自适应网格辐射流体力学软件 FLASH 的安装

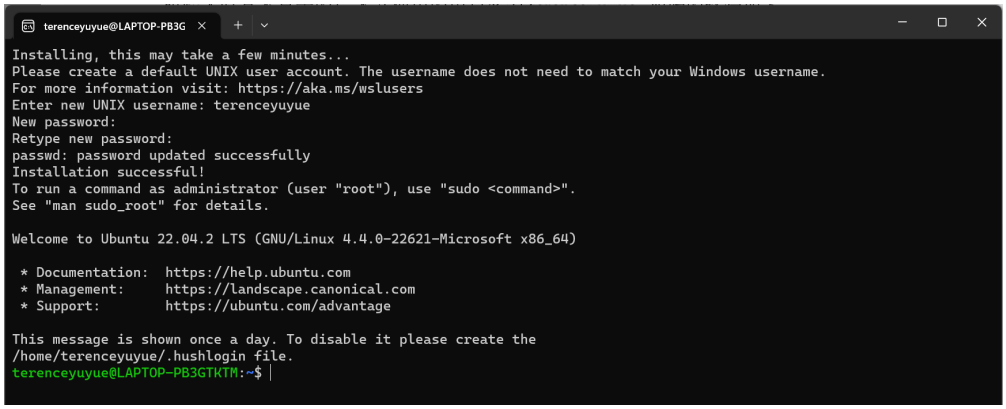
1 FLASH 的安装

安装过程参考了如下网页:

- https://github.com/astro-wjz/FLASH-code-installation
- https://zhuanlan.zhihu.com/p/259632602

1.1 Ubuntu 子系统的安装

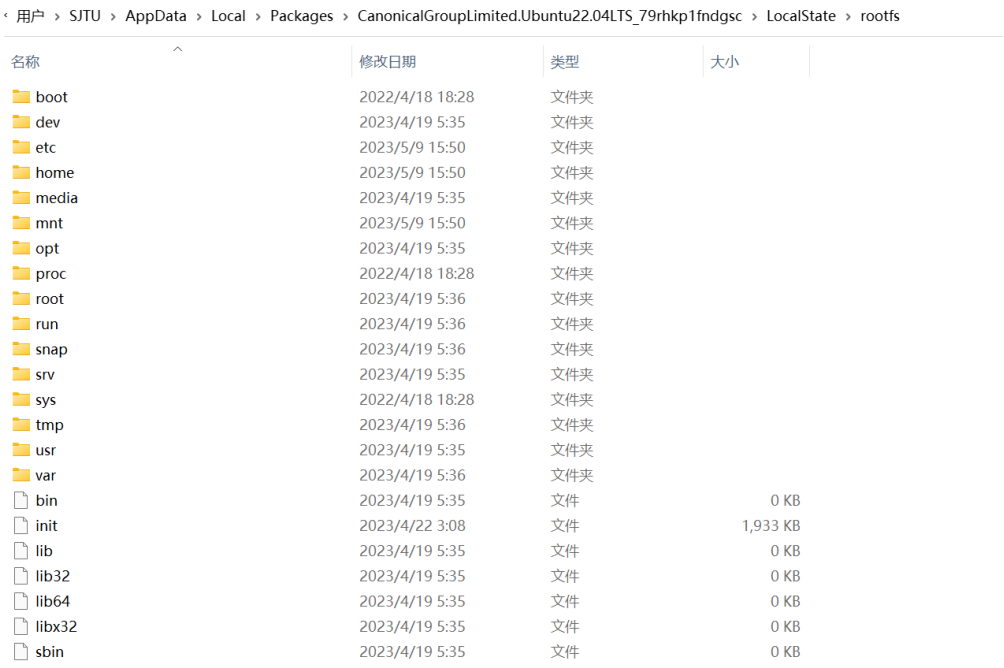
FLASH 是使用 Linux 系统的命令行形式进行操作的. Windows 系统上可以安装 Linux 子系统 (WSL) 来实现这些操作, 本文使用 Ubuntu 22.04.2 LTS. 安装方式非常简单, 只需要在 Microsoft Store 中搜索 Ubuntu 并下载安装即可. 初次使用会要求创建 UNIX username 和 password (注意密码输入时是不显示的). 本文使用的用户名为 terenceyuyue, 初始的终端如下



先简单了解一下 Ubuntu 中的目录结构, 这里的关键是根文件系统 rootfs, 它的位置为

`C:\Users\SJTU\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu22.04LTS_79rhkp1fndgsc\LocalState\rootfs`

其下的目录见下图



在 FLASH 的安装中, 本文使用 home 和 usr 两个目录. rootfs 下的目录相当于 C 盘和 D 盘等, 在进入这些目录时要加上斜杠“/”. home 称为用户目录, 其中有一个以用户名命名的子目录 terenceyuyue. 可直接用“cd ~”进入该子目录.

```
terenceyuyue@LAPTOP-PB3GTKTM:~$ pwd
/home/terenceyuyue
terenceyuyue@LAPTOP-PB3GTKTM:~$ cd /home
terenceyuyue@LAPTOP-PB3GTKTM:/home$ cd terenceyuyue
terenceyuyue@LAPTOP-PB3GTKTM:~$ cd ..
terenceyuyue@LAPTOP-PB3GTKTM:/home$ cd ~
terenceyuyue@LAPTOP-PB3GTKTM:~$
```

为了方便, 以下称 /home/terenceyuyue 为工作目录.

1.2 编译器的安装

安装前建议更新软件包

```
sudo apt-get update
```

为了避免每次都输 sudo, 可一开始进行

```
sudo passwd
su root
```

并创建和输入密码. 此时进入 root 模式. 以下默认进行了该步骤.

1. 安装 C/C++ 编译器

   ```
   apt install build-essential gdb
   ```

   检查是否安装成功

   ```
   gcc --version
   g++ --version
   gdb --version
   ```

   例如, gcc 的版本为: gcc (Ubuntu 11.3.0-1ubuntu1 22.04) 11.3.0

2. 安装 cmake

   ```
   apt install cmake
   cmake --version
   ```

   cmake 的版本为: cmake version 3.22.1

3. 安装 Fortran 编译器

   ```
   apt-get install gfortran
   gfortran --version
   ```

   Fortran 的版本为: GNU Fortran (Ubuntu 11.3.0-1ubuntu1 22.04) 11.3.0

使用 apt 方式安装的软件包的执行命令会存放在 /usr/bin 中.

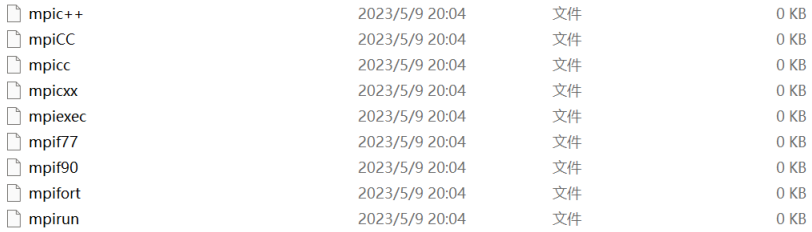
1.3 依赖包的安装

1.3.1 MPI 并行

FLASH 需要使用 MPI 并行, 可安装 mpich 或 openmpi. 这里使用 apt 方式安装 mpich (使用 openmpi 似乎有编译错误).

```
sapt-get install mpich
```

此时, /usr/bin 下会出现 mpi 开头的一些执行命令



FLASH 还需要 hdf5, hypre 和 zlib 等依赖包, 它们要使用源码方式安装. 源码安装的软件将放在 /usr/local 下.

1.3.2 zlib

zlib 软件包不一定是必须的 (至少对 Sedov 这个例子).

1. zlib 的源码可从如下网址下载:

   ```
   http://www.zlib.net
   ```

   下载的文件为 zlib-1.2.13.tar.gz, 并放在工作目录下.

2. 在工作目录下解压 zlib 压缩包

   ```
   tar -zxvf zlib-1.2.13.tar.gz
   ```

3. 进入解压后的文件并配置

   ```
   cd zlib-1.2.13
   ./configure --prefix=/usr/local/zlib
   ```

4. 编译并安装

   ```
   make
   make install
   ```

注 1.1 剩下的一个步骤是添加环境变量, 这里单独介绍一下.

(1) 可直接用命令行方式输入

```
export PATH=/usr/local/zlib/bin:$PATH
export INCLUDE=/usr/local/zlib/include:$INCLUDE
export LD_LIBRARY_PATH=/usr/local/zlib/lib:$LD_LIBRARY_PATH
```

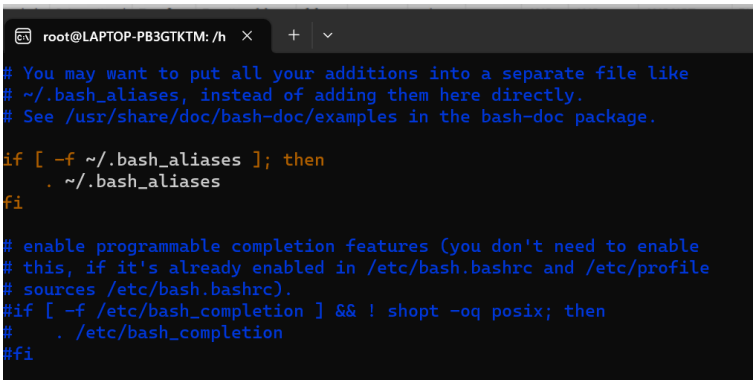
但它只是临时有效.

(2) 为了保证重启后有效, 需要在 .bashrc 中添加上面的语句.

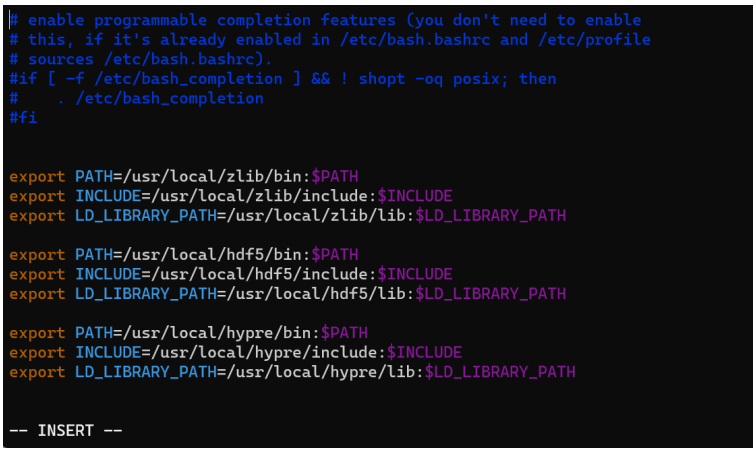
- 当前工作目录下有一个 .bashrc, 直接用 nodepad++ 打开它, 将上述语句添加到最后, 并在终端执行 source ~/.bashrc, 似乎不能奏效.
- 在 /etc/skel 下有一个 .bashrc, 用 nodepad++ 打开它, 将上述语句添加到最后, 并在终端执行 source /etc/skel/.bashrc, 则可以生效.

(3) 也可以在终端使用 vim 打开编辑.

- 在命令行输入 vim ~/.bashrc, 此时切换到如下界面



- 现在要把前面的语句加入到该界面的最后. 按动键盘上的 i, 底部会出现 – INSERT –, 这表明此时可进行编辑. 将上面的语句复制到最后 (通过键盘的上下左右定位光标).



这里把后面需要的一些环境变量也加进去了. 为了方便使用, 这里用文本给出

```
export PATH=/usr/local/zlib/bin:$PATH
export INCLUDE=/usr/local/zlib/include:$INCLUDE
export LD_LIBRARY_PATH=/usr/local/zlib/lib:$LD_LIBRARY_PATH

export PATH=/usr/local/hdf5/bin:$PATH
export INCLUDE=/usr/local/hdf5/include:$INCLUDE
export LD_LIBRARY_PATH=/usr/local/hdf5/lib:$LD_LIBRARY_PATH

export PATH=/usr/local/hypre/bin:$PATH
export INCLUDE=/usr/local/hypre/include:$INCLUDE
export LD_LIBRARY_PATH=/usr/local/hypre/lib:$LD_LIBRARY_PATH
```

- 按 Esc 退出 INSERT. 键盘输入 :x, 即可保存并退出.

1.3.3 hdf5

hdf5: FLASH 的数据输出依赖 hdf5, 它的安装过程如下:

1. 从如下网址下载源码:

   https://www.hdfgroup.org/downloads/hdf5/source-code/

   下载的文件为 hdf5-1.14.0.tar.gz, 并放在工作目录下.

2. 在工作目录下解压 hdf5 压缩包

   ```
   tar -zxvf hdf5-1.14.0.tar.gz
   ```

3. 进入解压后的文件并配置

   ```
   cd hdf5-1.14.0
   ./configure --enable-parallel --enable-fortran --enable-shared ...
       --prefix=/usr/local/hdf5 CC=mpicc FC=mpif90
   ```

   输入如下内容则配置成功

   ```
   Features:
   ---------
                        Parallel HDF5: yes
     Parallel Filtered Dataset Writes: yes
                  Large Parallel I/O: yes
                  High-level library: yes
   Dimension scales w/ new references: no
                   Build HDF5 Tests: yes
                   Build HDF5 Tools: yes
                    Build GIF Tools: no
                      Threadsafety: no
                Default API mapping: v114
       With deprecated public symbols: yes
               I/O filters (external):
                        Map (H5M) API: no
                           Direct VFD: no
                           Mirror VFD: no
                        Subfiling VFD: no
                  (Read-Only) S3 VFD: no
                (Read-Only) HDFS VFD: no
       Packages w/ extra debug output: none
                          API tracing: no
                Using memory checker: no
              Function stack tracing: no
                     Use file locking: best-effort
           Strict file format checks: no
         Optimization instrumentation: no
   ```

4. 编译并安装

   ```
   make
   make install
   ```

   编译大概需要 20 分钟. 注意, 如果编译出现问题, 可重新配置再编译. 成功后, /usr/local 目录下会生成 hdf5 文件夹. 若想检查是否安装成功, 可执行如下命令

   ```
   h5pcc -showconfig
   ```

   可能会出现如下提示

   ```
   Command 'h5pcc' not found, but can be installed with:
   apt install libhdf5-dev          # version 1.10.7+repack-4ubuntu2, or
   apt install libhdf5-mpich-dev    # version 1.10.7+repack-4ubuntu2
   apt install libhdf5-openmpi-dev  # version 1.10.7+repack-4ubuntu2
   ```

使用第二个命令, 并重新执行 h5pcc 即可, 结果如下

```
Features:
---------
                  Parallel HDF5: yes
Parallel Filtered Dataset Writes: yes
             Large Parallel I/O: yes
             High-level library: yes
               Build HDF5 Tests: yes
               Build HDF5 Tools: yes
                   Threadsafety: no
            Default API mapping: v18
  With deprecated public symbols: yes
         I/O filters (external): deflate(zlib),szip(encoder)
                            MPE:
                     Direct VFD: no
                     Mirror VFD: no
            (Read-Only) S3 VFD: yes
          (Read-Only) HDFS VFD: no
                       dmalloc: no
  Packages w/ extra debug output: none
                  API tracing: no
           Using memory checker: no
 Memory allocation sanity checks: no
         Function stack tracing: no
              Use file locking: best-effort
      Strict file format checks: no
   Optimization instrumentation: no
```

5. 添加环境变量: 见前面的说明.

6. 测试: 在 /usr/local/hdf5/share/hdf5_examples 下有一些测试例子, 其中的 run-all-ex.sh 是可执行文件, 用以测试例子. 进入该目录, 在命令行输入 `./run-all-ex.sh`, 即可出现测试结果, 最后一部分如下

```
##################  ex_table_10  ##################
Table has 5 fields and 16 records
zero  0     0     0.000000 0.000000
one   10    10    1.000000 10.000000
two   20    20    2.000000 20.000000
three 30    30    3.000000 30.000000
four  40    40    4.000000 40.000000
five  50    50    5.000000 50.000000
six   60    60    6.000000 60.000000
seven 70    70    7.000000 70.000000
zero  0     0     0.000000 0.000000
one   10    10    1.000000 10.000000
two   20    20    2.000000 20.000000
three 30    30    3.000000 30.000000
four  40    40    4.000000 40.000000
five  50    50    5.000000 50.000000
six   60    60    6.000000 60.000000
seven 70    70    7.000000 70.000000

##################  ex_table_11  ##################
Table has 6 fields and 8 records

##################  ex_table_12  ##################
Table has 4 fields and 8 records

##################  ex_ds1  ##################

Run hl fortran examples

##################  exlite  ##################

Finished running hl examples
Done
```

1.3.4 hypre

hypre 是大型线性方程组的求解器.

1. 从 github 上下载源码并放置在工作目录下:

```
git clone https://github.com/hypre-space/hypre
```

2. 进入 hypre/src 目录, 并如下配置

```
cd hypre/src
./configure --prefix=/usr/local/hypre CC=mpicc FC=mpif90
```

3. 编译并安装

```
make
make install
```

编译大概需要 3 分钟. 成功后, /usr/local 目录下会生成 hypre 文件夹.

4. 添加环境变量: 见 zlib 那里的说明.

5. 测试: 进入工作目录中的 hypre/src/examples, 将 Makefile 中的 HYPRE_DIR 路径写为

`HYPRE_DIR = /usr/local/hypre`.

该语句在本机上位于第 21 行, 如下图

```
 8  ##########################################################################
 9  CC         = mpicc
10  F77        = mpif77
11  CXX        = mpicxx
12  F90        = mpif90
13  CUF        = mpixlcuf
14  LINK_CC    = ${CC}
15  LINK_CXX   = ${CXX}
16  LINK_FC    = ${F77}
17  LINK_CUF   = ${CUF}
18
19  XL_DIR=$(dir $(shell which xlc))..
20
21  HYPRE_DIR = /usr/local/hypre
22
```

在 examples 目录下如下测试

```
make
mpirun -np 2 ./ex1
```

结果如下

```
root@LAPTOP-PB3GTKTM:/home/terenceyuyue/hypre/src/examples# mpirun -np 2 ./ex1
<C*b,b>: 1.800000e+01


Iters       ||r||_C     conv.rate  ||r||_C/||b||_C
-----    ------------    ---------  ------------
    1    2.509980e+00    0.591608    5.916080e-01
    2    9.888265e-01    0.393958    2.330686e-01
    3    4.572262e-01    0.462393    1.077693e-01
    4    1.706474e-01    0.373223    4.022197e-02
    5    7.473022e-02    0.437922    1.761408e-02
    6    3.402624e-02    0.455321    8.020061e-03
    7    1.214929e-02    0.357057    2.863616e-03
    8    3.533113e-03    0.290808    8.327628e-04
    9    1.343893e-03    0.380371    3.167586e-04
   10    2.968745e-04    0.220906    6.997400e-05
   11    5.329671e-05    0.179526    1.256215e-05
   12    7.308483e-06    0.137128    1.722626e-06
   13    7.411552e-07    0.101410    1.746920e-07
```

1.4 FLASH 的安装

FLASH 的安装需要在其官网上申请账号, 通过之后才能下载 (一般要个几天才能收到邮件). 官方网址为 https://flash.rochester.edu/site/

1. 解压下载的源码

```
tar -xzvf FLASH4.7.tar.gz
```

2. 进入 FLASH4.7 目录, 并如下配置 (以最基本的 Sedov 激波问题为例)

```
cd FLASH4.7
./setup Sedov -auto
```

这里可能出现如下提示: /usr/bin/env: 'python': No such file or directory. 这是因为目录中的是 python3, 而不是 python. 可通过符号链接解决, 过程如下: 在终端输入

```
python3 --version
whereis python3
```

本机提示的位置如下

python3: /usr/bin/python3 /usr/lib/python3 /etc/python3 /usr/share/python3 /usr/share/man/man1/python3.1.gz

创建符号链接:

```
sudo ln -s /usr/bin/python3 /usr/bin/python
```

重新配置: `./setup Sedov -auto`, 结果如下

```
    flashUtilities/general
    flashUtilities/interpolation/oneDim
    flashUtilities/nameValueLL
    flashUtilities/sorting/quicksort
    flashUtilities/system/memoryUsage/legacy
    monitors/Logfile/LogfileMain
    monitors/Timers/TimersMain/MPINative
    physics/Eos/EosMain/Gamma
    physics/Eos/localAPI
    physics/Hydro/HydroMain/unsplit/Hydro_Unsplit
    physics/Hydro/localAPI
Computing default values for options not specified on command line
Using Makefile.h: /home/terenceyuyue/FLASH4.7/sites/Prototypes/Linux/Makefile.h
generating Makefile
Copying data files: 3 copied
SUCCESS
```

3. Makefile 文件的修改: 注意上图中的倒数第 4 行给出了 Makefile 文件的位置

Using Makefile.h: /home/terenceyuyue/FLASH4.7/sites/Prototypes/Linux/Makefile.h

Makefile.h 中的内容如下

```
# FLASH makefile definitions for x86-64 Linux (GNU compilers)
#------------------------------------------------------------------------------
# Set the HDF5/MPI library paths -- these need to be updated for your system
#------------------------------------------------------------------------------

MPI_PATH   = /usr/local/mpich2/
HDF4_PATH  =
HDF5_PATH  = /usr/local/hdf5
HYPRE_PATH = /usr/local/hypre

ZLIB_PATH  =

PAPI_PATH  =
PAPI_FLAGS =

NCMPI_PATH = /usr/local/netcdf
MPE_PATH   =

```

这里 MPI_PATH 的路径应修改为 MPI_PATH = /usr.

4. 编译: 修改 Makefile.h 后重新配置, FLASH4.7 下会生成 object 文件夹. 进入该目录, 并编译

```
cd object
make
```

编译需要四五分钟的时间.

5. 运行案例: 终端输入如下命令

```
mpirun -np 4 ./flash4
```

可得如下输出

```
root@LAPTOP-PB3GTKTM:/home/terenceyuyue/FLASH4.7/object# mpirun -np 4 ./flash4
 Grid_init: resolution based on runtime params:
  lrefine          dx              dy
      1            0.125           0.125
      2            0.062           0.062
      3            0.031           0.031
      4            0.016           0.016
      5            0.008           0.008
      6            0.004           0.004
 MaterialProperties initialized
 Cosmology initialized
 Source terms initialized
 sim_rhoAmbient is   1.0000000000000000
  iteration, no. not moved =            0           0
 refined: total leaf blocks =            1
 refined: total blocks =            1
 [amr_morton_process]: Initializing surr_blks using standard orrery implementation
  iteration, no. not moved =            0           0
 refined: total leaf blocks =            4
 refined: total blocks =            5
  iteration, no. not moved =            0           0
 refined: total leaf blocks =           16
 refined: total blocks =           21
  iteration, no. not moved =            0           0
 refined: total leaf blocks =           28
 refined: total blocks =           37
  iteration, no. not moved =            0           2
  iteration, no. not moved =            1           0
 refined: total leaf blocks =           40
 refined: total blocks =           53
  iteration, no. not moved =            0           7
  iteration, no. not moved =            1           0
 refined: total leaf blocks =           52
 refined: total blocks =           69
  Finished with Grid_initDomain, no restart
 Ready to call Hydro_init
 Hydro initialized
 Gravity initialized
 Initial dt verified
 *** Wrote checkpoint file to sedov_hdf5_chk_0000 ****
 *** Wrote plotfile to sedov_hdf5_plt_cnt_0000 ****
 Initial plotfile written
 Driver init all done
       n         t         dt  (         x,          y,          z) |  dt_hydro
       1 1.0000E-10 2.0000E-10  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       2 3.0000E-10 4.0000E-10  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       3 7.0000E-10 8.0000E-10  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       4 1.5000E-09 1.6000E-09  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       5 3.1000E-09 3.2000E-09  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       6 6.3000E-09 6.4000E-09  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
       7 1.2700E-08 1.2800E-08  ( 0.498    ,  0.490    ,   0.00    ) |  1.012E-04
```

6. 结果查看: 见名称以 sedov 开头的文件, 如 sedov.dat.

注 1.2 重启电脑后, 有时候还会遇到如下错误:

./flash4: error while loading shared libraries: libhdf5.so.310: cannot open shared object file: No such file or directory

这还是 hdf5 环境变量的问题. 前面已经处理了这个问题, 不知道为什么还是遇到该提示. 在终端输入 vim ~/.bashrc 发现 .bashrc 文件中没有环境变量内容. 为此, 只好重复前面的步骤. 问题在哪还不清楚, 可能使用纯粹的 Linux 系统不会有该问题.