STAT243 Lecture 1.1 Introduction to UNIX, Computers, and Key Tools

Logic

本章目标：熟悉 UNIX 命令行、版本控制、计算机硬件基本组成、远程连接与常用文本编辑器，为后续编程与数据计算打下统一的环境与工作流基础。

---

1 UNIX command line basics（命令行基础）

Logic

大多数科学计算在 UNIX 系统（Linux / macOS）上进行。命令行提供可组合、可追踪、可自动化的工作方式。

- **UNIX** 指类 UNIX 系统（Linux、macOS）。科研计算常通过 **SSH** 登录到远程 UNIX 服务器。
- 建议尽快完成入门教程（如 UNIX 基础、Shell scripting），确保能独立完成基本文件操作与脚本运行。

1.1 常用概念

- **Shell**：如 `bash`、`zsh`，提供命令解析与脚本执行。
- **Path**：`/` 为根目录；`~` 为用户 home；`.` 当前目录；`..` 上级目录。
- **权限**：`rwx`，可用 `chmod` / `chown` 调整。
- **I/O 重定向**：`>` 覆盖输出、`>>` 追加输出、`<` 输入重定向、`|` 管道。

1.2 常见操作示例

Shell

```shell
# 查看路径与文件
pwd
ls -lha

# 创建、移动、复制、删除
mkdir -p project/src
mv a.txt project/
cp -r project project_bak
rm -rf project_bak

# 查看内容与搜索
cat file.txt
less file.txt
grep -n "pattern" file.txt

# 环境信息
uname -a
which python
```

Remark

在 Windows 环境进行科研计算时，建议使用 WSL（Windows Subsystem for Linux）或直接在远程 Linux 服务器上工作，以获得一致的 UNIX 体验。

---

2 Version control（版本控制：Git）

Logic

版本控制是现代科研与协作的核心。Git 基于逐行 diff 跟踪文本文件（代码、配置、文档）的演化。

- **Repository（仓库）**：项目文件与历史记录的集合。
- **Remote（远端）**：如 GitHub/GitLab；本地与远端通过 `fetch` / `push` 同步。

2.1 最小工作流

Shell

```shell
# 1) 克隆课程仓库（示例）
git clone https://github.com/berkeley-stat243/fall-2025

# 2) 同步远端更新
cd fall-2025
git pull

# 3) 本地开发常用
git status
git add <files>
git commit -m "feat: add notes for unit1"

# 4) 推送到远端（需提前设置权限）
git push origin main
```

Remark

GUI 客户端（如 GitHub Desktop）可降低上手难度，但必须理解基础命令与分支/合并概念，以便在冲突与回滚时自救。

---

3 Parts of a computer（计算机组成与性能瓶颈）

Logic

编程效率不仅取决于算法，也受硬件层次结构影响：CPU、缓存、内存与磁盘 I/O 的速度差异可能决定整体性能瓶颈。

- **CPU**：负责算术与逻辑运算；具备多级 **cache**（L1/L2/L3）。
- **Main memory（RAM）**：主存，容量中等，延迟低于磁盘但高于 cache。
- **Bus**：在 CPU、内存、外设之间传输数据的总线。
- **Disk（持久化存储）**：HDD/SSD，容量大但访问最慢；I/O 可能成为瓶颈。

3.1 典型瓶颈判断

- **Compute-bound**：CPU 算力成瓶颈。
- **Memory-bound**：内存带宽/访问延迟成瓶颈。
- **I/O-bound**：磁盘读写成瓶颈，需优化数据布局、批量 I/O、缓存与并行。

---

4 Connecting to other machines（远程连接与文件传输）

Logic

通过 SSH 进行远程登录与端口转发，通过 SCP/SFTP 可靠地搬运数据。

4.1 SSH 登录

Shell

```shell
# 登录到远程主机（示例用户名与主机）
ssh username@radagast.berkeley.edu

# 指定端口、开启压缩
ssh -p 22 -C username@host

# 首次连接会提示保存主机指纹；建议配置 SSH key 免密登录
```

4.2 SCP 文件传输

Shell

```shell
# 本地 → 远程
scp file.txt username@radagast.berkeley.edu:~/research/

# 远程 → 本地（重命名保存）
scp username@radagast.berkeley.edu:/data/file.txt \
    ~/research/renamed.txt
```

Remark

- 远程路径通常使用**绝对路径**或以 `~` 为基准的相对路径。
- 大量文件或断点续传场景，考虑 `rsync -avP` 以降低传输成本并便于续传。
- 若需图形界面，可选 SFTP 客户端（如 FileZilla、WinSCP）。

---

5 Editors（文本编辑器与 IDE）

Logic

科学计算需要**文本编辑器**，而非 Word 类富文本处理器。编辑器需良好支持代码、高亮、Lint、LSP 与可扩展性。

5.1 选择建议

- 传统 UNIX 编辑器：**emacs**, **vim**。
- 现代通用编辑器：**VS Code**（IDE 级别，支持 Python / R / Jupyter / Quarto，集成 Git 与调试；可用 GitHub Copilot）。
- 其他：Sublime Text（专有），Aquamacs Emacs（macOS），Notepad++（Windows），TextMate（macOS）。
- **RStudio**：R / Quarto / R Markdown 生态友好，亦可运行 Python 代码块。

Remark

- 不要用 Microsoft Word 或 Google Docs 编辑代码或 Markdown/Quarto/LaTeX。

- Windows 默认隐藏文件后缀，可能导致脚本扩展名判断错误，建议开启显示后缀。

5.2 (Optional) Basic emacs（快速上手）

- **Major modes**：针对不同文件类型启用特定编辑体验（Python、R、C、LaTeX 等）。
- **在终端中运行**：图形转发不便时可用 `emacs -nw file.txt`。
- **与解释器联动**：R 可用 ESS（Emacs Speaks Statistics）。

5.2.1 常用键位（部分）

- `C-x C-c` 退出；`C-x C-s` 保存；`C-x C-w` 另存。
- `C-s` 搜索；`ESC` 退出命令区。
- `C-a` 行首；`C-e` 行尾；`C-k` 剪切到行尾；`C-y` 粘贴（yank）。
- 选区：`C-space` 起始 → 移动光标到末尾 → 操作。

Example

若在 `git commit` 时默认打开 emacs 作为编辑器，输入 commit message 保存并退出：`C-x C-s` → `C-x C-c`。

5.3 (Optional) Basic vim（模式编辑器）

- 两种模式：**Normal**（导航/命令）与 **Insert**（插入文本）。
- 由 Normal 进入 Insert：按 `i`；退出回 Normal：`Esc`。
- 保存/退出/搜索：`:w`、`:x`、`:q`；`/pattern` 搜索；`n`/`N` 导航匹配。

Example

`git commit` 未加 `-m` 时默认进入 vim：按 `i` 进入 Insert，写 message；按 `Esc` 回 Normal，输入 `:wq` 保存退出。