 0x55555555551dc <main+12>  mov    %fs:0x28,%rax
0x55555555551e5 <main+21>  mov    %rax,-0x8(%rbp)
0x55555555551e9 <main+25>  xor   %eax,%eax
0x55555555551eb <main+27>  lea   0xel2(%rip),%rax      # 0x5555555556004
0x55555555551f2 <main+34>  mov    %rax,%rdi
0x55555555551f5 <main+37>  call   0x5555555555080 <printf@plt>
0x55555555551f9 <main+42>  lea   -0x10(%rbp),%rax
0x5555555555203 <main+51>  mov    %rax,%rsi
0x5555555555206 <main+54>  lea   0xdff(%rip),%rax      # 0x555555555600c
0x555555555520d <main+61>  mov    %rax,%rdi
0x5555555555210 <main+64>  mov    $0x0,%eax
0x5555555555215 <main+69>  call   0x5555555555090 <__isoc99_scanf@plt>

multi-thread Thread 0x7ffff7d877 In: main
(gdb) start
Temporary breakpoint 1 at 0x11dc: file test.c, line 8.
Starting program: /usr/os-lab/lab0/test
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Temporary breakpoint 1, main () at test.c:8

```

- `b fn` 在 `fn` 函数入口设立断点

```

Temporary breakpoint 1, main () at test.c:8
(gdb) b fn
Breakpoint 2 at 0x555555555199: file test.c, line 4.
(gdb)

```

- `c(continue)` 执行到程序第一次遇到 `fn`

```

0x55555555551d0 <main>    endbr64
0x5555555555189 <fn>    endbr64        %rbp
0x555555555518d <fn+4>   push  %rbp ,%rbp
0x555555555518e <fn+5>   mov    %rsp,%rbp          p
0x555555555518f <fn+8>   push  %rbx          rax
0x5555555555192 <fn+9>   sub   $0x18,%rsp          p
0x5555555555196 <fn+13>  mov    %edi,-0x14(%rbp)
B+> 0x5555555555199 <fn+16>  cmpl  $0x1,-0x14(%rbp) p,%rax      # 0x5555555556004
0x555555555519d <fn+20>  je    0x5555555551a5 <fn+28>
0x555555555519f <fn+22>  cmpl  $0x2,-0x14(%rbp)
0x55555555551a3 <fn+26>  jne   0x55555555551ac <fn+35> <printf@plt>
0x55555555551a5 <fn+28>  mov    $0x1,%eax
0x55555555552aa <fn+33>  jmp   0x5555555551ca <fn+65>
0x55555555551ac <fn+35>  mov    -0x14(%rbp),%eax p,%rax      # 0x555555555600c
0x55555555551af <fn+38>  sub   $0x1,%eax
0x55555555551b2 <fn+41>  mov    %eax,%edi
0x55555555551b4 <fn+43>  call   0x555555555189 <fn>090 <__isoc99_scanf@plt>
1b9 <fn+48>  mov    %eax,%ebx

multi-thread Thread 0x7ffff7d877 In: main
Starting program: /usr/os-lab/lab0/fn
Temporary breakpoint 1, main () at test.c:8
(gdb) b fn
Breakpoint 2 at 0x555555555199: file test.c, line 4.
(gdb) c
Continuing.
Input: 10

```

- `display n` 查看 `n` 值

```

(gdb) si
(gdb) display n
1: n = 10

```

- 之后我们可以通过反复 `c`, 嵌套调用 `fn` 函数, 可以观察到 `n` 的值在不断变化

```

Continuing.

Breakpoint 2, fn (n=8) at test.c:4
1: n = 8
(gdb) c
Continuing.

Breakpoint 2, fn (n=7) at test.c:4
1: n = 7
(gdb) c
Continuing.

Breakpoint 2, fn (n=6) at test.c:4
1: n = 6
(gdb)

```

- 这时可以运行 `bt` 查看函数的调用的栈帧和层级关系

```

(gdb) bt
#0  fn (n=6) at test.c:4
#1  0x0000555555551b9 in fn (n=7) at test.c:5
#2  0x0000555555551b9 in fn (n=8) at test.c:5
#3  0x0000555555551b9 in fn (n=9) at test.c:5
#4  0x0000555555551b9 in fn (n=10) at test.c:5
#5  0x000055555555224 in main () at test.c:14
(gdb)

```

- 最后运行 `finish`, 可以执行到函数末尾, 而 `n=6` 的末尾自然就是执行 `n=5`

```

(gdb) finish
Run till exit from #0  fn (n=6) at test.c:4
Breakpoint 2, fn (n=5) at test.c:4
1: n = 5

```

## 二、实验心得体会

这个实验总体还是比较顺利的，主要的问题就是一开始没有注意常见问题中提到的不能在 `/mnt` 目录下进行Linux源码的编译，导致编译了一两节课还没编译完，后面才发现此问题。这个实验做下来，整体对wsl的文件结构更熟悉了一些，同时也更熟悉了gdb的基本使用。

## 三、思考题

- 为了后续操作, 首先写一个简单的 `.c` 文件, 主要内容是一个简单的从1到10的循环, 并输出对应数字

```
#include <stdio.h>

int main() {
    int i;
    for (i = 1; i <= 10; i++) printf("%d\n", i);
    return 0;
}
```

- 之后使用 `riscv64-linux-gnu-gcc` 编译此文件, 之后利用 `riscv64-linux-gnu-objdump` 反编译前一步得到的编译产物并将输出重定向到 `new.txt` 文件中。其中 `riscv64-linux-gnu-gcc` 将C语言编译生成适用于RISC-V64位架构并在Linux上运行的可执行文件, `>` 表示重定向

```
● root@PiXe1Ran9E:/usr/os-lab/lab0# riscv64-linux-gnu-gcc -o new new.c
● root@PiXe1Ran9E:/usr/os-lab/lab0# riscv64-linux-gnu-objdump -d new > new.txt
```

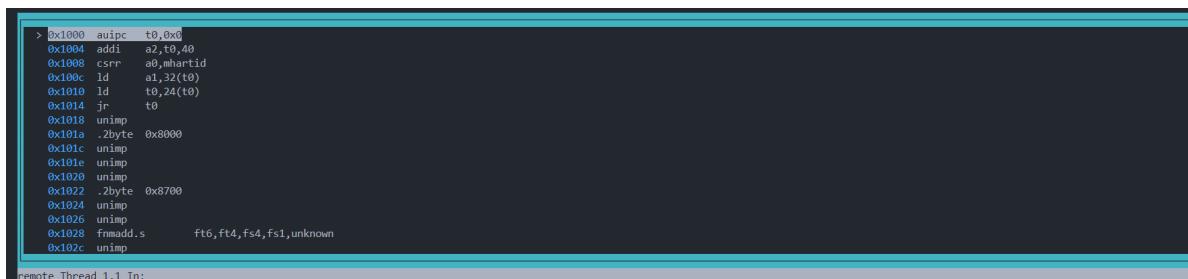
3. 之后我们可以打开 new.txt 查看汇编代码，可以观察到以下汇编代码完成了循环条件的判断和

printf

```
678: fec42783    lw a5,-20($0)
67c: 85be        mv a1,a5
67e: 00000517    auipc a0,0x0
682: 03a50513    addi a0,a0,58 # 6b8 <_IO_stdin_used+0x8>
686: f1bff0ef    jal ra,5a0 <printf@plt>
68a: fec42783    lw a5,-20($0)
68e: 2785        addiw a5,a5,1
690: fef42623    sw a5,-20($0)
694: fec42783    lw a5,-20($0)
698: 0007871b    sext.w a4,a5
69c: 47a9        li a5,10
69e: fce7dde3    bge a5,a4,678 <main+0x10>
```

4. 接下来进行Linux调试，首先需要按实验步骤5打开两个terminal，将gdb与qemu连接，之后运行下述指令

- 运行 layout asm 显示汇编代码



```
> 0x1000 auipc t0,0x0
0x1004 addi a2,t0,40
0x1008 csrr a0,mhartid
0x100c ld a1,32(t0)
0x1010 ld t0,24(t0)
0x1014 jr t0
0x1018 unimp
0x101a .2byte 0x8000
0x101c unimp
0x101e unimp
0x1020 unimp
0x1022 .2byte 0x8700
0x1024 unimp
0x1026 unimp
0x1028 fmadd.s ft6,ft4,fs4,fs1,unknown
0x102c unimp
```

- 在 0x80000000 处下断点并查看所有已下的断点，发现成功设置断点

```
(gdb) break *0x80000000
Breakpoint 1 at 0x80000000
(gdb) info b
Num      Type            Disp Enb Address          What
1        breakpoint       keep y  0x0000000080000000
```

- 在 0x80200000 处下断点后清除 0x80000000 处的断点，发现成功删除

```
(gdb) d 1
(gdb) delete 1
No breakpoint number 1.
(gdb) info b
Num      Type            Disp Enb Address          What
2        breakpoint       keep y  0x0000000080200000
```

- 继续运行直到触发 0x80200000 处的断点，并单步调试一次。发现成功在断点处停止，同时单步调试成功

```
z      breakpoint       keep y  0x0000000080200000
(gdb) c
Continuing.

Breakpoint 2, 0x0000000080200000 in ?? ()
(gdb) si
0x0000000080200002 in ?? ()
```

- 最后 quit 退出qemu

5. 使用 make 工具清除 Linux 的构建产物，我们进入 linux-6.11-rc7 目录，运行 make clean

```
● root@PiXe1Ran9E:/usr/linux-6.11-rc7# make clean
  CLEAN  drivers/firmware/efi/libstub
  CLEAN  drivers/gpu/drm/radeon
  CLEAN  drivers/scsi
  CLEAN  drivers/tty/vt
  CLEAN  init
  CLEAN  kernel
  CLEAN  lib/raid6
  CLEAN  lib
  CLEAN  security/apparmor
  CLEAN  security/selinux
  CLEAN  usr
  CLEAN  .
  CLEAN  modules.builtin modules.builtin.modinfo .vmlinuz.export.c
```

## 6. vmlinu 和 image 的联系和区别

- 区别：
    - `vmlinux` (其中 `vm` 指的是 `virtual memory`, Linux 支持虚拟内存) 是编译 Linux 内核得到的最原始的内核文件, 是 `elf` 格式(Executable and Linkable Format)的文件, 未经压缩, 文件比较大, 多存放在 PC 上。 `vmlinux` 主要用于加载和运行 Linux 内核。计算机启动时, `Boot Loader` 会加载 `vmlinux` 文件, 首先将其复制到系统内存中, 之后开始执行代码。 `vmlinux` 包含了 Linux 的所有代码和数据结构, 还包含调试符号等信息(如函数名称、变量名称等)
    - `image` 是 Linux 内核镜像文件, 但仅包含可执行的二进制数据
  - 联系：用户对 Linux 内核源码进行编译, 会先生成 `vmlinux`, 之后由于此文件过大, 要经过 `objcopy` 处理成只包含二进制数据的内核代码, 去除掉不需要的文件信息, 即 `image`