认识大家
为什么
课程概论
计算环境
学习建议
版本控制
对 FAQ 和讲义作贡献

总论

续本达

清华大学 工程物理系

2025-06-23 清华

自我介绍

2005-2009 在数理基科学习（科协）

2009-2018 在日本神冈地下实验室，学习研究中微子和暗物质

2018-至今 在工程物理系近代物理研究所任教

主要工作：锦屏中微子实验

合作研究

1. JUNO 江门中微子实验
2. 日本 SuperK 超级神冈实验

研究兴趣：中微子质量、核子衰变、地球中微子

爱好：大数据分析、高性能计算、系统架构运维

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队

助教

王宇逸 物理系本科毕业，工物系博士生
武益阳 物理系本科毕业，工物系博士生
徐闯 工物系本科毕业、工物系博士生
郝传晖 物理系本科毕业，工物系博士生
刘逸祺 工物系本科毕业、工物系博士生
胡宇阳 工物系本科毕业、工物系准博士生
温欣洋 工物系本科毕业、物理系准博士生
张珺铫 物理系本科生（物理系科协）
何宇峰 工物系本科生（工物系科协）
杨哲涵 工物系本科生（工物系科协）
刘骥安 工物系本科生

教学团队（续）

顾问

陈晟祺 计算机系本科毕业、计算机系博士生
翁俊 工物系本科毕业、工物系博士生

小助教

- 请踊跃报名。

算力支持

物理系科协（somnia）、工物系科协（cat）

我为什么在这里?

培养方案要求?

- 本课程的“性价比很低”!（由某巨佬结课后评价道）

班主任的观察

（学长）每年都劝退（学弟学妹），每年都劝不住.....然后劝不住的学完开始劝退下一届

听说这门课讲 Python?

- 现在猴子都会 Python，我不会的话会很焦虑

我为什么在这里?

培养方案要求?

- 本课程的“性价比很低”!（由某巨佬结课后评价道）

班主任的观察

（学长）每年都劝退（学弟学妹），每年都劝不住……然后劝不住的学完开始劝退下一届

听说这门课讲 Python?

- 现在猴子都会 Python，我不会的话会很焦虑

我为什么在这里?

培养方案要求?

- 本课程的“性价比很低”!（由某巨佬结课后评价道）

班主任的观察

（学长）每年都劝退（学弟学妹），每年都劝不住……然后劝不住的学完开始劝退下一届

听说这门课讲 Python?

- 现在猴子都会 Python，我不会的话会很焦虑

“实验物理”和“大数据方法”哪个更重要

- 课程名的中心语在“大数据方法”，所以后者重要？
- 服务于“实验物理”，所以前者重要？
- 思考：如果两者去掉其一，你能否接受？

形式逻辑与实验同等重要

- 计算机是形式逻辑演绎机，计算机的原理植根于逻辑。

爱因斯坦 1953 给 J.S.Switzer 的信

西方科学的发展是以两个伟大的成就为基础：希腊哲学家发明形式逻辑体系 (在欧几里得几何中)，以及 (在文艺复兴时期) 发现通过系统的实验可能找出因果关系。

“实验物理”和“大数据方法”哪个更重要

- 课程名的中心语在“大数据方法”，所以后者重要？
- 服务于“实验物理”，所以前者重要？
- 思考：如果两者去掉其一，你能否接受？

形式逻辑与实验同等重要

- 计算机是形式逻辑演绎机，计算机的原理植根于逻辑。

爱因斯坦 1953 给 J.S.Switzer 的信

西方科学的发展是以两个伟大的成就为基础：希腊哲学家发明形式逻辑体系 (在欧几里得几何中)，以及 (在文艺复兴时期) 发现通过系统的实验可能找出因果关系。

“实验物理”和“大数据方法”哪个更重要

- 课程名的中心语在“大数据方法”，所以后者重要？
- 服务于“实验物理”，所以前者重要？
- 思考：如果两者去掉其一，你能否接受？

形式逻辑与实验同等重要

- 计算机是形式逻辑演绎机，计算机的原理植根于逻辑。

爱因斯坦 1953 给 J.S.Switzer 的信

西方科学的发展是以两个伟大的成就为基础：希腊哲学家发明形式逻辑体系 (在欧几里得几何中)，以及 (在文艺复兴时期) 发现通过系统的实验可能找出因果关系。

释义

什么是“实验物理”

- 一切费电的物理！
  “理论学家费纸，实验学家费电，理论实验物理学家费米”
  - 欧洲核子中心 (CERN) 一年用电约 $1.3\,\mathrm{TW\,h}$。
    北京东城区西城区 2016 年用电总量 $10.1\,\mathrm{TW\,h}$，人口 200 万。
- 观察物理现象的学科
  - 控制物理现象发生的环境？是 → “实验”(狭义)；否 → “观测”。
  - 实验物理的目标：发现物理规律，包括测量物理常数。
    从取得的数据出发，进行统计推断，证伪物理规律的假设。

什么是“大数据”

商业概念。“大”是相对的，大数据是指无法用一台计算机处理的数据。

释义

什么是“实验物理”

- 一切费电的物理！
  “理论学家费纸，实验学家费电，理论实验物理学家费米”
  - 欧洲核子中心 (CERN) 一年用电约 $1.3\,\mathrm{TW\,h}$。
    北京东城区西城区 2016 年用电总量 $10.1\,\mathrm{TW\,h}$，人口 200 万。
- 观察物理现象的学科
  - 控制物理现象发生的环境？是 $\rightarrow$“实验”(狭义)；否 $\rightarrow$“观测”。
  - 实验物理的目标：发现物理规律，包括测量物理常数。
    从取得的数据出发，进行统计推断，证伪物理规律的假设。

什么是“大数据”

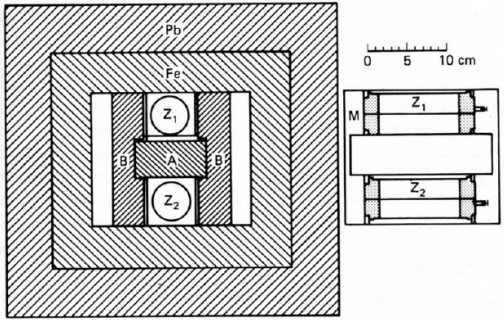
商业概念。“大”是相对的，大数据是指无法用一台计算机处理的数据。

逻辑电路的起源

- 盖革计数器，电离辐射通过时给出脉冲信号
- 真空管：源于阴极射线管。

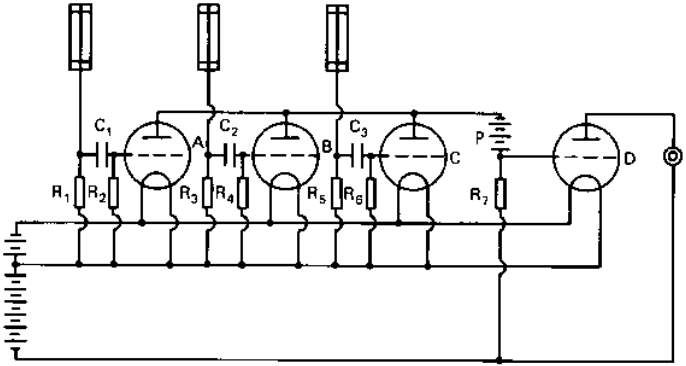
宇宙射线的发现

Bothe-Kolhoerster 实验 (Bothe 1954 诺贝尔物理奖)



- 用“符合”的方法发现了宇宙射线，证明并为高能带电粒子。

真空三极管的出现与应用



- Rossi 盖革计数器与真空三极管组合，形成宇宙射线的测量逻辑。

实验物理与计算机

- 需要构造：顶端探测器无计数，中间有计数的逻辑
  - 真空电子管“与”、“或”、“非”等逻辑门出现
  - 真空电子管通用电子计算机：ENIAC（1945）
- 中子和辐射输运问题
  - 第一代电子计算机大多用来进行 Metropolis Monte Carlo 算法
  - 模拟预测氢弹中的输运过程
- 半导体逻辑电路
  - 晶体管替代真空电子管
  - 计算机功耗和体积大幅减小
- Monte Carlo 算法成为大型物理实验设计的一部分

计算历史：大型机

- 一台大型计算机，用户通过终端 (Terminal) 连接

计算历史：个人计算机

- 1981 年

计算历史：个人计算机组成集群

- 1990s 个人计算机组成集群 "beowulf"

高性能计算与超级计算机

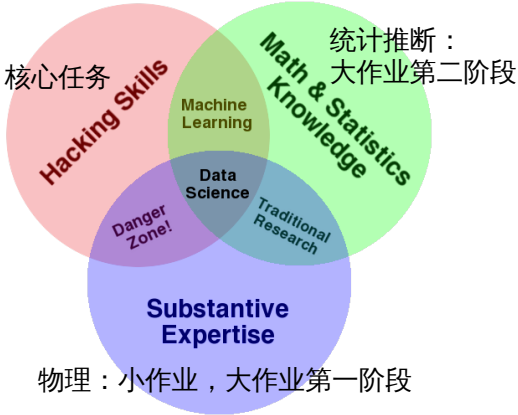
- 更大的计算机集群，优化节点间的通信，共享存储。
- 软件环境如何配置？

大物理与大数据

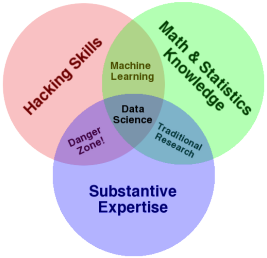
- 大型物理实验，产生大数据，多台计算机协同工作：分布式计算。
- 跨地理的分布式计算称为“网格”(grid)，在早期促进和影响了互联网的发展。
  - FAST 天眼：每年 $\sim 1\,\mathrm{PB} = 1 \times 10^{15}\,\mathrm{B}$。
  - JUNO 江门中微子实验：每年 $\sim 2\,\mathrm{PB}$ 。
- CERN 在 90 年代决定使用 Intel 民用计算机组成的集群来完成 LHC 的计算任务，是个人电脑级硬件集群成为科学计算主流的标志。
  - X86(Intel 和 AMD) 计算集群是当今大数据工业界的主流配置。
- 近年来，大数据方法独立于实验物理，被工业界广泛采用，与神经网络结合，催生了“深度神经网络”，或“深度学习”。
  - 一系列新方法被应用到实验物理。

此课在知识体系中的地位

- 这是一个数据的时代。
- 数据时代需要 黑客技术 Hacking Skills、数理统计 Math & Statistics Knowledge 、专业知识 Substantive Expertise 等多方面技能。



技能树



1. 本课程目标为数理大类的同学补齐 黑客技术 。
2. 警惕 Danger Zone! 。
   黑客技术 人类掌握工具的最高水平
   数学与统计 人类理性思维的最高水平，欢迎选修《概率统计分析及量测技术》
3. 专业知识 根据自己的兴趣选择。
   - 物理学的思维方式和实践方法是一切专业知识的标杆。
4. 本科时代要积极认真学习以上硬知识。

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - XY ，XY ？
- 1 类学科：努力确认变量之间影响的趋势
- A 类学科：努力算出具体的数字
- Z 类学科：没有不确定度的结果不是科学结论！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
- A 类学科：努力算出具体的数字
- Z 类学科：没有不确定度的结果不是科学结论！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - XY ，XY ？
- 1 类学科：努力确认变量之间影响的趋势
- A 类学科：努力算出具体的数字
- Z 类学科：没有不确定度的结果不是科学结论！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 这个数字是 1.2 还是 1.3？
  - 这个数字是 1.2 ± 0.1 还是 1.2 ± 0.2？
- Z 类学科：没有不确定度的结果不是科学结论！
  - 这个数字是 1.2 ± 0.1 还是 1.2 ± 0.2？
  - 这个数字是 1.2 ± 0.1 ± 0.2 还是 1.2 ± 0.2 ± 0.1？

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
- Z 类学科：没有不确定度的结果不是科学结论！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年？
- Z 类学科：没有不确定度的结果不是科学结论！
  - 这个学科，还是不要学了吧！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年？
- Z 类学科：没有不确定度的结果不是科学结论！

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年？
- Z 类学科：没有不确定度的结果不是科学结论！
  - 电子磁矩，以玻尔磁子 μ_B 为单位

$$g/2 = 1.00115965218073(28)$$

Ref. Phys. Rev. Lett. 100, 120801 (2008)

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年？
- Z 类学科：没有不确定度的结果不是科学结论！
  - 电子磁矩，以玻尔磁子 μ_B 为单位

$$g/2 = 1.00115965218073(28)$$

Ref: Phys.Rev.Lett.100:120801,2008

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素？
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定？
  - $X\bar{Y}$ ，$\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字
  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年？
- Z 类学科：没有不确定度的结果不是科学结论！
  - 电子磁矩，以玻尔磁子 μ_B 为单位

$$g/2 = 1.00115965218073(28)$$

Ref: Phys.Rev.Lett.100:120801,2008

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don't Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don't Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don’t Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

- 最佳工具会随时间变化，因此课程名不是 Python 数据处理与科学计算

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don’t Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

- 最佳工具会随时间变化，因此课程名不是 ~~Python~~ 数据处理与科学计算
- 推论：入手研究之前，应当优先调研现有工具。

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don’t Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

- 最佳工具会随时间变化，因此课程名不是 ~~Python 数据处理与科学计算~~
- 推论：入手研究之前，应当优先调研现有工具。

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don’t Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

- 最佳工具会随时间变化，因此课程名不是 ~~Python 数据处理与科学计算~~
- 推论：入手研究之前，应当优先调研现有工具。

课程内容

- 以实验物理为主题，学习大数据方法的基础。
  1. 理解数据的科学精神，从一开始养成良好的科研习惯
  2. 掌握典型工具 Git, Python 和常用命令行工具，鼓励自学
  3. 培养工具的品位，针对具体场景选择合适的工具

计划安排

- 实验物理的大数据方法 (1) 五教 5103
  - 第一周 Git，Python 基础
  - 第二周 Python 科学计算与可视化
- 实验物理的大数据方法 (2) 五教 5203
  - 第三周 数据处理的命令行工具
  - 第四周 基于关系代数的数据组织
- 答疑：课后 4:55–5:30

课程内容

- 以实验物理为主题，学习大数据方法的基础。
  1. 理解数据的科学精神，从一开始养成良好的科研习惯
  2. 掌握典型工具 Git, Python 和常用命令行工具，鼓励自学
  3. 培养工具的品位，针对具体场景选择合适的工具

计划安排

- 实验物理的大数据方法 (1) 五教 5103

  第一周 Git，Python 基础

  第二周 Python 科学计算与可视化
- 实验物理的大数据方法 (2) 五教 5203

  第三周 数据处理的命令行工具

  第四周 基于关系代数的数据组织
- 答疑：课后 4:55–5:30

课程评估 按百分制记分 录入成绩时映射为等级

平时作业 62%

- 覆盖课堂学习的知识要点
  - 平时作业中的 80% 为黑盒测试，20% 为白盒测试。
  - 白盒测试重点考察：Git commit 的内容必须有意义，程序书写规范

大作业 30%

- 取材自中微子实验物理的真实场景。也可以由同学自行提出。
  - 分两个阶段，对应前两周“生成”和后两周“分析”，互为逆过程。

捌分 8%

- 小助教 (A)，贡献 FAQ 内容 (B)，审校讲义内容 (C)，课堂贡献 (D)，娱乐作业 (E)
- 计入总评 $\min(8, \sqrt{A^2 + B^2 + C^2 + D^2 + E^2})$

教材

- 自编讲义：《实验物理的大数据方法》

参考资料

- 往年课程录像：`http://hep.tsinghua.edu.cn/~orv/teaching/physics-data/`

辅助资料

- Allen Downey, Think Python 2e
- Fernando Perez et al., Scipy Lecture Notes
- Jeroen Janssens, Data Science at the Command Line
- David MacKay, Information Theory, Inference, and Learning Algorithms
- `https://learnxinyminutes.com/` 从已经有编程基础迅速入门 Python。

有智慧地提问

- 遇到困难多求助
  - 特别是那些可以把人“卡住”的“小”问题。一定不要自己扛，及时求助：老师、助教、小助教。
  - 学长忠告 不要问我是怎么知道的系列
    - 不要用某度，用 Google 或 Bing 海外版
    - 以 CSDN 为代表的中文网站 99% 的信息都是错的（或过时的）
    - stackexchange 系列英文网站 70% 信息可信，其它英文网站 50% 可信
    - 官方文档 100% 可信
    - 综合多种信息来源，作出自己的判断，不要轻信小学生博主。

求助时要有智慧 作者 Eric Raymond

- 原则：让他人以最小的努力复现出你的问题
  1. 不要这样说话：救命!
     ~~我的程序坏了！哪位大神快来帮帮我！急！在线等！哭了！~~
  2. 使用最简洁的语言，借助最简单的例子描述问题
- 原则：努力让你的问题使更多的人受益
  1. 非隐私问题，尽可能公开提问，公开讨论。不要私信

有智慧地提问

- 遇到困难多求助
  - 特别是那些可以把人“卡住”的“小”问题。一定不要自己扛，及时求助：老师、助教、小助教。
  - 学长忠告 不要问我是怎么知道的系列
    - 不要用某度，用 Google 或 Bing 海外版
    - 以 CSDN 为代表的中文网站 99% 的信息都是错的（或过时的）
    - stackexchange 系列英文网站 70% 信息可信，其它英文网站 50% 可信
    - 官方文档 100% 可信
    - 综合多种信息来源，作出自己的判断，不要轻信小学生博主。

求助时要有智慧 作者 Eric Raymond

- 原则：让他人以最小的努力复现出你的问题
  1. 不要这样说话：救命!
     ~~我的程序坏了！哪位大神快来帮帮我！急！在线等！哭了！~~
  2. 使用最简洁的语言，借助最简单的例子描述问题
- 原则：努力让你的问题使更多的人受益
  1. 非隐私问题，尽可能公开提问，公开讨论。不要私信

有智慧地提问

- 遇到困难多求助
  - 特别是那些可以把人“卡住”的“小”问题。一定不要自己扛，及时求助：老师、助教、小助教。
  - 学长忠告 ~~不要问我是怎么知道的系列~~
    - 不要用某度，用 Google 或 Bing 海外版
    - 以 CSDN 为代表的中文网站 99% 的信息都是错的（或过时的）
    - stackexchange 系列英文网站 70% 信息可信，其它英文网站 50% 可信
    - 官方文档 100% 可信
    - 综合多种信息来源，作出自己的判断，不要轻信小学生博主。

求助时要有智慧 作者 Eric Raymond

- 原则：让他人以最小的努力复现出你的问题
  1. 不要这样说话：~~救命!~~
     ~~我的程序坏了！哪位大神快来帮帮我！急！在线等！哭了！~~
  2. 使用最简洁的语言，借助最简单的例子描述问题
- 原则：努力让你的问题使更多的人受益
  1. 非隐私问题，尽可能公开提问，公开讨论。不要私信

如何使用 Gitlab 提问和解答

- issue（议题）
  - 创建要讨论的问题；
  - 注意指定 issue 的负责人。

GitLab issue

- `https://git.tsinghua.edu.cn/physics-data/faq/-/issues/`
- `https://git.tsinghua.edu.cn/physics-data/lecture/-/issues`
- 创建 issue
  - 标题：概括问题
  - 正文：详细描述问题
    1. 给出复现问题的方法，提供 最小可复现单元
    2. （可选）分析问题，提出可能的解决方法
- 回复 issue ，参与讨论
- 关注 issue ，打开“Notifications”

GNU 环境

- POSIX 是 *Portable Operating System Interface* 的缩写，是关于计算机操作系统的国际标准。它规定了操作系统的基本工具和程序接口。
- GNU 是 *GNU is not Unix* 的缩写，是自由软件运动形成的 POSIX 环境。
- 科学研究中时间和空间跨度都很大，需要使用公开的国际标准化接口，才能满足 复现 透明 和 最佳工具 原则。

可使用 GNU 环境的操作系统

GNU/Linux 天然大佬。请帮助周围的同学，并自荐小助教。

Apple macOS 使用 UTM 虚拟机。

- 安装 Debian 12

Microsoft Windows 本身不满足，但可以使用以下扩展

- Windows Subsystem for Linux (WSL)；
- 安装 Debian 12

权威指南：`https://physics-data.meow.plus/faq/`

GNU 环境的基本操作

- ls list structure, 列出当前路径中的文件
- cd change directory, 改变当前路径
- sudo 使用管理员权限执行操作
- apt Advanced Package Tool 软件管理器
- nano 文件编辑器，与 emacs, vi, VSCode 等互换

GNU 环境的基本操作

- ls list structure, 列出当前路径中的文件
- cd change directory, 改变当前路径
- sudo 使用管理员权限执行操作
- apt Advanced Package Tool 软件管理器
- nano 文件编辑器，与 emacs, vi, VSCode 等互换

GNU 环境的基本操作

- ls list structure, 列出当前路径中的文件
- cd change directory, 改变当前路径
- sudo 使用管理员权限执行操作
- apt Advanced Package Tool 软件管理器
  - apt update
  - apt install [package]
  - apt search [keyword]
- nano 文件编辑器，与 emacs, vi, VSCode 等互换
  - ctrl + o 保存
  - ctrl + x 退出
  - ctrl + w 搜索

GNU 环境的基本操作

- **ls** list structure, 列出当前路径中的文件
- **cd** change directory, 改变当前路径
- **sudo** 使用管理员权限执行操作
- **apt** Advanced Package Tool 软件管理器
  - `apt update`
  - `apt install nano diffutils patch`
- **nano** 文件编辑器，与 emacs, vi, VSCode 等互换

GNU 环境的基本操作

- ls list structure, 列出当前路径中的文件
- cd change directory, 改变当前路径
- sudo 使用管理员权限执行操作
- apt Advanced Package Tool 软件管理器
  - `apt update`
  - `apt install nano diffutils patch`
- nano 文件编辑器，与 emacs, vi, VSCode 等互换

GNU 环境的基本操作

- **ls** list structure, 列出当前路径中的文件
- **cd** change directory, 改变当前路径
- **sudo** 使用管理员权限执行操作
- **apt** Advanced Package Tool 软件管理器
  - `apt update`
  - `apt install nano diffutils patch`
- **nano** 文件编辑器，与 emacs, vi, VSCode 等互换
  - 查找 ^W 、替换 ^R 、保存 ^O 、退出 ^X
    注：^W 代表按住 Ctrl 键，再按 W。

GNU 环境的基本操作

- **ls** list structure, 列出当前路径中的文件
- **cd** change directory, 改变当前路径
- **sudo** 使用管理员权限执行操作
- **apt** Advanced Package Tool 软件管理器
  - `apt update`
  - `apt install nano diffutils patch`
- **nano** 文件编辑器，与 emacs, vi, VSCode 等互换
  - 查找 `^W` 、替换 `^R` 、保存 `^O` 、退出 `^X`
    注：`^W` 代表按住 Ctrl 键，再按 W。

GNU 环境的基本操作

- **ls** list structure, 列出当前路径中的文件
- **cd** change directory, 改变当前路径
- **sudo** 使用管理员权限执行操作
- **apt** Advanced Package Tool 软件管理器
  - `apt update`
  - `apt install nano diffutils patch`
- **nano** 文件编辑器，与 emacs, vi, VSCode 等互换
  - 查找 `^W` 、替换 `^R` 、保存 `^O` 、退出 `^X`
    注：`^W` 代表按住 Ctrl 键，再按 W。

命令参数标准

- GNU 标准定义了推荐的参数格式
- `git config --global user.name "Benda Xu"`
- 命令的各个部分由 半角 空格分隔
  - 不作为分隔符的空格由 半角 引号标出
- 短参数：一条短线接一个字母
  - `ls -a`
  - `kill -9 xxxx`
  - `sudo -s`
  - 可以缩写在一起
    - `ls -lrta`
    - `sudo -sE`
- 长参数：两条短线接一个单词
  - `ls --color=tty`
  - `git show --pretty=short --show-signature`
  - 由 GNU 推荐，是在 POSIX 标准之上的扩展
- 更多参数，使用 `man` 查看文档 `man ls`, `man git`

学术共同体意识

- 我们在一个月的时间里，共同理解物理和逻辑，沉淀下来我们的探索过程

网站的使用

- 收藏夹

邮件

- 客户端

编辑器

程序编辑器三大流派

- Emacs
- Vi
- Visual Studio Code

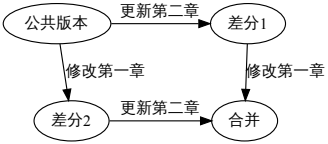
版本控制

- “我和同学合写文章，结果他是在昨天的版本上改的！我还得重改！”

发展历史

石器时代 实验报告-v1, 实验报告-v2, 实验报告-v2.2, 实验报告-v2.2-续本达更新 20190629 .....

青铜时代 diff, patch 文本差分算法



铁器时代 版本控制服务：CVS，SVN

现代 分布式版本控制：Git

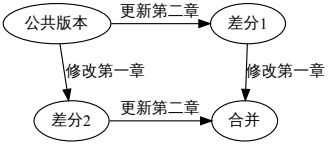
版本控制

- “我和同学合写文章，结果他是在昨天的版本上改的！我还得重改！”

发展历史

石器时代 实验报告-v1, 实验报告-v2, 实验报告-v2.2, 实验报告-v2.2-续本达更新 20190629 .....

青铜时代 diff, patch 文本差分算法



铁器时代 版本控制服务：CVS，SVN

现代 分布式版本控制：Git

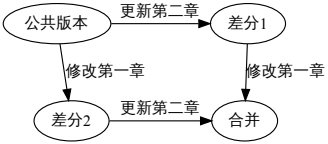
版本控制

- “我和同学合写文章，结果他是在昨天的版本上改的！我还得重改！”

发展历史

石器时代 实验报告-v1, 实验报告-v2, 实验报告-v2.2, 实验报告-v2.2-续本达更新 20190629 .....

青铜时代 diff, patch 文本差分算法



铁器时代 版本控制服务：CVS，SVN

现代 分布式版本控制：Git

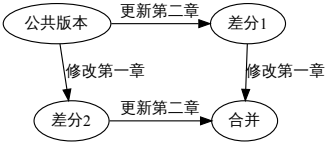
版本控制

- “我和同学合写文章，结果他是在昨天的版本上改的！我还得重改！”

发展历史

石器时代 实验报告-v1, 实验报告-v2, 实验报告-v2.2, 实验报告-v2.2-续本达更新 20190629 .....

青铜时代 diff, patch 文本差分算法



铁器时代 版本控制服务：CVS，SVN

现代 分布式版本控制：Git

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

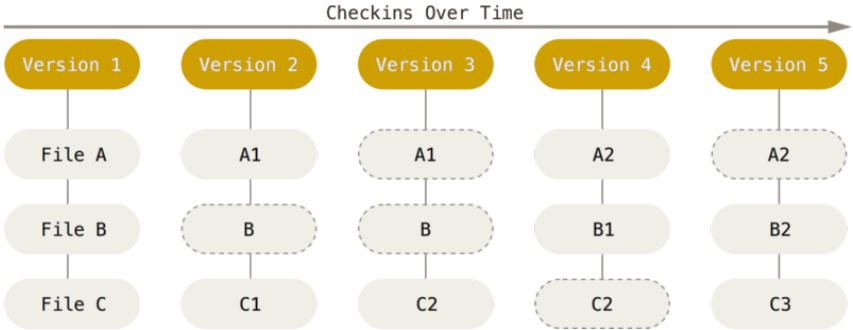
Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup，解法-20190608？重复存储，一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决？

Git 基础 安装：apt install git

- 使用 Git，养成良好的习惯，就能解决以上问题。
- Git 由 Linus Torvalds 发明，用来支撑全世界 5000 名以上 Linux 操作系统内核开发者的协作。是我们的 最佳工具 。



- 在任意时间我们都可以选择给工作区进行一次“快照”。
- 相邻快照之间，是代表文件改动的差分量。

例子：撰写讲义

```
2019-06-29 17:32 Jiajie Chen o [master] [o
2019-06-29 16:51 Jiajie Chen o Fix Mac OS
2019-06-29 15:54 Benda Xu    o Python 和 g
2019-06-29 15:15 Benda Xu    o update team
2019-06-29 15:11 Benda Xu    o 可预见的坑
2019-06-29 15:08 Benda Xu    o update team
2019-06-29 12:50 Benda Xu    o Round 1 sli
2019-06-28 22:27 Benda Xu    o staged slid
2019-06-28 21:54 Benda Xu    o 大作业更新
2019-06-28 21:51 Benda Xu    o start to wr
2019-06-26 16:20 Benda Xu    o 暂不宜注明
2019-06-25 20:24 Fugoes      o fix typos
2019-06-25 20:10 Benda Xu    o 周二周简要
2019-06-25 19:55 Benda Xu    o 第一周大纲
2019-06-25 17:26 Chen        o Update pd.m
2019-06-25 16:48 Benda Xu    o 大作业调整
2019-06-24 15:56 Benda Xu    o 命令行，淡
2019-06-24 15:37 Benda Xu    o 平时作业也
2019-06-24 15:33 Benda Xu    o Add a patch
[main] ff1c4ba8a6a1afdd3952bf6ccdb78b0535%
```

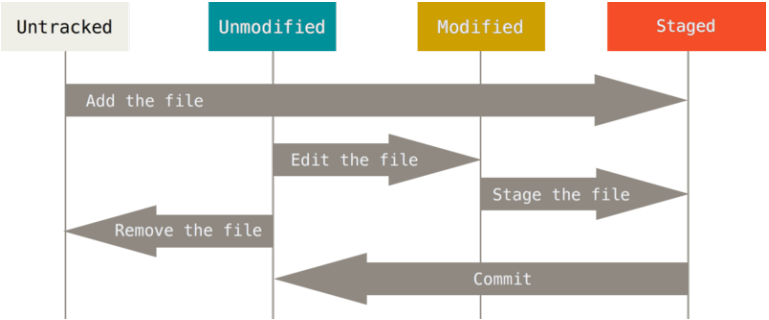
```
commit ff1c4ba8a6a1afdd3952bf6ccdb78b050d
Author:     Benda Xu <heroxbd@gentoo.org>
AuthorDate: Fri Jun 28 21:54:24 2019 +080
Commit:     Benda Xu <heroxbd@gentoo.org>
CommitDate: Fri Jun 28 21:54:24 2019 +080

    大作业更新

---
 pd.md | 9 +++++++--
 1 file changed, 7 insertions(+), 2 delet

diff --git a/pd.md b/pd.md
index 7ce81ta..282fcd8 100644
--- a/pd.md
+++ b/pd.md
@@ -40,8 +40,13 @@
 * https://ghost-hunter.net9.org
 * 课赛结合，大作业备选
[diff] ff1c4ba8a6a1afdd3952bf6ccdb78b061
```

```
0$ zsh  1$ zsh  2-$ zsh  4$* zsh                    proton  0.54 0.52 0.55  21:48
[ proton ]          [ sh ] 2:ssh  4:ssh  6:zsh  7:man- 8:zsh  9:zsh          [ 21:48 ]
```

- 查看每次提交的改动。工具 `tig`，可由 `apt install tig` 安装。

Git 三种状态

已提交（committed）改动安全保存在数据库中
已修改（modified）自上一次提交，有文件被修改
已暂存（staged）已修改的文件被作了标记，将被包含在下一轮提交中



作业：Git 练习

- ☒ DONE 你会在 清华 GitLab 收到一个新仓库，是今天的作业
- ☒ DONE 设置本地到 清华 GitLab 的访问权限
  - ☒ 生成 SSH 密钥对
  - ☒ 上传到 `https://git.tsinghua.edu.cn`
- ☒ DONE 同步你的 Git 仓库，输入你的基本信息

  我的姓名：
  我的学号：
  我的 Python 版本信息：

  课程感言：
- ☐ TODO 查看修改 (diff)，预估得分
- ☐ TODO 添加修改 (add)，提交 (commit)，推送 (push)

Git 基本命令

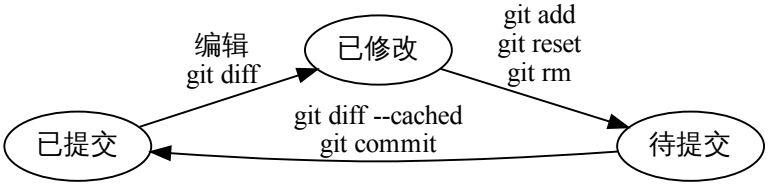
diff	查看改动
status	查看状态
add	添加文件
commit	提交
log	查看历史
pull	从远程下载
push	推送到远程
merge	合并分支

内建手册 man

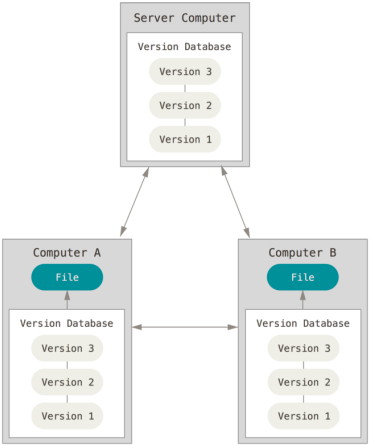
命令的具体含义可以参考 man 命令，例如，

```
$ man git add
```

Git 的状态与命令


已修改
已提交
待提交
编辑
git diff
git add
git reset
git rm
git diff --cached
git commit

Git 仓库间通信



- 建议：通过 SSH (secure shell) 通信。

SSH 通信

- 通信的认证问题：如何证明我是“我”？
  - 手机验证码
  - 一次性密码（one-time password，OTP）
  - 用户名密码
  - 非对称加密（银行 U 盾）
- 生成 SSH 非对称密钥对
  - 一个公钥：公布，`id_rsa.pub`
  - 一个私钥：私密，`id_rsa`
  - 在网页界面把 `id_rsa.pub` 交给 `git.tsinghua.edu.cn`
  - `git.tsinghua.edu.cn` 用我的公钥加密挑战码，只有我的私钥才能解密
    - `git.tsinghua.edu.cn` 知道了我是“我”
    - 我也需要知道它是真的 `git.tsinghua.edu.cn`
  - 建立了与 `git.tsinghua.edu.cn` 的可靠通信

SSH 通信

- 通信的认证问题：如何证明我是“我”？
  - 手机验证码
  - 一次性密码（one-time password，OTP）
  - 用户名密码
  - 非对称加密（银行 U 盾）
- 生成 SSH 非对称密钥对
  - 一个公钥：公布，`id_rsa.pub`
  - 一个私钥：私密，`id_rsa`
  - 在网页界面把 `id_rsa.pub` 交给 `git.tsinghua.edu.cn`
  - `git.tsinghua.edu.cn` 用我的公钥加密挑战码，只有我的私钥才能解密
    - `git.tsinghua.edu.cn` 知道了我是“我”
    - 我也需要知道它是真的 `git.tsinghua.edu.cn`
  - 建立了与 `git.tsinghua.edu.cn` 的可靠通信

提交说明 commit message

- `git commit` 时需要输入一段话概括本次 commit 所作的修改
- 目标：让他人和未来的自己一目了然，只通过读此信息就知道自己做了什么

commit 样例

- 不错的例子：“完善了个人信息页面，并填写了课程感言”
- 短的 ，把事情讲清楚，把涉及到的人讲清楚。
- 较长的 ，把所有的改动概括清楚。
- 更长的 ，在 commit 里说明为什么要这样做，引用相关的讨论。

反例

1. 数数：1,2,3
2. 复读：Update,Update,Update
3. 欺负键盘：aaaaa,wwwww,asdf

- 不太好的例子：“完成作业”

精心撰写提交说明

- 说明既要给人类读，也要易于自动处理
- 第一行：高度概括改动
- 空一行后，撰写具体改动内容
  - 形式不限，可分段、分要点。
  - 篇幅不限，以描述清楚为目标。
- 备注：感谢其他贡献者。

何时提交 commit ？

- 完成一件事，有一个较独立的“版本”：添加一个功能，修复一个问题，重构一块代码。
- 初学者可尝试多提交，可以充分练习 Git 操作，也可以获得更细致的开发历史。
  - 不要 把多个不相关的修改攒成一次巨大的提交。

来自学长的忠告

作业得分不重要，重要的是优雅地得分

我看到的无意义 commit message 有：“这是一个描述”，“xX”，“1234567899”等等。由于 Git 的特性，后续的提交没有办法覆盖这些无意义的 commit ，所以这个操作在很大程度上是不可逆的。也就是说，批阅作业的助教可以看到这些无意义的提交信息，然后或许会给你的白盒扣上 2 分。虽然我相信批改作业的助教不会这么残忍，但请一定要有“提交信息要有意义”的意识。

请查看白盒标准：

`https://physics-data.meow.plus/faq/rules/whitebox/`

- 如果想要自己玩一下 Git 的相关操作（强烈推荐这么做，尤其是要学会 add 和 commit 如何取消。我个人对于一个操作是否可逆是有追求的），可以自己在 GitLab 上面创建一个项目练习。

Git 与编辑器的整合

禁止使用 Gitlab 网页上传和修改工具

- 远远次于命令行的功能，无推广价值
- 禁止使用，防止养成不良习惯

在命令行调用相应的编辑器

- `git` 会调用 环境设定 的编辑器。临时调整：使用 `EDITOR=xxxxx` 前缀
  - `EDITOR="code --wait"` 见 李禹锋的 issue 88
  - `EDITOR="vim"`
  - `EDITOR="emacsclient"` 见 黄宇同的 issue 109
- Debian 上永久调整系统的默认编辑器，管理员权限选择需要的编辑器。

```
update-alternatives --config editor
```

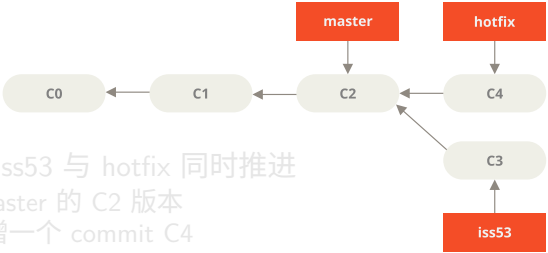
- 配置 git 的编辑器：查看文档配置，关键字是 `core.editor`

```
man git config
```

Git 分支

参考 `https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging`

- 一直往 master (或 main) 提交：线性历史
- `git branch iss53` 创建新分支，名为 iss53
- `git switch iss53` 切换至 iss53 分支
- `git switch --create hotfix` 创建新分支 hotfix 并切换

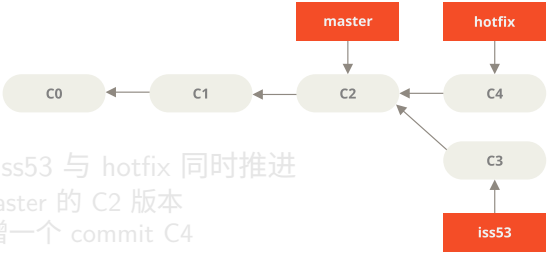


- 从 C2 开始，iss53 与 hotfix 同时推进
  - 都基于 master 的 C2 版本
  - hotfix 新增一个 commit C4
  - iss53 新增 C3
- 将 iss53 push 到远程: `git push --set-upstream origin iss53`

Git 分支

参考 https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

- 一直往 master (或 main) 提交：线性历史
- `git branch iss53` 创建新分支，名为 iss53
- `git switch iss53` 切换至 iss53 分支
- `git switch --create hotfix` 创建新分支 hotfix 并切换

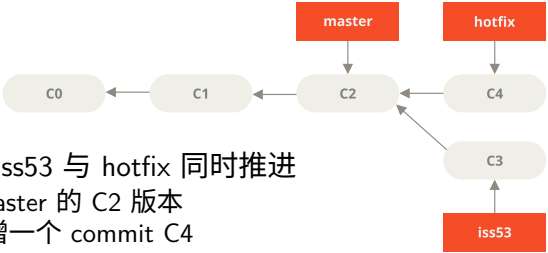


- 从 C2 开始，iss53 与 hotfix 同时推进
  - 都基于 master 的 C2 版本
  - hotfix 新增一个 commit C4
  - iss53 新增 C3
- 将 iss53 push 到远程: `git push --set-upstream origin iss53`

Git 分支

参考 https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

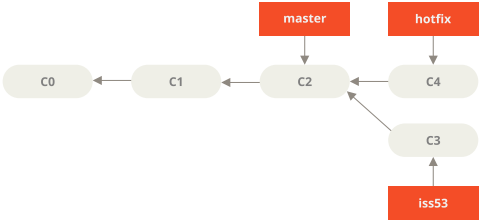
- 一直往 master (或 main) 提交：线性历史
- `git branch iss53` 创建新分支，名为 iss53
- `git switch iss53` 切换至 iss53 分支
- `git switch --create hotfix` 创建新分支 hotfix 并切换

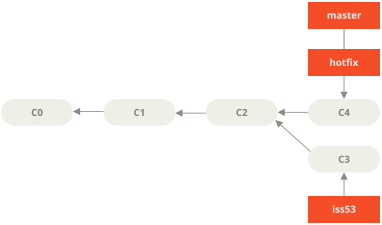


- 从 C2 开始，iss53 与 hotfix 同时推进
  - 都基于 master 的 C2 版本
  - hotfix 新增一个 commit C4
  - iss53 新增 C3
- 将 iss53 push 到远程: `git push --set-upstream origin iss53`

Git 分支的结果：分叉

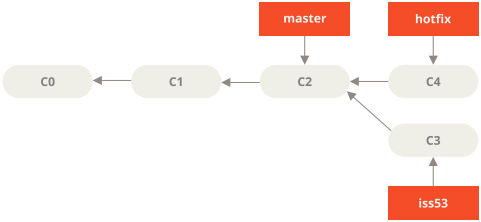
- 由于 hotfix 紧迫，hotfix 率先被 master 接受
- `git merge --ff hotfix` 快进合并 (Fast-Forward)，拿来 原始 的 hotfix commit (相当于直接在 master 上改动)

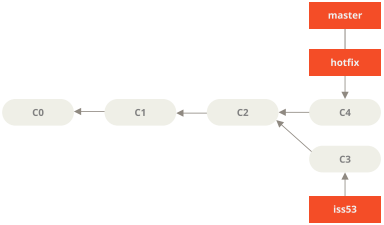




Git 分支的结果：分叉

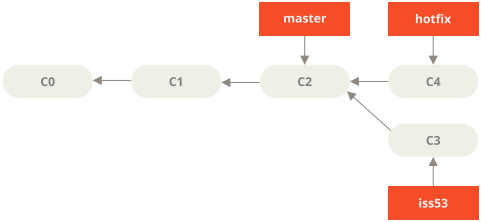
- 由于 hotfix 紧迫，hotfix 率先被 master 接受
- `git merge --ff hotfix` 快进合并 (Fast-Forward), 拿来 原始 的 hotfix commit (相当于直接在 master 上改动)

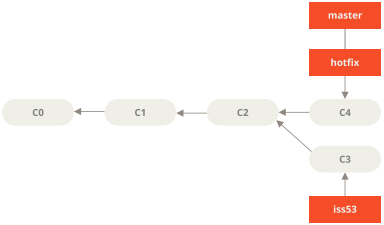




Git 分支的结果：分叉

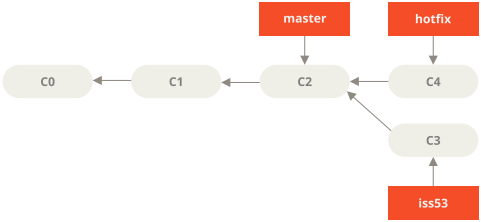
- 由于 hotfix 紧迫，hotfix 率先被 master 接受
- `git merge --ff hotfix` 快进合并 (Fast-Forward), 拿来 原始 的 hotfix commit (相当于直接在 master 上改动)

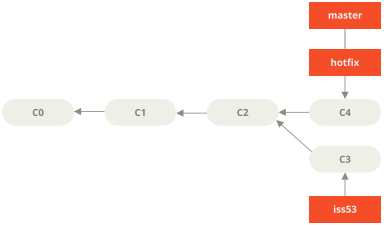




Git 分支的结果：分叉

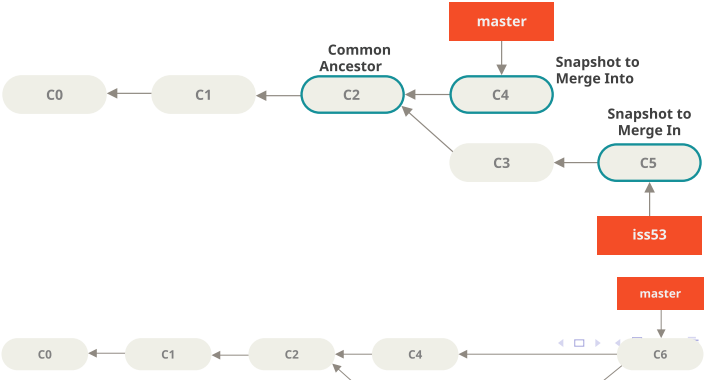
- 由于 hotfix 紧迫，hotfix 率先被 master 接受
- `git merge --ff hotfix` 快进合并 (Fast-Forward), 拿来 原始 的 hotfix commit (相当于直接在 master 上改动)





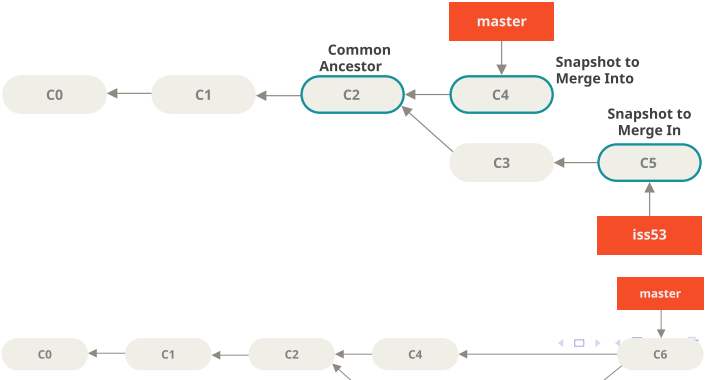
Git merge：合并 (狭义) I

- 出现分叉需要合并，回忆平行四边形法则
- 比较 master 的最新版 (C4) 和 iss53 最新版 (C5) 的差异，取两家之长，处理一次冲突
- 在 master 上 `git merge iss53`
- 生成一个 Merge commit C6，历史不再线性。
- master 的开发者负责解决冲突



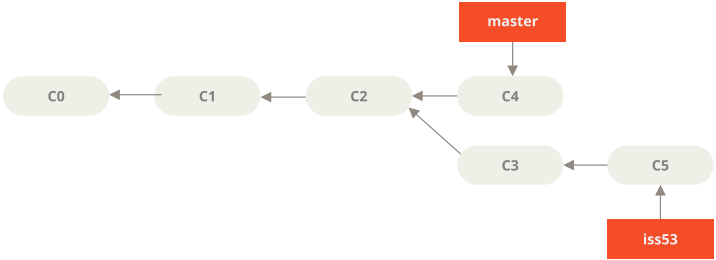
Git merge：合并 (狭义) I

- 出现分叉需要合并，回忆平行四边形法则
- 比较 master 的最新版 (C4) 和 iss53 最新版 (C5) 的差异，取两家之长，处理一次冲突
- 在 master 上 `git merge iss53`
- 生成一个 Merge commit C6，历史不再线性。
- master 的开发者负责解决冲突



Git rebase：换祖先

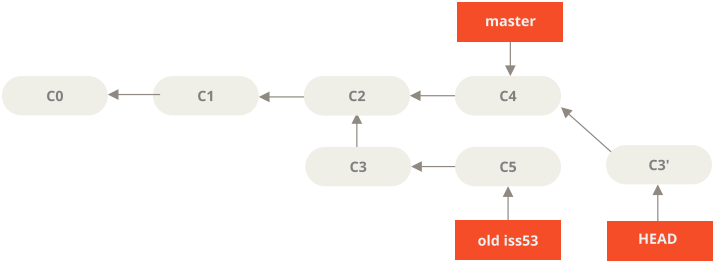
- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发



- 在 iss53 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 iss53 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突（C4 vs C3 & C3' vs C5'），但在 iss53

Git rebase：换祖先

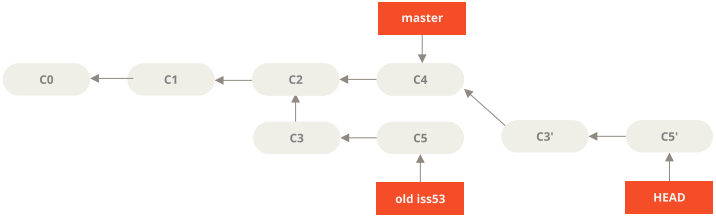
- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发



- 在 `iss53` 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 `iss53` 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突（C4 vs C3 & C3' vs C5'），但在 iss53

Git rebase：换祖先

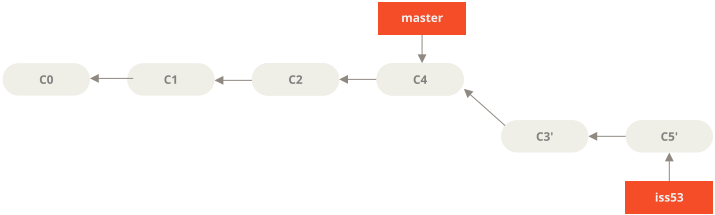
- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发



- 在 `iss53` 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 `iss53` 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突 (C4 vs C3 & C3' vs C5')，但在 iss53 分支上操作

Git rebase：换祖先

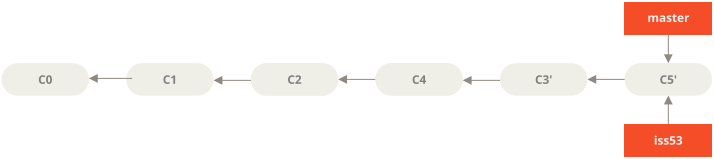
- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发



- 在 `iss53` 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 `iss53` 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突 (C4 vs C3 & C3' vs C5')，但在 iss53 分支上操作

Git rebase：换祖先

- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发



- 在 `iss53` 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 `iss53` 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突 (C4 vs C3 & C3' vs C5')，但在 iss53 分支上操作
- `git rebase -i master` 还能允许你进一步修改历史，例如合并 commit、打磨 commit

Git rebase：换祖先

- 思考：让 iss53 的祖先从 C2 切换到 C4 不就没有分叉了吗？基于新版的 C4 重新做一遍 iss53 的开发

- 在 `iss53` 分支 `git rebase C4` 或 `git rebase master`
- 基于 C4，将 C3 稍作修改变成 C3'
- 基于 C3'，将 C5 稍作修改变成 C5'
- rebase 修改历史！结束后旧的 `iss53` 历史消失，不要在 `master` rebase！
- `git switch master; git merge --ff iss53` 快进合并。最终合并出 线性 历史
- 需要处理 iss53 上所有 commit 的冲突 (C4 vs C3 & C3' vs C5')，但在 iss53 分支上操作
- `git rebase -i master` 还能允许你进一步修改历史，例如合并 commit、打磨 commit

Git 自动冲突处理 I

- 如果 C3、C5 与 C4 不干涉 (修改的不是一个文件 or 修改的位置距离较远) 则程序可以自动处理

C4: 补全最后一句的上文 (hotfix):

```
 今日中国，强敵环伺，科技卡膀，海疆未靖，幼苗未长，此誠危急存亡之秋也。
 告诉学生們，去修 birkar 的課，交論文，得分最高的，F一个华为手表。
 概率統計，前十名，北京大學F八名，清華大學F有
 做學問一定要在有學問的朋友氛圍中，時時砥F成長。
 我到落杉机，不熟地方
-我宣布他已經不是我的學生了
+已經到了無F的地步。我宣布他已經不是我的學生了
```

Git 自动冲突处理 II

- 如何把 C3 修改成 C3'？回忆平行四边形法则
- 如果 C3、C5 与 C4 不干涉 (修改的不是一个文件 or 修改的位置距离较远) 则程序可以自动处理

C3: 加入“躲在遊戲屋”语录 (iss53):

```
 今日中国，强敵环伺，科技卡膀，海疆未靖，幼苗未长，此誠危急存亡之秋也。
+我們学生躲在Ⓕ戲屋，不觉得羞愧？
 告诉学生們，去修 birkar 的課，交論文，得分最高的，Ⓕ一个华为手表。
 概率統計，前十名，北京大學Ⓕ八名，清華大學Ⓕ有
 做學問一定要在有學問的朋友氛圍中，時時砥Ⓕ成長。
 我到落杉机，不熟地方
 我宣布他已經不是我的學生了
```

Git 自动冲突处理 III

- 如何把 C3 修改成 C3'？回忆平行四边形法则
- 如果 C3、C5 与 C4 不干涉 (修改的不是一个文件 or 修改的位置距离较远) 则程序可以自动处理

C3' (rebase C3 upon C4):

```
 今日中国，强敵环伺，科技卡膀，海疆未靖，幼苗未长，此誠危急存亡之秋也。
+我們学生躲在F戲屋，不觉得羞愧？
 告诉学生們，去修 birkar 的課，交論文，得分最高的，F一个华为手表。
 概率統計，前十名，北京大學F八名，清華大學F有
 做學問一定要在有學問的朋友氛圍中，時時砥F成長。
 我到落杉机，不熟地方
 我宣布他已經不是我的學生了
```

Git 手动冲突处理 I

C5: 加一句鼓励的话 (iss53)

```
 概率統計，前十名，北京大學F八名，清華大學F有
 做學問一定要在有學問的朋友氛圍中，時時砥F成長。
 我到落杉机，不熟地方
-我宣布他已經不是我的學生了
+你们都比我聪明，只是妄自菲薄而已!
+我宣布他已經不是我的學生了!
```

`git switch iss53; git rebase master`

```
Auto-merging record.txt
CONFLICT (content): Merge conflict in record.txt
error: could not apply 262ef6c... C5: 加一句鼓励的话
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abo
Could not apply 262ef6c... C5: 加一句鼓励的话
```

Git 手动冲突处理 II

```
Unmerged paths:
(use "git restore --staged <file>..." to unstage)
(use "git add <file>..." to mark resolution)
        both modified:   record.txt
```

record.txt:

```
<<<<<<< HEAD
已經到了無Ⓕ的地步。我宣布他已經不是我的學生了
=======
你们都比我聪明，只是妄自菲薄而已!
我宣布他已經不是我的學生了!
>>>>>>> 262ef6c (C5: 加一句鼓励的话)
```

需要揣摩两个分支作者的意图，最终合并成一个合理的版本：

```
你们都比我聪明，只是妄自菲薄而已!
已經到了無Ⓕ的地步。我宣布他已經不是我的學生了!
```

Git 手动冲突处理 III

- 处理后 `git add record.txt; git rebase --continue`

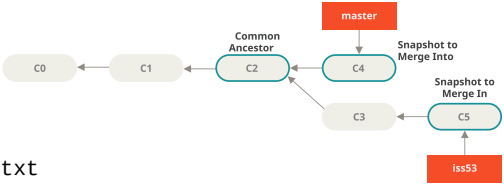
C5' (rebase C5 upon C3'):

```
 概率統計，前十名，北京大學Ⓕ八名，清華大學Ⓕ有
 做學問一定要在有學問的朋友氛圍中，時時砥Ⓕ成長。
 我到落杉机，不熟地方
-我宣布他已經不是我的學生了
+你们都比我聪明，只是妄自菲薄而已！
+我宣布他已經不是我的學生了！
```

- 有多少 commit 需要 rebase (`iss53` 分支有多少 commit) 就需要处理多少次 (自动 or 手动)

Git merge 冲突处理与结果 I

- 回到分叉状态，在 master (C4) 上 `git merge iss53`



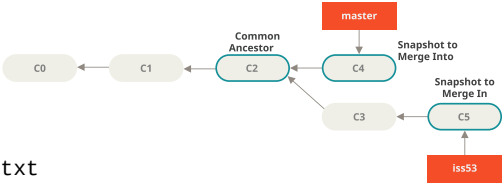

```
Auto-merging record.txt
CONFLICT (content): Merge conflict in record.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<<< HEAD
已經到了無F的地步。我宣布他已經不是我的學生了
=======
你们都比我聪明，只是妄自菲薄而已!
我宣布他已經不是我的學生了!
>>>>>>> iss53
```

- 解决好冲突后 `git merge --continue` 生成 merge commit C6
- Merge commit 有两个祖先，单个差分 (`git log -p`) 无法表达

Git merge 冲突处理与结果 I

- 回到分叉状态，在 master (C4) 上 `git merge iss53`



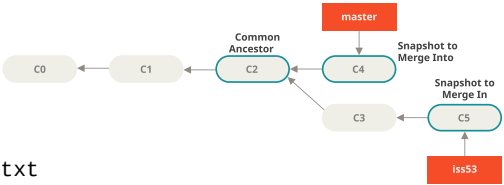

```
Auto-merging record.txt
CONFLICT (content): Merge conflict in record.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<<< HEAD
已經到了無F的地步。我宣布他已經不是我的學生了
=======
你们都比我聪明，只是妄自菲薄而已!
我宣布他已經不是我的學生了!
>>>>>>> iss53
```

- 解决好冲突后 `git merge --continue` 生成 merge commit C6
- Merge commit 有两个祖先，单个差分 (`git log -p`) 无法表达

Git merge 冲突处理与结果 I

- 回到分叉状态，在 master (C4) 上 `git merge iss53`




```
Auto-merging record.txt
CONFLICT (content): Merge conflict in record.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<<< HEAD
已經到了無F的地步。我宣布他已經不是我的學生了
=======
你们都比我聪明，只是妄自菲薄而已!
我宣布他已經不是我的學生了!
>>>>>>> iss53
```

- 解决好冲突后 `git merge --continue` 生成 merge commit C6
- Merge commit 有两个祖先，单个差分 (`git log -p`) 无法表达

Git merge 冲突处理与结果 II

使用 `git log --diff-merges=combined` 查看 merge commit 相对两个祖先的差分

```
commit 3773ad5beca2111981908d010e7356fa5d381a67 (HEAD -> main)
Merge: 37bb2ab 5f9c706
Author: Yiyang Wu <xgreenlandforwyy@gmail.com>
Date:   Fri Jan 31 19:22:46 2025 +0800

    Merge branch 'iss53'

diff --cc record.txt
index 046a6d4,598b092..219a08a
--- a/record.txt
+++ b/record.txt
@@@ -1,6 -1,8 +1,9 @@@
  今日中国，强敌环伺，科技卡脖，海疆未靖，幼苗未长，此誠危急存亡之秋也。
+ 我們学生躲在Ⓔ戴屋，不觉得羞愧？
  告诉学生們，去修 birkar 的課，交論文，得分最高的，Ⓕ一个华为手表。
  概率統計，前十名，北京大學Ⓔ八名，清華大學Ⓕ有
++這種成績，使人汗Ⓕ！如此成績，如何招生？
  做學問一定要在有學問的朋友氛圍中，時時砥Ⓕ成長。
  我到落杉机，不熟地方
-  已經到了無Ⓕ的地步。我宣布他已經不是我的學生了
+ 你们都比我聪明，只是妄自菲薄而已！
 -我宣布他已經不是我的學生了！
++已經到了無Ⓕ的地步。我宣布他已經不是我的學生了！
```

彩蛋：这个 merge commit 塞了私货

权衡 merge vs rebase

参考 `https://git-scm.com/book/en/v2/Git-Branching-Rebasing`

Rebase vs. Merge

- `git merge` 只处理一次 (大) 冲突，产生非线性历史
  - 遮住其他分支后 merge commit 视作单个祖先，获得简单的线性历史，可以排除低质量 commit 干扰 bisect
  - 难理清 commits 的逻辑顺序
  - revert 更困难
  - 不遮住分支 bisect 困难
- `git rebase` 处理多次 (小) 冲突，产生线性历史：
  - 所有 commits 逻辑顺序清晰，revert 容易
  - bisect 可以检查所有合并进来的 commits
  - 所有 commits 在一个历史中，低质量 commits 会干扰 bisect
- 判据: master 的喜好；分支的所有 commits 是否达到和 master 一样的水准
  - 分支充斥半成品 commits、revert(s)，只有最终 commit 达到了 master 水准: merge
  - 分支 commits 质量和 master 一致，rebase + 快进 merge
  - 分支充斥半成品但 master 不愿 merge：要求分支作者在 rebase 时打磨干净，提升水准: rebase + 快进 merge

权衡 merge vs rebase

参考 `https://git-scm.com/book/en/v2/Git-Branching-Rebasing`

Rebase vs. Merge

- `git merge` 只处理一次 (大) 冲突，产生非线性历史
  - 遮住其他分支后 merge commit 视作单个祖先，获得简单的线性历史，可以排除低质量 commit 干扰 bisect
  - 难理清 commits 的逻辑顺序
  - revert 更困难
  - 不遮住分支 bisect 困难
- `git rebase` 处理多次 (小) 冲突，产生线性历史：
  - 所有 commits 逻辑顺序清晰，revert 容易
  - bisect 可以检查所有合并进来的 commits
  - 所有 commits 在一个历史中，低质量 commits 会干扰 bisect
- 判据: master 的喜好；分支的所有 commits 是否达到和 master 一样的水准
  - 分支充斥半成品 commits、revert(s)，只有最终 commit 达到了 master 水准: merge
  - 分支 commits 质量和 master 一致，rebase + 快进 merge
  - 分支充斥半成品但 master 不愿 merge：要求分支作者在 rebase 时打磨干净，提升水准: rebase + 快进 merge

权衡 merge vs rebase

参考 `https://git-scm.com/book/en/v2/Git-Branching-Rebasing`

Rebase vs. Merge

- `git merge` 只处理一次 (大) 冲突，产生非线性历史
  - 遮住其他分支后 merge commit 视作单个祖先，获得简单的线性历史，可以排除低质量 commit 干扰 bisect
  - 难理清 commits 的逻辑顺序
  - revert 更困难
  - 不遮住分支 bisect 困难
- `git rebase` 处理多次 (小) 冲突，产生线性历史：
  - 所有 commits 逻辑顺序清晰，revert 容易
  - bisect 可以检查所有合并进来的 commits
  - 所有 commits 在一个历史中，低质量 commits 会干扰 bisect
- 判据: master 的喜好；分支的所有 commits 是否达到和 master 一样的水准
  - 分支充斥半成品 commits、revert(s)，只有最终 commit 达到了 master 水准: merge
  - 分支 commits 质量和 master 一致，rebase + 快进 merge
  - 分支充斥半成品但 master 不愿 merge：要求分支作者在 rebase 时打磨干净，提升水准: rebase + 快进 merge

权衡 merge vs rebase

参考 `https://git-scm.com/book/en/v2/Git-Branching-Rebasing`

Rebase vs. Merge

- `git merge` 只处理一次 (大) 冲突，产生非线性历史
  - 遮住其他分支后 merge commit 视作单个祖先，获得简单的线性历史，可以排除低质量 commit 干扰 bisect
  - 难理清 commits 的逻辑顺序
  - revert 更困难
  - 不遮住分支 bisect 困难
- `git rebase` 处理多次 (小) 冲突，产生线性历史：
  - 所有 commits 逻辑顺序清晰，revert 容易
  - bisect 可以检查所有合并进来的 commits
  - 所有 commits 在一个历史中，低质量 commits 会干扰 bisect
- 判据: master 的喜好；分支的所有 commits 是否达到和 master 一样的水准
  - 分支充斥半成品 commits、revert(s)，只有最终 commit 达到了 master 水准: merge
  - 分支 commits 质量和 master 一致，rebase + 快进 merge
  - 分支充斥半成品但 master 不愿 merge：要求分支作者在 rebase 时打磨干净，提升水准: rebase + 快进 merge

权衡 merge vs rebase

参考 `https://git-scm.com/book/en/v2/Git-Branching-Rebasing`

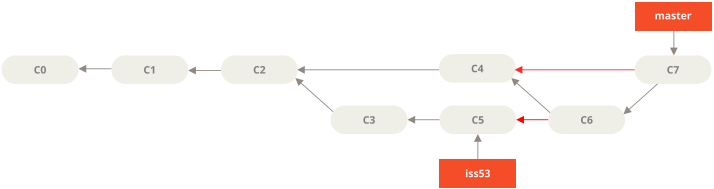
Rebase vs. Merge

- `git merge` 只处理一次 (大) 冲突，产生非线性历史
  - 遮住其他分支后 merge commit 视作单个祖先，获得简单的线性历史，可以排除低质量 commit 干扰 bisect
  - 难理清 commits 的逻辑顺序
  - revert 更困难
  - 不遮住分支 bisect 困难
- `git rebase` 处理多次 (小) 冲突，产生线性历史：
  - 所有 commits 逻辑顺序清晰，revert 容易
  - bisect 可以检查所有合并进来的 commits
  - 所有 commits 在一个历史中，低质量 commits 会干扰 bisect
- 判据: master 的喜好；分支的所有 commits 是否达到和 master 一样的水准
  - 分支充斥半成品 commits、revert(s)，只有最终 commit 达到了 master 水准: merge
  - 分支 commits 质量和 master 一致，rebase + 快进 merge
  - 分支充斥半成品但 master 不愿 merge：要求分支作者在 rebase 时打磨干净，提升水准: rebase + 快进 merge

Git merge 但由待合并分支处理冲突

`git merge` 由 master 处理冲突并敲定 merge commit。如果 master 可能要求 `iss53` 处理冲突?

- 可以 rebase，但也许工作量更大
- 可以先 `git merge master` 把冲突解决，再由 master `git merge --no-ff iss53`



- 红色部分表示 merge commit 的默认祖先 (first-parent)
- 如果 `git merge --ff iss53` 则最终 merge commit 是 C6，默认祖先指向 C5 (iss53 分支)，在 revert 和 bisect 时带来困扰

基于 Gitlab merge request (或 Github pull request) 的开发流程 I

gitlab 命令行工具: `apt install glab`, 登录: `glab auth login`

```
# 首先在 Gitlab fork 需要的上游仓库 foo/bar.git 到 my/bar.git
# 之所以 fork 是因为一般人无法往原仓库 push 分支
glab repo fork # 或在网页上点击 fork
git clone git@git.tsinghua.edu.cn:my/bar.git # 拉取 fork 仓库, 进入 master 分支
cd bar
git switch --create my-dev # 取一个合适的分支名字
# 一通编辑代码
git add xxx
git commit
git push --set-upstream origin HEAD # push 自己的分支
glab mr create # 或者根据命令行提示信息, 在网页发起 merge request
# 上游若要求 rebase, 则需要先同步 fork 仓库 my/bar.git 的 master 分支
git switch master
git pull # 同步 master, 产生分叉
git switch my-dev
git rebase master
git push --force # rebase 后破坏了 my-dev 的历史, my/bar.git 的 my-dev 远端需要被覆盖
```

参考资料

- Git 提交说明的白盒采分项：
  https://physics-data.meow.plus/faq/whitebox/#git
- 写好提交说明的七大原则 by cbeams
  https://chris.beams.io/posts/git-commit/
- 提交说明编写指南 by 阮一峰
  http://www.ruanyifeng.com/blog/2016/01/commit_message_change_log.html

有突出贡献的同学可得 8% 左右的总评鼓励

- 课程 FAQ
  - `https://git.tsinghua.edu.cn/physics-data/faq/`
  - 提出好问题，整理问题解答，提交 merge request，……
- 课程讲义
  - 修正错别字、语言错误、常识错误
  - 指出逻辑不明的地方
  - 补充课堂讲授但讲义遗漏的内容
  - ……
- 贡献方式
  - ☒ Git
  - ☒ GitLab issue
  - ☐ GitLab Merge Request

分支

例子

- 关系
- merge request（合并请求）
  - 提交代码审核。

Git 仓库间通信的各阶段

fork Gitlab 类 Git 平台的术语，对应于 `git clone`

- 在本人帐号创建复本，用于自己修改和提交
- 提交说明的备注：关闭哪个 issue

remote Git 的远程仓库

```
git remote -v
origin  git@git.tsinghua.edu.cn:physics-data/lecture.git (fetch)
origin  git@git.tsinghua.edu.cn:physics-data/lecture.git (push)
```

pull, push 从远程下载和上传改动，单元为差分 commit

merge 与其他仓库或分支通过交换差分来整合

merge request GitLab 平台术语，发送 merge 请求。请他人整合自己的差分贡献。