

v1.5.3_MiniOS使用qemu运行和调试

版本v0.1

最后修改时间 2024-10-28

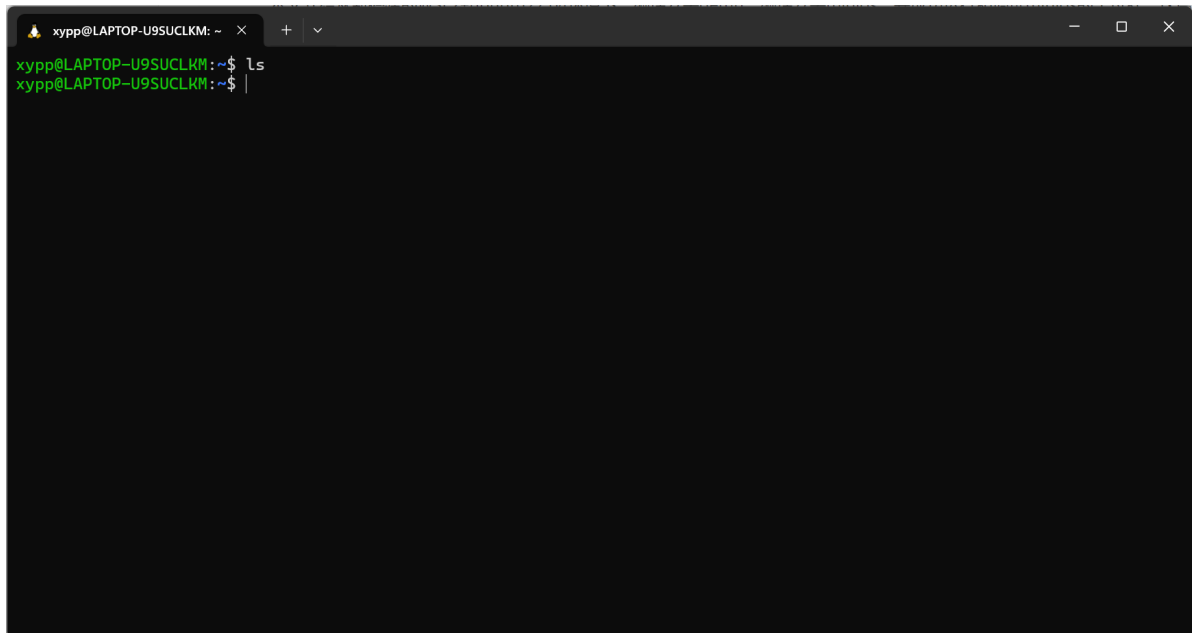
本文介绍从新搭建的WSL2+ubuntu22.04环境下，编译安装qemu，编译安装minios，并成功运行和调试minios的全过程。这篇文章适合刚刚接触qemu和minios的小白，为保证实践过程中尽量少的出现错误，请尽量严格按照本文的步骤进行操作。

1 起始环境

起始环境为使用WSL2刚刚安装成功ubuntu22.04

1.1 安装WSL2+ubuntu22.04

具体可以参考 [安装WSL2和Ubuntu22.04版本](#) 这篇文章进行安装



1.2 更换镜像源

建议使用本文列出的镜像源，使用其他镜像源在后续安装工具包可能会出现依赖错误

- 备份 `sources.list` 文件

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

- 换源

```
sudo vim /etc/apt/sources.list
```

将sources.list清空后，将下列代码粘贴到sources.list后保存退出

```
deb http://mirrors.aliyun.com/ubuntu/ jammy main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-security main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-updates main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-proposed main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-backports main restricted
universe multiverse
```

- 更新软件列表

```
sudo apt-get update
sudo apt-get upgrade
```

2 安装工具包

2.1 安装编译和调试工具

```
sudo apt-get install net-tools make build-essential gdb nasm
```

2.2 安装qemu所需依赖

以下命令摘自qemu官网<https://wiki.qemu.org/Hosts/Linux>

```
sudo apt-get install git libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev
ninja-build
```

为避免后续使用qemu启动minios后，只输出一行提示信息: `VNC server running on 127.0.0.1:5900` 的问题，还要安装下面依赖用来支持 `SDL`

```
sudo apt-get install libsdl1.2-dev libsdl2-dev
```

还要安装对应的gcc 32位的库，使用 `multilib` 这个库可以在64位的机器上产生32位的程序或者库文件

```
sudo apt-get install gcc-multilib
```

3 qemu

安装的版本是qemu6.2.0

进入用户根目录，创建并进入qemu文件夹

```
cd ~  
mkdir qemu  
cd qemu
```

3.1 源码方式下载qemu

- 下载源码

```
wget https://download.qemu.org/qemu-6.2.0.tar.xz
```

- 解压缩

```
tar xvJf qemu-6.2.0.tar.xz
```

- 进入qemu-6.2.0目录

```
cd qemu-6.2.0
```

```
xypp@LAPTOP-U9SUCLKM:~/qemu/qemu-6.2.0$ ls  
COPYING          blockjob.c        fsdev              meson.build        qemu-edid.c        scsi  
COPYING.LIB      bsd-user          gdb-xml            meson_options.txt  qemu-img-cmds.hx   semihosting  
Kconfig          capstone          gdbstub.c          migration           qemu-img.c          slirp  
Kconfig.host     chardev           gitdm.config       module-common.c    qemu-io-cmds.c     softmmu  
LICENSE          configs           hmp-commands-info.hx monitor            qemu-io.c           storage-daemon  
MAINTAINERS       configure         hmp-commands.hx   nbd                 qemu-keymap.c      stubs  
Makefile         contrib          hw                  net                  qemu-nbd.c          subprojects  
README.rst       cpu.c            include            os-posix.c         qemu-options.hx    target  
VERSION          cpus-common.c    io                  os-win32.c         qemu.nsi            tcg  
accel            crypto            iothread.c         page-vary-common.c qemu.sasl            tests  
audio            disas            job-qmp.c          page-vary.c        qga                  thunk.c  
authz            disas.c          job.c               pc-bios             qobject             tools  
backends         docs              libdecnumber       plugins              qom                  trace  
block            dtc               linux-headers      po                   replay               trace-events  
block.c          dump              linux-user          python               replication.c       ui  
blockdev-nbd.c   ebpf              memory_ldst.c.inc  qapi                 roms                  util  
blockdev.c       fpu               meson               qemu-bridge-helper.c scripts              version.rc
```

3.2 configure

进入qemu-6.2.0目录后执行以下命令

```
./configure
```

确保SDL support开启

```
xypp@LAPTOP-U9SUCLKM: ~  
Dependencies  
SDL support : YES  
SDL image support : NO  
GTK support : NO  
pixmap : YES 0.40.0  
VTE support : NO  
slirp support : internal  
libtasn1 : NO  
PAM : NO  
iconv support : YES  
curses support : NO  
virgl support : NO  
curl support : NO  
Multipath support : YES  
VNC support : YES  
VNC SASL support : NO  
VNC JPEG support : NO  
VNC PNG support : YES 1.6.37  
OSS support : YES  
ALSA support : YES 1.2.6.1  
PulseAudio support : YES 15.99.1  
JACK support : NO  
brlapi support : NO  
vde support : NO  
netmap support : NO  
l2tpv3 support : YES  
Linux AIO support : NO  
Linux io_uring support : NO  
ATTR/XATTR support : YES
```

3.3 编译

执行 `make` 就可以进行编译了

```
make
```

编译时间会很长，可以添加参数进行多核编译，`nproc` 表示系统cpu核心的数量

```
xypp@LAPTOP-U9SUCLKM:~/qemu/qemu-6.2.0$ nproc  
12  
xypp@LAPTOP-U9SUCLKM:~/qemu/qemu-6.2.0$ |
```

以下命令使用 `make` 工具并行编译项目，并行作业的数量等于系统中可用的处理器数量。这样可以充分利用多核处理器的性能，加快编译速度。

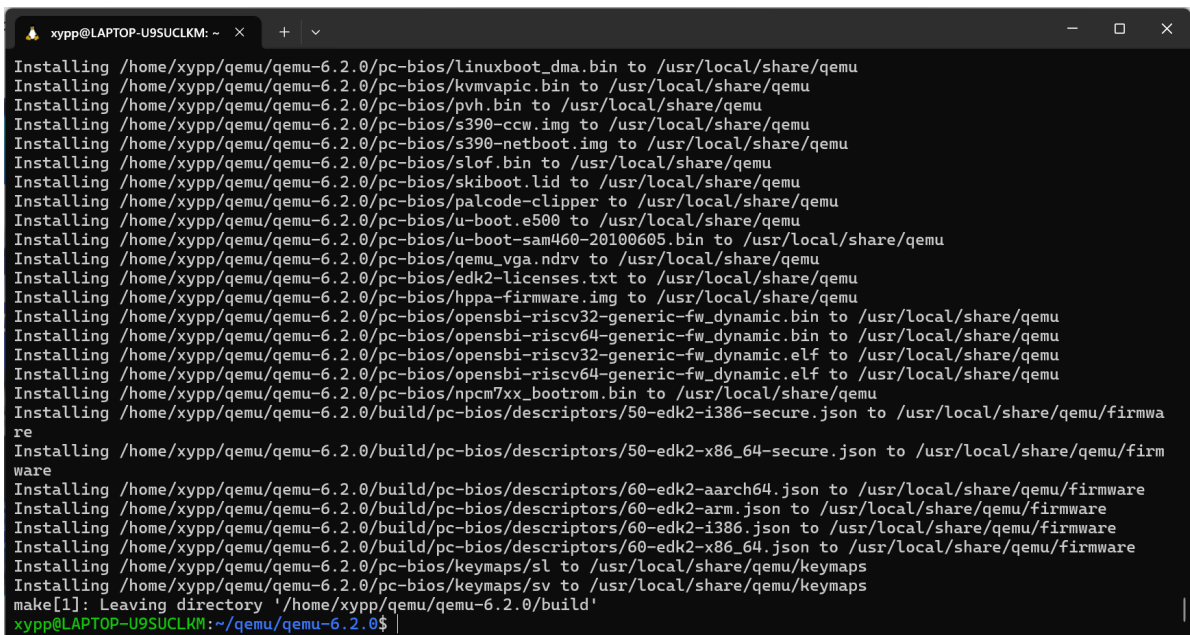
```
make -j`nproc`
```

```
xypp@LAPTOP-U9SUCLKM: ~  
[9585/9607] Linking target tests/qtest/device-plugin-test  
[9586/9607] Compiling C object tests/qtest/rtc-test.p/rtc-test.c.o  
[9587/9607] Linking target tests/qtest/fw_cfg-test  
[9588/9607] Linking target tests/qtest/hd-geo-test  
[9589/9607] Compiling C object tests/qtest/cpu-plugin-test.p/cpu-plugin-test.c.o  
[9590/9607] Compiling C object tests/qtest/vmgenid-test.p/boot-sector.c.o  
[9591/9607] Linking target tests/qtest/drive_del-test  
[9592/9607] Compiling C object tests/qtest/vmgenid-test.p/vmgenid-test.c.o  
[9593/9607] Compiling C object tests/qtest/tco-test.p/tco-test.c.o  
[9594/9607] Compiling C object tests/qtest/vmgenid-test.p/acpi-utils.c.o  
[9595/9607] Compiling C object tests/qtest/q35-test.p/q35-test.c.o  
[9596/9607] Compiling C object tests/qtest/prom-env-test.p/prom-env-test.c.o  
[9597/9607] Compiling C object tests/qtest/pnv-xscom-test.p/pnv-xscom-test.c.o  
[9598/9607] Linking target tests/qtest/rtc-test  
[9599/9607] Linking target tests/qtest/cpu-plugin-test  
[9600/9607] Linking target tests/qtest/m48t59-test  
[9601/9607] Linking target tests/qtest/tco-test  
[9602/9607] Linking target tests/qtest/vmgenid-test  
[9603/9607] Linking target tests/qtest/q35-test  
[9604/9607] Linking target tests/qtest/prom-env-test  
[9605/9607] Linking target tests/qtest/pnv-xscom-test  
[9606/9607] Linking target tests/qtest/rtas-test  
[9607/9607] Linking target tests/qtest/virtio-ccw-test  
make[1]: Leaving directory '/home/xypp/qemu/qemu-6.2.0/build'  
changing dir to build for make "...  
make[1]: Entering directory '/home/xypp/qemu/qemu-6.2.0/build'  
[1/177] Generating QAPI test (include) with a custom command  
[2/144] Generating qemu-version.h with a custom command (wrapped by meson to capture output)  
make[1]: Leaving directory '/home/xypp/qemu/qemu-6.2.0/build'  
xypp@LAPTOP-U9SUCLKM:~/qemu/qemu-6.2.0$ |
```

3.4 安装

安装qemu使用 `make install` 命令，但直接 `make install` 会出现错误，请使用以下命令

```
sudo make install
```



```
xypp@LAPTOP-U9SUCLKM: ~  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/linuxboot_dma.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/kvmvapic.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/pvh.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/s390-ccw.img to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/s390-netboot.img to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/slof.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/skiboot.lid to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/palcode-clipper to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/u-boot.e500 to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/u-boot-sam460-20100605.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/qemu_vga.ndrv to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/edk2-licenses.txt to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/hppa-firmware.img to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/opensbi-riscv32-generic-fw_dynamic.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/opensbi-riscv64-generic-fw_dynamic.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/opensbi-riscv32-generic-fw_dynamic.elf to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/opensbi-riscv64-generic-fw_dynamic.elf to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/npcm7xx_bootrom.bin to /usr/local/share/qemu  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/50-edk2-i386-secure.json to /usr/local/share/qemu/firmwa  
re  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/50-edk2-x86_64-secure.json to /usr/local/share/qemu/firm  
ware  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/60-edk2-aarch64.json to /usr/local/share/qemu/firmware  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/60-edk2-arm.json to /usr/local/share/qemu/firmware  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/60-edk2-i386.json to /usr/local/share/qemu/firmware  
Installing /home/xypp/qemu/qemu-6.2.0/build/pc-bios/descriptors/60-edk2-x86_64.json to /usr/local/share/qemu/firmware  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/keymaps/sl to /usr/local/share/qemu/keymaps  
Installing /home/xypp/qemu/qemu-6.2.0/pc-bios/keymaps/sv to /usr/local/share/qemu/keymaps  
make[1]: Leaving directory '/home/xypp/qemu/qemu-6.2.0/build'  
xypp@LAPTOP-U9SUCLKM: ~/qemu/qemu-6.2.0$
```

4 minios

- 创建并进入code目录

```
cd ~  
mkdir code  
cd code
```

- 从gitee下载minios源码到code目录后，进入minios目录

```
cd minios
```

4.1 创建一些空文件夹

- 创建 `hd` 目录用于存放镜像

```
mkdir hd
```

- 创建 `root` 目录用于挂载 `/dev/loop1p2`

```
mkdir root
```

- 创建 `iso` 目录用于挂载 `/dev/loop0p1`

```
mkdir iso
```

4.2 制作启动镜像文件

使用 `dd` 创建一个镜像文件，为了与makefile对应，镜像文件名称为 `fat32_boot.img`，具体命令如下

```
dd if=/dev/zero of=hd/fat32_boot.img bs=512 count=204800
```

```
xypp@LAPTOP-U9SUCLKM:~/code$ cd minios/
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ls
Makefile      doc                launch-qemu-init-gdb.sh  launch-qemu.sh  umount_fat0.sh
README.md     generate-test-script.sh launch-qemu-shell_0-gdb.sh mount_fat0.sh   user
bochsrc       launch-qemu-gdb.sh  launch-qemu-test_1-gdb.sh os               utils
xypp@LAPTOP-U9SUCLKM:~/code/minios$ mkdir hd
xypp@LAPTOP-U9SUCLKM:~/code/minios$ mkdir root
xypp@LAPTOP-U9SUCLKM:~/code/minios$ mkdir iso
xypp@LAPTOP-U9SUCLKM:~/code/minios$ dd if=/dev/zero of=hd/fat32_boot.img bs=512 count=20480
20480+0 records in
20480+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.100701 s, 104 MB/s
xypp@LAPTOP-U9SUCLKM:~/code/minios$
```

4.3 使用fdisk对镜像进行分区

使用 `fdisk` 对硬镜像进行分区

```
fdisk hd/fat32_boot.img
```

1. 创建第一个分区：

- 输入 `n` 创建新分区。
- 选择 `p` 创建主分区。
- 输入分区号 `1`。
- 输入起始扇区 `2048`。
- 输入结束扇区 `4095`。
- 输入 `a` 将该分区设置为可启动分区。

2. 创建第二个分区：

- 输入 `n` 创建新分区。
- 选择 `p` 创建主分区。
- 输入分区号 `2`。
- 输入起始扇区 `4096`。
- 输入结束扇区 `106495`。

3. 创建扩展分区：

- 输入 `n` 创建新分区。
- 选择 `e` 创建扩展分区。
- 输入分区号 `3`。
- 输入起始扇区 `106496`。
- 输入结束扇区 `204799`。

4. 在扩展分区内创建逻辑分区：

- 输入 `n` 创建新分区。
- 输入起始扇区 `108544`。
- 输入结束扇区 `204799`。

5. 保存分区表并退出：

- 输入 `w` 保存分区表并退出 `fdisk`。

使用以下命令查看分区结果

```
fdisk -l hd/fat32_boot.img
```

```
xypp@LAPTOP-U9SUCLKM:~/code/minios$ fdisk -l hd/fat32_boot.img
Disk hd/fat32_boot.img: 100 MiB, 104857600 bytes, 204800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x5bb2a6ab

Device                Boot  Start    End  Sectors  Size Id Type
hd/fat32_boot.img1 *      2048    4095     2048    1M 83 Linux
hd/fat32_boot.img2        4096  106495   102400   50M 83 Linux
hd/fat32_boot.img3       106496  204799    98304   48M  5 Extended
hd/fat32_boot.img5       108544  204799    96256   47M 83 Linux
xypp@LAPTOP-U9SUCLKM:~/code/minios$
```

4.4 编译

- 分区后就可以执行 `make all` 命令编译minios了

```
make all
```

4.5 安装

- 安装成功后执行 `make install` 命令安装minios

```
sudo make install
```

下图说明安装成功

```
user/user/test/4_ipc/shmat/run.sh => /test/4_ipc/shmat/run.sh
user/user/test/4_ipc/shmctl/run.sh => /test/4_ipc/shmctl/run.sh
user/user/test/2_fs/read/run.sh => /test/2_fs/read/run.sh
user/user/test/2_fs/open/run.sh => /test/2_fs/open/run.sh
user/user/test/2_fs/opendir/run.sh => /test/2_fs/opendir/run.sh
user/user/test/2_fs/run.sh => /test/2_fs/run.sh
user/user/test/2_fs/rmdir/run.sh => /test/2_fs/rmdir/run.sh
user/user/test/2_fs/getcwd/run.sh => /test/2_fs/getcwd/run.sh
user/user/test/2_fs/write/run.sh => /test/2_fs/write/run.sh
user/user/test/2_fs/mkdir/run.sh => /test/2_fs/mkdir/run.sh
user/user/test/2_fs/lseek/run.sh => /test/2_fs/lseek/run.sh
user/user/test/2_fs/unlink/run.sh => /test/2_fs/unlink/run.sh
user/user/test/2_fs/mount/run.sh => /test/2_fs/mount/run.sh
user/user/test/2_fs/close/run.sh => /test/2_fs/close/run.sh
user/user/test/3_pthread/run.sh => /test/3_pthread/run.sh
user/user/test/3_pthread/pthread_self/run.sh => /test/3_pthread/pthread_self/run.sh
user/user/test/3_pthread/pthread_cond_broadcast/run.sh => /test/3_pthread/pthread_cond_broadcast/run.sh
user/user/test/3_pthread/pthread_exit/run.sh => /test/3_pthread/pthread_exit/run.sh
user/user/test/3_pthread/pthread_mutex_unlock/run.sh => /test/3_pthread/pthread_mutex_unlock/run.sh
user/user/test/3_pthread/pthread_join/run.sh => /test/3_pthread/pthread_join/run.sh
user/user/test/3_pthread/pthread_mutex_destroy/run.sh => /test/3_pthread/pthread_mutex_destroy/run.sh
user/user/test/3_pthread/pthread_cond_wait/run.sh => /test/3_pthread/pthread_cond_wait/run.sh
user/user/test/3_pthread/pthread_cond_timewait/run.sh => /test/3_pthread/pthread_cond_timewait/run.sh
user/user/test/3_pthread/pthread_mutex_lock/run.sh => /test/3_pthread/pthread_mutex_lock/run.sh
user/user/test/3_pthread/pthread_create/run.sh => /test/3_pthread/pthread_create/run.sh
user/user/test/3_pthread/pthread_cond_destroy/run.sh => /test/3_pthread/pthread_cond_destroy/run.sh
user/user/test/3_pthread/pthread_cond_init/run.sh => /test/3_pthread/pthread_cond_init/run.sh
user/user/test/3_pthread/pthread_mutex_trylock/run.sh => /test/3_pthread/pthread_mutex_trylock/run.sh
user/user/test/3_pthread/pthread_mutex_init/run.sh => /test/3_pthread/pthread_mutex_init/run.sh
user/user/test/3_pthread/pthread_cond_signal/run.sh => /test/3_pthread/pthread_cond_signal/run.sh
user/user/t_mnt.sh => /t_mnt.sh
sudo umount root
xypp@LAPTOP-U9SUCLKM:~/code/minios$
```

安装成功后会多一些文件

```
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ls
Makefile  bochsrc      hd           kernel.gdb.bin  launch-qemu-shell_0-gdb.sh  mount_fat0.sh  umount_fat0.sh
README.md doc          iso          launch-qemu-gdb.sh  launch-qemu-test_1-gdb.sh  os             user
b.img     generate-test-script.sh  kernel.bin  launch-qemu-init-gdb.sh  launch-qemu.sh             root           utils
```

4.6 拷贝镜像文件

minios的启动脚本 `launch-qemu.sh` 中需要两块硬盘镜像，需要将安装后新生成的 `b.img` 文件拷贝一份，新镜像名称为 `test2.img`

```
launch-qemu.sh X
home > xypp > code > minios > launch-qemu.sh
4 # deleted by mingxuan 2019-5-17
5 # qemu-system-i386 -fda a.img -hda 80m.img -boot order=a -ctrl-grab \
6 # -monitor stdio
7
8 # modified by mingxuan 2019-5-17
9 # qemu-system-i386 -m 2048 -hda b.img -boot order=a -ctrl-grab \
10 # -monitor stdio -usbdevice tablet
11
12 # qemu-system-i386 -drive id=disk,file=b.img,if=none -device ich9-ahci,id=ahci \
13 # -device ide-hd,drive=disk,bus=ahci.0 -monitor stdio
14 # cp b.img a.img
15 # cp b.img c.img
16 # cp b.img d.img
17 # cp b.img e.img
18
19 # qemu-system-i386 \
20 # -drive id=disk,file=b.img,if=none -device ich9-ahci,id=ahci -device ide-hd,drive=disk,bus=ahci.0 \
21 # -drive id=disk1,file=c.img,if=none -device ide-hd,drive=disk1,bus=ahci.1 \
22 # -drive id=disk2,file=d.img,if=none -device ide-hd,drive=disk2,bus=ahci.2 \
23 # -hda a.img -hdb e.img
24 #cp b.img c.img
25 qemu-system-i386 \
26 -device ich9-ahci,id=xiaofeng \
27 -device ide-hd,drive=disk,bus=xiaofeng.0 -drive id=disk,file=b.img,if=none,media=disk \
28 -device ide-hd,drive=disk2,bus=xiaofeng.1 -drive id=disk2,file=test2.img,if=none,media=disk \
29 -rtc base=utc \
30 -boot menu=on \
31 -serial file:minios.log \
32 -monitor stdio
33
```

执行以下命令进行拷贝操作

```
cp b.img test2.img
```

4.7 启动minios

完成以上所有步骤并且没有报错的话，就可以运行minios的启动脚本 `launch-qemu.sh` 了

```
sudo ./launch-qemu.sh
```

运行以上命令之后，会弹出minios的启动界面


```
launch-qemu.sh X
home > xypp > code > minios > launch-qemu.sh
4 # deleted by mingxuan 2019-5-17
5 # qemu-system-i386 -fda a.img -hda 80m.img -boot order=a -ctrl-grab \
6 # -monitor stdio
7
8 # modified by mingxuan 2019-5-17
9 # qemu-system-i386 -m 2048 -hda b.img -boot order=a -ctrl-grab \
10 # -monitor stdio -usbdevice tablet
11
12 # qemu-system-i386 -drive id=disk,file=b.img,if=none -device ich9-ahci,id=ahci \
13 # -device ide-hd,drive=disk,bus=ahci.0 -monitor stdio
14 # cp b.img a.img

问题 输出 终端 窗口

user/user/test/3_pthread/pthread_join/run.sh =>
user/user/test/3_pthread/pthread_mutex_destroy/run.sh =>
user/user/test/3_pthread/pthread_cond_wait/run.sh =>
user/user/test/3_pthread/pthread_cond_timewait/run.sh =>
user/user/test/3_pthread/pthread_mutex_lock/run.sh =>
user/user/test/3_pthread/pthread_create/run.sh =>
user/user/test/3_pthread/pthread_cond_destroy/run.sh =>
user/user/test/3_pthread/pthread_cond_init/run.sh =>
user/user/test/3_pthread/pthread_mutex_trylock/run.sh =>
user/user/test/3_pthread/pthread_mutex_init/run.sh =>
user/user/test/3_pthread/pthread_cond_signal/run.sh =>
user/user/t_mnt.sh => /t_mnt.sh
sudo umount root

xypp@LAPTOP-U9SUCLKM:~/code/minios$ ls
Makefile bochsrtc hd kernel.gdb.bin launch-qemu-shell_0-gdb.sh mount_fat0.sh umount_fat0.sh
README.md doc iso launch-qemu-gdb.sh launch-qemu-test_1-gdb.sh os user
b.img generate-test-script.sh kernel.bin launch-qemu-init-gdb.sh launch-qemu.sh root utils

xypp@LAPTOP-U9SUCLKM:~/code/minios$ cp b.img test2.img
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ls
Makefile bochsrtc hd kernel.gdb.bin launch-qemu-shell_0-gdb.sh mount_fat0.sh test2.img utils
README.md doc iso launch-qemu-gdb.sh launch-qemu-test_1-gdb.sh os umount_fat0.sh user
b.img generate-test-script.sh kernel.bin launch-qemu-init-gdb.sh launch-qemu.sh root user

xypp@LAPTOP-U9SUCLKM:~/code/minios$ ./launch-qemu.sh
WARNING: Image format was not specified for 'b.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'test2.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
QEMU 6.2.0 monitor - type 'help' for more information
(qemu)

-----Kernel Initialization Begins-----
ABAR: FEBF1080 interrupt: 10B
SATA drive found at port 0
SATA drive found at port 1:Message queue initialization done.
-----Processes Begin-----
main:total_mem_size=1885000
[TTY #0]
init:total_mem_size=1878000
time:2024-10-07 20:50:33
minios:~ $ _
```

如果出现上图的启动界面，说明minios启动成功。

4.8 调试minios

调试minios需要运行调试脚本 `1launch-qemu-gdb.sh`

```
sudo ./launch-qemu-gdb.sh
```

运行以上命令后，需要开启另一个终端进行调试，执行以下命令进入gdb环境

```
cd ~/code/minios
sudo gdb kernel.gdb.bin
```

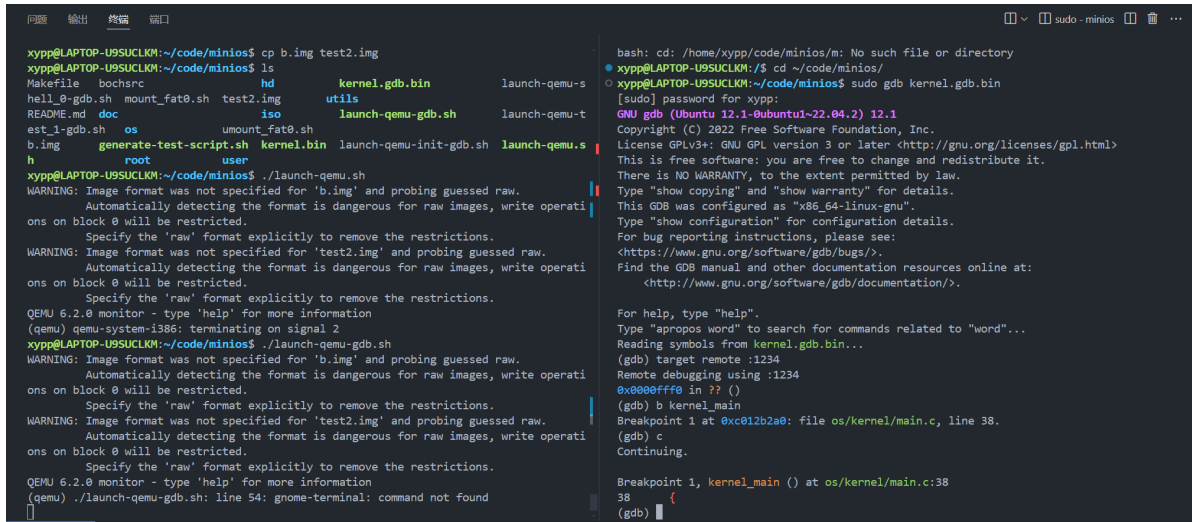
```
xypp@LAPTOP-U9SUCLKM:~/code/minios$ sudo gdb kernel.gdb.bin
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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 kernel.gdb.bin...
(gdb)
```

在gdb命令界面执行命令 `target remote :1234` 和qemu建立连接

```
target remote :1234
```

连接成功后，就可以顺利调试了



```
xypp@LAPTOP-U9SUCLKM:~/code/minios$ cp b.img test2.img
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ls
Makefile  bochsrtc  hd          kernel.gdb.bin  launch-qemu-s
hell_0-gdb.sh  mount_fat0.sh  test2.img  utils
README.md  doc          iso          launch-qemu-gdb.sh  launch-qemu-t
est_1-gdb.sh  os          umount_fat0.sh
b.img      generate-test-script.sh  kernel.bin  launch-qemu-init-gdb.sh  launch-qemu.s
h          root        user
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ./launch-qemu.sh
WARNING: Image format was not specified for 'b.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'test2.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
QEMU 6.2.0 monitor - type 'help' for more information
(qemu) qemu-system-i386: terminating on signal 2
xypp@LAPTOP-U9SUCLKM:~/code/minios$ ./launch-qemu-gdb.sh
WARNING: Image format was not specified for 'b.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'test2.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
QEMU 6.2.0 monitor - type 'help' for more information
(qemu) ./launch-qemu-gdb.sh: line 54: gnome-terminal: command not found

bash: cd: /home/xypp/code/minios/m: No such file or directory
xypp@LAPTOP-U9SUCLKM:~/code/minios/$ cd ~/code/minios/
xypp@LAPTOP-U9SUCLKM:~/code/minios/$ sudo gdb kernel.gdb.bin
[sudo] password for xypp:
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (c) 2022 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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 kernel.gdb.bin...
(gdb) target remote :1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb) b kernel_main
Breakpoint 1 at 0xc012b2a0: file os/kernel/main.c, line 38.
(gdb) c
Continuing.

Breakpoint 1, kernel_main () at os/kernel/main.c:38
38 {
(gdb)
```