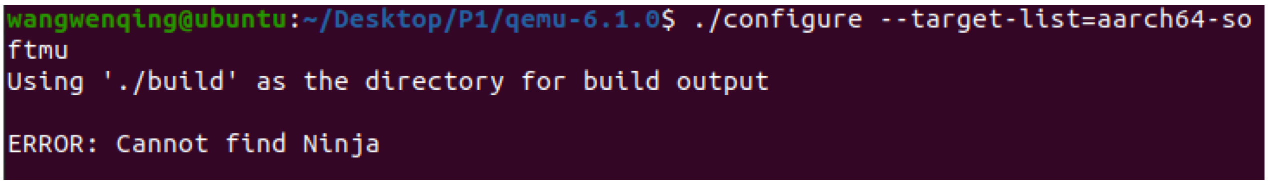
# project 1

## 1. 在 Ubuntu 上安装能够运行 aarch64 (64-bit ARM ISA)应用的 Qemu 虚拟机（qemu-aarch64）

**安装步骤**



```
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ ./configure --target-list=aarch64-so
ftmu
Using './build' as the directory for build output

ERROR: Cannot find Ninja
```

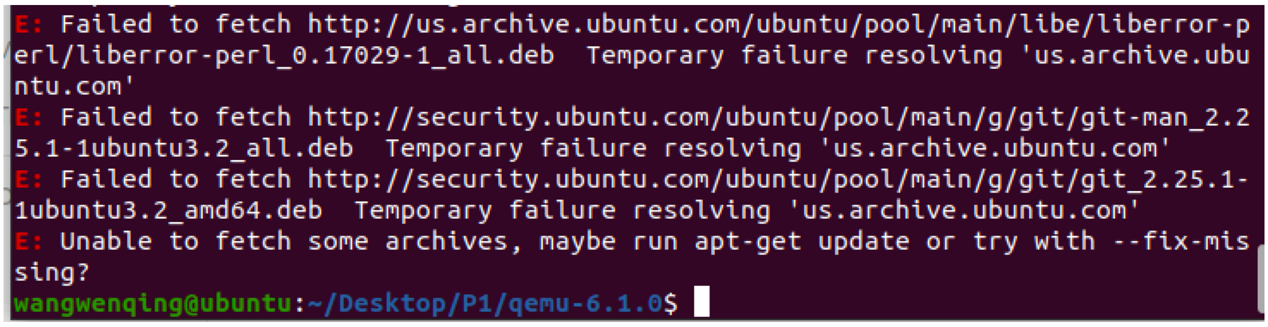
解决方法

参考链接https://blog.csdn.net/kelxLZ/article/details/111084537

https://www.cxyzjd.com/article/WMX843230304WMX/102628133

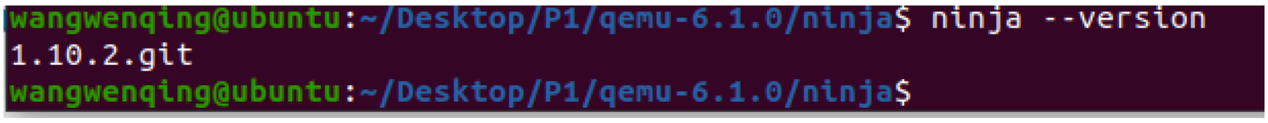
https://zhuanlan.zhihu.com/p/345232459

安装ninja过程中报错



```
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/libe/liberror-p
erl/liberror-perl_0.17029-1_all.deb  Temporary failure resolving 'us.archive.ubu
ntu.com'
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/g/git/git-man_2.2
5.1-1ubuntu3.2_all.deb  Temporary failure resolving 'us.archive.ubuntu.com'
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/g/git/git_2.25.1-
1ubuntu3.2_amd64.deb  Temporary failure resolving 'us.archive.ubuntu.com'
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis
sing?
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$
```

Ninja 安装参考https://blog.csdn.net/qiuguolu1108/article/details/103842556

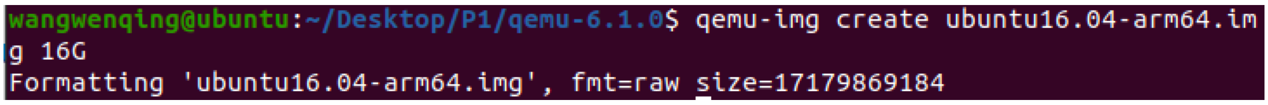


```
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0/ninja$ ninja --version
1.10.2.git
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0/ninja$
```

安装成功

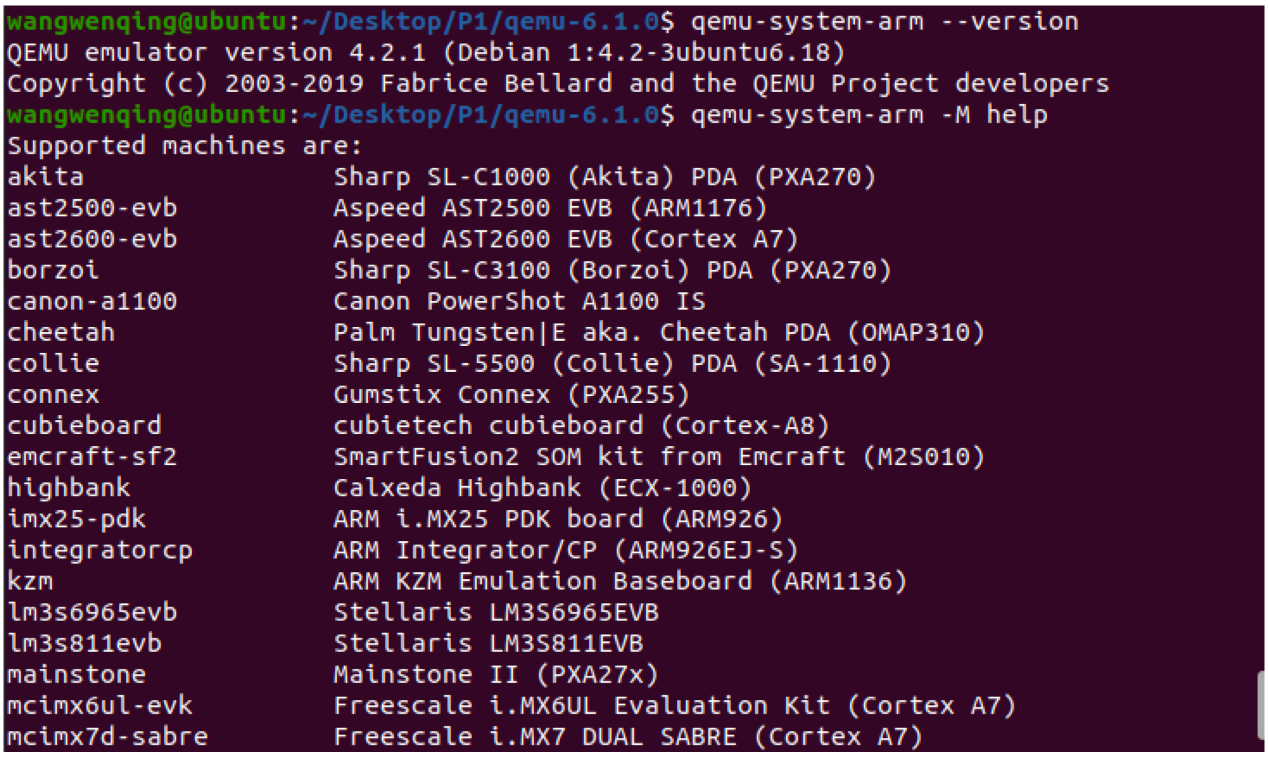
```
sudo apt-get install libglib2.0-dev
sudo apt-get install libpixman-1-dev
```

```
make
sudo make install
qemu-img create ubuntu16.04-arm64.img 16G
```



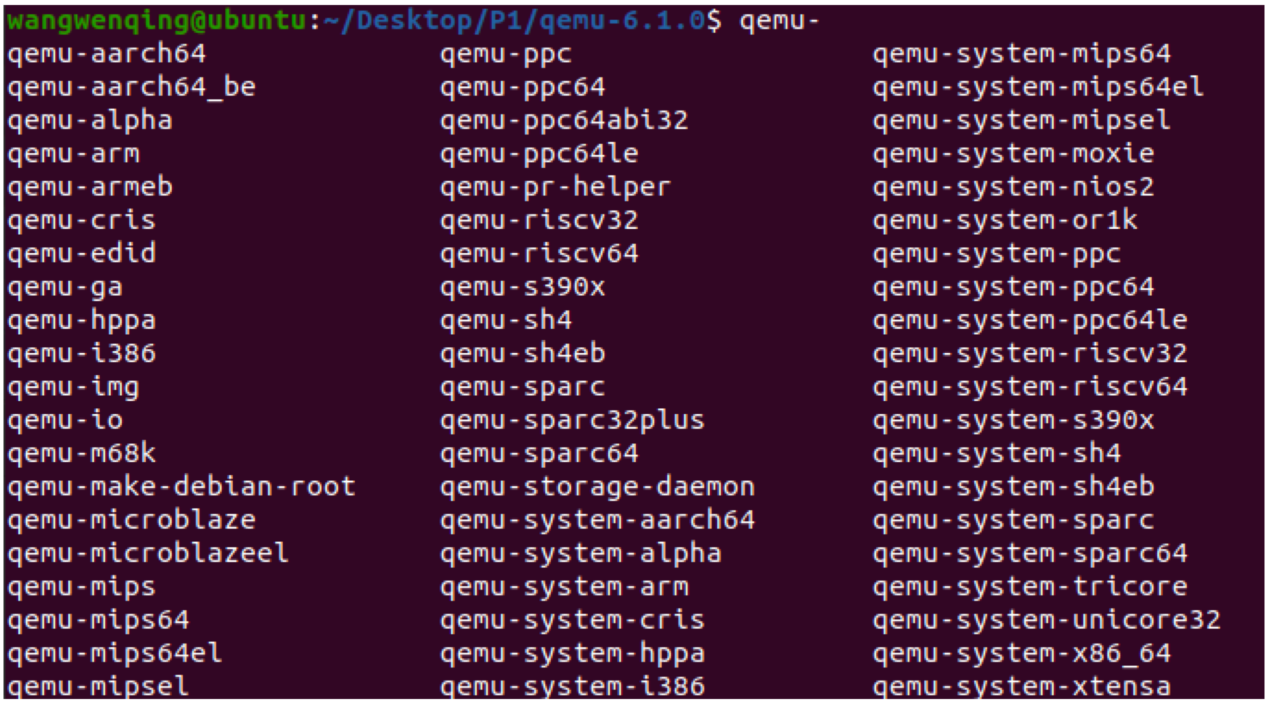
```
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ qemu-img create ubuntu16.04-arm64.im
g 16G
Formatting 'ubuntu16.04-arm64.img', fmt=raw size=17179869184
```

查看Qemu版本以及Qemu支持的开发板



```
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ qemu-system-arm --version
QEMU emulator version 4.2.1 (Debian 1:4.2-3ubuntu6.18)
Copyright (c) 2003-2019 Fabrice Bellard and the QEMU Project developers
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ qemu-system-arm -M help
Supported machines are:
akita                Sharp SL-C1000 (Akita) PDA (PXA270)
ast2500-evb          Aspeed AST2500 EVB (ARM1176)
ast2600-evb          Aspeed AST2600 EVB (Cortex A7)
borzoi               Sharp SL-C3100 (Borzoi) PDA (PXA270)
canon-a1100          Canon PowerShot A1100 IS
cheetah              Palm Tungsten|E aka. Cheetah PDA (OMAP310)
collie               Sharp SL-5500 (Collie) PDA (SA-1110)
connex               Gumstix Connex (PXA255)
cubieboard           cubietech cubieboard (Cortex-A8)
emcraft-sf2          SmartFusion2 SOM kit from Emcraft (M2S010)
highbank             Calxeda Highbank (ECX-1000)
imx25-pdk            ARM i.MX25 PDK board (ARM926)
integratorcp         ARM Integrator/CP (ARM926EJ-S)
kzm                  ARM KZM Emulation Baseboard (ARM1136)
lm3s6965evb          Stellaris LM3S6965EVB
lm3s811evb           Stellaris LM3S811EVB
mainstone            Mainstone II (PXA27x)
mcimx6ul-evk         Freescale i.MX6UL Evaluation Kit (Cortex A7)
mcimx7d-sabre        Freescale i.MX7 DUAL SABRE (Cortex A7)
```

安装成功的验证

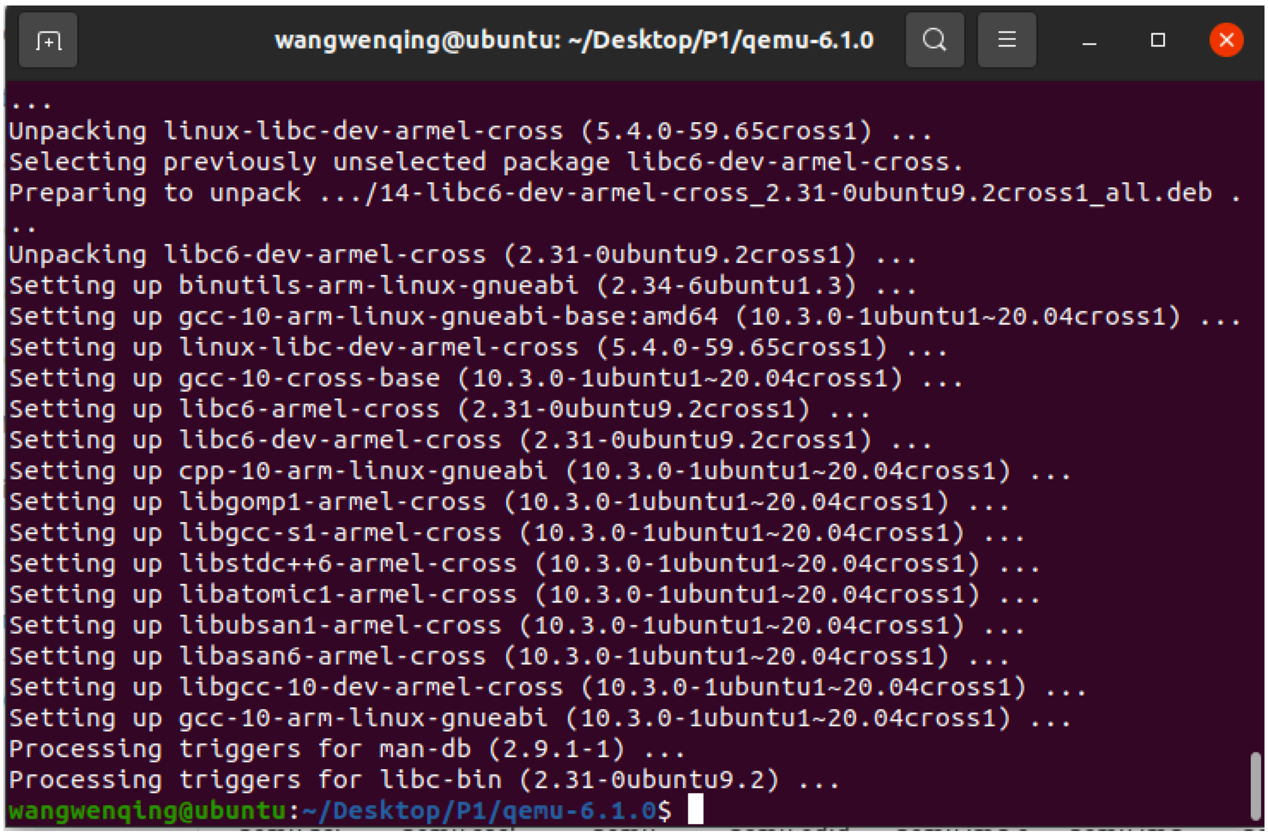


```
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ qemu-
qemu-aarch64             qemu-ppc                 qemu-system-mips64
qemu-aarch64_be          qemu-ppc64               qemu-system-mips64el
qemu-alpha               qemu-ppc64abi32          qemu-system-mipsel
qemu-arm                 qemu-ppc64le             qemu-system-moxie
qemu-armeb               qemu-pr-helper           qemu-system-nios2
qemu-cris                qemu-riscv32             qemu-system-or1k
qemu-edid                qemu-riscv64             qemu-system-ppc
qemu-ga                  qemu-s390x               qemu-system-ppc64
qemu-hppa                qemu-sh4                 qemu-system-ppc64le
qemu-i386                qemu-sh4eb               qemu-system-riscv32
qemu-img                 qemu-sparc               qemu-system-riscv64
qemu-io                  qemu-sparc32plus         qemu-system-s390x
qemu-m68k                qemu-sparc64             qemu-system-sh4
qemu-make-debian-root    qemu-storage-daemon      qemu-system-sh4eb
qemu-microblaze          qemu-system-aarch64      qemu-system-sparc
qemu-microblazeel        qemu-system-alpha        qemu-system-sparc64
qemu-mips                qemu-system-arm          qemu-system-tricore
qemu-mips64              qemu-system-cris         qemu-system-unicore32
qemu-mips64el            qemu-system-hppa         qemu-system-x86_64
qemu-mipsel              qemu-system-i386         qemu-system-xtensa
```

qemu-system-aarch64就是我们要使用的，用于模拟ARM64平台的qemu工具。

## 2. 安装 aarch64 的 GCC 工具链(gcc-10-aarch64-linux-gnu)

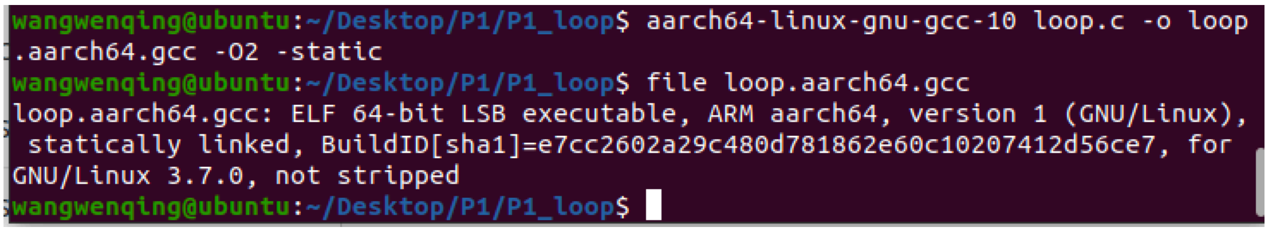
sudo apt-get install gcc-10-aarch64-linux-gnu



```
...
Unpacking linux-libc-dev-armel-cross (5.4.0-59.65cross1) ...
Selecting previously unselected package libc6-dev-armel-cross.
Preparing to unpack .../14-libc6-dev-armel-cross_2.31-0ubuntu9.2cross1_all.deb .
..
Unpacking libc6-dev-armel-cross (2.31-0ubuntu9.2cross1) ...
Setting up binutils-arm-linux-gnueabi (2.34-6ubuntu1.3) ...
Setting up gcc-10-arm-linux-gnueabi-base:amd64 (10.3.0-1ubuntu1~20.04cross1) ...
Setting up linux-libc-dev-armel-cross (5.4.0-59.65cross1) ...
Setting up gcc-10-cross-base (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libc6-armel-cross (2.31-0ubuntu9.2cross1) ...
Setting up libc6-dev-armel-cross (2.31-0ubuntu9.2cross1) ...
Setting up cpp-10-arm-linux-gnueabi (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libgomp1-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libgcc-s1-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libstdc++6-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libatomic1-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libubsan1-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libasan6-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up libgcc-10-dev-armel-cross (10.3.0-1ubuntu1~20.04cross1) ...
Setting up gcc-10-arm-linux-gnueabi (10.3.0-1ubuntu1~20.04cross1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
wangwenqing@ubuntu:~/Desktop/P1/qemu-6.1.0$ 
```

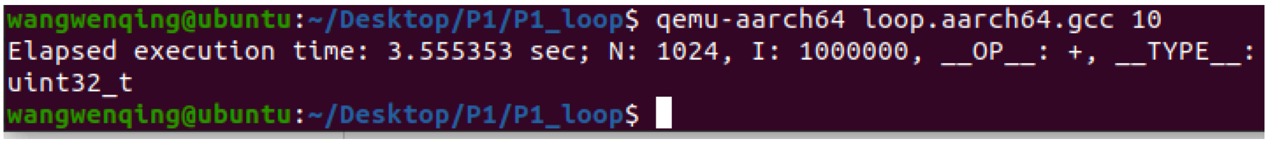
## 3. 用 aarch64 的 GCC 工具链交叉编译 loop.c (-O2)，生成可执行文件 loop.aarch64.gcc，并用 qemu-aarch64 运行loop.aarch64.gcc

注意需要使用静态编译



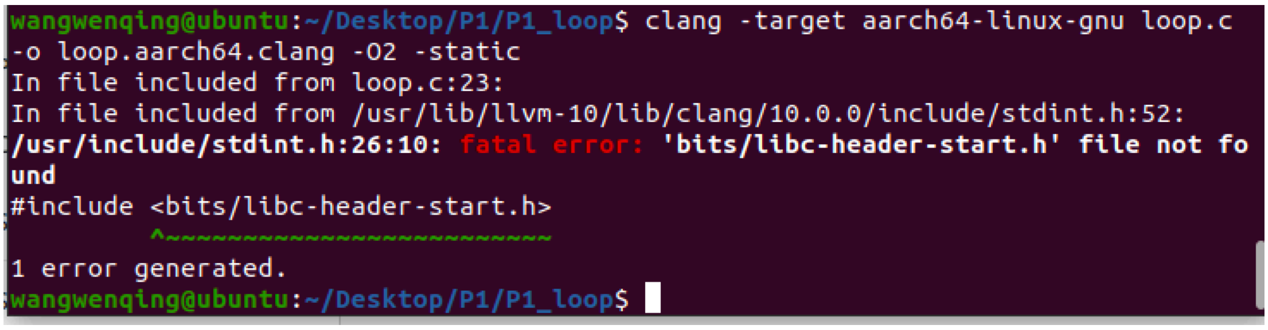
```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ aarch64-linux-gnu-gcc-10 loop.c -o loop
.aarch64.gcc -O2 -static
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ file loop.aarch64.gcc
loop.aarch64.gcc: ELF 64-bit LSB executable, ARM aarch64, version 1 (GNU/Linux),
 statically linked, BuildID[sha1]=e7cc2602a29c480d781862e60c10207412d56ce7, for
GNU/Linux 3.7.0, not stripped
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ 
```

用 qemu-aarch64 运行loop.aarch64.gcc



```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 10
Elapsed execution time: 3.555353 sec; N: 1024, I: 1000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ 
```

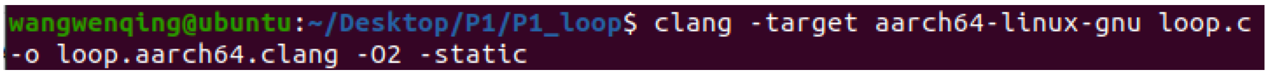
## 4.用 clang 交叉编译 loop.c(-O2)，生成可执行文件loop.aarch64.clang，并用 qemu-aarch64 运行loop.aarch64.clang



```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ clang -target aarch64-linux-gnu loop.c
-o loop.aarch64.clang -O2 -static
In file included from loop.c:23:
In file included from /usr/lib/llvm-10/lib/clang/10.0.0/include/stdint.h:52:
/usr/include/stdint.h:26:10: fatal error: 'bits/libc-header-start.h' file not fo
und
#include <bits/libc-header-start.h>
         ^~~~~~~~~~~~~~~~~~~~~~~~~~
1 error generated.
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ 
```

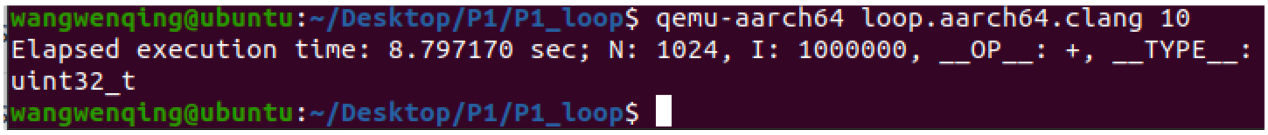
安装提供多平台编译的库文件

sudo apt-get install gcc-multilib



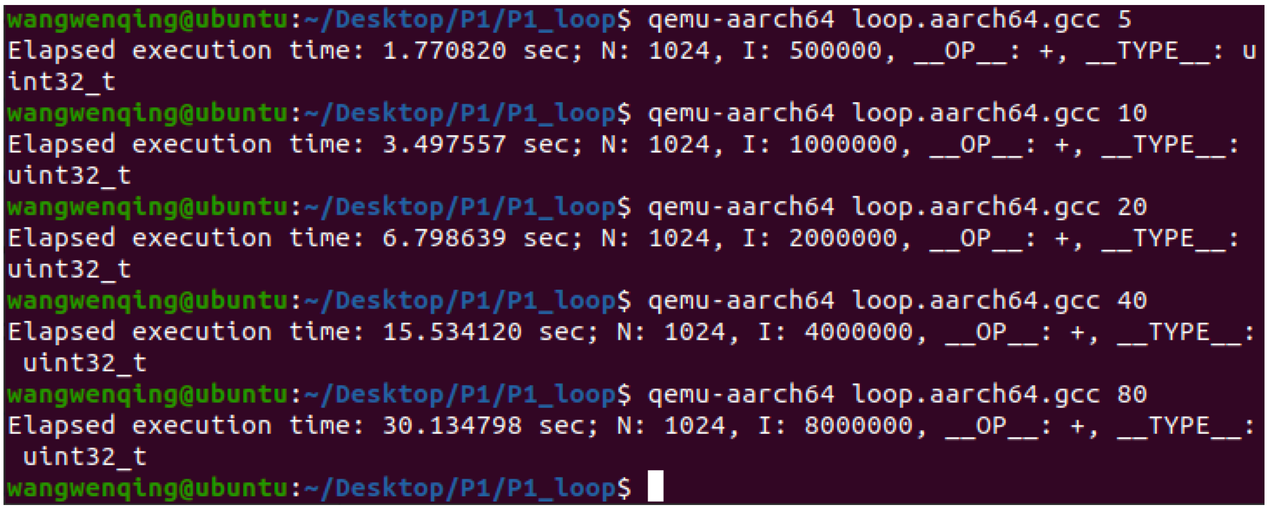
```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ clang -target aarch64-linux-gnu loop.c
-o loop.aarch64.clang -O2 -static
```

用 qemu-aarch64 运行loop.aarch64.clang

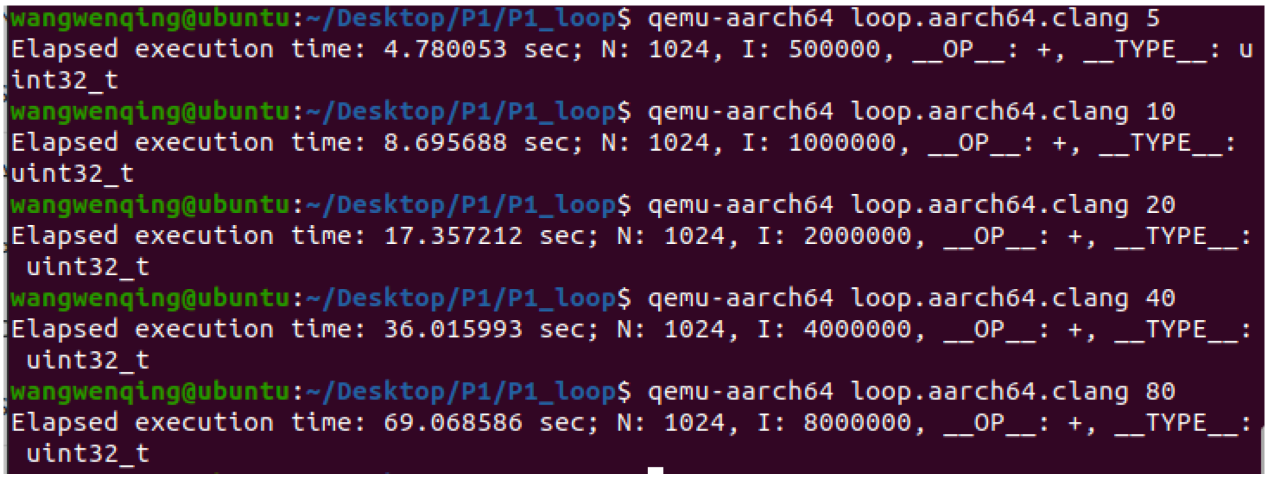


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 10
Elapsed execution time: 8.797170 sec; N: 1024, I: 1000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$
```

**5.用 qemu-aarch64 分 别 运 行 前 面 编 译 出 来 的loop.aarch64.gcc 和 loop.aarch64.clang（分别用参数5、10、20、40、80 进行测试），记下每次测试的执行时间并以图形方式呈现。**

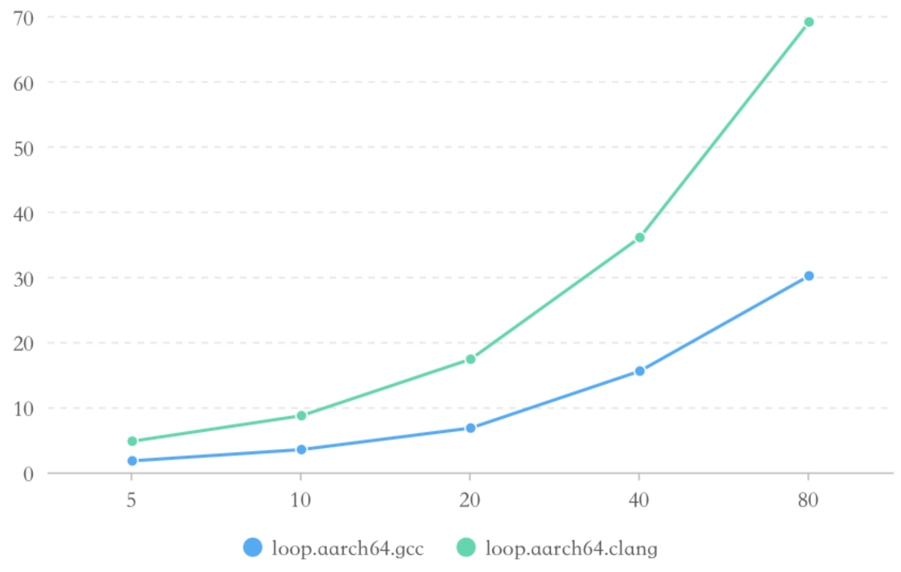


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 5
Elapsed execution time: 1.770820 sec; N: 1024, I: 500000, __OP__: +, __TYPE__: u
int32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 10
Elapsed execution time: 3.497557 sec; N: 1024, I: 1000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 20
Elapsed execution time: 6.798639 sec; N: 1024, I: 2000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 40
Elapsed execution time: 15.534120 sec; N: 1024, I: 4000000, __OP__: +, __TYPE__:
 uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.gcc 80
Elapsed execution time: 30.134798 sec; N: 1024, I: 8000000, __OP__: +, __TYPE__:
 uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$
```

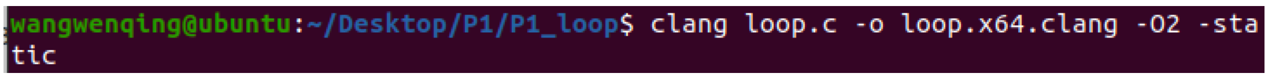


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 5
Elapsed execution time: 4.780053 sec; N: 1024, I: 500000, __OP__: +, __TYPE__: u
int32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 10
Elapsed execution time: 8.695688 sec; N: 1024, I: 1000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 20
Elapsed execution time: 17.357212 sec; N: 1024, I: 2000000, __OP__: +, __TYPE__:
 uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 40
Elapsed execution time: 36.015993 sec; N: 1024, I: 4000000, __OP__: +, __TYPE__:
 uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 loop.aarch64.clang 80
Elapsed execution time: 69.068586 sec; N: 1024, I: 8000000, __OP__: +, __TYPE__:
 uint32_t
```

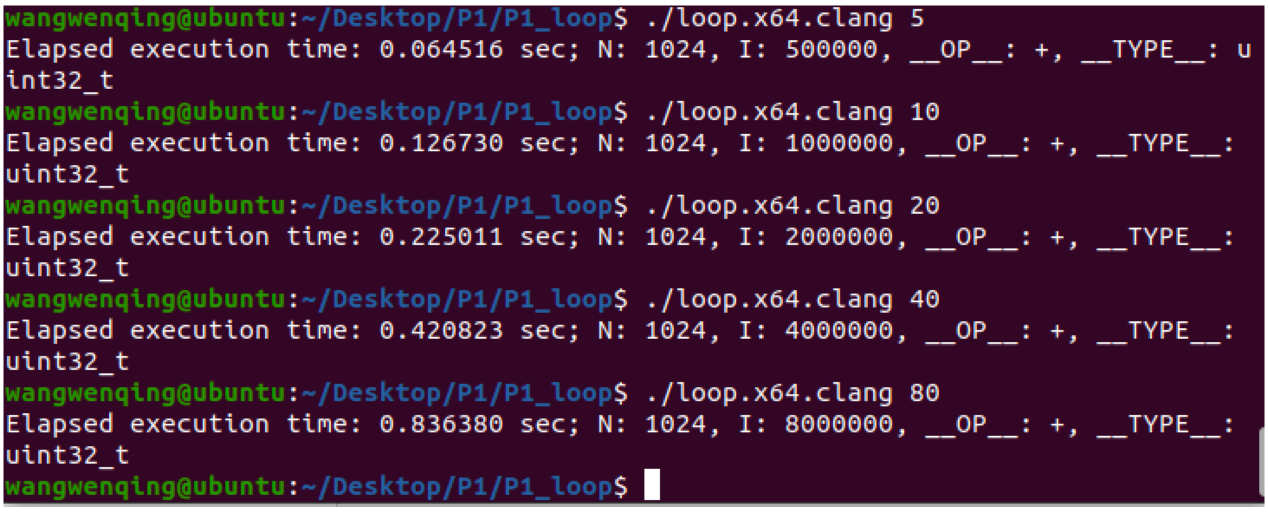
## qemu-aarch64编译结果



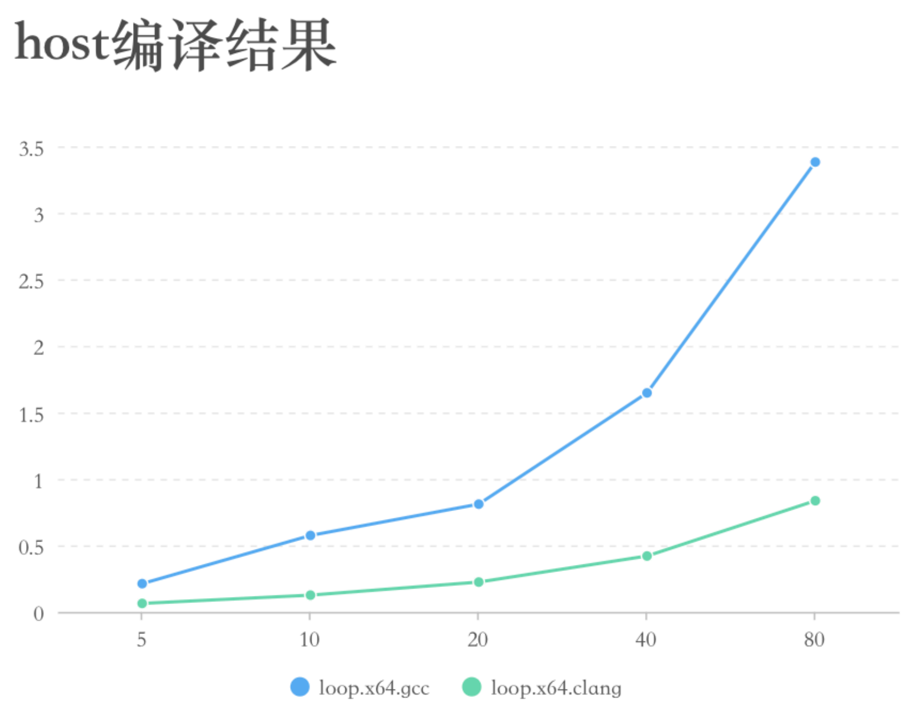
**6.用 host 机器上的 gcc 和 clang 分别编译(-O2)出loop.x64.gcc 和 loop.x64.clang，并对这两个执行文件分别用参数 5、10、20、40、80 进行测试，记下每次测试的执行时间并以图形方式呈现，进而与前一步 qemu 仿真测试的结果进行比较。**



```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ clang loop.c -o loop.x64.clang -O2 -sta
tic
```

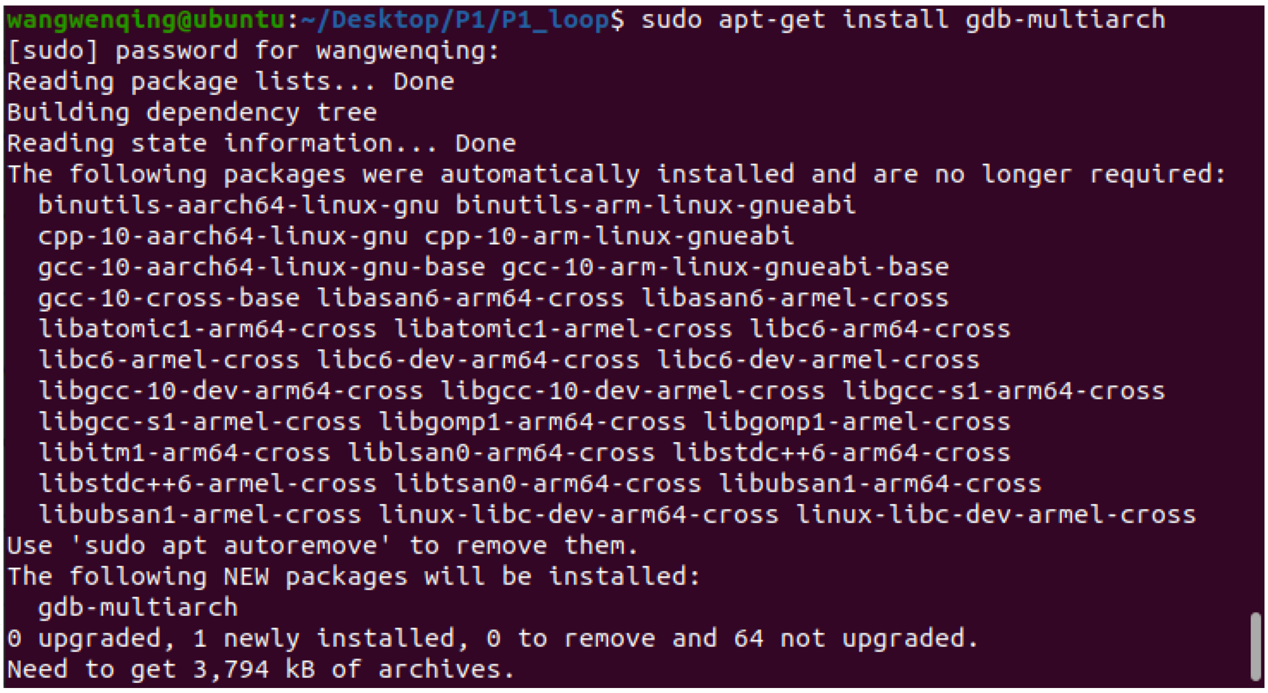


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ./loop.x64.clang 5
Elapsed execution time: 0.064516 sec; N: 1024, I: 500000, __OP__: +, __TYPE__: u
int32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ./loop.x64.clang 10
Elapsed execution time: 0.126730 sec; N: 1024, I: 1000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ./loop.x64.clang 20
Elapsed execution time: 0.225011 sec; N: 1024, I: 2000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ./loop.x64.clang 40
Elapsed execution time: 0.420823 sec; N: 1024, I: 4000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ./loop.x64.clang 80
Elapsed execution time: 0.836380 sec; N: 1024, I: 8000000, __OP__: +, __TYPE__:
uint32_t
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ 
```



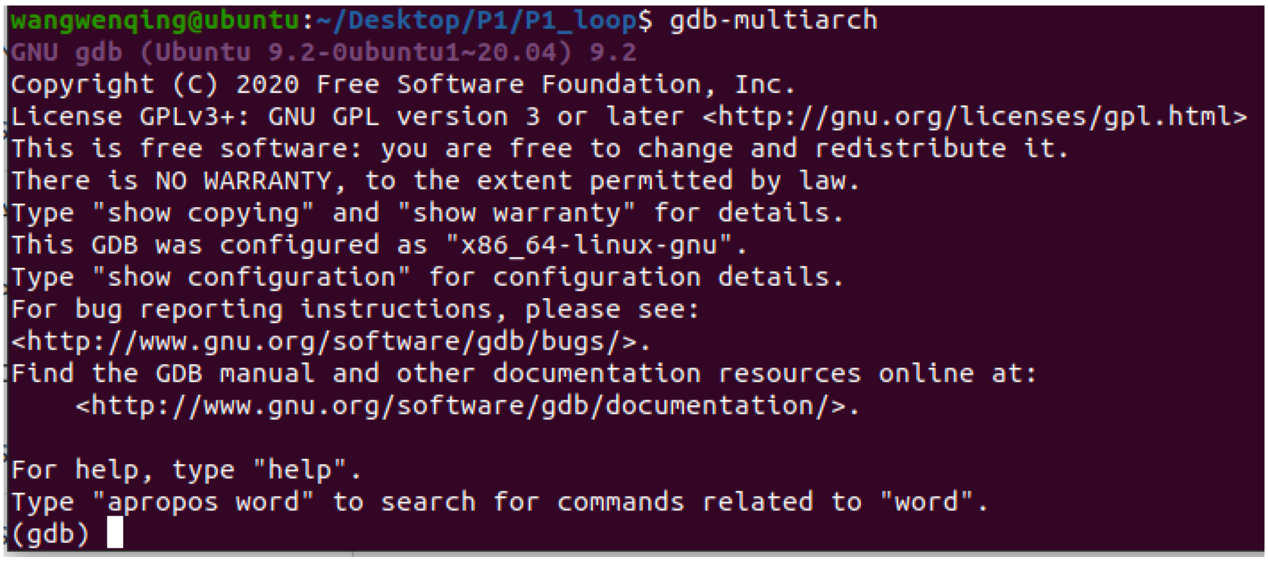
对比发现，不管是host模式还是qemu仿真测试情况，随着参数的增大，运行时间上升。相同参数，不管是gcc还是clang 编译，host编译时间要远小于qemu编译时间。此外，在host模式下，gcc的运行时间一直要高于clang且上升速度更快，而在则相反。

## 7.安装支持多 ISA 的 gdb 调试器



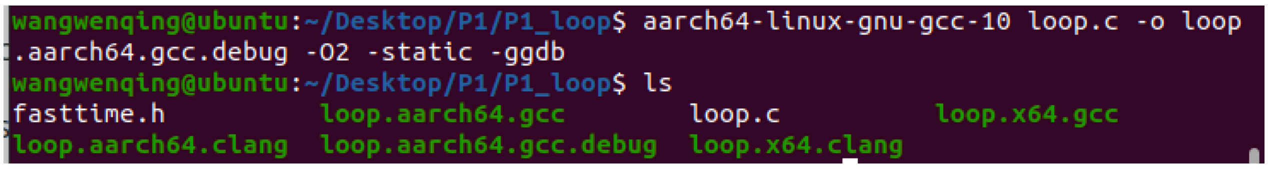
```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ sudo apt-get install gdb-multiarch
[sudo] password for wangwenqing:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  binutils-aarch64-linux-gnu binutils-arm-linux-gnueabi
  cpp-10-aarch64-linux-gnu cpp-10-arm-linux-gnueabi
  gcc-10-aarch64-linux-gnu-base gcc-10-arm-linux-gnueabi-base
  gcc-10-cross-base libasan6-arm64-cross libasan6-armel-cross
  libatomic1-arm64-cross libatomic1-armel-cross libc6-arm64-cross
  libc6-armel-cross libc6-dev-arm64-cross libc6-dev-armel-cross
  libgcc-10-dev-arm64-cross libgcc-10-dev-armel-cross libgcc-s1-arm64-cross
  libgcc-s1-armel-cross libgomp1-arm64-cross libgomp1-armel-cross
  libitm1-arm64-cross liblsan0-arm64-cross libstdc++6-arm64-cross
  libstdc++6-armel-cross libtsan0-arm64-cross libubsan1-arm64-cross
  libubsan1-armel-cross linux-libc-dev-arm64-cross linux-libc-dev-armel-cross
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  gdb-multiarch
0 upgraded, 1 newly installed, 0 to remove and 64 not upgraded.
Need to get 3,794 kB of archives.
```

安装成功验证


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ gdb-multiarch
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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:
<http://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".
(gdb) 
```

## 8. 用 gdb-multiarch 结 合 qemu-aarch64 对loop.aarch64.gcc.debug 进行源码级调试

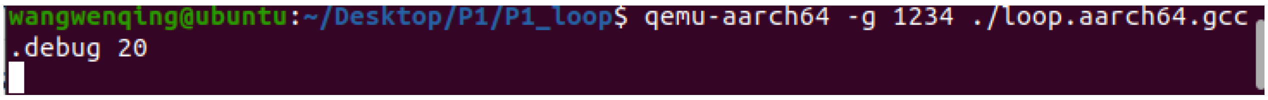
编译生成带调试信息的 loop.aarch64.gcc.debug


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ aarch64-linux-gnu-gcc-10 loop.c -o loop
.aarch64.gcc.debug -O2 -static -ggdb
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ ls
fasttime.h          loop.aarch64.gcc        loop.c          loop.x64.gcc
loop.aarch64.clang  loop.aarch64.gcc.debug  loop.x64.clang
```

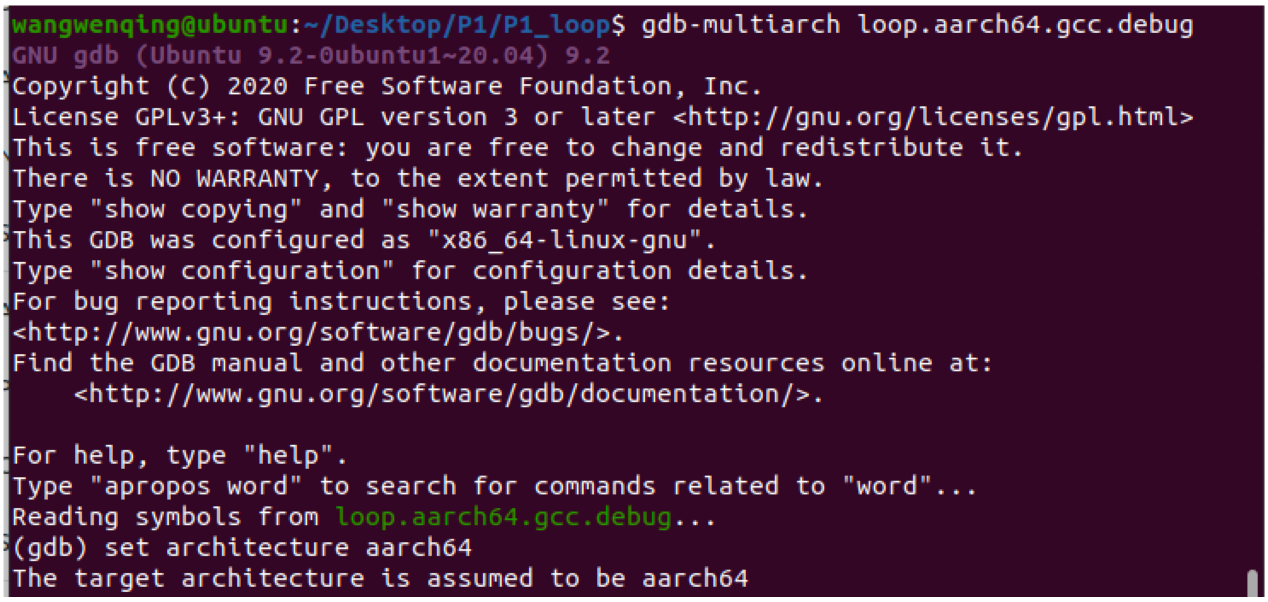
调试过程

https://zhuanlan.zhihu.com/p/47783910

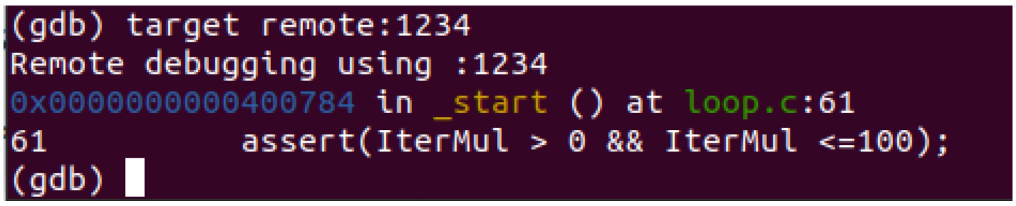
1.使用qemu-aarch64 -g传入端口后，并传入参数，等待gdb连接


```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ qemu-aarch64 -g 1234 ./loop.aarch64.gcc
.debug 20
```

2.启动gdb-multiarch并进入gdb后设置架构为aarch64

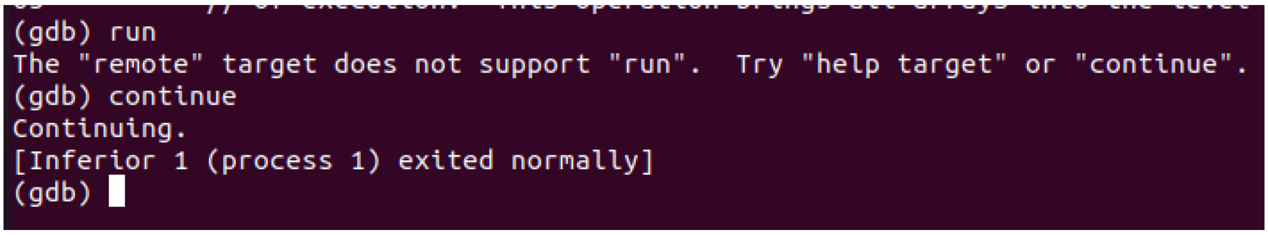

```
wangwenqing@ubuntu:~/Desktop/P1/P1_loop$ gdb-multiarch loop.aarch64.gcc.debug
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute 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:
<http://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 loop.aarch64.gcc.debug...
(gdb) set architecture aarch64
The target architecture is assumed to be aarch64
```

3.连接qemu的内核



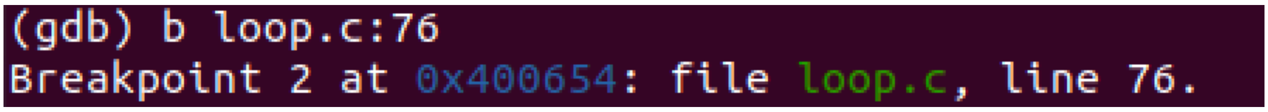
```
(gdb) target remote:1234
Remote debugging using :1234
0x0000000000400784 in _start () at loop.c:61
61            assert(IterMul > 0 && IterMul <=100);
(gdb) 
```

4.run和continue可以看到程序正常运行结束



```
(gdb) run
The "remote" target does not support "run".  Try "help target" or "continue".
(gdb) continue
Continuing.
[Inferior 1 (process 1) exited normally]
(gdb) 
```

5.打断点

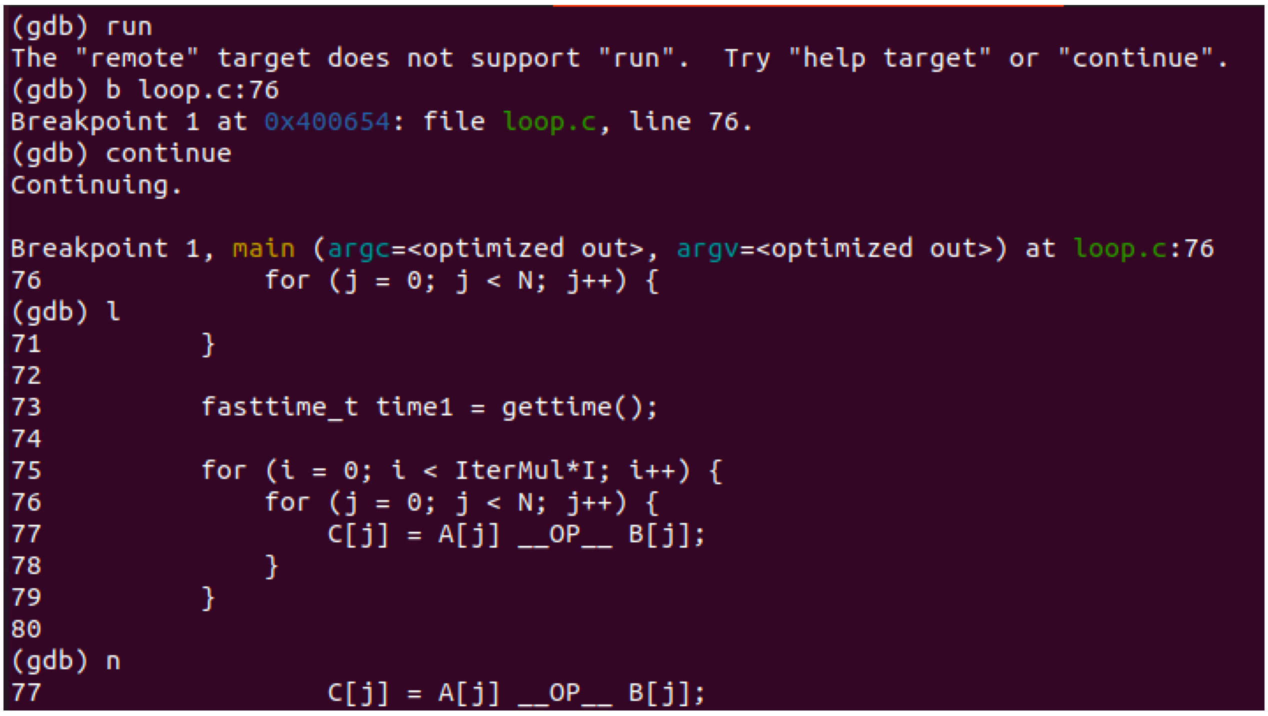


```
(gdb) b loop.c:76
Breakpoint 2 at 0x400654: file loop.c, line 76.
```

6.调试

单步调试n

查看当前代码l



```
(gdb) run
The "remote" target does not support "run".  Try "help target" or "continue".
(gdb) b loop.c:76
Breakpoint 1 at 0x400654: file loop.c, line 76.
(gdb) continue
Continuing.

Breakpoint 1, main (argc=<optimized out>, argv=<optimized out>) at loop.c:76
76                for (j = 0; j < N; j++) {
(gdb) l
71            }
72
73            fasttime_t time1 = gettime();
74
75            for (i = 0; i < IterMul*I; i++) {
76                for (j = 0; j < N; j++) {
77                    C[j] = A[j] __OP__ B[j];
78                }
79            }
80
(gdb) n
77                    C[j] = A[j] __OP__ B[j];
```