针对 VASP 的材料计算教程
为材料计算而生

Jiaqi Z.[1]

2024 年 8 月 27 日

目录

前言

“为材料计算而生”，是抱着多大的觉悟说出这种话啊。这只是一本书，有办法背负其他人的人生吗？

真是满脑子只想着自己呢……[1]

俗话说得好，好记性不如烂笔头，这句话在任何时候都显得格外贴切。尤其是在科研领域（特别是材料科学这样规范和流程化的学科），记录的重要性更是不言而喻。随着计算任务的不断增加，我们掌握的计算方法和参数也日益繁多。将这些知识、操作和方法记录下来，不仅能够帮助我们避免遗忘，还能在遇到类似问题时快速查找，而不必在海量的网络搜索结果中苦苦寻觅。

正是基于这样的考虑，我们结合自己科研团队在材料计算方面遇到的一些实际问题，整理编写了这本书。我们编写这本书的目的有两个：一是为了方便自己，在面对类似问题时能够迅速回忆起解决方案；二是为了通过集体的智慧，汇聚大家的方法和思路，以便在遇到新问题时能够迅速找到答案。

这本书的诞生，既是为了服务于材料计算，也是因材料计算而生。既然如此，我们为何不称它为“为材料计算而生”呢？

我们衷心希望这本书能够惠及更多的人，无论是我们团队的新成员，还是其他团队的老师或同学，都能从中获得帮助。

同时，我们也清楚地认识到自己的能力和知识是有限的，书中的内容难免会有疏漏或错误。我们诚挚地希望读者在使用过程中能够提出宝贵的意见和建议，或者分享你们的经验，共同促进我们的成长和进步。

最后，再次感谢您阅读并使用这本书。

Jiaqi Z.

2024 年 8 月青岛

[1] 以上内容改编自动漫《BanG Dream! It’s MyGO!!!!!》中丰川祥子的台词

如何联系作者

可通过以下任意一种方式联系:

- GitHub 的 Issue, 这是最直接的方式[2];
- email, 请发送邮件至 zhangjq00@sdust.edu.cn 或 zhangjq_sd@163.com

如何使用这本书

在使用时，请按照如下方法：

1. 根据研究问题，寻找合适的章节；如果没有，可以在 GitHub 上提交 Issue 或者贡献 Pull Request;
2. 在每一节开始，会介绍本节的内容和知识点，查看是否与你的研究问题符合；如果不符合，返回第 1 步重新查找新的章节;
3. 阅读这一节内容，并试着针对自己的问题进行操作（或简单检查自己操作是否正确）。如果报错或出现异常结果，进行第 4 步；如果成功，进行第 5 步;
4. 在该节后面的“错误处理”部分，会介绍如何处理报错或异常结果，并给出解决方案。请查找是否有你需要的解决方案，并尝试解决。如果已经解决，进入第 5 步；否则重新查找新的解决方案；若所有解决方案都无法解决，请提交 Issue 或者贡献 Pull Request;
5. 放下教程，继续你的研究；或者阅读这一章其他内容，了解其他相关内容。

上述步骤可能（也一定）会重复许多次

关于本笔记的版权使用说明

- 本书可**免费用于学习, 科研等非商业活动**;

[2]GitHub 仓库地址：https://github.com/JackyZhang00/Computational-Materials-Tutorial

- 可以以**非商业目的进行传播**, 但在传播过程中**必须保证内容的完整性** (截止到最新发布时, 包括但不限于仓库内 Latex 源码, pdf 文件等. 下同), 需**保证作者信息完整**, 不得进行修改;
- 本书**不可用于任何商业用途** (如确有需要, 需联系作者);
- 除在 GitHub 仓库以 pull request 形式进行编辑修改外, **不允许进行修改并公开传播私自修改版本** (以 GitHub 仓库版本为标准版本);
- 本书著作权归作者 (Jiaqi Z.) 所有, 其他进行创作的人员也可获得著作权, 其他著作权所有者不得违反上述版权说明;
- **如因违反上述说明传播而造成不良影响, 与作者和其他创作者无关**, 特此声明;
- 以上说明解释权归 Jiaqi Z. 所有, 且如有后续更新, 以 GitHub 仓库最新版说明为准.

创作者名单

感谢以下人员参与贡献了内容：
Jiaqi Z.

为创作者而作的教程

0.1 关于如何使用 LaTeX 编写模板

本节作者：Jiaqi Z.

在本节，你将要学到：

- 一些基本的 LaTeX 语法
- 如何输入公式
- 如何插入代码
- 如何插入图片
- 如何插入创建索引

对于创作者而言，这一部分可以帮助你快速了解 LaTeX 的基本语法，帮助你按照规范编写正确的文件。同时对于普通读者而言，这一部分也是你了解本书内容样式的一部分。

因此，任何人都应当阅读这一部分[3]。

0.1.1 文章结构

请在编写正文内容时，以“节”（section）为单位创建 tex 文档，同时为方便引用，请在每个小节的后面按照`\label{sec:节标题}`的格式创建标签。

若需要添加小节，使用`\subsection{`小节标题`}`命令，同时类似于上方关于节标题标签的创建规范，以`\label{subsec:`节标题`-`小节标题`}`的格式创建标签，方便他人引用。

[3]对于创作者而言，还应当试着从 GitHub 上寻找源码阅读

对于更小一级的小节（`\subsubsection{}`），对标签不作规范。事实上，我们不建议在引用时涉及到这一层级。通常涉及到小节即可。

注意：若你需要修改某一节（或小节）的标题，编译后需要确认是否与他人的标签产生冲突（这通常出现在他人提前按照原有格式引用后发生了修改，从而导致无法指向正确标签）。因此，你需要检查编译后文件是正确的，至少要求通过编译，在必要时需要修改他人代码当中的引用标签与新标签一致。

对于正文内容，请使用正常的 LaTeX 语法。例如，当你希望对某一段文字进行强调时，请使用`\emph{}`语句。例如，*这是一句强调的话*在代码中体现为`\emph{这是一句强调的话}`。你并不需要关注具体的强调格式–LaTeX 会按照统一的格式进行编排。

当你希望分段时，使用空行即可。

对于条目,请在必要的时候使用`itemize`环境（没有顺序列表）或`enumerate`环境（有顺序列表）。在环境内使用`\item`进行编号。环境之间可以嵌套。例如：

- 列表 1
- 列表 2
    1. 列表 2-1
    2. 列表 2-2
- 列表 3

在代码中体现为：

```
\begin{itemize}
    \item 列表1
    \item 列表2
    \begin{enumerate}
        \item 列表2-1
        \item 列表2-2
    \end{enumerate}
    \item 列表3
\end{itemize}
```

大多数用法与基本 LaTeX 一样，少有的需要特别注意的是波浪线符号 ~，如果使用习惯的打法\~则会显示较小，因此，在输入时请使用修改后的波浪线符号\texttilde，或者使用更简洁的形式\ttilde。例如，在使用\~/bin显示的结果为~/bin 而使用\ttilde/bin显示为 ~/bin

0.1.2 一些特殊的格式

在有些时候，会希望在书后添加一些辅助性的文字说明，可以使用脚注。脚注应当使用命令\footnote{}创建，例如，

```
这里有一段文字\footnote{这里是说明性的脚注}。
```

编译后效果为

这里有一段文字[4]。

此外，为了激发读者思考，在编写时可以添加一些简单的思考题贯穿于正文中。这些思考题不应当很难（对于较难的题目，可以放置在一节后），应当做到读者经过简单的思考（约 1 分钟以内）即可得到正确答案。此时在编写时应当将答案放置在题目后面，考虑到避免读者直接看到答案，所有答案都按照特定的格式排版。在编写时应当使用\answer{}命令，例如：

```
这里是思考题。


\answer{倒着看便是答案}
```

排版的效果是

这里是思考题。

【答案】：倒着看便是答案

0.1.3 一些特殊的环境

为统一教程格式，当你希望添加一段让读者注意的文字时，请使用环境attention例如，下面的语句

```
\begin{attention}
    当你写注意语句时，不需要在前面加任何符号。
\end{attention}
```

[4]这里是说明性的脚注

在编译后的结果为：

注意：当你写注意语句时，不需要在前面加任何符号。

类似地，对于一些补充性质的内容，可以使用extend环境，例如：

```
\begin{extend}
    这是一段补充的内容，同样不需要在前面加任何符号。
\end{extend}
```

编译后的结果为：

补充：这是一段补充的内容，同样不需要在前面加任何符号。

0.1.4 数学公式

当你希望添加数学公式时，请使用equation环境。同时，在使用\label语句进行标签注明时，请如同代码所示那样，添加“节标题”避免冲突且方便引用。

```
\begin{equation}
    \label{eqn:数学公式-1}
    a^2+b^2=c^2
\end{equation}
```

$$a^2+b^2=c^2 \tag{1}$$

在引用公式时，请使用如\ref{eqn:数学公式-1}方式进行交叉引用。**请勿直接在正文内写编号**以免出现引用错误。

注意：在引用公式时，所引用的公式尽量保持在本节内。同时，为避免他人引用，请在编写完成后尽量不要修改相关标签。

为避免公式删除导致的错误，如确实需要引用其他章节的公式，一个较合理的做法是将其他章节的公式在使用时拷贝至当前章节，同时另起标签名。之后的引用限制在当前章节内。

如需要添加多行公式，请使用gather环境或align环境。例如，

$$a^2+b^2=c^2 \tag{2}$$

$$\int_a^b f(x)\,\mathrm{d}x = F(b)-F(a) \tag{3}$$

图 1: 图片标题

0.1.5 图片

当添加图片前，请首先在相关文件夹内创建名为fig的文件夹，在插入图片时如正常 LaTeX 代码插入即可。即使用下面的代码方式

```
\begin{figure}
    \centering
    \includegraphics[width=1\linewidth]{图片路径}
    \caption{图片标题}
    \label{fig:图片-标签}
\end{figure}
```

注意：在引用图片时，务必使用\ref{fig:图片-标签}进行引用。例如，图 *1*。不要使用“如上图”、“如下图”的表述形式，以免图片位置发生移动造成指代不明。

0.1.6 代码

代码使用lstlisting环境。

```
SYSTEM = 0
ISMEAR = 0
SIGMA = 0.05
NSW = 1
```

如果希望在代码左侧添加行号以示说明,请在引用环境的右侧添加numbers=left设置，即采用下面的代码：

```
\begin{lstlisting}[numbers=left]
    ..
```

得到效果如下所示

```
1 SYSTEM = 0
2 ISMEAR = 0
3 SIGMA = 0.05
4 NSW = 1
```

注意：若你希望在代码块中添加代码名称（例如文件名），可以使用caption选项进行说明，例如，下面的代码：

```
\begin{lstlisting}[caption=简单的INCAR文件]
    ..
```

实际编译后结果为

Listing 1: 简单的 INCAR 文件

```
SYSTEM = 0
ISMEAR = 0
SIGMA = 0.05
NSW = 1
```

0.1.7 添加索引

建议在编写过程中，为文中的程序关键字创建索引。为方便使用，可以使用命令\keyword{}。例如，在 VASP 讲解 SIGMA 函数时，可以使用下面的代码

```
\keyword{SIGMA}
```

从而在表述关键字的同时创建索引。同时，也可以使用\keywordin{}{}的形式创建带有所属关系的索引，例如

```
\keywordin{INCAR}{SIGMA}
```

便是创建了属于 INCAR 条目下的 SIGMA 索引。

注意：在编写时，鼓励使用索引方便他人根据关键字直接检索。在创建索引时，请提前确认现有的索引表是否有现成的索引。例如，\keyword{SIGMA}和\keywordin{INCAR}{SIGMA}会得到不同的结果。

同时，在编写过程中已经将输出和索引集成在一个命令中，你不需要特地再编写一个输出的命令如 *SIGMA*\keyword{SIGMA}，只需要使用 \keyword{SIGMA} 便可在输出 *SIGMA* 的同时创建索引。

对于一些特殊的内容，可能不希望给出索引（通常是命令或者文件名等），它们既不属于关键字，也不属于简单的英文单词。为了将它们区分，使用\code{}命令进行编写。例如，\code{cd \$OLDPWD}的输出结果为 cd $OLDPWD

注意：在一些特殊的情况下（例如上面的例子），可能会包含 LaTeX 本身的特殊符号（如 $ 本身作为公式符号）。在输入时，应当使用反斜杠 \ 作为转义字符。

Part I

Linux 基础

Chapter 1

Linux 命令行操作

Contents

1.1 认识 Linux 目录

本节作者：Jiaqi Z.

在本节，你将要学到：

- Linux 命令格式
- 如何在 Linux 当中表示目录
- 绝对路径和相对路径
- 如何快速表示当前目录和上一级目录

1.1.1 命令格式

与 Windows 使用可视化界面不同，Linux 大多时候使用命令行（shell）进行操作。因此，在 Linux 的学习过程中，一个最重要的任务，就是掌握一些常见的 Linux 命令。对于大多数科研课题组而言，Linux 系统都是在远程云端服务器上，因此在本地往往只需要一个终端程序即可连接到服务器。一些常见的终端软件包括 Xshell、MobaXterm、甚至 VS Code[1]等。

注意：如果你熟悉其他操作系统，可能听闻过类似于 Windows Server，或者 Linux 的 Ubuntu 这样的操作系统。明明也可以使用可视化界面，为什么在科研过程中从来不会用到它们呢？（更严谨地说，在远程服务器上）。实际上，当使用可视化界面进行远程连接时，所产生的网络资源消耗是巨大

[1]对于 VS Code 而言，可能需要扩展插件（例如 Remote-SSH）的支持

的，通常需要更大的带宽，而使用命令行就可以提高数据传输效率。此外，更重要的一点是，使用命令行可以很容易实现批量处理，这在后续的章节会介绍到。

在 Linux 当中，输入命令通常采用的格式是命令 [-选项] [参数]，其中中括号表示这个部分是**可选的**，即可以没有的。例如，当我们希望列出当前目录下所有文件时，可以使用 ls直接输出，也可以使用 ls -l 以列表格式输出。

注意：在后面可能会看到选项有多个的情况，此时为了简化，可以将选项合并在一起。例如，*ls -l -a* 可以简化为 *ls -la*。

命令与选项、参数之间是以空格进行分割，且这个空格不能省略。

1.1.2 目录表示方法

在 Linux 当中，所有目录都是以根目录/为起点，任何目录都是根目录的子目录。根目录下存在一些固定的目录（这些目录通常有特定的含义），例如，在根目录下有一个叫做 bin 的目录（通常写作/bin），它存放的都是**二进制文件**，也就是系统可以执行的程序文件。

注意：在 *Linux* 当中，任何命令实际上都是可执行程序。你可以在*/bin* 目录下看到后面所学的所有 *Linux* 终端命令。

另一个比较重要的位置是家目录/home，它存放的是用户个人文件。在这一目录下，你可以看到系统所注册的所有用户名。但是，这些文件夹大多数是无法查看的[2]。对于用户自己的家目录，通常也可以表示为 ~。通常来说，当你使用终端等连接登录时，默认的所在目录就是家目录 ~

1.1.3 绝对路径和相对路径

任何目录在操作时都具有两种表示方式，**绝对路径**和**相对路径**。正如1.1.2所介绍的那样，任何目录都是从根目录开始的。因此在描述一个目录时，可以从根目录（即/）开始。例如，若你在你的家目录下有一个叫做 vasp 的目录，那么它的绝对路径就是/home/< 你的用户名 >/vasp。

随着层级逐渐增多，这种表示方法也会越来越复杂，因此，在表示一个目录时，默认也可以从当前所在目录开始算起（即**相对路径**）。例如，若你刚

[2]这涉及到 Linux 操作权限的问题，通常来说，权限分为三组，即所有者权限、所属组权限和其他用户权限。对于/home 目录下而言，所有目录都是对所有者（即这个用户本身）提供全部权限，而其他人无法访问、修改。

刚进入终端，此时所在目录就是 ~ 目录，即/home/< 你的用户名 > 下，此时若想表示 vasp，则只需要使用 vasp 即可。

注意：在这种情况下，你可以将目录 vasp 理解为 < 当前所在目录 >/vasp，即等价于/home/< 你的用户名 >/vasp。

千万不要写成/vasp，它表示根目录下的 vasp 目录。如果你希望特别强调当前目录，可以使用符号.（一个点）表示“当前目录”，即可以写成./vasp

然而，在这种情况下，回到**当前目录的上一级目录**是麻烦的，即在目前所学范围内，只能使用绝对路径。好在 Linux 提供了一个命令：..（两个点）表示**上一级目录**。因此，如果你当前处在目录/home/< 你的用户名 >/vasp 当中，则.. 表示/home/< 你的用户名 >

同理，../.. 表示父目录的父目录，在上面的例子中即为/home 目录。

注意：在终端当中，..（两个点）表示父目录（即上一级目录），而一个点. 表示当前目录。

这些符号（指令）在后续关于目录操作中都是可以使用的。

看到这里，可以思考下面的问题：如果在你的家目录下有两个目录 python 和 vasp，此时你在/home/< 你的用户名 >/vasp 目录下，如何可以快速表示 python 目录呢（不能使用绝对路径）?

【答案】：../python 即可表示/home/< 你的用户名 >/python 目录

1.2 目录操作

本节作者：Jiaqi Z.

在本节，你将要学到：

- 如何显示当前目录下所有文件
- 如何创建目录
- 如何切换至其他目录

1.2.1 显示目录文件 ls

在这一节以及下一节，我们将讨论如何对目录和文件做基本的操作。无论是哪一种，一个最基本的前提是**知道当前目录有哪些文件和目录**，从而才能进行后续操作（例如编辑、删除、移动、进入目录等）

在 Linux 当中，列出一个目录下所有文件使用的是 `ls` 命令。在没有任何参数与选项的前提下，它输出的结果就是**当前所在目录下的所有文件和目录**。以 1.1.3一节最后的例子为例，家目录下有 `vasp` 和 `python` 两个目录，当在家目录下执行 `ls` 命令时，结果如下：

```
$ ls
vasp python
```

同时，`ls` 支持在后面添加一个参数表示要输出的目录。例如，在这一例子下，若在家目录当中执行命令 `ls vasp`，将会输出 `vasp` 目录下的所有文件和目录。利用`..` 表示上一级目录的用法，若当前处在 `~/vasp` 目录下，使用 `ls ..` 便可得到上一级目录（即家目录）下的所有文件和目录。

ls -l

下面介绍两个常见的 `ls` 选项，首先是`-l` 选项，它表示**以列表形式输出结果**。例如，还是上面的例子，使用这一命令的结果为：

```
$ ls -l
total 0
drwxrwxr-x 2 zjq zjq 6 Aug 12 16:35 python
drwxrwxr-x 2 zjq zjq 6 Aug 12 16:35 vasp
```

补充：每一个文件的输出结果可以分为 *9* 个部分，分别是：权限、文件硬链接数或目录子目录数、拥有者用户名、拥有者所在组、文件大小、文件修改月份、日期、时间、文件名。

关于权限，可以将其分成四部分：第一部分（一个字符）表示文件类型（这里的 *d* 表示目录），第二部分（三个字符）表示拥有者权限（*rwx* 表示可

读可写可执行），第三部分（三个字符）表示组用户权限，第四部分（三个字符）表示其他用户权限（*r-x*）表示可读，可执行但不可编辑。

对于文件硬链接数和目标子目录数，对于初始创建的文件而言，通常为 *1*，而对于目录而言，默认为 *2*（因为有两个子目录 . 和 ..）

有时，也可以使用 `ll`代替指令 `ls -l`，其二者是完全等价的。

`ls -a`

`-a` 选项表示**列出所有文件，包括隐藏文件**。例如，在 `~/vasp` 目录下，使用 `ls -a` 命令，结果为：

```
$ ls -a
.  ..extend
```

补充：正如前面所介绍的那样，任何一个空目录都会默认有两个隐藏目录–自身和它的上一级目录。而这也解释了 *1.1.3*一节所介绍的 . 和 .. 的本质，它们实际上就是任何当前目录下的两个子目录。

注意：前面所介绍的*-l* 选项和*-a* 选项是可以合并使用的，此时可以将两个选项之间以空格分割，如 *ls -l -a*，或者将两个选项写在一起 *ls -la*

当选项写在一起时，选项的排列顺序不重要。

与最开始介绍 *ls* 后面加参数表示目录一样，带有选项的 *ls* 同样可以在后面添加参数，例如，*ls -a vasp* 表示列出当前目录下的 *vasp* 子目录下的所有文件和目录（包括隐藏文件）

1.2.2 关于隐藏文件

补充：隐藏文件是指在文件名前面加上 . 的，例如*.bashrc*。

隐藏文件在 *Linux* 当中的常见用途有：

- 配置文件
- 临时文件
- 缓存文件
- 等

总而言之，隐藏文件是为了防止误操作而存在的。（这可能与一些人认为的“隐藏文件是避免别人看到”不同）事实上，哪怕在 *Windows* 操作系统中，隐藏文件也是存在且方便查看的[3]

1.2.3 创建目录 mkdir

如果所有操作都在家目录下进行，那文件管理就太复杂了。试想一下，在科研里面算了好几年的结果，全部“一股脑”堆在一起，既难找，也容易忘记当时是做了什么。因此，一个好的目录管理至关重要。而前提，就是知道如何创建目录。

在 Linux 当中，创建目录的方法是使用 `mkdir` 命令。与前面介绍的 `ls`，以及后面要介绍的 `cd` 不同的是，`mkdir` 必须带有一个参数，表示**创建的目录路径**。对于刚开始接触的初学者，一个最简单的命令格式是：`mkdir < 目录名 >`，其中表示在当前目录下创建一个名为 `< 目录名 >` 的目录。例如，希望在当前目录下创建一个名为 `ML` 的目录，则可以使用命令 `mkdir ML`。

正如前面所介绍的路径，`mkdir` 后面的路径也可以是绝对路径或相对路径。无论是哪种形式，其含义是一样的，即**在你所描述的路径下创建目录**。利用这种方法，你可以在更远的层级关系下创建目录。例如，在 `~/vasp/lattice/Fe` 目录下创建 `~/python/ML/plot` 目录。

注意：你所写的路径名，应当是你所要创建的目录。这句话似乎有点绕，举个例子，如果你希望在 */home/zjq/vasp* 下创建一个名为 *lattice* 的目录，则你需要运行的命令是 *mkdir /home/zjq/vasp/lattice*。注意到，后面的路径实际上就是你要创建的目录。

1.2.4 切换目录 cd

在 Linux 当中，切换目录使用的命令是 `cd`，通常来说，后面需要配合一个参数，表示**要切换到哪里**。例如，使用命令 `cd /home` 则是将当前目录切换到`/home` 目录下。配合以`..`，可以使用 `cd ..` 切换到上一级目录。

思考：如果使用 `cd .`，会得到什么结果？

【答案】：这个命令的含义是**切换到当前目录**，最终效果就是什么也不发生

[3]在 Windows 操作系统中，可以通过右键-属性-隐藏的方式将文件或文件夹设置为隐藏；相对地，对 Windows10 操作系统而言，可以通过文件夹菜单栏的“查看”-“隐藏的项目”找到那些隐藏文件。只不过在 Linux 当中，隐藏文件使用前面加点 . 的方式设置，但无论如何，隐藏文件永远不是不让别人看见的方法，如果想达成这一目的，正确的方法是设置权限。

特殊的，对于家目录而言，除了可以使用 `cd ~` 外，Linux 也支持直接使用 `cd`，不添加任何参数实现这一功能，即二者是等价的。

1.2.5 错误处理

-bash: cd: < 目录名 >: Not a directory

`cd` 后面的参数必须是目录，不能是文件。如果参数是文件，则会报该错误。

如果不知道哪个是目录，哪个是文件，可以使用 `ls -l` 查看第一个字符（文件类型），如果第一个字符是 `d`，则表示目录，如果是`-`，则表示文件[4]。例如，

```
$ ls -l
total 4
-rw-rw-r-- 1 zjq zjq 4 Aug 12 17:12 INCAR
drwxrwxr-x 2 zjq zjq 6 Aug 12 16:35 python
drwxrwxr-x 2 zjq zjq 6 Aug 12 16:35 vasp
```

表示 `INCAR` 是文件,而 `vasp` 和 `python` 是目录。如果执行了 `cd INCAR`，则会报错。

-bash: cd: < 目录名 >: Permission denied

这表明你尝试进入一个你没有权限的目录。例如，在`/home` 目录下，有 `ljk` 和 `zjq` 两个目录，分别表示两个用户。如果执行 `ls -l` 则会发现：

```
$ ls -l
total 32
drwx------ 13 ljk  ljk   4096 Aug 5 17:34 ljk
drwx------ 75 zjq  zjq   4096 Aug 12 17:12 zjq
```

[4]在一些比较新的终端程序中，可能会将文件和目录以不同颜色区分。例如，在 MobaXterm 当中，默认情况下文件是白色，目录是蓝色。当然，这些颜色设置都是可以通过 Settings-Terminal-Default color settings 设置颜色主题，这里所说的这一例子为“Dark background / Light text”主题

很显然，每个目录只有目录拥有者自己可以访问。例如，作为用户 `zjq`，当尝试执行 `cd ljk` 时，则会报错。

补充：这种情况有一个特例：*root* 用户。对于 *root* 用户而言，可以进入任何目录。但通常来说，*root* 用户是由服务器管理者所持有的，作为一般用户而言，不需要也不应该进入没有权限的目录，或者执行没有权限的操作[5]。

mkdir: cannot create directory < 目录名 >: No such file or directory

虽然我们说可以用绝对路径或相对路径在更远的层级关系下创建目录。但这一操作的前提是，这个目录的上一级目录需要存在。例如，当你执行 `mkdir vasp/lattice/Fe` 时，首先需要确保目录 `vasp` 和 `vasp/lattice` 存在，才会创建 `vasp/lattice/Fe`。如果你要创建的目录其上一级目录不存在，则会报错。

一个很自然的解决方法是：一层一层创建。这种方法虽然麻烦，但可以确保目录是清晰的[6]。

mkdir: missing operand

很显然，你在使用 `mkdir` 时没有给任何参数。正如 1.2.3所说的那样，在调用 `mkdir` 时必须提供一个参数表示要创建的目录路径。

1.3 文件操作

本节作者：Jiaqi Z.

在本节，你将要学到：

- 如何移动文件（目录），如何给文件（目录）重命名
- 如何删除文件（目录）

[5] “删库跑路”这件事对于一般用户来说是不可能的

[6] 如果你确实想要一个快捷的方法，可以使用选项`-p`。这一选项可以在遇到没有的目录时自动为你创建。例如，上面的例子也可以直接使用 `mkdir -p vasp/lattice/Fe`，但这一操作需要保证输入内容是正确的。一旦有内容输错，则极有可能造成目录结构混乱。

- 如何复制文件（目录）

这一节，我们专注于文件的相关操作。类似于 Windows 的基本操作，Linux 的文件操作也无外乎就是**移动、删除、复制**。同时，这一节的许多命令对于文件和目录都是适用的，但可能会有一个注意事项，这往往会出错。

1.3.1 移动文件 mv

在 Linux 当中，移动文件使用的命令是 mv。其基本用法是 mv < 源文件路径 > < 目标文件路径 >。例如，我们在 vasp 目录下，希望将里面的 OUTCAR 移动至上一级目录，可以使用 mv OUTCAR ..。类似地，对于更复杂的文件移动，只不过在描述路径时稍微复杂一点，其他的步骤没有什么不同。

如果你足够敏感，也许会发现一点问题：**为什么前面的命令，OUTCAR 是文件，而.. 是目录**？两个难道不应该统一吗？

对于这个问题，可以分两个部分讨论：如果前面是文件，后面也是文件，例如 mv OUTCAR ../OUTCAR，这个命令与前面的命令效果是完全等价的。但是，有趣的地方在于，如果你试着执行 mv OUTCAR ../INCAR 的话，你会发现，Linux 将 OUTCAR 移动到.. 的同时，还将其改名为 INCAR。

进一步想一下，如果我们现在直接写成 mv OUTCAR INCAR 的话，可以将其看作把当前目录下的 OUTCAR 移动至当前目录，同时改名为 INCAR，总的效果就是，文件被重命名为 INCAR。

注意：正如你所见到的那样，*Linux* 没有单独的重命名文件命令，而是通过 *mv* 命令来完成。

进一步，如果前后两个参数都是目录会发生什么呢？很简单，**就是将前面的目录移动至后面的目录**，从效果上看，近似于将第一个参数的目录看作文件。

注意：与文件移动类似的操作，如重命名，对目录的移动同样成立。

1.3.2 如何删除文件 rm

相比于移动文件需要两个参数，删除文件的命令 rm 只需要一个参数即可，也许你也能猜到这个参数的含义，即 rm < 删除的文件路径 >。例如，要删除当前目录下的 INCAR 文件，只需要执行 rm INCAR 即可。同样

的，你也可以使用更复杂的绝对路径或相对路径，例如，删除上一级目录下的 OUTCAR 文件，可以使用 rm ../OUTCAR。

补充：与 Windows 不同，Linux 删除文件通常是直接删除，而不是放在所谓的回收站内。因此，在删除文件时务必小心。

在有些版本的 Linux(例如 Ubuntu)当中，删除的文件被移动至/home/<用户名 >/.local/share/Trash/files 当中，这个目录起到的临时的回收站功能，但你不应该寄希望于这个功能，而是仔细检查删除文件的正确性，并做好合适的备份。

对于删除目录而言，情况有点特殊，需要使用 rm -r命令删除一个目录，此时后面所接参数为目录的路径，例如，删除当前目录下的 vasp 目录，则可以使用 rm -r vasp。

注意：-r 选项通常表示递归，例如，在 rm -r 当中，表示递归删除，从而达到删除一个目录的效果。在删除目录时会连同里面的所有内容都删除掉，因此需要特别小心。

如果担心删除错误的文件，可以在选项中使用-i。rm -i表示在删除时询问是否删除。

对于空目录而言，Linux 还提供了一个命令 rmdir，其用法为 rmdir <目录路径名 >，可以删除一个空目录。

1.3.3 如何复制文件 cp

复制文件的命令为 cp，其用法与移动文件 mv 几乎完全一样，无非就是将移动改为复制。简单来说，语法就是 cp < 源文件路径 > < 目标文件路径 >，类似于 1.3.1当中所介绍的重命名方法，使用 cp 命令同样可以做到复制的同时重命名。例如，cp vasp/OUTCAR ../INCAR 表示将 vasp 目录下的 OUTCAR 文件复制到上一层目录，并重命名为 INCAR

如果想要复制一个目录，也需要使用选项 cp -r。例如，cp -r vasp/ python/表示将 vasp 目录复制到当前目录并重命名为 python。

注意：我们在上面的命令当中使用 vasp/和 python/表示两个目录。其中使用了符号/作为结尾，这个符号通常强调该路径是个目录。对于 Linux 本身而言，有没有这个符号并没有区别。例如，cp -r vasp python 也可以表示上面的操作。我们这么写只是为了强调这两个路径是目录而不是文件。

1.3.4 一次性处理多个文件

前面介绍的 rm，cp，mv，以及在 1.2一节所介绍的 mkdir，都是可以针对多个文件同时操作的。以 rm 为例，如果想同时删除多个文件，只需要在后面添加多个参数即可，其中参数之间以空格分割。例如，rm INCAR KPOINTS 表示删除当前目录下的 INCAR 文件和 KPOINTS 文件。对于 mkdir 创建多个目录而言，也是一样的用法，例如，使用 mkdir vasp ML 表示在当前目录下创建 vasp 目录和 ML 目录。

对于 cp 和 mv 而言，情况稍有不同。它们自身就需要两个参数，第一个是源路径，第二个是目标路径。如果有多个文件需要处理，Linux 默认**最后一个路径为目标路径，前面的所有参数都是源路径**。例如，cp INCAR KPOINTS POSCAR POTCAR .. 表示将 INCAR，KPOINTS，POSCAR 和 POTCAR 复制到上一级目录中。

注意：对于 cp 和 mv 而言，若一次性移动多个文件，则最后一个参数必须是目录。这就意味着不能进行重命名操作。

1.3.5 错误处理

rmdir: failed to remove < 路径名 >: Directory not empty

使用 rmdir 命令时，**只能用来删除空目录**。当要删除的目录不是空目录时，执行该命令则会报错。使用 rm -r < 路径名 > 往往是删除非空目录的常见方法。

cp: -r not specified; omitting directory < 路径名 >

当使用 cp 复制目录时，需要添加-r 选项。如果没有添加这一选项则会报错。

cp: target < 路径名 > is not a directory

这通常出现在尝试使用 cp 复制多个文件时，最后的参数**必须是目录**。如果此时不是目录，则会报错。

rm: cannot remove < 路径名 >: Is a directory

类似于 `cp` 复制目录，使用 `rm` 删除目录时，也需要添加`-r` 选项。特别地，对于一次性删除多个文件，如果在删除文件的同时也存在把目录删除的情况，也需要添加这一选项。

1.4 查看文件

本节作者：Jiaqi Z.

在本节，你将要学到：

- Linux 文件类型
- 如何查看 Linux 文件

这一节看似知识点不多，但命令还是挺多的。因此，一节只讲这一部分内容完全足够了。

1.4.1 Linux 文件类型

补充：在 1.2.1当中，我们介绍了 `ls -l` 命令可以以列表形式查看文件。当时仅仅提到，第一个字符如果是 `d` 则表示目录，如果是-则表示普通文件。在这一部分，我们稍微详细介绍一下更多的文件类型。

- *普通文件（-）：就是普通的文件，通常可以分为文本文件，可执行文件和压缩文件等；*
- *目录（d）：在 Linux 当中，目录也是一种文件，该文件下存放的是这一目录下的 **inode** 号（又名索引节点）和文件名等信息。当执行打开文件时，Linux 实际上是通过 inode 号找到当前文件所在 block（8 个磁盘扇区组成一个 block），从而执行文件；*
- *设备文件，又分为块设备文件（b）和字符设备文件（c）。其中前者可以以“块”为单位进行访问（例如硬盘，软盘等），而后者则是以“字节流”的方式访问（例如字符终端、键盘等）。一般来说，设备文件存放在/dev/目录下；*

- 链接文件（*l*）：一般情况下指的是符号链接（软链接），类似于 *Windows* 操作系统下的“快捷方式”。创建符号链接的方法是使用 *ln*的 *ln -s*选项[7]，例如，*ln -s INCAR INCAR_link* 表示创建了一个指向 *INCAR* 文件的链接文件 *INCAR_link*。当源文件删除时，符号链接文件也会删除；

- 管道文件（*p*）：通常用于进程间的通信，创建方法是 *mkfifo*命令[8]，即 *mkfifo fifo_file*。

- 套接字文件（*s*）：用于通信（尤其是网络上的通信）。简单来说，这是为了避免多个进程或多个 *TCP* 连接同时在一个 *TCP* 协议端口传输数据造成混淆。一般来说，套接字文件包含目的 *IP* 地址，传输层使用协议（*TCP* 或 *UDP*）和使用的端口号，利用套接字文件将三个参数组合起来，从而在传输过程中实现并发服务。

1.4.2 查看文件内容 cat，tac

在整个 Linux 使用过程中，最关键的仍然是**普通文件**和**目录**。虽然其他文件对于操作系统本身而言也很重要，但对于非计算机相关专业的科研用户而言，其意义不大。上面的介绍仅仅是作为一个补充。下面地内容将专注在文件本身。首先一个关键的问题是：**如何查看文件内容**。

注意：当然，从文件操作本身来说，第一件事应当是创建文件。但是，创建文件需要的内容较多（例如，需要一些 vi 编辑器的使用，可能还需要重定向命令，在后面的章节再详细介绍。

如果是初学者，希望可以尽快上手的话，你可以试着在 Windows 本地用记事本创建一个文本文件，并在里面随意输入一些你喜欢的文字（建议使用英文，对于中文等非 ASCII 字符而言，可能会出现乱码。），然后利用远程终端将文件发送至服务器（对于 MobaXterm 而言，在终端左侧有一个目

[7] 相对地还有“硬链接”，直接使用 `ln` 即可。对于硬链接而言，二者本质上是一个文件（类似于做了备份），当其中一个删除时，另一个不会删除；当其中一个文件修改时，另一个也会同时修改

[8] 也许你会疑惑为什么是 fifo 而不是管道的单词 pipe。事实上，FIFO 是一种数据缓存器执行方法，即“先进先出”（First In First Out）。作为数据缓存器，其与普通存储器的区别是没有外部读写地址线，这样使用起来非常简单，但缺点就是只能顺序写入数据，顺序的读出数据。数据地址在内部由指针自动加 1 实现，而不能通过地址线寻找地址。而 Linux 进程间的通信大多就是采用这种通信方式，这种方式也是管道的特性。相对的，还有一种 LIFO，即“后进先出”（Last In First Out），通常“堆栈”（Stack）就是采用这种方法。

录列表，你可以直接将文件拖拽至相应的目录中；对于其他终端软件，请参考其软件具体的操作方法）。后面对文件的查看操作，都可以对这个文本文件进行。

首先需要了解的是，如何查看完整的文件。在 Linux 当中，查看文件内容的命令是 `cat`，其基本用法是 `cat < 文件路径名 >` 例如，对于位于当前目录下的 `INCAR` 文件，可以使用 `cat INCAR` 查看其内容。

`cat` 命令有一些常用选项，例如，可以使用 `cat -n`或 `cat -b`显示行号，二者的区别在于前者会显示所有行号，而后者只对有内容的行显示行号。如果文本中空行内容太多，可以使用 `cat -s`对空行进行压缩，使其缩减为一个空行。

相对地，命令 `tac` 也是查看所有内容，只不过它是从最后一行倒着输出。可以看出，`tac` 本身就是命令 `cat` 倒着写。例如，`tac INCAR` 表示从最后一行开始输出 `INCAR` 文件。

注意：命令 *cat* 并不是单词“猫”的意思，而是连接 *concatenate* 的缩写。正如单词所表示的那样，*cat* 最原始的功能，是连接多个文件。例如，有一个文件叫 *a*，另一个文件叫 *b*，执行命令 *cat a b*，则会将两个文件内容按照顺序连接起来并输出。

1.4.3 关于文件后缀名

对于熟悉 Windows 的用户而言，看到上面（包括之前的所有示例）也许都会有一个疑惑：在 Linux 当中，文件名为什么没有后缀？事实上，后缀名的重要性仅仅是 Windows 操作系统给你的一个“错觉”，让你误以为**后缀名很重要**。事实上，**Windows 操作系统的文件名后缀并不会影响这个文件本身**。

例如，在 Windows 操作系统下创建一个文本文件，后缀名为`.txt`，此时试着将其后缀名改为 `*.mp3` 或者 `*.jpg` 等，并再次在打开方式中用记事本打开（如果使用默认的打开方式一定会出现错误，例如音乐播放器或者图片查看器等）。可以发现，用记事本打开的结果与之前的文本文件内容是完全一样的。

注意：虽然表示后缀名的 . 可以任意放置，但有一个地方比较特殊–文件名开头。对于以 . 开头的文件名而言，它表示的含义是隐藏文件（这在 *1.2.2*一节介绍过了）

对于 Windows 操作系统而言，使用后缀名往往是为了决定文件的打开

方式（取决于 Windows 特有的**注册表**）；而 Linux 文件大多都是文本文件（甚至系统配置也是文本文件），因此在 Linux 当中，文件后缀名就变得不重要了。也正因如此，在 Linux 当中你可以类似于 Windows 后缀名的方式创建任何的后缀（`*.jpg`,`*.xyz` 甚至 `*.zjq`,`*.ykn` 都是可以的），在 Linux 看来，它们仅仅是文件名的一部分。

甚至，在 Linux 当中，大多时候文件都是没有后缀名的。这也就是之前的 `INCAR` 和 `OUTCAR` 为什么没有后缀。对于从 Windows 创建的文本文件上传至服务器而言，可能还留有所谓的后缀名 `*.txt`，你完全可以使用 `mv` 命令将后缀名删去，丝毫不影响文件本身和其他命令的运行。

1.4.4 按页查看文件 more，less

使用 `cat` 和 `tac` 查看文件，都是“一股脑”输出到终端里，对于比较短的文件而言，这种方法是可行的；如果这个文件很长，则要上下翻页就会比较繁杂。

对于多页的文件而言，Linux 可以使用 `more` 命令查看。基本用法是 `more <` 文件路径名 `>`。例如，使用 `more ../band/OUTCAR` 就可以查看上一级目录下的 `band` 目录下的 `OUTCAR` 文件。在查看过程中，可以**使用空格进行翻页，使用回车进行下一行查看**。

在查看过程中，可以随时使用 `q` 键退出。

对于一些需要往回翻页查看的文件，可以使用 `less` 命令。基本调用格式与 `more` 类似，即 `less <` 文件路径名 `>`。与 `more` 不同的是，`less` 命令可以向上翻页（使用 `Page Up` 键或者 `b` 键）[9]

注意：对于 more 而言，实际上也可以通过 b 键实现向前翻一页的效果。但相比于 less 而言，more 的自由性并不是太高。而且，使用 b 向前翻页的效果对于管道文件无法实现。

此外，less 还有更复杂的“搜索功能”，例如，可以使用符号/< 字符串 > 的方法实现向下搜索，使用符号?< 字符串 > 的方法实现向上搜索。同时，less 的其他命令都是在显示文件后的操作，并不是类似于之前的“选项”（即使用-的形式），这种方法与 vi 的使用类似。

无论是 `more` 还是 `less`，都可以使用 `q` 键退出显示文件。

[9]除次之外，也可以使用 `d` 向后翻半页，使用 `u` 向前翻半页。

1.4.5 取头部 head和取尾部 tail

有时，可能会希望仅仅查看一个文件的开头或者结尾。此时可以使用 Linux 操作系统下的 head 和 tail 命令。这两个命令的基本调用方法都是一样的，即 head < 文件路径名 > 和 tail < 文件路径名 >。例如，使用 head POSCAR 就可以查看当前目录下 POSCAR 文件开头几行，同理，使用 tail relax/OSZICAR 就可以查看 relax 目录下的 OSZICAR 文件结尾几行。

注意：通常情况下，直接调用 *head* 和 *tail* 得到的都是开头（或结尾）*10* 行的内容。在有些时候，可能会希望输出更多行，或者少输出几行避免混乱。此时可以使用参数 *head -n*和 *tail -n*实现，其基本格式为 *head -n < 行数 > < 文件路径名 >* 和 *tail -n < 行数 > < 文件路径名 >*，这一选项表示输出指定的行数。例如，*head -n 5 POSCAR* 可以查看 *POSCAR* 文件开头 *5* 行。对于 *tail* 同理。

除此之外，*head* 和 *tail* 还提供了选项 *head -c*和 *tail -c*，表示输出开头（或结尾）多少个字符的内容，格式与上面*-n* 选项类似，即 *head -c < 字符数 > < 文件路径名 >* 和 *tail -c < 字符数 > < 文件路径名 >*。

1.4.6 错误处理

cat: < 文件名 >: Is a directory

cat 命令仅限于查看文件内容，若后面所接内容为一个目录，例如，cat vasp/则会报错

输入 cat 命令后忘记输入文件名直接回车，输入文件名后结果只输出了文件名，并没有输出内容

当直接调用 cat 而没有接任何参数时，表示将终端标准输入所读取到的内容输出到终端。对于普通调用 cat < 文件路径名 > 而言，是将读取到的文件输出到终端。若没有任何参数，则会读取后面输入的内容。

退出的方法则是使用 ctrl+d 键结束当前输入，或者使用 ctrl+c 键强制终止当前命令。

cat: < 文件名 >: No such file or directory

文件路径不存在，检查一下路径（尤其是当前工作路径）是否正确。

head（或 tail）: invalid number of lines: < 文件名 >

当你使用 `head -n` 或 `tail -n` 时，后面的行数是必须提供的一个参数。若没有提供行数，则会报错。

head（或 tail）: cannot open < 文件名 > for reading: No such file or directory

文件路径不存在，检查一下路径（尤其是当前工作路径）是否正确。

head（或 tail）: error reading < 文件名 >: Is a directoryy

类似于使用 `cat` 打开目录，使用 `head` 或 `tail` 打开目录就会报这种错误。

使用 `more` 查看文件，输出 * < 文件名 >: directory ***

这是因为试着用 `more` 查看目录而不是文件。

使用 `less` 查看文件，输出许多奇怪的路径，不是想要的内容

如果你仔细看一下里面的内容就会发现，当你用 `less` 查看目录时，输出的是这个目录下所有的文件和目录（包括隐藏文件）。事实上，使用 `less < 目录路径 >` 得到的结果和使用 `ls -l < 目录路径 >` 是一样的。只不过前者是单独输出的，而后者是直接输出在终端里。

Missing filename ("less –help" for help)

在调用 `less` 时忘记提供文件路径了。

more: bad usage Try 'more –help' for more information.

与上面的错误类似，在调用 `more` 时忘记提供文件路径了。

1.5 压缩与解压缩

本节作者：Jiaqi Z.

在本节，你将要学到：

- 如何压缩文件
- 如何解压缩文件

1.5.1 备份和压缩

补充：虽然在许多场合，会将 *Linux* 的一些使用 *tar*的操作说成是压缩文件和解压缩文件，但这个表述实际上是不贴切的。事实上，*tar* 的本意是 *tape archive*，指的是“磁带存档”，是为将若干个文件归档到磁带上，从而方便备份而设计的。而压缩文件实际上在 *tar* 当中经历了另外的步骤，即 *gzip* 压缩，或者是 *bzip2* 压缩等。这些在命令上都是通过额外的选项实现的。

但是，由于现在大多数时候都是习惯于将两个步骤合二为一，包括使用 *gzip* 压缩后得到的 *.gz* 文件也可以在 *Windows* 操作系统下解压缩，从而极大方便了文件之间的跨系统传输。因此，在通常情况下，我们使用到的都是“压缩”。这里之所以给出两者的不同仅作为补充扩展用，在后续表述中，往往不做区分，一律表述为“压缩”和“解压缩”。

1.5.2 使用 tar命令压缩文件

tar 命令在使用时通常会配合许多选项，在官方文档中，选项就有 50 个左右甚至更多，因此，我们不可能在这里完全介绍完所有的选项。对于一般的科研工作而言，只需要掌握几个最基本的选项即可。

首先一个最基本的选项是 tar -c，表示**创建备份文件**。通常仅有这一个参数是不够的，还需要配合以如 tar -f参数，这一参数表示**指定备份文件**。结合这两个选项，可以得到一个备份文件的基本模式为：tar -cf < 备份文件路径 > < 要备份的文件 1 路径 > < 要备份的文件 2 路径 > ...。例如，tar -cf vasp.tar INCAR KPOINTS POSCAR POTCAR 表示将当前目录下的 INCAR,KPOINTS,POTCAR 和 POSCAR 备份至当前目录的 vasp.tar 当中。

注意：*-cf* 后面的参数，除第一个是备份文件路径外，后面所有参数都是要备份的文件路径。

正如 1.5.1所说的关于压缩和备份的区别一样，我们这里所做的仅仅是

备份。对于真正的压缩，我们还需要添加一个压缩格式[10]。对于常见的 gzip 压缩格式而言，使用的选项是 `tar -z`。因此，一个完整的压缩命令可以表示为 `tar -czf <压缩文件路径> <要压缩的文件 1 路径> <要压缩的文件 2 路径> ...`。

注意：一般情况下，使用 *gzip* 压缩的文件后缀名都是 *.gz*。

对于上面所提到的备份例子，你能想到它的压缩命令是什么吗？

【答案】：tar -czf vasp.tar.gz INCAR KPOINTS POSCAR POTCAR

1.5.3 解压缩

相对地，有了压缩过程，就一定会有**解压缩**过程。首先，先忽略掉压缩格式（即 gzip 等相关内容），仅仅从备份的角度，考虑它的逆过程，也就是**还原文件**。

在 `tar` 当中，可以使用选项 `tar -x`实现备份文件的还原。例如，在开始的备份操作中，可以使用 `tar -xf vasp.tar` 实现对备份文件 `vasp.tar` 的还原。对于解压缩过程，选项完全类似，只需要使用 `tar -xzf` 即可。例如，对上面的 `vasp.tar.gz` 进行解压缩，可以使用 `tar -xzf vasp.tar.gz`。

1.5.4 查看压缩文件

这里所说的查看压缩文件，主要指的是查看**压缩包内的文件**，从更广义的角度看，就是查看所谓的“备份”文件。

首先，在 `tar` 里面有一个选项 `tar -v`，可以在压缩（解压）过程中查看压缩（解压）的文件。例如，上面的压缩和解压命令，分别可以写成 `tar -cvf vasp.tar INCAR KPOINTS POSCAR POTCAR` 和 `tar -xvf vasp.tar`。对于 gzip 格式的压缩和解压缩，只需要在参数里额外添加`-z` 即可。

另外一种可能的场景是已经存在一个压缩文件（例如 `vasp.tar`），需要查看里面的内容。虽然直接解压查看是一种办法，但如果发现不是想要的文件再删除，就稍显麻烦（尤其是有多个文件时）。因此，Linux 提供了一种类似于 Windows 直接查看压缩包文件的方法，即使用 `tar -t`表示列出压缩包内文件。

[10]所谓的**压缩格式**，在 Windows 系统下常见的如 zip、rar 等，而在 Linux 操作系统下，最常见的是 gzip，当然也有如 bzip、xz 等。

注意：在使用 *tar -t* 时，往往需要配合以*-f* 参数指定压缩文件名。其完整用法为 *tar -tf <* 压缩文件路径 *>*。例如，使用 *tar -tf vasp.tar* 可以查看 *vasp.tar* 压缩文件中的文件列表。

无论是普通的备份文件，还是使用 *gzip* 压缩的文件，都是使用 *tar -tf* 查看（没有选项*-z*）。

使用 *tar -tvf* 同样可以得到文件列表，只是输出的内容更详细（类似于 *ls -l* 的输出结果）

除此之外，查看压缩文件还有一种方法，使用 less命令。通过 less < 压缩文件路径 > 可以直接查看压缩文件内容，其形式上类似于 tar -tvf 和 ls -l。

1.5.5 压缩文件的追加与合并

虽然已经非常小心地创建了压缩文件，但有时还是会有遗漏。例如，当你将 INCAR,POSCAR,KPOINTS 和 POTCAR 已经添加到 vasp.tar 之后，突然发现还应当把 CONTCAR 添加进去。如果此时文件还保留着，当然，重新使用 tar -cf vasp.tar ... 也是可以的（其中... 表示五个文件路径）。但是，如果之前的文件已经删除了呢？解压后再重新压缩也不是不可行，但总是麻烦一步。

在 tar 的选项中，提供了一个选项 tar -r表示**将文件追加到压缩文件内**。例如，上面的例子，可以直接使用 tar -rf vasp.tar CONTCAR 即可将 CONTCAR 添加到 vasp.tar 中（哪怕原先的四个原始文件删除了也没关系）。

上面的例子是将文件追加到压缩文件内，如果是**将压缩文件内的所有文件全部追加到另一个压缩文件里呢**？可以使用 tar -A选项。其格式为 tar -Af < 追加的目标压缩文件路径 > < 追加的压缩文件路径 >。例如，我们已经有了包含 INCAR,KPOINTS,POSCAR,POTCAR 的压缩文件 vasp.tar，此时又有一个压缩文件 result.tar，里面包含有 OUTCAR,CONTCAR，如何将其合并到共同的 vasp.tar 当中呢？可以使用 tar -Af vasp.tar result.tar。

注意：这里的选项*-A* 是大写字母，千万不要写成小写字母。二者的含义不同，对于小写字母 *tar -a*，表示根据后缀来决定压缩格式。例如，使用 *tar -caf vasp.tar.gz INCAR* 将会以 *gzip* 格式创建压缩文件。

同时，使用*-A* 合并压缩文件时，只能对两个文件进行合并。

1.5.6 错误处理

tar: < 压缩文件路径 >: Cannot stat: No such file or directory
tar: Exiting with failure status due to previous errors

通常这是因为在调用 tar 时错误放置了压缩文件路径和被压缩的文件路径的位置。在使用 tar 进行压缩时，第一个参数是压缩文件路径，第二个参数是被压缩的文件路径。

例如,对前面的例子,如果使用的是 tar -czvf INCAR KPOINTS POSCAR POTCAR vasp.tar.gz，就会报错。

tar: Refusing to write archive contents to terminal (missing -f option?)
tar: Error is not recoverable: exiting now

在使用 tar 进行压缩（或解压）时，需要给定选项-f 并指定压缩文件名，例如 tar -cf vasp.tar INCAR。如果没有选项-f 则会报错。

tar: Cowardly refusing to create an empty archive

这意味着你在压缩文件时试图压缩空的文件。这通常是因为你没有指定压缩文件（例如，直接调用 tar -cf vasp.tar 就会报错）。

还有一种可能是你错用了压缩选项-c 和解压缩选项-x。例如，也许上面的命令你是想解压 vasp.tar，那么你需要的命令是 tar -xf vasp.tar。

tar: < 压缩文件路径 >: file is the archive; not dumped

这可能是因为你在压缩文件时对压缩文件本身进行压缩，这可能会造成递归压缩。例如，tar -cf vasp.tar vasp.tar 时就会报错。

但是,压缩文件本身是可以被压缩的。例如,tar -cf vasp.tar result.tar 是允许的，这执行的操作是将 result.tar 文件压缩至压缩包 vasp.tar 当中。

Part II

VASP 计算

Chapter 2

声子谱计算

Contents

2.1 什么是声子、声子谱

本节作者：Isay K.

在本节，你将要学到：

- 声子
- 声子谱

2.1.1 声子

声子（Phonon），即“晶格振动的简正模能量量子”，是晶体中**原子振动**的量子化描述。

在固体物理学中，声子是晶格振动的准粒子，其携带能量和动量，并且可以像粒子一样进行相互作用。

声子是**简谐近似**下的产物，如果振动太剧烈，超过小振动的范围，那么晶格振动就要用非简谐振动理论描述。

声子并不是一个真正的粒子，声子可以产生和消灭，有相互作用的声子数不守恒，声子动量的守恒律也不同于一般的粒子，并且声子不能脱离固体存在。声子只是格波激发的量子，在多体理论中称为集体振荡的元激发或准粒子。

声子的化学势为零，属于**玻色子**，服从玻色-爱因斯坦统计。声子本身并不具有物理动量，但是携带有准动量，并具有能量，它的能量等于 $\hbar\omega_q$。

声子可以分为以下两类：

- 声学支：与晶格的纵向和横向振动相关，类似于声波，表示原胞的整体振动。
- 光学支：与晶格的非均匀振动相关，通常与电荷的重新分布有关，表示原胞内原子间的相互振动。

如果一个材料的原胞中有 N 个原子，那么声子谱就会有 3N 支，其中 3 条声学支，3N-3 条光学支。

2.1.2 声子谱

声子谱，也称为声子色散关系，是描述声子能量与动量之间关系的图表。

声子谱通常在第一布里渊区内绘制，因为其包含了所有可能的声子模式。

通常，使用声子谱研究体系的动力学稳定性，使用分子动力学研究体系的热力学稳定性。

声子谱的其他物理意义：

- 电子-声子耦合：在半导体和超导体中，电子-声子耦合相互作用对材料的电子性质至关重要；
- 声子散射：在金属和半导体中，声子散射是影响电子迁移率的关键因素；
- 热容：声子谱可以解释材料在不同温度下的热容行为
- ...

2.2 计算方法简介

本节作者：Isay K.

在本节，你将要学到：

- 密度泛函微扰理论（DFPT）
- 有限位移法（Finite Displacement Method）
- 适用情境比较

2.2.1 密度泛函微扰理论（DFPT）

DFPT 是一种基于第一性原理的方法，它直接从周期性边界条件的 Kohn-Sham 波函数计算出声子谱。在 DFPT 中，通过计算原子间相互作用的微扰来得到力常数矩阵，这是描述晶格动力学性质的关键量。

补充：1987 年，Baroni、Giannozzi 和 Testa 提出了一种新的晶格动力学性质计算方法–微扰密度泛函方法 (Density Function Perturbation Theory)。

DFPT 通过计算系统能量对外场微扰的响应来求出晶格动力学性质。该方法最大的优势在于它不限定微扰的波矢与原胞边界 *(super size)* 正交，不需要超原胞也可以对任意波矢求解。因此可以应用到复杂材料性质的计算上。此外，能量对外场微扰的响应不仅可以推导出声子的晶体性质，还能求出弹性系数、声子展宽、拉曼散射截面等性质, 这种方法本身就能算出 *Born effective charge dielectric constant,* 可以很好的预言 *LO-TO splitting* 甚至 *Kohn anomalies*。这些优势使得 *DFPT* 一经提出就被广泛应用到了半导体、金属和合金、超导体等材料的计算上。比较常用的程序是 *pwscf* 和 *abinit*, *castep* 等采用的是一种 *linear response theory* 的方法（或者称为 *density perturbation functional theory*，*DFPT*），直接计算出原子的移动而导致的势场变化，再进一步构造出动力学矩阵。

2.2.2 有限位移法（Finite Displacement Method）

有限位移法通过在超原胞中引入原子的有限位移来模拟晶格振动。这种方法基于位移-响应理论，通过计算原子位移后系统的受力来构造动力学矩阵

补充：直接法，或称 *frozen-phonon* 方法，是通过在优化后的平衡结构中引入原子位移，计算作用在原子上的 *Hellmann-Feynman* 力，进而由动力学矩阵算出声子色散曲线。用该方法计算声子色散曲线最早开始于 *80* 年代初，由于计算简便，不需要特别编写的计算程序，很多小组都采用直接法计算材料性质。直接法的缺陷在于它要求声子波矢与原胞边界 *(super size)* 正交，或者原胞足够大使得 *Hellmann-Feynman* 力在原胞外可以忽略不计。这使得对于复杂系统，如对称性高的晶体、合金、超晶格等材料需要采用超原胞。超原胞的采用使计算量急剧增加，极大的限制了该方法的使用。这种方法不能很好的预言 *LO-TO splitting,* 只有在计算了 *Born effective charge* 和 *dielectric constant* 之后，进一步考虑了 *non-analyticity term*，才能计算出；但 *Direct Method* 本身并不能给出 *Born effective charge* 和 *dielectric constant*，所以这也是它的一个缺陷。

2.2.3 适用情境比较

DFPT 适用情境：

- 需要高精度声子谱的系统，尤其是小到中等大小的晶胞；

- 研究者希望避免有限位移法可能引入的系统误差时。

注意：*DFPT* 方法计算成本较高，尤其对于大晶胞或高对称点附近的计算。

有限声子法适用情境：

- 当计算资源有限或需要对多种材料进行筛选时；
- 对于大晶胞材料的初步声子谱分析。

总得来说，对于较重的任务，DFPT 方法可能会造成内存溢出，且 DFPT 方法由于其特性而无法进行并行计算，而有限声子法可以并行。对于较小的体系，可以根据需要和组内资源选择方法。

补充：建议优先使用有限位移法。

补充：一些教程中有时候将有限位移法又称为冷冻声子法或直接法。但笔者并没有找到更官方的资料说明有限位移法和冷冻声子法是同一种方法，谨奉上 *PHONOPY* 官网供读者自行分辨：*https://phonopy.github.io/phonopy/index.html*

参考：http://muchong.com/html/200802/723527.html

2.3 计算软件 PHONOPY

本节作者：Isay K.

在本节，你将要学到：在本节，你将要学到：

- 开始安装之前
- PHONOPY 快速安装
- PHONOPY 使用方法

2.3.1 开始安装之前

由于本教程面向的群体是计算小白（包括笔者也是通过本教程记录一下自己掉的坑），所以在开始安装软件之前，我们强烈建议先咨询组内老师或师兄师姐：服务器上是否已经配置了相应的软件？

补充：当然也可以使用 *module avail* 命令自己检查系统内已安装的软件，如果没有找到的话再咨询更有自主性哦。

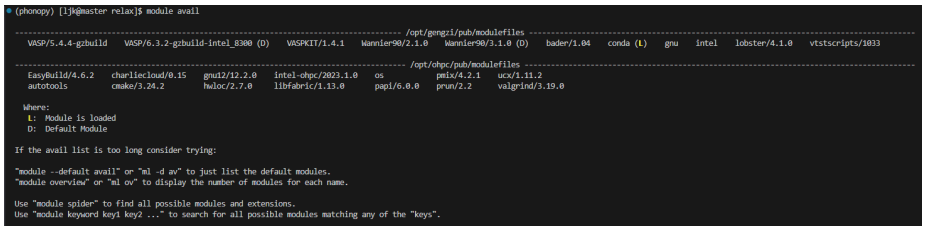

图 2.1: 检查是否含有所需软件

通常情况下，组内服务器的**根目录**下已经配置了相应的软件，这个时候再在自己的**用户目录**下进行配置的话，一方面在使用过程中可能会出现命令的冲突，另一方面也是一种时间、精力和资源的浪费。

如果组内确实并没有安装，或者你是传说中的开山大弟子，又或者是自学，请放心进入第二小节。

2.3.2 PHONOPY 快速安装

参考:https://blog.csdn.net/qq_41866202/article/details/124407208?spm=1001.2101.3001.666
task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ECtr-1-124407208-blog-
139391379.235%5Ev43%5Epc_blog_bottom_relevance_base9&depth_1-utm_source=distribute.p
task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ECtr-1-124407208-blog-
139391379.235%5Ev43%5Epc_blog_bottom_relevance_base9&utm_relevant_index=1

2.3.3 PHONOPY 使用方法

加载 PHONOPY 环境

1. 加载 Anaconda 应用：`module load conda`
2. 激活 PHONOPY 环境：`conda activate PHONOPY`

命令详解

进入下面的官网之后点击 Command options 即可看到所有功能。

https://phonopy.github.io/phonopy/index.html

注意：在进行扩胞时的标准是：扩胞后的原子达到 80 100 个，每个晶轴方向大于 10，不然得到的声子谱中很容易出现虚频。

2.4 具体计算步骤

本节作者：Isay K.

在本节，你将要学到：

- 计算声子谱前的结构优化
- DFPT 方法计算声子谱
- 有限位移法计算声子谱

2.4.1 计算声子谱前的结构优化

注意：在计算声子时需要先对原胞结构做高精度的结构优化，不然得到的声子谱中很容易出现虚频。

我们以 $TiTe_2$ 为例，以下是高精度优化的具体参数。

Listing 2.1: INCAR

```
Global Parameters
ISTART = 1          (Read existing wavefunction, if there)
ISPIN =  1          (Non-Spin polarised DFT)
LREAL = .FALSE.     (Projection operators: automatic)
ENCUT =  380       (Cut-off energy for plane wave basis set, in
     eV)

LWAVE = .FALSE.      (Write WAVECAR or not)
LCHARG = .FALSE.     (Write CHGCAR or not)
ADDGRID= .TRUE.     (Increase grid, helps GGA convergence)
LASPH = .TRUE.      (Give more accurate total energies and
    band structure calculations)
PREC  = Accurate    (Accurate strictly avoids any aliasing or
    wrap around errors)
NCORE = 8
ISYM = 0
```

```
Lattice Relaxation
NSW    =  300         (number of ionic steps)
ISMEAR = 0            (gaussian smearing method )
SIGMA =  0.03         (please check the width of the smearing)
IBRION = 2            (Algorithm: 0-MD, 1-Quasi-New, 2-CG)
ISIF   =  3           (optimize atomic coordinates and lattice
    parameters)
IOPTCELL = 1 0 0 1 1 0 0 0 0
EDIFF = 1E-08
EDIFFG = -1E-03   (Ionic convergence, eV/A)
```

为了保证优化精度足够高，其中需要注意的是：

1. EDIFF 表示电子收敛标准，至少要取 1E-06，体系小的话尽量取 1E-08；
2. EDIFFG 取负值时表示力收敛标准，取 1E-03；
3. ADDGRID 表示是否添加额外网格提高精度，设定为.TRUE.；
4. PREC 表示“精度”模式，设定为 Accurate（准确）；
5. NSW 表示电子优化步数，取 300 防止计算中断；
6. ENCUT 可以自行做测试，详见 VASP 计算-结构优化章节（先别去找，我没写）；

另外，其中 ISIF=3 表示既优化晶格又优化原子坐标，配合 IOPTCELL 可以实现晶轴的单独固定，以达到计算二维材料的目的，详见 VASP 计算-结构优化章节（也还没写）。

下面的其他输入文件没有需要特别说明的，如有疑问请参考 VASP 计算-结构优化章节（哈哈，又是这）或参考 VASP 官网：https://www.vasp.at/wiki/index.php/The_VA

Listing 2.2: KPOINTS

```
A
0
Gamma
24   24   1
```

```
0.0  0.0  0.0
```

Listing 2.3: POSCAR

```
TiTe2-1m1
   1.00000000000000
     3.7458432095936396 -0.0000184453725456 0.0000000000000000
    -1.8729216047968198 3.2439861579338527 0.0000000000000000
     0.0000000000000000 0.0000000000000000 18.0000000000000000
   Ti   Te
     1    2
Direct
  0.0000000000000000 0.0000000000000000 0.5127400160000022
  0.6666666132020026 0.3333334158033583 0.6098496477195335
  0.3333334157979960 0.6666666131966403 0.4156403382804701

  0.00000000E+00 0.00000000E+00 0.00000000E+00
  0.00000000E+00 0.00000000E+00 0.00000000E+00
  0.00000000E+00 0.00000000E+00 0.00000000E+00
```

Listing 2.4: POTCAR

```
（结构优化章节）
```

提交任务进行计算，得到 CONTCAR 为优化后的更合理的结构，作为后续声子计算的初始晶胞。（后续小节中提到“初始晶胞”均指优化后得到的晶胞，为避免歧义在此说明。）

2.4.2 DFPT 方法计算声子谱

1.`mkdir method_DFPT`
新建文件夹。

2.cp relax/CONTCAR method_DFPT/POSCAR

将上一步高精度结构优化得到的 CONTCAR 复制进文件夹内，并重命名为 POSCAR。

3.cd method_DFPT

进入新文件夹。

4.module load conda conda activate phonopy

加载 conda 模块，并激活 phonopy 环境，详情可参考 2.3.3

5.phonopy -d --dim="6 6 1"

使用 PHONOPY 进行 6×6 的扩胞。

此时会产生数个名为 POSCAR-0? 的位移文件，以及名为 SPOSCAR 的扩胞后的结构。

DFPT 方法使用的是 SPOSCAR，而有限位移法使用的是这些位移文件。

注意：笔者研究的是 *2D* 结构，仅对两个方向进行扩胞，读者可根据需要自行调整。扩胞的标准是扩胞后达到 *80 100* 个原子，且晶轴长度大于 *10* 埃，不然得到的声子谱中很容易出现虚频。

6.mkdir vasp-calculations

新建文件夹用于后续计算。

补充：此处根据个人习惯不同，也可以不新建文件夹。将 *POSCAR* 重命名为 *POSCAR-unit*，将第 *5* 步新产生的 *SPOSCAR* 重命名为 *POSCAR*，直接在当前文件夹中进行计算。

7.cp SPOSCAR vasp-calculations/POSCAR

将第 5 步新产生的 SPOSCAR 复制进文件夹，并重命名为 POSCAR。

8. 准备其它基本文件：

Listing 2.5: INCAR

```
SYSTEM = TiTe2
#ISIF = 3
NSW = 1
IBRION = 8

LWAVE = F
LCHARG = F
```

```
ENCUT = 380
EDIFF = 1E-8
EDIFFG =-1E-3
ISMEAR = 0

LREAL = F
SIGMA = 0.03

PREC = A
ADDGRID = .TRUE.
```

Listing 2.6: KPOINTS

```
A
0
Gamma
3   3   1
0.0  0.0  0.0
```

注意：因为该计算使用的是扩胞之后的结构，所以 K 点没有必要取太大。

Listing 2.7: POTCAR

```
老方法
```

9.`sbatch sub.vasp`
提交任务进行计算。
10.`cd ..`
返回 method_DFPT 文件夹。
11.`cp vasp-calculations/vasprun.xml .`
将 vasprun.xml 复制到当前文件夹。

12.`phonopy --fc vasprun.xml`
使用 phonopy 读取 vasprun.xml 生成力常数文件 FORCE_CONSTANTS。
13.`vi band.conf`
编辑 band.conf 文件：

Listing 2.8: band.conf

```
ATOM_NAME =Ti Te
DIM = 6 6 1
BAND =0 0 0 0.5 0 0 0.33333 0.33333 0 0 0 0
BAND_POINTS = 101
FORCE_CONSTANTS = READ
```

1. DIM 根据体系的扩胞大小设置，如扩胞扩到 332，就设置成 332。
2. BAND 和能带的取点是一样的，也可以用 vaspkit 生成。
3. FORCE_CONSTANTS 一定设置成 READ。
4. 更多设置可以看 PHONOPY 官网。

14.`phonopy -p -s band.conf`
使用 phonopy 读取 band.conf 文件，作图并保存。
15.`phonopy-bandplot --gnuplot > phonon.out`
将数据导出方便后续用 Origin 等软件重新绘图。
补充:旧版本的 phonopy 的导出命令为:bandplot --gnuplot> phonon.out

2.4.3 有限位移法计算声子谱

1.`mkdir method_yxwy`
2.`cp relax/CONTCAR method_yxwy/POSCAR`
3.`cd method_yxwy`
4.`phonopy -d --dim="6 6 1"`
前四步和 DFPT 法完全相同。
5.`for i in 01..12; do mkdir $i; cp POSCAR-0$i $i/POSCAR;done`

假如第四步产生了 12 个位移文件，使用 for 循环生成 12 个文件夹，并将对应的位移 POSCAR 移入文件夹重命名为 POSCAR。

6. 准备其它基本文件：

Listing 2.9: INCAR

```
ADDGRID = .TRUE.
PREC = Accurate
IBRION = -1
ENCUT = 380
EDIFF = 1E-8
EDIFFG = -1E-3

ISMEAR = 0
SIGMA = 0.03

IALGO = 38

LREAL = .FALSE.
LWAVE = .FALSE.
LCHARG = .FALSE.

NCORE = 4
```

注意：有限位移法的单个计算实际上就是高精度的静态自洽。

Listing 2.10: KPOINTS

```
Automatic mesh
0
Gamma
3 3 1
0 0 0
```

注意：同 *DFPT* 法一样，有限位移法使用的也是扩胞之后的结构，所以 *K* 点没有必要取太大。

7.`for i in 01..12; do cp INCAR KPOINTS POTCAR sub.vasp $i; cd $i; sbatch sub.vasp; cd $OLDPWD;done`

将基本文件复制进各个小文件夹中并进行计算。

8.`phonopy -f 01..12/vasprun.xml`

计算全部结束后,使用 phonopy 读取全部的计算文件夹中的 vasprun.xml，生成 FORCE_SETS 文件。

9.`vi band.conf`

Listing 2.11: band.conf

```
ATOM_NAME =Ti Te
DIM = 6 6 1
BAND =0 0 0 0.5 0 0 0.33333 0.33333 0 0 0 0
BAND_POINTS = 101
FORCE_SETS = READ
```

与 DFPT 方法唯一不同的部分在于将 FORCE_CONSTANTS=READ 改成 FORCE_SETS=READ。

10.`phonopy -p -s band.conf`

使用 phonopy 读取 band.conf 文件，作图并保存。

11.`phonopy-bandplot --gnuplot > phonon.out`

将数据导出方便后续用 Origin 等软件重新绘图。

2.5 声子谱分析

本节作者：Isay K.

我还不太会，等我学学。

http://pubs.acs.org/doi/abs/10.1021/acs.jpcc.5b04669

2.6 错误处理

vasprun.xml 没有必要信息

Part III

Python 与机器学习

索引

参考文献