一些废话

大家好，我是 ~~suzumiya.tech.sast.phys.tsinghua~~ 来自物理系科协技术组的盘笛 ~~suzumiya~~

很高兴认识大家。

众所周知 heroxbd 老师给物理壬小学期开了一门 小 课叫做 实验物理的大数据方法 ，我们这个培训的目的就是帮大家 ~~无痛~~ 迈过第一道门槛，学到一些技术。

~~甚么，竟然不能从第一节课就带着你慢慢入门？~~

Fly_Bitch.jpg

关于培训的 Motivation 以及基本内容

在这次培训的相关信息放出之后，有人质疑

> “相关的技术文档如此丰富，续老师的 FAQ 也很友好，为什么还需要培训呢？”

~~首先，你得先有相关的一些知识，或者至少有一点点大概的理解才能看懂文档~~。这也是上手技术的“最初一公里”，在技术方面，初学者往往是最痛苦的，~~因为我也是这么过来的~~，在日常中或者是网上，一些技术讨论会默认你大概领会一些很杂的前置概念，不知不觉的使用技术名词，如果你并不了解，可能会被各种术语淹没，不知所措。所以找个懂哥带带路先走两步，带到能够知道需要向哪个方向努力，并能自主寻找方式解决问题的程度总是好的。我希望自己努力让这个讲座做到这一点。

本次培训的目的，即以尽量通俗实用的语言介绍如下东西（下面是大致框架目录在网页左侧边栏，可点击跳转）：

- 了解物理学家用 **计算机文件（Computer File）**、**文件系统（File System）** 以及简单的 **操作系统（Operating System, OS）** 知识
- 了解 GNU/Linux
  - 多词辨析：Terminal/Shell/Bash/Console/Powershell/Cmd/Linux/GNU/POSIX/Unix/Debian/Ubuntu/WSL...
  - Linux 的文件系统
  - 常用的命令
    - `ls`, `rm`, `cp`, `mv`, `clear`;
    - 包管理器 `apt`
  - 编辑器
    - 编辑器简介
- 小学期生存法则（一些经验）
  - 遇到问题怎么办？
  - 如何提问？
  - ...

可以根据上述大纲

- 判断自己的需要，有取舍的学习
- 判断自己是否需要参加线下技术培训，如果没法让你学到东西，我会很愧疚浪费了你的时间。

关于讲稿用语风格的一点注（狡）释（辩）

我想让讲稿/培训变得有趣一点，不至于搞得大家睡着

~~“考完了 sei 想听你在这 bb 这些破玩意儿啊”~~

所以写作/讲解风格会比较有个人特色（高情商），说白了就是嗯整烂活，有点长也是因为嗯整烂活。如果你讨厌这种文风 ~~那还真是抱歉呢（笑）~~ 看给出的参考文献吧，而参考文献则是我认为写的比我好得多的教程。

~~请各位轻点喷我~~

参考文献里面也有很多在我熬过小学期的时候帮我大忙的东西。希望也能帮到你们（たぶん…

*关于讲稿知识的 "严谨性" 问题

我很喜欢上这门课的 heroxbd 老师所说的以下观点

> 学习技能时，**理论支持是辅助的，练习是核心的。**
>
> 在无法理解一个新概念时，最重要的是**在练习中体验获得第一手经验。**一个还**不能透彻理解的工具**，如果我会用，就已经**掌握了它的重要接口部分**。这与基础数学和物理课程的演绎风格很不一样，**要求我们在不求甚解时，掌握工具之间关系的全貌**，是“归纳”式的学习。这些内容的共同点是“恰好在物理实验的数据处理中有用”。此类经验性的技能，只要能以某种方式实现，满足“四个原则”注释：详情见 heroxbd 老师第一节课内容 即可，**允许和鼓励“不求甚解”。**
>
> 归纳学习可以类比婴儿学习人类语言的过程。一开始，他所了解的概念都是分立的。随着他长大，联系才被慢慢建立起来。这种想法与物理实验很像，因为只要涉及现实世界，就没有完整的模型。不论是光谱仪，还是量热计，其中总会残留无法解释的现象，被我们按照系统误差处理了。

本文所说的很多概念深究下去都能写本书，但是 **只懂一点点的状态** 并不妨碍我去 **理解并熟悉怎么用**。计算机**技术**的灵魂就是**封装抽象**，我愿称该种精神为技术的哲学。

文件，文件系统，操作系统

什么是文件

甚么，你问我文件是什么？~~我不到啊！~~

- ~~一些官话：**文件是以计算机硬盘为载体所存储的信息集合**~~
- Windows 版本：文件当然就是你打开电脑之后的桌面上、~~AB~~ CDEFG 盘里面各种小东西啦！
- macOS 版本：文件当然就是你打开电脑之后桌面上，**目录** 下的各种小东西啦！

为了真正理解文件以及文件系统，首先让我简单介绍一下硬盘：

- 机械硬盘 (HDD, Hard Disk Drive)：用磁介质和磁头存储和检索数字信息的信息载体
- 固态硬盘 (SSD，Solid State Drive)：用集成电路（常采用闪存技术）来存储和检索数字信息的信息载体

而文件就可以简单的理解为

- 机械硬盘上面磁道上某一组磁介质极化方向的集合
- 固态硬盘上面的一组电容的电平高低的集合。

极化形式和电平以 0，1 分类（正向磁化 or 反向磁化，高电平 or 低电平）。在硬盘里，我们使用 0 和 1 来记录信息。而将我们所熟悉的信息 (文本信息，声音信息，图像信息等) 转化为能在硬盘上存储的信息，则需要将原有的信息进行**编码**，将信息转化为二进制 bits 序列。同样的，我们读取的时候也顺着编码方式进行**解码**，从而将信息还原出来。解码的过程所依赖的软件就是对应文件格式的**阅读器**，比如我们使用 Windows Media Player，WPS，Office 等打开不同的文件，都是在使用这个思路。

在计算机中文件通常有**文件名**用来区分不同的文件，其中“.”后面称为**后缀名**，标注的是文件的**编码方式**。我们可以依据此对计算机中的文件进行简单的分类

纯文本文件

对 **纯文字信息** 进行编码的文件，比如 .txt, .cpp, .html, .py, .md 等等。

人们为了用尽可能少的内存编码尽可能多的字符，设计了许多字符与二进制之间的映射方案。

对文本，常见的有 ASCII，GBK，UTF-8 等 (很蛋疼的一件事就是一些汉语文本在不同的编码方案下情况是不同的...因此需要指定解码方式，否则可能会出现乱码)

> 例子：下面是选自励志小故事《吾亦死》的一段字符串的 UTF-8 编码结果
>
> ```
> 字符串：
> 好啊！来啊！打在胸上！
> UTF-8:
> 好啊！来啊！打在胸上！
> ```
>
> 采取十六进制表示，~~这个可以有！（赞赏）~~

媒体文件

对 **声音，图像信息** 等进行编码的文件，以及多种信息组合成的文件，比如：

- 声音文件 .mp3, .wav 等
- 图像 .jpg, .png 等
- 图像 + 声音组合成的视频文件 .mp4, .avi 等
- 图像，文字，视频，声音组合成的 .pptx, .docx, .pdf 等等
- 为了减少空间占用的压缩文件.rar, .zip 等

我们再一次使用编码解码的思路进行理解，我们使用 Media Player 等多媒体播放器，都是在对相关文件进行解码

程序文件

高级语言 (C，C++，python 等) 的源代码文件例如 .cc, .py 等本质上只是**文本文件**。~~所以记事本写代码这种事情当然是可以的~~

而我们这里说的程序文件，是这些东西，比如：~~原神.exe~~，~~星穹铁道.app~~，Mathematica.exe，，，（省略号文明，逗号野蛮）

那么这些文件又是什么呢？学过 C，C++ 的同学都知道（什么，你不知道？那请看我写的另一点学术垃圾，点这里），对于编译型语言，写好代码之后需要把代码进行编译链接，转化为一条一条的**机器指令序列**，这些指令通常是按照某些规则组织起来的一串**命令**，用来告诉 CPU 该干什么：调用哪个寄存器，把哪两个数据进行加法，把什么数据写入到哪个寄存器中等等。*

之后把这些命令和相关的支持（比如一些运行时等）打包在一起形成**可执行文件**，.exe/.app 等文件就是程序文件的一种。点到为止，程序就是能够让计算机帮你做事的文件，你给文件一定的输入，程序高效率的给你完成某样任务，并给出输出。

* *注释：这里涉及到 **指令集** 等等计算机指令架构的知识，也就是**通用计算机**的思想：利用有限完备指令集，即一些有限的，通用的操作，将大任务转化为这些操作的有序序列，只需依次进行这些基本操作，就可以完成任务*

小结以及与课程衔接

总之，文件本质上就是用 0 和 1 按照一定的规则编码信息得到的。我们需要根据不同文件的编码方式（对应文件类型的编码方式有相关的行业标准，例如：音频、视频等编码的 MPEG-4 规范，数字图像有损压缩的 JPEG 规范等），借助相关的软件完成对文件的解码，获得我们所需要的信息。

在大型物理实验中会产生海量的实验数据（所谓 dark ♂ 数 ♂ 据 时代），如何以一种易于被理解和检查的方式（heroxbd 老师所说的 **透明原则**）在文件中组织这些数据是一个很重要的课题。

在实验物理的大数据课程第六讲中，heroxbd 老师会以 .csv, .hdf5, .json 三种文件为例子，讲述在这三种文件中，数据是以一种怎样的组织形式存储在文件之中的（**编码**），我们又该怎么使用 Python 中的包装好的工具去读取这些数据以便后续的处理（**解码**）。

虽然文件的各种后缀名是很多很多的（有些实验工程由于数据过于庞大，还会自主设计数据文件编码规范，维护相关的读取工具。例如：粒子物理领域的欧洲核子中心 CERN 设计的数据分析框架 ROOT），但是无论怎么变化，**文件**的相关基本的理念是不会变的，“要把文件的理念先搞懂！”。

操作系统简介 (OS, Operating System)

我们平时使用的例如 QQ，微信，浏览器 Chrome，游戏平台 Steam，科学计算软件 Mathematica 等等**程序**，都可以简单理解为在一个叫做**操作系统**的程序基础上实现的**更高层次功能的程序**。

而**操作系统**则上接高层次软件，下链接硬件

- 承担了对于硬件的调度，将硬件封装成一个又一个程序的**接口**供调用（硬件抽象）
- 协调不同软件同时运行时，对于一些内存以及计算资源的竞争（并发问题）
- 提供高层次软件公用的基础工具的程序 (比如编译器，数据库管理，存储器格式化，**文件系统管理**，用户身份验证，驱动管理，网络连接等) 等等更加基础和公共的功能。
- ...

实现上面功能的关键部分就是我们称作**操作系统内核（Kernel）**的东西，内核是操作系统最基本的部分。它是为众多**应用程序**提供对计算机硬件的**安全访问**的一部分软件。可以理解成**覆盖在硬件上的第一层软件**，把硬件变成了黑箱，抽象出来一些可以调用的接口。

什么是并发？什么是这些乱七八糟的概念？没有理解不要担心，只需要知道操作系统连接着我们的硬件资源和软件即可。

POSIX

Portable Operating System Interface，操作系统的国际标准。只要我在一个满足 POSIX 的操作系统上写一个程序，那么任何满足 POSIX 的操作系统都能跑这个程序，满足了 heroxbd 老师的 **复现原则**.

GNU/Linux，macOS，或者其他的类 Unix 系统都满足 POSIX 标准，很可惜，Windows 不是一个满足 POSIX 的操作系统，因此，对于 Windows 用户我们需要 WSL 来填充我们对于 POSIX 的需求（待续）

CLI vs GUI

- Q1:**灵魂拷问，计算机一定需要鼠标吗？当我们在使用鼠标的时候，我们在干什么？**

  A1:当我们移动鼠标的时候，鼠标上面的**传感器**会将移动的位移等物理信息转换为**数字信息**，并通过连接线或者蓝牙等实现数据的传输，数据会进入 **鼠标驱动软件** 进行数据的处理，这个软件通过**操作系统**，调动计算机的 CPU, GPU. 计算相应的小箭头移动的方向和速率等信息，并渲染（画）出**图形**到显示器上；

  当我们双击一个文件的时候，同样会通过上面类似的过程，让**操作系统**帮忙调动**文件系统**打开相应的**文件**，随后经过 CPU, GPU 的计算将相应的文件以图形的形式渲染到显示器上，比如弹出了一个窗口。

可以看到，我们与计算机操作系统的交互是以鼠标 + 显示器显示图形的方式进行的，这种与计算机交互的形式就是 **图形用户界面**（GUI Graphical User Interface）是上个世纪人机交互的伟大创造之一，大大降低了计算机的使用成本，飞入寻常百姓家

- Q2:**那在图形用户界面出现之前，人们使用的是什么呢？**

  A2: ~~当然是打孔卡~~ 当然是命令行界面 (CLI Command Line Interface) 了！下面我简单介绍一下他的工作原理

  我们通过键盘向 Shell 输入按照一定规则组织的字符串，也即**命令**。**Shell**也是一个**程序**，他会解析（读懂）我们所输入的字符串命令，并与**操作系统**交互，告诉计算机做相应的事情。

  Shell 如其名一般，是覆盖在操作系统内核上的一层新的软件层，承担人与操作系统交互的功能。我们需要 Shell 是因为内核本身不提供任何交互的方式

  其实 Shell 也有非 CLI 的形式，不过我们在这里默认讨论 CLI

- Q3:**GUI 那么傻瓜，为啥还要 CLI**
  - GUI 需要进行大量的图形学运算，相比 CLI 速度会慢不少
  - CLI 与 计算机本身的一些精神更加靠近一些，通过掌握它你会更高效的使用计算机处理问题，会更习惯使用计算机思维进行思考。
  - ~~因为小学期要用~~

文件结构

好了，现在我们理解了计算机里面是以一种怎么样的形式**组织存储信息**的，也就是**文件**，也大概了解了操作系统是个什么东西，下面一个十分迫切的需求就是了解，操作系统是以一种什么样的形式**组织文件**的。理解这个，在 GNU/Linux 中十分的重要。因为他的哲学就是

> “一切皆文件”

根目录，家目录，文件路径

首先让我们从自己所使用的操作系统中感受以下文件系统

- 对于 Windows 用户，我们最熟悉的就是各种盘，C 盘，D 盘，E 盘等等；每一个**盘符**里面由文件夹和各种文件组合成一种树状结构。这样的模式叫做**多根逻辑存储结构**
- 对于 macOS 和 GNU/Linux 用户，我们最熟悉的则是一个 `/` 下面放了很多文件夹，之后在这个 `/` 下面由很多的文件和文件夹，组成一个树状结构，这样的模式叫做**单根逻辑存储结构**

而对于文件所处在这样一个文件系统之中的位置，我们可以使用**路径**来进行描述。由于路径比较重要也用的比较多，下面我会以 GNU/Linux 的文件结构来详细介绍这个概念，其他的差不太多。

- 根目录：顾名思义，就是在这个文件结构最底层的那个目录，也即 `/`，可以理解成你的硬盘

```
# GNU/Linux or MacOS etc.
/

# Windows
C:/
D:/
...
```

- 家目录：
  - **用户**：事实上，你的电脑是可以有很多人同时使用的，这些叫做**用户**，每一个用户都有自己的目录（展示 Windows 的家目录），在 GNU/Linux 中使用 `~/` 表示
  - **Root 用户**：在大型服务器中，会开很多用户，同时为了不让这些用户乱搞事，我们会限制他们的一些权限（比如一些文件不能删除，修改，读取等等），而在所有用户之上有一个超级用

户，你可以把它理解成**网管**，他拥有操纵计算机资源的最高权限。

```
# 家目录
~/

# 绝对路径(见下)
/home/name/
```

- 绝对路径：路径从盘符/根目录开始

```
/home/laplacian/academic
```

- 相对路径：描述某一个文件相对于你当前所处路径的位置
  - `./` 当前文件夹
  - `../` 前一级的文件夹
  - `../..` 上上级别，以此类推

```
# 我所处的目录（以绝对路径表示）
/home/laplacian/academic

# 放在 /home/laplacian 下的文件夹 life 的相对路径可以写成
../life

# 有一个放在 academic 文件夹中的文件 linux.pdf 可以写成
./linux.pdf # 相对路径
/home/laplacian/academic/linux.pdf # 绝对路径
/linux.pdf # 错误
```

Linux 系统的简单介绍

好了，我们终于终于进入关键部分了，也就是 Linux 操作系统的简单介绍。对于 macOS 用户实际上不需要安装 Linux 环境，因为 macOS 本身是 POSIX 的。我们需要 WSL 只是因为 Windows 不是 POSIX 的。

安装 Terminal/WSL

请参考以下这些教程，首先以满足课程需求的安装过程为主，进入答疑环节

注意，由于 WSL2 中会涉及到 Hyper-V 的兼容性问题，如使用 WSL2 可能会导致 VirtualBox 等安卓模拟器以及像 VMware 等与 Hyper-V 不兼容的虚拟机无法使用，请自行决定是否升级到 WSL2

[实验物理大数据方法 FAQ-windows 环境配置]

[这个是 OIwiki 的教程]

*一些常见的概念辨析

- Unix：一种操作系统
- GNU：GNU is Not Unix, 被认证成 Unix 要付费，开源软件运动
- distro. (Linux 发行版)：我们日常中说的 Linux 一般是某种发行版，而发行版是什么呢？发行版为一般用户预先集成好的 Linux 操作系统及各种**应用软件**。可以想想，我们在拿到一台新电脑时也不是只给我们一个裸的 Windows 内核，会预装网络软件，图形界面等等。Linux 发行版也是一样的道理，这些操作系统通常由 Linux 内核、以及来自 GNU 计划的大量的函数库组成，常见的有

Ubuntu，Debian，CentOS, Arch...

- WSL: Windows Subsystem for Linux，在 windows 上提供 POSIX 环境
- Terminal：终端，在计算机发展的早期指连接到计算机上的，输入输出等外部设备。现在由于 PC 的普及和进化，生活中已经很少看到大型的计算机了。所以 Terminal 进化为了终端**模拟器**（简称终端），模拟终端的功能捕捉输入，渲染输出，管理 Shell。
- Console：控制台（特殊的终端），是系统管理员的终端，拥有更高的权限。
- CLI Shell：Shell 如其名一般，是覆盖在操作系统内核上的一个**程序**，承担人与操作系统交互的功能，他会解析（读懂）我们所输入的字符串命令，并与**操作系统**交互，告诉计算机做相应的事情。
  - **解析**你输入的字符串：寻找字符串对应的**程序**，并运行该程序
    - 如果你输入的是一个**绝对路径**，则 shell 会按照路径寻找并运行，如果是一个单词则会前往**环境变量**中寻找

      ```
      ./my-console.exe # 运行当前文件夹下的程序 my-console.exe
      ls # 运行环境变量中的 ls 程序
      ```

  - **环境变量**：是一系列放**程序文件**的**文件夹**，如果我直接输入的字符串是这些文件夹中某一个文件夹里的一个程序，则可直接执行，因此我们称这些环境变量中的**程序**为**命令行程序**，简称**命令**，使用下面的命令可以看自己有哪些环境变量

    ```
    echo $PATH
    ```

  - 注意，在不同的发行版上，命令会有所不同！

```
# deb 系
apt install xxx # 下载 xxx
# CentOS
yum install xxx # 下载 xxx
```

- Bash：既然 Shell 是一类程序，那么只要能使得我们能与操作系统内核交互的程序都是一种 Shell，bash 就是众多 Linux Shell 中的的一种，全称为 Bourne Again Shell，除了 Bash，还有 macOS 上 常用的 zsh，比较现代的 fish 等，使用下面命令查看自己安装了哪些 shell

  ```
  echo $SHELL # 查看当前发行版默认 shell
  cat /etc/shells # 查看当前发行版上可用的 shell 有哪些
  ```

- PowerShell：Windows 自带的 Shell，与 Windows 内核交互

Linux 的文件系统

~~你已经具备的理解大部分知识所需要的前置概念，现在可以自己查看一些相应的文档了！~~ 其实后面就是教会一些常用的命令了，太理论的铺垫我想已经做的差不多了

进入的初始界面如下

```
yjsp@yjsp-pasokon:~/ # 用户名@计算机名
```

其中的 `~/` 就是家目录的省略，全称为

```
/home/yjsp/
```

在根目录下面有很多操作系统的相关文件，这就是 Linux 的哲学 **一切皆文件**

- 普通文件是文件
- 目录/文件夹也是文件
- 硬件设备也是文件 `/dev`，
- `/bin` 文件夹中存放了各种命令程序
- `/home` 文件夹中存放的是用户的文件

  ...

想进一步深入了解根目录下各个文档的功能的，请看 Linux 的系统目录结构

常见命令

```
ls #列出当前目录的所有文件
ls -t # 按照修改时间顺序列出当前目录的所有文件
ls /home/yjsp # 列出给出目录下的所有文件
```

我们可以给命令加入参数，调节功能

```
rm my-console.md # 删除给出的文件（彻底删除）
rm -rf /var/log # 删除给定的文件夹 /var/log
cp my-console.log ./log # 把本目录下 my-console.log 文件复制到本目录下的 ./log 文件夹中
mv ./my-console.log ./log # 把本目录下 my-console.log 文件移动到本目录下的 ./log 文件夹
中
mv my-console.log my-terminal.log # 相当于将文件重命名
clear # 清屏
```

此外

- 有一些命令需要你拥有更高的权限才能运行，，此时可以在运行不了的命令前面加上 `sudo`，
- {% label ^C %}（即 {% label ctrl + c %}），可以强制终止当前运行的命令
- {% label Tab %}，可以补全命令
- 方向键上下，可以快速查看之前的命令
- 方向键左右，可以更改光标位置。

这里只列出了最基础最常用的命令，还有很多命令在课程上会进行教学，也有很多相关的资料可以进行自学。

环境，包管理器 和 `apt`

- **Q1：当我们配置环境的时候，我们在配置一些什么？**

  相信很多人都玩过一个东西叫做 Minecraft，如果你使用启动器，他的 Java 版本在某些启动器情况下需要你手动下载 Java 环境。因为 Java 要跑在 JVM（Java Virtual Machine）上面

  这个过程我们也可以把他称作配环境，简单来说，你写出来的任何代码要能跑起来，都需要相关的**环境**对其进行支持。

  比如，

  - 我需要在 Linux 上跑 C++，我就需要下载**编译器**等相关支持，在 Linux 我们可以使用 GNU 开源编译器 GNU GCC；
  - 我需要在 Linux 上跑 python，我就需要下载**解释器**等相关支持
  - 我需要在 WSL 上使用 python 画图，并弹出绘制结果的图片，我就需要配置让 WSL 能够调用我 windows 窗口的环境，这也是为什么我们要安装 X11 服务器 VcXsrv（当然也有一些更加友好的，不需要安装 VcXsrv 的工具，强烈推荐 Jupyter，感兴趣的同学可以自行研究）
- **Q2：既然我们需要配环境，那 Linux 上面该怎么下载相应的文件呢？**

在 Linux 上我们下载软件也是通过命令进行的，Linux 提供了一个 **包管理器** 来帮助我们 **管理（下载，升级，删除）** 软件包。

在 deb 系发行版的 Linux 下，最常用的包管理器是 `apt` ，`apt`最底层的是 `dpkg`，但是我们不推荐使用 `dpkg` 对你的软件包进行管理。因为 dpkg 不能联网安装/更新软件包，也不会自动的下载你下载的软件包所依赖的一些其他的软件包。

下面我们会较为详细的介绍 `apt`.

- 当我们运行 `apt` 时，`apt` 会按照储存在你系统上的一个**索引文件**去下载相应版本的软件，因此要获得最新版的软件，建议在每次下载前使用`sudo apt update`更新。
- 如果我们需要要安装软件，只需要运行命令 `sudo apt install <包名（可包含多个包，用空格分隔）>` 即可
- 卸载软件则可以使用 `sudo apt remove <包名>`， `sudo apt autoremove`。
- 若要更新软件（更新大部分软件），一般的流程都是如下

```
sudo apt update
sudo apt upgrade
```

1. 大部分软件包操作都需要 `root` 权限，需要使用 `sudo` 或者在 `root` 用户下执行。
2. `apt` 内部默认软件源服务器在国外，因此访问很慢，请参考 FAQ 换 TUNA 镜像

下面是一个小练习

> **Check Point!**
>
> 请在网上自行学习一下如何在 Linux 中压缩文件以及解压文件，需要下载什么 GNU 软件，并自己尝试压缩一个文件和解压一个文件

编辑器简介

- **Q：啥叫编辑器啊？**
  我们把用来修改**文本文件**的软件叫做**编辑器**，比如在 Windows 上面的记事本，有挺多同学使用的 VScode 也是一个轻量编辑器，而且有众多的插件，支持连接远程等功能。

  选择合适的编辑器就像厨师挑一把称心的刀，完全取决于 coder 对工具的品味，~~所以我选择 VScode~~

  宗 教 战 争 不 可 避

  在 Linux 上面，常用的编辑器有

  - GNU nano：一般是自带的
  - vim 学习曲线陡峭，但是一旦心领神会便能够以飞快地速度写代码
  - Emacs（本人没用过，xbd 老师使用 Emacs）
  - **VScode** 最热编辑器，有众多插件支持个性化设计，例如 vim 插件等等...

这里真心建议各位打开一边摸索一边使用，不要着急一口气吃胖子，~~你折腾不坏编辑器的~~。

小学期生存法则

不知不觉这个网页写了四百多行了，确实又长又臭，不过很感谢读到这里的你，希望我以我浅薄的对这些东西的理解能帮助到你。~~干面来点真正的私货~~，线下是完整版（）

如何问问题？

问一个技术问题需要让对方能够复现你的错误，请给出你所使用的环境配置相关的参数以及相关报错信息等。~~**实在不清楚建议往 wyy 学长的 issue 里学（doge）**~~，我觉得问的比较规范的一个问题 #issue3060 ，按照这样问，~~学长还喷你们你就把这个甩到他脸上~~

遇到问题怎么办？

多上网，多用搜索引擎。

github issue 和 stackoverflow 以及官方文档会是你的好朋友

文档要善用搜索，查找你需要的函数名称，从他给出的例子入手照着修改嵌入到自己的代码里会更快学会东西一些。

多和同学讨论，~~多抱大腿~~

看看 FAQ 里面的遇到问题时你该怎么办

References

- 实验物理大数据方法资源课程网页
- 实验物理大数据方法FAQ
- Missing Semester of Your CS Education, MIT 开给 CS 学生的教会你如何精通一些开发工具的课程，写的很 CS 但是确实十分的实用，可以自行判断是否需要。
- 菜鸟教程 中文社区里面较为优质的学习资料
- StackOverflow，基本上多数你遇到的问题别人之前都遇到过，善用论坛，善用搜索
- 杨希杰 - Linux 讲座 - 2021 信院联合暑培，这是我的相关知识的启蒙之作，2021 年暑假好像是第一次也是唯一一次信息学院三系（软无雷）联合暑培，当时笔者参加了开始的几次，虽然后面小学期电设课直接灭掉了我所有剩下的暑假的时间，但是光是那几次就让我收获颇丰。杨学长这份讲稿是本讲稿的重要参考资料。
- 计算机系学生科协技能引导文档，~~我的评价是写的比 You Know Who 系写的好~~，贵系一如既往的发扬开源精神。相关教程可读性很高，~~大概是因为贵的 Fly Bitch 现象太严重了~~。

鸣谢

感谢基科 21 的 Zeauw 贡献了一部分讲义的内容

感谢物理 01 的 yfzhao，mhliu 提供了一些点子以及指出讲义的缺陷

感谢 heroxbd 老师对培训的支持