

Ubuntu 中 lammmps 串并行安装带 reax/meam/poems 全解读

(本 lammmps 在 Ubuntu13.04 下安装亲测多次，其余版本类似或相同)

目 录

| | |
|---|----|
| 1 并行安装普通 lammmps 包 | 2 |
| 1.1 下载需要的安装包 | 2 |
| 1.2 检查机器是否已经安装 c 和 fortran 编译器 | 2 |
| 1.3 安装 fftw | 3 |
| 1.4 安装 mpich | 3 |
| 1.4 安装 lammmps | 4 |
| 1.5 安装 lammmps 后并行运算 example | 7 |
| 2 特殊 lammmps 库安装包的编译 | 8 |
| 2.1 关于 lammmps 库文件的说明 | 8 |
| 2.2 关于 lammmps 库文件安装包的查看 | 9 |
| 2.3 lammmps 中 reax 安装包的编译(串行) | 9 |
| 2.4 lammmps 中 meam 和 poems 安装包的编译(并行) | 13 |
| 3 VMware 虚拟机给 ubuntu 安装 VMware tools | 17 |
| 4 关于 ubuntu 更新源地址问题 | 17 |
| 5 关于 ubuntu 版本的说明 | 18 |

常用的几个 ubuntu 命令：

`sudo nautilus`; 获取访问系统文件夹的权限;

`cp`; 文件复制;

`mv`; 文件移到;

`cd`; 打开文件;

1 并行安装普通 lammps 包

1.1 下载需要的安装包

(1) **fftw-2.1.5.tar.gz**, 可以到这里下, <http://www.fftw.org/download.html>, 当然更高版本的也可以, 如最新的 **fftw-3.3.4.tar.gz**, 但是后面安装过程要把 **fftw2** 改成 **fftw3**。

(FFTWT 是计算离散 Fourier 变换 (DFT) 的快速 C 程序的一个完整集合, 它可计算一维或多维、实和复数据以及任意规模的 DFT。FFTW 还包含对共享和分布式存储系统的并行变换。FFTW 由麻省理工学院计算机科学实验室超级计算技术组开发。)

(2) **mpich2_1.4.1.orig.tar.gz**, 可以到这里下, <https://www.mpich.org/downloads/>, 当然更高版本的也可以, 如最新的 **mpich-3.1.4.tar.gz**, 但是后面安装过程要把 **mpich2** 改成 **mpich3**。

(通过安装 MPICH 构建 MPI 编程环境, 从而进行并行程序的开发。MPICH 是 MPI (Message-Passing Interface) 的一个应用实现, 支持最新的 MPI-2 接口标准, 是用于并行运算的工具。)

(3) **lammps.tar.gz**, 版本号 28 Jun 2014, <http://lammps.sandia.gov/download.html>

(LAMMPS 即 Large-scale Atomic/Molecular Massively Parallel Simulator, 可以翻译为大规模原子分子并行模拟器, 主要用于分子动力学相关的一些计算和模拟工作。)

1.2 检查机器是否已经安装 c 和 fortran 编译器

Linux 一般有 gcc, g++, gfortran 和 intel 的 fortran, 可以在终端中用 **which g++** 和 **which gfortran** 查看是否存在安装目录。(注意后文中所有用黄色突出显示的都是终端中输入的)

这里用 ubuntu 源里的 g++ 和 gfortran, 保证联网的情况下用以下命令:

```
sudo apt-get install build-essential
```

```
sudo apt-get install g++
```

```
sudo apt-get install gfortran
```

其中 gfortran 是 fortran 编译器包, 而 build-essential 是 GNU c/c++ 命令行编译器包, 安装后可用下面命令测试一下。

查询 g++ 是否安装:

```
whereis g++
```

1.3 安装 fftw

采用自定义安装目录的方法，一般将 fftw 安装到/usr/local/fftw2 内，这样如果要卸载软件，直接删除该目录即可。

终端输入:

```
cd /usr/local/src
```

 (进入/usr/local/src 文件夹，注意这个是 fftwcd 临时存放文件夹，不是安装目录，服务器中最好放在自己的文件夹下面)

```
sudo tar xzvf ~/Downloads/fftw-2.1.5.tar.gz
```

 (解压文件，后面的解压目录~/Downloads/fftw-2.1.5.tar.gz 为你实际放压缩包地址)

```
cd fftw-2.1.5
```

 (进入 fftw-2.1.5 文件夹)

```
sudo ./configure --prefix=/usr/local/fftw2 --enable-float
```

 (进行安装前注册)

/usr/local/fftw2 是安装目录，可根据需要进行更改，这个路径对后面的 Makefile.g++有影响;

```
sudo make
```

 (预编译)

```
sudo make install
```

 (安装 fftw-2.1.5 在/usr/local/fftw2 下)

1.4 安装 mpich

终端输入:

```
cd /usr/local/src
```

 (进入/usr/local/src 文件夹，注意这个是 fftwcd 临时存放文件夹，不是安装目录，服务器中最好放在自己的文件夹下面)

```
sudo tar xzvf ~/Downloads/mpich2-1.4.1p1.tar.gz
```

 (解压文件，~/Downloads/mpich2-1.4.1p1.tar.gz 也是你实际存放压缩包地址)

```
cd mpich2-1.4.1p1
```

 (进入 mpich2-1.4.1p1 文件夹)

```
sudo ./configure --prefix=/usr/local/mpich2
```

 (进行安装前注册，/usr/local/mpich2 是安装目录，可根据需要进行更改，这个路径对后面的 Makefile.g++有影响)

```
sudo make
```

 (预编译)

```
sudo make install
```

 (安装 mpich2-1.4.1p1 即安装在/usr/local/fftw2 下)

如果上述方法在你的电脑中执行失败，请参考如下简单方法（联网的情况下）:

```
sudo apt-get install mpich2
```

```
cd ~
```

```
touch .mpd.conf
```

```
chmod 600 .mpd.conf
```

```
echo "MPD_SECRETWORD=mr45-j9z">.mpd.conf (mpich 即安装在/usr/include/mpich2 下)
```

1.4 安装 lammmps

`cd /home/lammmps/` 该目录为你实际实际存放 lammmps 压缩包的地址,建议放到主文件下。

```
gunzip lammmps.tar.gz
```

```
tar xvf lammmps.tar
```

```
mv lammmps-22Mar13 lmp
```

```
cd /lmp/src
```

下面就是重要的 Makefile.g++的编译了。先敲两行命令调出 Makefile。

```
cd ~ /lmp/src/MAKE
```

```
gedit Makefile.g++
```

这里的東西比較難改,我已經做好了一個如果路徑一樣可以直接用我的 Makefile.g++; 如果路徑不一樣,黃色突出部分需要修改。

```
# g++ = RedHat Linux box, g++4, MPICH2, FFTW

SHELL = /bin/sh

# -----
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC = g++
CCFLAGS = -g -O
SHFLAGS = -fPIC
DEPFLAGS = -M

LINK = g++
LINKFLAGS = -g -O
LIB =
SIZE = size

ARCHIVE = ar
ARFLAGS = -rc
SHLIBFLAGS = -shared

# -----
```

```
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use
# if you change any -D setting, do full re-compile after "make clean"

# LAMMPS ifdef settings, OPTIONAL
# see possible settings in doc/Section_start.html#2_2 (step 4)

LMP_INC = -DLAMMPS_GZIP

# MPI library, REQUIRED
# see discussion in doc/Section_start.html#2_2 (step 5)
# can point to dummy MPI library in src/STUBS as in Makefile.serial
# INC = path for mpi.h, MPI compiler settings
# PATH = path for MPI library
# LIB = name of MPI library

MPI_INC = -DMPICH_SKIP_MPICH2 -I/usr/include/mpich2 # 即 mpi.h 的路径
MPI_PATH = -L/usr/lib # 即 libmpich.a 的路径
MPI_LIB = -lmpich -lmpi -lpthread

# FFT library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 6)
# can be left blank to use provided KISS FFT library
# INC = -DFFT setting, e.g. -DFFT_FFTW, FFT compiler settings
# PATH = path for FFT library
# LIB = name of FFT library

FFT_INC = -DFFT_FFTW2 -I/usr/local/fftw2/include #即 fftw.h 的路径
FFT_PATH = -L/usr/local/fftw2/lib # 即 libfftw.a 的路径
FFT_LIB = -lfftw # 理解为将-l换成lib，后面添加.a后缀，就是libfftw.a这个库文件

# JPEG and/or PNG library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 7)
# only needed if -DLAMMPS_JPEG or -DLAMMPS_PNG listed with LMP_INC
# INC = path(s) for jpeglib.h and/or png.h
# PATH = path(s) for JPEG library and/or PNG library
# LIB = name(s) of JPEG library and/or PNG library

JPG_INC =
JPG_PATH =
JPG_LIB =

# -----
# build rules and dependencies
```

```
# no need to edit this section

include Makefile.package.settings
include Makefile.package

EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(JPG_INC)
$(PKG_SYSINC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(JPG_PATH) $(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(JPG_LIB) $(PKG_SYSLIB)

# Path to src files

vpath %.cpp ..
vpath %.h ..

# Link target

$(EXE): $(OBJ)
    $(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o $(EXE)
    $(SIZE) $(EXE)

# Library targets

lib: $(OBJ)
    $(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

shlib: $(OBJ)
    $(CC) $(CCFLAGS) $(SHFLAGS) $(SHLIBFLAGS) $(EXTRA_PATH) -o $(EXE) \
    $(OBJ) $(EXTRA_LIB) $(LIB)

# Compilation rules

%.o:%.cpp
    $(CC) $(CCFLAGS) $(SHFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
    $(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
sinclude $(DEPENDS)
```

将 Makefile.g++ 仔细修改好后保存，开始安装 lammps

```
cd /mnt/lmp/src
```

```
make clean-all
```

```
make g++
```

 (lammps 开始安装，如果安装成功，最后在 src 目录下可生成 lmp_g++ 的可执行文件)

```
mv lmp_g++ lmp
```

 (改名为 lmp 可以复制到桌面常用)

1.5 安装 lammps 后并行运算 example

a: 终端输入

```
sudo mv ~/lmp/src/lmp /usr/bin
```

```
cd /mnt/lmp/examples/shear
```

 (一定要进入需要计算文件的文件夹中)

```
cp ~/mpich2/bin/mpirun ~/lmp/e*/shear
```

```
cp ~/lmp/src/lmp ~/lmp/e*/shear
```

 (拷贝在同一个文件夹)

```
mpirun -np 2 ./lmp<in.shear
```

 (lammps 开始计算)

b: 终端输入 (方法二，比较简单，直接给 mpirun 和 lmp 的绝对路径，不需要拷贝了)

```
cd /mnt/lmp/e*/shear
```

```
/opt/mpich/bin/mpirun -np 4 /mnt/lmp/src/lmp<in.shear
```

另外，如果嫌每次需要打出 mpirun 绝对路径太麻烦，可以在 ~/.bashrc 的末尾添加以下部分，将其路径添加到 \$PATH 环境变量中。

```
cd
```

```
gedit .bashrc
```

```
# Source global definitions
```

```
if [ -f /etc/bashrc ]; then
```

```
    . /etc/bashrc
```

```
fi
```

```
export PATH=/usr/local/mpich2/bin:$PATH # 将新路径添加到现有的 $PATH 中
```

保存退出后，关闭终端，然后用 Ctrl-Alt-t 热键重新打开一个新的终端使 .bashrc 的设置生效，

可以用 `echo $PATH` 命令检查一下，/usr/local/mpich2/bin 这个路径应该会包含其中。

2 特殊 lammps 库安装包的编译

2.1 关于 lammps 库文件的说明

lammps 中 lib 目录下含有 **atc**, **awpmd**, colvars, **cuda**, **gpu**, linalg, meam, poems 和 reax 文件夹（红色字体的包因为无法排错而没有安装，因此也就不需要编译这几个库文件），为了尽可能安装 lammps 所有的包，每个都需要进去编译。

以 linalg 为例，进去后可以发现其中的 Makefile 只有 Makefile.gfortran 和 Makefile.mingw_cross 这两种，文件名的后缀就是这个 Makefile 适用的编译器。由于此前已经安装了 gfortran 编译器，因此需要使用命令 `make -f Makefile.gfortran` 进行编译。

这里需要说明：lib 下的各包，除了 awpmd、cuda 和 gpu 只有 Makefile.openmpi 外，其它所有包中要么有 Makefile.g++，要么有 Makefile.gfortran，因此在这里使用 g++ 和 gfortran 这两个编译器就足够了。下表中为各库文件家中各编译器版的 Makefile，第一个即为我所用的 Makefile。

| LIB | Makefile的后缀 |
|--------------|--|
| atc | g++, icc, serial, lammps |
| awpmd | openmpi, lammps |
| colvars | g++, femi, mingw32-cross, lammps |
| cuda | 无后缀, common, cudalib, defaults, lammps |
| gpu | femi, lens, lincoln, linux, linux_opencl, longhorn, mac, mac_opencl, serial, serial_opencl, lammps |
| linalg | gfortran, mingw_cross |
| meam | gfortran, g95, ifort, pgf90, tbird, lammps, lammps.gfortran, lammps.glory, lammps.ifort |
| poems | g++, icc, storm, lammps |
| reax | gfortran, g77, g95, ifort, pgf90, redsky, tbird, lammps, lammps.gfortran, lammps.ifort |

各包中均包含 README 文件，里面指出如果编译成功了，会生成静态库文件 lib*.a 和配置文件 Makefile.lammps，它俩为接下来的 lammps 编译所用。

atc 是需要 BLAS(Basic Linear Algebra Subroutines)和 LAPACK(Linear Algebra Routines)的，如果系统没有安装这两个东西，有两种解决办法：要么编译 lib 中的 linalg 并利用它做伪 BLAS 和 LAPACK，然后再供 atc 编译时调用。但在对 atc 使用 `make -f Makefile.g++` 编译时提示错误：mpi.h 没有那个文件或目录。要么 apt-get 安装了 liblapack-dev 和 libblas-dev

之后再编译 atc 即可。从这里可以看到 atc 是需要数学库的支持的，Intel 的 Math Kernel Library(简称 MKL)就是针对自家 CPU 优化的数学库，如果是 Intel 的 CPU 需要首选这个。

awpmd 只有 Makefile.openmpi 可供选择，为此我还用原始码安装了 openmpi-1.6.2，然后成功编译并得到 libawpmd.a 和 Makefile.lammps 文件，但在随后对 lammps 编译时，如果选择了 USER-AWPMD 包，编译时会报错：/usr/bin/ld: cannot find -lawpd。

reax 的问题比较怪，即可以成功编译，并得到 libreax.a 和 Makefile.lammps 这两个文件，但在对 lammps 编译时，如果选择了 REAX 包，编译时会报错：/usr/bin/ld: cannot find -lifcore -lsvml -lompstub -limf。这里指出可以在编译好 lib/reax 后修改其中的 Makefile.lammps 文件，将其中 reax_SYSLIB 后的内容从 -lifcore -lsvml -lompstub -limf 修改为 -lgfortran，这样可以成功编译得到 lmp 可执行文件，但运行它却提示“已杀死”。

2.2 关于 lammps 库文件安装包的查看

查看已经安装了那些包：

```
make package-status
```

需要特殊安装就：

```
make yes-meam
```

```
make yes-reax
```

```
make yes-poems    即 make yes-*
```

也可以 `make yes-all`，会因缺少文件出错所以不建议，也可都卸载 `make no-all`

(几个特殊的 package:meam, poems, reax, gpu, usercd-atc 需要特别安装，如下)

同时注意，因为你用的是 gfortran 编译器，所以里面有个 Makefile.lammps 的文件需要修改一下，poems 不用，因为它用的是 g++编译器。

2.3 lammps 中 reax 安装包的编译(串行)

前提已经装好 lammps，并改好名字为 lmp，因为并行会出现“已杀死”的错误，故只能串行。

1) 编译 STUBS

```
cd lmp/src/STUBS
```

```
make
```

(到 STUBS 中看是否已经生成了 lmpi.a 文件)

2) 编译需要安装的包

```
cd lmp/src
```

终端输入 `make package-status`，可以查看当前安装包的安装情况。

```
make yes-reax
```

 准备安装 reax

3) 生成安装包库文件

```
cd lmp/lib/reax
```

```
make -f Makefile.gfortran
```

 (查看 reax 文件下是否生成了 libreax.a 库文件)

4) 编译好库后，就是最关键的修改 `Makefile.serial` 文件了。

```
cd lmp/src/MAKE
```

```
gedit Makefile.serial
```

修改 (只修改黄色突出部分) 及注释 `Makefile.serial` 如下:

```
# serial = RedHat Linux box, g++4, no MPI, no FFTs

SHELL = /bin/sh

# -----
# compiler/linker settings    (编译器和链接 设置)
# specify flags and libraries needed for your compiler    (一般情况下不需要修改这一部分)

CC =      g++
CCFLAGS = -O
SHFLAGS = -fPIC
DEPFLAGS =    -M

LINK =      g++
LINKFLAGS = -O
LIB =
SIZE =      size

ARCHIVE = ar
ARFLAGS = -rc
SHLIBFLAGS = -shared

# -----
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use
# if you change any -D setting, do full re-compile after "make clean"

# LAMMPS ifdef settings, OPTIONAL
```

```
# see possible settings in doc/Section_start.html#2_2 (step 4)

LMP_INC = -DLAMMPS_GZIP

# MPI library, REQUIRED
# see discussion in doc/Section_start.html#2_2 (step 5)
# can point to dummy MPI library in src/STUBS as in Makefile.serial
# INC = path for mpi.h, MPI compiler settings    (mpi.h的路径, MPI编译器的设置)
# PATH = path for MPI library    (MPI库的路径)
# LIB = name of MPI library    (MPI库的名称)

MPI_INC =      -I./STUBS
MPI_PATH =     -L./STUBS
MPI_LIB =     -lmpi_stubs

# FFT library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 6)
# can be left blank to use provided KISS FFT library
# INC = -DFFT setting, e.g. -DFFT_FFTW, FFT compiler settings
# PATH = path for FFT library
# LIB = name of FFT library

FFT_INC =
FFT_PATH =
FFT_LIB =

# JPEG library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 7)
# only needed if -DLAMMPS_JPEG listed with LMP_INC
# INC = path for jpeglib.h
# PATH = path for JPEG library
# LIB = name of JPEG library

JPG_INC =
JPG_PATH =
JPG_LIB =

# additional systems libraries needed by LAMMPS package libraries    (lammps提供的额外系统库)
# these settings are IGNORED if the corresponding LAMMPS package    (如果编译这些库, 以下的这些设置就会被忽略)
# SYSLIB = names of libraries
reax_SYSLIB = -lgfortran
# SYSPATH = paths of libraries
```

```
reax_SYSPATH = -L/home/lmp/lib/reax (libreax.a库文件的目录)

# 以下区域不需要进行修改
# -----
# build rules and dependencies
# no need to edit this section

include Makefile.package.settings
include Makefile.package

EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(JPG_INC)
$(PKG_SYSINC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(JPG_PATH)
$(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(JPG_LIB) $(PKG_SYSLIB)

# Link target

$(EXE): $(OBJ)
    $(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o
$(EXE)
    $(SIZE) $(EXE)

# Library targets

lib: $(OBJ)
    $(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

shlib: $(OBJ)
    $(CC) $(CCFLAGS) $(SHFLAGS) $(SHLIBFLAGS) $(EXTRA_PATH) -o $(EXE) \
    $(OBJ) $(EXTRA_LIB) $(LIB)

# Compilation rules

%.o:%.cpp
    $(CC) $(CCFLAGS) $(SHFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
    $(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
sinclude $(DEPENDS)
```

5) 安装 Makefile.serial 文件

修改后保存，进入到 src 目录下：

```
cd lmp/src
```

```
make serial
```

如果成功，可在 src 目录下生成 lmp_serial 的可执行文件。

6) 算例计算

将生成的 lmp_serial 的可执行文件拷到你要计算的文件中，终端输入

```
./lmp_serial<in.***
```

 (***)为 in 文件的名称

2.4 lammmps 中 meam 和 poems 安装包的编译(并行)

前提已经安装好 lammmps，fftw2 和 mpich2。

1) 编译需要安装的包

```
cd lmp/src
```

终端输入 `make package-status`，可以查看当前安装包的安装情况。

```
make yes-meam
```

 准备安装 meam

```
make yes-poems
```

 准备安装 poems

2) 修改库文件中的 Makefile.lammmps 文件并编译

```
cd /mnt/lmp/lib/meam
```

```
gedit Makefile.lammmps
```

将其中的内容与下文核实：

```
meam_SYSINC =
```

```
meam_SYSLIB = -lfeore -lsvml -lompstub -limf 画线部分改为-lgfortran
```

```
meam_SYSPATH = -L/opt/intel/fcc/10.0.023/lib 删除画线部分
```

之后保存关闭，在终端输入：

```
make -f Makefile.gfortran
```

 (查看 meam 文件下是否生成了 libmeam.a 库文件)

同理，

```
cd /mnt/lmp/lib/poems
```

```
make -f Makefile.g++
```

 (查看 poems 文件下是否生成了 libpoems.a 库文件)

3) 编译 Makefile.g++文件

这里选用 Makefile.g++进行编译。先敲两行命令调出 Makefile。

```
cd ~ /lmp/src/MAKE
```

```
gedit Makefile.g++
```

```
# g++ = RedHat Linux box, g++4, MPICH2, FFTW

SHELL = /bin/sh

# -----
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC = g++
CCFLAGS = -g -O
SHFLAGS = -fPIC
DEPFLAGS = -M

LINK = g++
LINKFLAGS = -g -O
LIB =
SIZE = size

ARCHIVE = ar
ARFLAGS = -rc
SHLIBFLAGS = -shared

# -----
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use
# if you change any -D setting, do full re-compile after "make clean"

# LAMMPS ifdef settings, OPTIONAL
# see possible settings in doc/Section_start.html#2_2 (step 4)

LMP_INC = -DLAMMPS_GZIP

# MPI library, REQUIRED
# see discussion in doc/Section_start.html#2_2 (step 5)
# can point to dummy MPI library in src/STUBS as in Makefile.serial
# INC = path for mpi.h, MPI compiler settings
# PATH = path for MPI library
# LIB = name of MPI library
```

```
MPI_INC =      -DMPICH_SKIP_MPICH2 -I/usr/include/mpich2  # 即 mpi.h 的路径
MPI_PATH =      -L/usr/lib      # 即 libmpich.a 的路径
MPI_LIB =      -lmpich -lmpi -lpthread

# FFT library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 6)
# can be left blank to use provided KISS FFT library
# INC = -DFFT setting, e.g. -DFFT_FFTW, FFT compiler settings
# PATH = path for FFT library
# LIB = name of FFT library

FFT_INC =      -DFFT_FFTW2 -I/usr/local/fftw2/include  #即 fftw.h 的路径
FFT_PATH =      -L/usr/local/fftw2/lib  # 即 libfftw.a 的路径
FFT_LIB =      -lfftw  # 理解为将-l换成lib，后面添加.a后缀，就是libfftw.a这个库文件

# JPEG and/or PNG library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 7)
# only needed if -DLAMMPS_JPEG or -DLAMMPS_PNG listed with LMP_INC
# INC = path(s) for jpeglib.h and/or png.h
# PATH = path(s) for JPEG library and/or PNG library
# LIB = name(s) of JPEG library and/or PNG library

JPG_INC =
JPG_PATH =
JPG_LIB =

# additional systems libraries needed by LAMMPS package libraries （lammps提供的额外系统库）
# these settings are IGNORED if the corresponding LAMMPS package （如果编译这些库，以下的这些设置就会被忽略）
# SYSLIB = names of libraries
# SYSPATH = paths of libraries

meam_SYSLIB = -lmeam -lpthread -lgfortran （lpthread, lgfortran 是编译工具库）
poems_SYSLIB = -lpoems -lpthread -lgfortran

meam_SYSPATH = -L/home/lmp/lib/meam （libmeam.a库文件的目录）
poems_SYSPATH = -L/home/lmp/lib/poems （libpoems.a库文件的目录）

# -----
# build rules and dependencies
# no need to edit this section
```

```
include Makefile.package.settings
include Makefile.package

EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(JPG_INC)
$(PKG_SYSINC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(JPG_PATH) $(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(JPG_LIB) $(PKG_SYSLIB)

# Path to src files

vpath %.cpp ..
vpath %.h ..

# Link target

$(EXE): $(OBJ)
    $(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o $(EXE)
    $(SIZE) $(EXE)

# Library targets

lib: $(OBJ)
    $(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

shlib: $(OBJ)
    $(CC) $(CCFLAGS) $(SHFLAGS) $(SHLIBFLAGS) $(EXTRA_PATH) -o $(EXE) \
    $(OBJ) $(EXTRA_LIB) $(LIB)

# Compilation rules

%.o:%.cpp
    $(CC) $(CCFLAGS) $(SHFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
    $(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
sinclude $(DEPENDS)
```

将 Makefile.g++仔细修改好后保存，开始安装 lammps


```
cd /mnt/lmp/src
```

```
make clean-all
```

```
make g++
```

 (lammers 开始安装，如果安装成功，最后在 src 目录下可生成 lmp_g++的可执行文件)

```
mv lmp_g++ lmp
```

 (改名为 lmp 可以复制到桌面常用)

3 VMware 虚拟机给 ubuntu 安装 VMware tools

- 1) 选择虚拟机菜单栏--安装 VMware tools
- 2) 然后在 Ubuntu 系统中弹出的 VMware tools 窗口中找到 VMwaretools-9.6.0-1294478.tar.gz
- 3) 将 VMwaretools-9.6.0-1294478.tar.gz 存放到自己的文件夹下，提取
进入到文件下，终端输入：

```
sudo tar xvzf VMwaretools-9.6.0-1294478.tar.gz
```
- 4)

```
cd vmware-tools-distrib
```
- 5)

```
sudo ./vmware-install.pl
```
- 6) 之后根据提示依次点击即可。
- 7) 最后需要重启一下才能生效。

4 关于 ubuntu 更新源地址问题

我们都知道在 ubuntu 中有一个强大的 apt-get install 功能，可以直接联网傻瓜式的下载很多必要的软件，如 build-essential, g++, gfortran 等，但是由于高校网络的问题，众多时候我们在使用

```
sudo apt-get install build-essential
```

 的时候会出现无法定位安装的问题，网上的一直说法是联网后

```
sudo apt-get update
```

 来进行更新，但是发现还是会遇到问题。这主要是因为各高校网络与 ubuntu 端口的不匹配问题导致的。下文将针对北科大校园网进行说明与源更新。

1、首先备份 Ubuntu 13.04 源列表

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

 (备份下当前的源列表，有备无患嘛)

2、修改更新源

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

 (打开 Ubuntu 13.04 源列表文件)

3、将下面的代码粘贴进去

```
deb http://ftp.ustb.edu.cn/pub/ubuntu/ precise main multiverse restricted universe
deb http://ftp.ustb.edu.cn/pub/ubuntu/ precise-backports main multiverse restricted universe
```

```
deb http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-proposed main multiverse restricted universe
deb http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-security main multiverse restricted universe
deb http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-updates main multiverse restricted universe
deb-src http:// ftp.ustb.edu.cn/pub/ubuntu/ precise main multiverse restricted universe
deb-src http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-backports main multiverse restricted universe
deb-src http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-proposed main multiverse restricted universe
deb-src http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-security main multiverse restricted universe
deb-src http:// ftp.ustb.edu.cn/pub/ubuntu/ precise-updates main multiverse restricted universe
restricted universe
```

这里用蓝色突出部分需要特别注意，必须与你安装的 ubuntu 版本一致，并且在 ftp.ustb.edu.cn/pub/ubuntu/ 中能找到。将 sources.list 修改结束后，在终端中输入：

```
sudo apt-get update
```

进行更新，更新后以上软件应该都能安装了。

5 关于 ubuntu 版本的说明

Ubuntu 每 6 个月发布一个新版本，而每个版本都有代号和版本号，其中有 LTS 是长期支持版。版本号基于发布日期，例如第一个版本，4.10，代表是在 2004 年 10 月发行的。

Ubuntu 历史版本一览表

























| 版本号 | 代号 | 发布时间 |
|-----------------------|------------------|--------------|
| 15.04 | Vivid Vervet | 即将发布（2015/4） |
| 14.10 | Utopic Unicorn | 2014/10/23 |
| 14.04 LTS | Trusty Tahr | 2014/04/18 |
| 13.10 | Saucy Salamander | 2013/10/17 |
| 13.04 | Raring Ringtail | 2013/04/25 |
| 12.10 | Quantal Quetzal | 2012/10/18 |
| 12.04 LTS | Precise Pangolin | 2012/04/26 |
| 11.10 | Oneiric Ocelot | 2011/10/13 |
| 11.04（Unity 成为默认桌面环境） | Natty Narwhal | 2011/04/28 |
| 10.10 | Maverick Meerkat | 2010/10/10 |

北京科技大学版，只供学习交流

| | | |
|---------------|------------------|------------|
| 10.04 LTS | Lucid Lynx | 2010/04/29 |
| 9.10 | Karmic Koala | 2009/10/29 |
| 9.04 | Jaunty Jackalope | 2009/04/23 |
| 8.10 | Intrepid Ibex | 2008/10/30 |
| 8.04 LTS | Hardy Heron | 2008/04/24 |
| 7.10 | Gutsy Gibbon | 2007/10/18 |
| 7.04 | Feisty Fawn | 2007/04/19 |
| 6.10 | Edgy Eft | 2006/10/26 |
| 6.06 LTS | Dapper Drake | 2006/06/01 |
| 5.10 | Breezy Badger | 2005/10/13 |
| 5.04 | Hoary Hedgehog | 2005/04/08 |
| 4.10 (初始发布版本) | Warty Warthog | 2004/10/20 |

大家下载安装的过程中最好选用 LTS 长期支持的，并且在各自的校园网 FTP 等中查找，非常方便。

如北科大 <ftp.ustb.edu.cn/pub/ubuntu/> 中就包括如下版本：

| 名称 | 大小 | 修改日期 |
|---|--------|---------------------|
|  [上级目录] | | |
|  10.04 | 0 B | 10/3/20 上午12:00:00 |
|  10.04.4 | 0 B | 12/2/17 上午12:00:00 |
|  12.04 | 0 B | 12/3/2 上午12:00:00 |
|  12.04.5 | 0 B | 14/8/8 上午12:00:00 |
|  14.04 | 0 B | 14/3/28 上午12:00:00 |
|  14.04.3 | 0 B | 15/8/7 上午2:44:00 |
|  14.10 | 0 B | 14/9/26 上午12:00:00 |
|  15.04 | 0 B | 15/3/27 上午6:53:00 |
|  FOOTER.html | 22 B | 06/2/2 上午12:00:00 |
|  HEADER.html | 2.2 kB | 15/8/7 上午3:33:00 |
|  cdicons/ | | 12/9/21 上午12:00:00 |
|  dapper/ | | 11/7/18 上午12:00:00 |
|  edubuntu/ | | 13/5/30 上午12:00:00 |
|  favicon.ico | 1.1 kB | 11/6/16 上午12:00:00 |
|  include/ | | 14/8/21 上午12:00:00 |
|  jigit/ | | 13/5/30 上午12:00:00 |
|  lucid/ | | 13/9/27 上午12:00:00 |
|  precise/ | | 15/6/4 上午5:11:00 |
|  releases | 0 B | 10/3/18 上午12:00:00 |
|  robots.txt | 49 B | 09/10/29 上午12:00:00 |
|  trusty/ | | 15/8/7 上午3:45:00 |
|  utopic/ | | 14/10/23 上午12:00:00 |
|  vivid/ | | 15/8/18 上午5:41:00 |