“Linux 生物信息技术基础”课程小组总结报告

组：G12　次：9　组长：马小凯　执笔：马小凯

1. 时间

2023 年 5 月 7 日

2. 方式

线上腾讯会议

3. 主题

Git 在 Linux 上的使用

4. 内容

本次讨论了 Git 的一些基本命令

- **直接记录快照，而非差异比较**
  - Git 更像是把数据看作是对小型文件系统的一系列快照
  - 在 Git 中，每当你提交更新或保存项目状态时，它基本上就会对当时的全部文件创建一个快照并保存这个快照的索引
  - 为了效率，如果文件没有修改，Git 不再重新存储该文件，而是只保留一个链接指向之前存储的文件
- **三种文件状态**
  - 已提交（**committed**）
    - 数据已经安全地保存在本地数据库中
  - 已修改（**modified**）
    - 修改了文件，但还没保存到数据库中
  - 已暂存（**staged**）
    - 对一个已修改文件的当前版本做了标记，使之包含在下次提交的快照中
- **安装 Git**
  - **sudo dnf install git-all**
  - **git clone git://**git.kernel.org/pub/scm/git/git.git
    - 使用 git 来获取 git 的更新
- **配置 Git**
  - 初次运行 Git 前需要定制 Git 环境
  - Git 自带一个 git config 的工具来帮助设置控制 Git 外观和行为的配置变量，存储在
    - **/etc/gitconfig** 文件：包含系统上每一个用户及他们仓库的通用配置
    - **~/.gitconfig** 文件：只针对当前用户
    - 当前使用仓库的 Git 目录中的**.git/config**：针对该仓库
    - 每一个级别会覆盖上一级别的配置

- **git config --list --show-origin**
  - 查看 Git 当前仓库的所有的配置以及它们所在的文件
- **设置用户名和邮件地址**
  - **git config --global user.email ""**
  - **git config --global** user.name **""**
  - 针对特定项目使用不同的用户名称与邮件地址时，可以在项目目录下运行没有 --global 选项的命令来配置

- **获取 Git 仓库**
  - **在已存在的目录中初始化仓库**
    - **git init**
      - 在该目录中创建一个名为 **.git** 的子目录，含有初始化的 Git 仓库中所有的必须文件
  - **克隆现有的仓库**
    - **git clone <url>** *name*
      - 在当前目录下创建一个名为 *name*（可选）的目录，在这个目录下初始化一个 **.git** 文件夹
- **记录更新**
  - **git status**
    - 查看哪些文件处于什么状态，展示当前目录下任何处于未跟踪与修改状态的新文件
    - Changes to be committed：这行下面的文件为已暂存状态
    - **-s**
      - 缩短状态命令的输出
      - ?? ： 未跟踪文件
      - A ： 新添加到暂存区中的文件
      - M ： 修改过的文件
      - 左栏指明了暂存区的状态，右栏指明了工作区的状态
  - **git add** *file*
    - 开始跟踪 *file* 文件，或者把已跟踪的文件放到暂存区
    - 也可用于合并时把有冲突的文件标记为已解决状态
    - 如果是目录的路径，该命令将递归地跟踪该目录下的所有文件
  - **git commit -m "..."**
    - 提交放在暂存区域的快照
    - **-m** 后写提交说明，也可不加**-m** 在 vim 编辑器中输入提交说明
    - **-a** 可自动 add 所有已经跟踪过的文件
    - **--amend** 可以新的提交覆盖上一次提交，用于提交忘记交的文件

- **git rm** *file*
  - 移除文件，下一次提交时，该文件不再纳入版本管理
  - **-f**
    - 强制删除之前修改过或已经放到暂存区的文件，防止误删尚未添加到快照的数据
  - **--cached**
    - 从暂存区域移除，但仍然希望保留在当前工作目录中
- **git reset HEAD** *file*
  - 从暂存区中取消这次修改的文件，用于撤销，将两次更改分两次 commit
- **git mv** *file_from file_to*
  - 重命名文件
  - 若用其他方式改名，注意先 rm 后 add
- **git log**
  - 按时间先后顺序列出所有的提交，最近的更新排在最上面
- **git checkout --** *file*
  - 撤消对文件的修改，还原成上次提交时的样子

- **远程仓库**
  - **git remote**
    - 列出指定的每一个远程服务器的简写
    - **-v**
      - 显示需要读写远程仓库使用的 Git 保存的简写与其对应的 URL
    - **add** *name <url>*
      - 添加一个新的远程 Git 仓库，简写为 *name*，链接为 url
    - **show** *<remote>*
      - 列出远程仓库的 URL 与跟踪分支的信息
    - **rename** *name newname*
      - 重命名远程仓库 name 的简写
    - **remove** *<remote>*
      - 移除一个远程仓库，删除和这个远程仓库相关的远程跟踪分支以及配置信息
  - **git fetch** *<remote>*
    - 访问远程仓库，从中拉取所有你还没有的数据
  - **git pull**
    - 自动抓取远程分支后合并到当前分支，通常会从最初克隆的服务器上抓取数据并自动尝试合并到当前所在的分支
  - **git push** *<remote> <branch>*

- 推送到上游，本地远程分支名不同用冒号隔开