第七讲课前预习

从一条历史到协作交付

迷你 Git · 分支 · 快照 · SSH · 真实 Git 交付

预习目标

看懂知识骨架，准备好电脑环境，带着问题来上课。

讲师：王众一
自动化系科协学创部

建议预习时间：20–30 分钟 · 面向零基础同学

预习导航

建议预习路线

第一遍（10 分钟）： 先看第 2–7 节，重点追踪状态怎样变化； **第二遍（10 分钟）：** 完成第 8–11 节的电脑、账号与资料检查； **最后（5 分钟）：** 用第 12 节自测，把答不清的问题带到课堂。

1 这份预习讲义怎么用

你不需要在课前完成什么

你**不需要**独立实现一个版本控制系统，也不需要提前背下所有 Git 命令。课堂上会用一个教学用的“迷你 Git”观察状态变化，再把同一套思路迁移到真实 Git、SSH 和课程交付中。

1. 用本讲义建立一张知识地图，知道每个术语在整条链路中的位置；
2. 确认电脑能运行 C++ 编译器、Git、SSH 与 PowerShell；
3. 准备好课程资料、平台账号和一个干净的实验目录；
4. 记下暂时不理解的地方，课堂上用实验验证。

1.1 预习完成后的最低标准

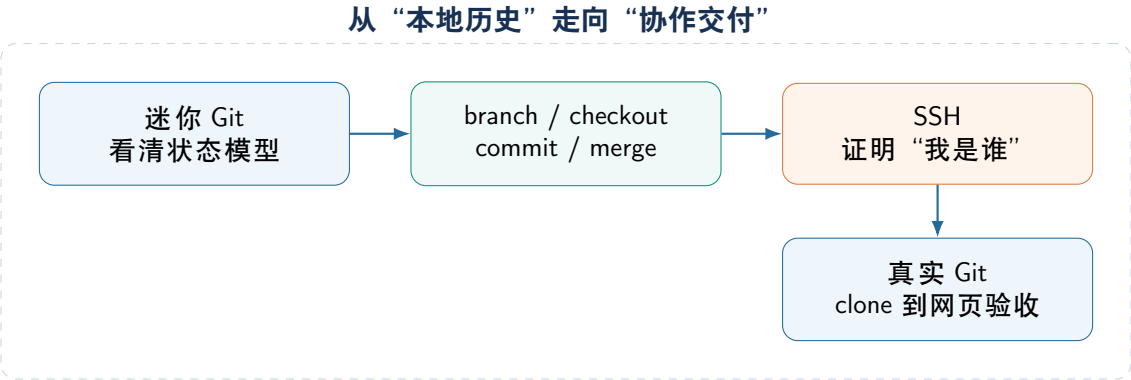
读完后，你应该能用自己的话回答下面三个问题：

- 分支为什么更像“名字”，而不是一整份项目副本？
- 为什么切换分支之前必须先检查工作区是否安全？
- 为什么 `ssh -T` 成功，仍然不等于作业已经提交？

不要把私钥内容发给任何人；不要在不清楚远程分支状态时使用 `force push`；不要在重要项目目录中试验会覆盖文件的命令。

2 先看到整张知识地图

2.1 本讲的一条主线



第一层：数据模型
工作区、暂存区、提交快照、HEAD、分支引用和 parent 链。

第二层：安全操作
每条命令先检查条件，再改变最少的状态，并保护失败前后的不变量。

第三层：身份认证
SSH 用密钥证明身份，但它不替你创建提交，也不替你推送代码。

第四层：交付闭环
clone、修改、add、commit、push，最后回到网页核对结果。

2.2 贯穿全讲的判断方式

面对一条命令，不急着背语法，先问：

1. 它**读取**哪些状态？
2. 它会**修改**哪些状态？
3. 执行前必须满足哪些**安全条件**？
4. 成功后哪些**不变量**必须仍然成立？
5. 失败时，仓库有没有保持原样？

3 迷你 Git 的核心状态模型

3.1 六个必须先认清的对象

工作区 working tree

你眼前正在编辑的文件。它可能与最近一次提交不同。

暂存区 stage

本次准备提交的候选内容。它不是永久历史。

提交 commit

一次完整、只读的项目快照，并记录父提交和说明信息。

HEAD

迷你 Git 中保存“当前分支名”，先指向名字，不直接保存最新编号。

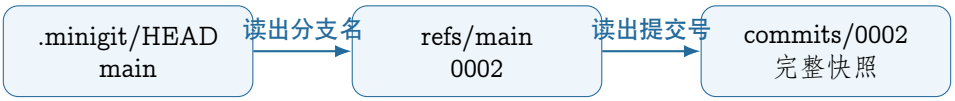
分支引用 ref

一个名字对应一个提交编号，例如 main -> 0002。

parent 链

每个提交指向父提交，连续连接后形成历史关系。

3.2 “当前位置”不是一个值，而是一条解析链



HEAD 指向分支名，分支名指向提交，提交再通过 parent 指向更早的提交。

4 四个操作怎样改变状态

4.1 branch：只增加一个名字

执行前 验证仓库存在、分支名合法、同名分支不存在。
状态变化 新分支 ref 指向当前提交；HEAD 不变；工作区不变。
核心不变量 创建分支不等于切换分支。

创建前：HEAD -> main -> 0002
创建后：HEAD -> main -> 0002，同时新增 feature -> 0002

4.2 checkout：换名字，也恢复文件

执行前 暂存区必须为空；工作区必须与当前提交一致；目标分支存在。
状态变化 先安全恢复目标快照，再让 HEAD 指向目标分支。
核心不变量 被拒绝时，HEAD、分支 ref 和工作区都不能被改到一半。

4.3 commit：产生快照，只推进当前分支

执行前 本次提交内容明确，提交编号来自全局 NEXT_ID。
状态变化 创建完整快照，记录 parent，再移动当前分支 ref。
核心不变量 其他分支不跟着移动；历史关系不能靠编号大小猜测。

4.4 merge：能证明安全时才前进

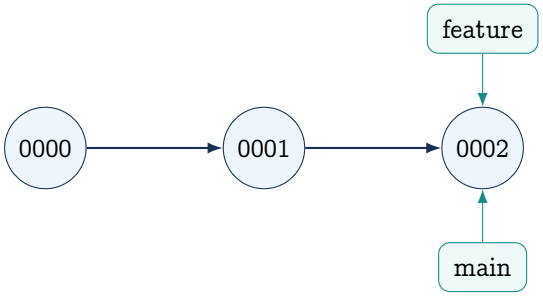
本讲范围 只讨论快进合并 fast-forward。
安全证明 沿目标提交的 parent 链向前走，能找到当前提交。
状态变化 恢复目标快照，并把当前分支 ref 移到目标提交；不创建新提交。

提交编号只表示创建顺序，不能表达分叉后的祖先关系。是否可以快进，必须检查 parent 链。

5 把四个操作串成一段历史

5.1 课堂实验将经历的状态变化

1. 在 main 上产生第一次提交 0001；
2. 创建 feature，它和 main 同时指向 0001；
3. checkout 到 feature，修改文件并提交为 0002；
4. checkout 回 main，工作区恢复为 0001 的快照；
5. 把 feature 快进合并到 main，于是 main -> 0002；
6. 再人为制造分叉，观察快进合并被安全拒绝。



快进完成后，两个名字可以暂时指向同一个提交

5.2 实验中真正要观察的证据

HEAD
当前分支名有没有按预期改变？

refs
哪一个分支移动了，哪一个没有动？

提交快照
切换后工作区内容是否与目标快照一致？

失败现场
命令被拒绝后，原有状态是否完整保留？

学习方式提醒

本讲不是把命令当作“咒语”背下来，而是通过文件内容和状态变化，为每条结论找到可检查的证据。

6 从迷你 Git 迁移到真实 Git

6.1 概念相似，但实现并不相同

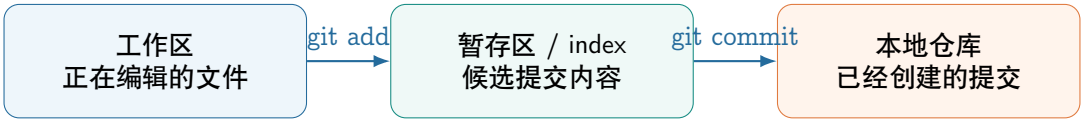
迷你 Git

用直观目录保存 HEAD、refs、stage 和完整快照，方便初学者直接观察。

真实 Git

使用对象数据库、索引、引用和更复杂的合并规则；不能把 .minigit 的布局当作真实 .git。

6.2 真实 Git 的三个本地位置



6.3 再加上远程平台，才构成交付链路

课程平台仓库 $\xrightarrow{\text{git clone}}$ 本地仓库 $\xrightarrow[\text{交付确认}]{\text{编辑、add、commit}}$ 新的本地提交 $\xrightarrow{\text{git push}}$ 远程分支 $\xrightarrow{\text{网页检查}}$

终端显示 push 成功之后，仍要打开网页核对：仓库是否正确、分支是否正确、最新提交是否出现、文件是否齐全。

7 SSH 在协作链路中的位置

7.1 SSH 解决的是身份认证

SSH 密钥对可以粗略理解为：

- **私钥**留在自己的电脑上，用来证明身份，绝不能公开；
- **公钥**可以上传到课程 Git 平台，让平台知道该信任哪把私钥；
- 连接时，平台验证你是否拥有对应私钥，而不是让私钥离开电脑。

文件名以 .pub 结尾的通常是公钥；没有 .pub 的私钥文件不能发到群里、不能粘贴进作业、不能提交到 Git 仓库。

7.2 三件容易混淆的事

SSH 认证成功
说明平台认出了你；不说明你已经 clone、commit 或 push。

Git 提交成功
说明本地历史中出现了提交；不说明远程平台已经收到。

Git 推送成功
说明远程分支被更新；仍应通过网页完成最终验收。

7.3 第一次连接为什么要核对主机指纹

第一次连接新主机时，SSH 会询问是否信任该主机。不要条件反射输入 yes；应先对照教师或平台公布的主机名与指纹，确认连接目标没有被替换。

8 课前电脑准备清单

8.1 硬件、系统与账号

☐ 至少预留 1 GB 可用空间
☐ 能使用校园网或课程要求的网络

☐ 能正常登录教师指定 Git 平台
☐ 已确认课程组或仓库访问权限
☐ 记得账号的主要邮箱或用户名

☐ PowerShell
☐ 支持 C++17 的 g++
☐ Git
☐ OpenSSH Client 与 ssh-keygen

☐ VS Code 或其他纯文本编辑器
☐ 已下载教师发放的第七讲资料
☐ 资料能正常解压，路径容易找到

8.2 推荐的实验目录

Windows 同学可提前建立一个路径短、容易定位的目录，例如：

```
D:\CourseWork\lesson7
```

这只是推荐路径，不是强制要求。请避免直接在重要项目、桌面杂乱目录或其他课程仓库中做覆盖文件实验。

不要把私钥永久留在公共机房电脑中；不要在共享电脑上直接设置不属于自己的全局 Git 身份信息。

9 用五分钟检查工具链

9.1 在 PowerShell 中执行

下面的命令只检查工具是否存在，不会修改仓库：

```
$PSVersionTable.PSVersion

Get-Command g++
Get-Command git
Get-Command ssh
Get-Command ssh-keygen

g++ --version
git --version
ssh -V
```

四个 Get-Command 都能找到程序；g++ –version 和 git –version 能输出版本；ssh -V 能显示 OpenSSH 版本。

9.2 检查 Git 身份信息

```
git config --global user.name
git config --global user.email
```

如果输出为空，个人电脑可按教师要求设置：

```
git config --global user.name ” 你的姓名或平台用户名”
git config --global user.email ” 你的课程邮箱”
```

引号中的内容必须换成你自己的信息。若使用共享电脑，请在课程仓库内改用 git config user.name 与 git config user.email，不要写入全局配置。

9.3 只查看现有 SSH 文件

```
Get-ChildItem $env:USERPROFILE\.ssh -Force -ErrorAction SilentlyContinue
```

没有任何输出并不表示出错，只说明你可能还没有创建过 SSH 密钥。除非教师已经明确给出命名和上传要求，否则可以留到课堂上统一完成。

10 做一次最小 C++ 编译测试

10.1 建立测试文件

在推荐的实验目录中新建 hello.cpp，内容如下：

```
#include <iostream>

int main() {
    std::cout << "toolchain ready\n";
    return 0;
}
```

10.2 编译并运行

```
g++ -std=c++17 -Wall -Wextra -pedantic .\hello.cpp -o .\hello.exe
.\hello.exe
```

如果看到：

```
toolchain ready
```

说明最基本的 C++17 编译链可以工作。

10.3 失败时先保留这些信息

- 完整命令与完整错误文本；
- Get-Location 的输出；
- g++ –version 的输出；
- Get-ChildItem -Force 的输出；
- 操作系统版本，以及你使用的终端名称。

课堂资料不要求课前编译通过

上面的 hello.cpp 只检查编译器。教师发放的迷你 Git 项目如果还有依赖、缺少文件或需要课堂说明，不要求你在预习阶段自行修完；请记录错误并带到课堂。

11 上课前的资料与目录检查

11.1 课程资料建议包含

以教师最终发放内容为准，通常可以看到以下类别：

- 讲义或预习 PDF；
- 迷你 Git 教学项目源码；
- include/、src/、tests/ 等源码目录；
- README.md、构建说明或课堂操作教程；
- 教师提供的 Git 平台地址、仓库地址或加入课程组的方式。

11.2 到课时最好已经做到

☐ 能在 PowerShell 中切换到实验目录。
☐ g++、git、ssh 均可被找到。
☐ 最小 C++17 测试程序可以编译运行。
☐ 能登录教师指定的 Git 平台。
☐ 已下载并解压课程资料。
☐ 知道自己的资料放在哪个绝对路径。
☐ 已记录尚未解决的报错，而不是只截取最后一行。

11.3 这些内容建议留到课堂统一完成

- 按统一命名规则生成课程专用 SSH 密钥；
- 核对并接受课程 Git 主机的首次连接指纹；
- clone 教师指定仓库并确认目标分支；
- 运行会改变迷你 Git 仓库状态的完整实验；
- 第一次向教师平台 push 并进行网页验收。

12 带着这些问题来上课

12.1 课前自测

先不要翻答案，用一两句话回答：

1. HEAD、分支名和提交编号之间是什么关系？
2. 创建分支以后，当前分支会自动改变吗？
3. checkout 为什么不能只改 HEAD？
4. commit 时，应该移动所有分支，还是只移动当前分支？
5. 快进合并为什么要检查 parent 链？
6. SSH、公钥、commit、push 分别解决什么问题？
7. push 成功以后，还需要在网页上检查什么？

12.2 预习答案要点

- HEAD 先给出当前分支名，分支 ref 再给出提交编号。
- 创建分支只增加名字，不自动切换。
- checkout 既要切换名字，也要安全恢复目标快照。
- commit 只推进当前分支。
- parent 链表达祖先关系，编号大小不能表达分叉关系。
- SSH 负责认证；commit 创建本地历史；push 更新远程分支。
- 网页上要核对仓库、分支、最新提交与文件是否完整。

你能看懂本讲知识地图，工具链检查基本通过，课程资料和账号已经就绪，并且知道哪些操作必须留到课堂上在教师引导下完成。