Linux系统操作基础

Basics of Linux System Operations

主讲人：张陌尘

邮箱：zhangmochen2019d@big

主讲时间：2025年7月10日

主讲地点：浙江海宁

目录 | CONTENTS

01 Linux系统概览与终端基础

Linux操作系统概览及其特点

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统

系统名称	说明	代表意义	
MS-DOS	微软早期命令行操作系统	原始命令行环境的代表	
UNIX	Unix系统，最早的多用户多任务系统之一	现代系统的祖先	
Windows	微软的图形化系统	主流商业桌面系统	
macOS	Unix-like 图形系统	BSD系图形系统代表	
Linux	开源类Unix操作系统	发展自Unix，免费、稳定、强大	

Linux系统发行版本介绍

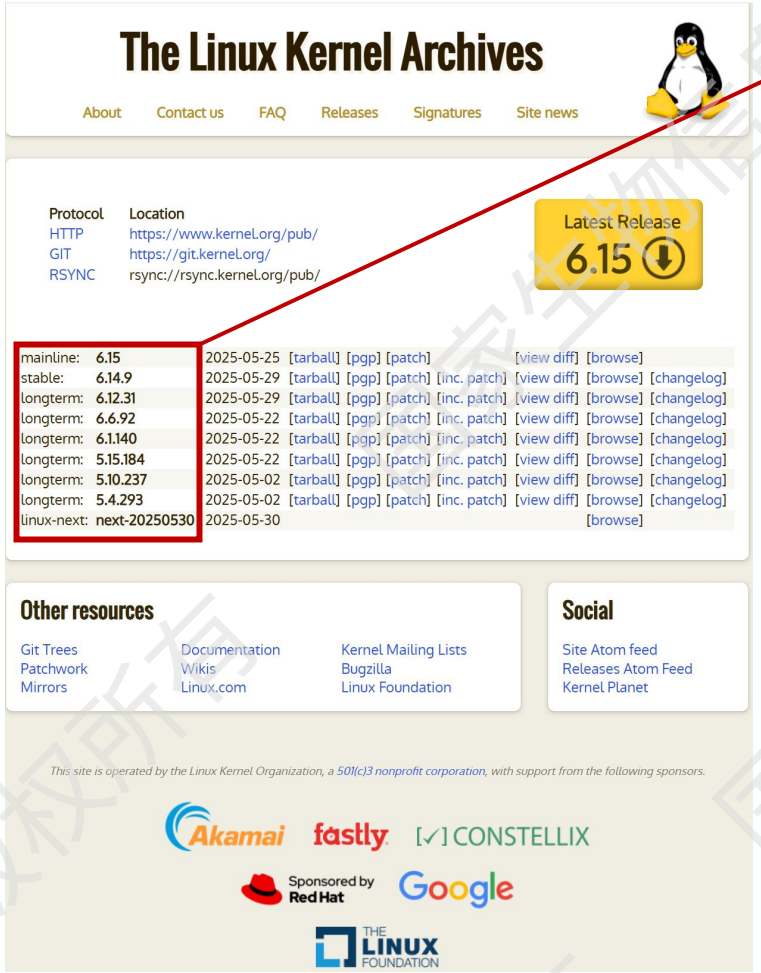
Linux指的是操作系统内核，不是一整套完整系统。所谓Linux发行版，是将Linux内核与应用软件打包形成的完整系统

LINUX内核

	Debian	Fedora	SUSE	其他发行版
发行家族	自由、稳定	Red Hat维护	德国SUSE维护	高阶定制系统
发行版本	Ubuntu Linux Mint	RHEL CentOS/Oracle Linux	SLES（商业版） openSUSE（社区版）	Arch Gentoo
	科研、服务器与日常桌面	强调前沿技术	商业支持完善	内核级优化或极简

Linux内核的版本结构和更新逻辑

在官方网站中会定期更新Linux内核版本，小更新改末位修订号，积累改动升次版本号，大改动才更新主版本号



Linux内核版本编号：主版本号.次版本号.修订号

修订号

当内核仅进行较小幅度的更新时，通常通过增加末位修订号进行版本迭代。

次版本

当功能改动和优化逐步积累到一定程度时，会更新次版本号。

主版本

当引入重大变更或架构调整时，则会升级主版本号。

Linux内核的官方网站：http://www.kernel.org

Linux常见发行版介绍

Linux发行版是将Linux内核、系统工具、包管理器以及可选的桌面环境打包而成的操作系统，其差异主要源于目标定位和维护方式。选择适合自身需求的发行版，才是最明智的选择。

什么是Linux发行版?

Linux内核

操作系统的核心

系统工具

管理系统的工具

包管理器

安装软件的工具

桌面环境

图形用户界面

（可选）

Linux发行版 = Linux内核+系统工具+包管理器+桌面环境（可选）

常见的Linux发行版家族

分类方式	示例	特点
按发行家族	Debian、Red Hat、SUSE等	相同家族共享包管理器和系统结构
按目标用途	桌面版、服务器版、嵌入式、科研用途等	Ubuntu Desktop、CentOS、OpenWRT
按支持类型	社区版 vs 商业版	Fedora vs RHEL，openSUSE vs SLES

没有最好的发行版，只有最适合你需求的发行版

Linux系统的特点

Linux系统凭借其开源免费、多用户多任务、安全稳定、可移植与可定制等优势，成为服务器、科研、高性能计算和嵌入式开发等领域的首选操作系统。因此，绝大多数生物信息分析软件都是为Linux系统开发的。

开源免费

Linux 采用 GPL 等开源许可证发布，用户可以免费获取、使用、修改和分发。这种开源模式促进了大规模社区协作和快速迭代。

多用户与多任务

在同一台机器上，多个用户可以同时登录并运行各自的进程；每个用户的程序也可以并行执行，彼此之间相互隔离、互不干扰。

稳定性与高可用性

由于 Linux 内核设计成熟，常用于服务器与超级计算集群，长期运行不易崩溃。即使长时间连续运行，系统性能也能保持稳定。

安全性强

基于Unix权限体系，Linux提供多层级的用户与组权限管理、文件系统访问控制，可以实现细粒度的安全策略，抵御常见攻击。

可移植性与跨平台支持

Linux 内核可运行在从嵌入式设备、ARM 单板机，到 x86 服务器、超级计算机等多种硬件架构上，具备出色的可移植性。

高度可定制与模块化

用户可以根据实际需求裁剪内核功能或添加模块，也可选择不同的发行版、桌面环境（KDE等）以及包管理器（apt等），灵活性极高。

丰富的软件生态

Linux 社区与各大发行版的软件仓库中汇集了数以万计的开源应用，从服务器架构（如MySQL）到开发工具（如R），都有完善支持。

命令行工具强大

标准的Shell（如Bash）和大量文本处理工具（awk、sed、grep等）使得批量数据处理、自动化脚本编写非常高效，尤其适合科研与运维场景。

绝大多数生物信息分析软件都是为Linux系统开发的

生物信息学软件普遍依赖Linux系统，因为其在高性能计算、稳定性和工具兼容性方面远优于图形化操作系统。

为什么选Linux?

1. 大数据量需讲求性能

- 生物序列、甲基化、转录组数据体量巨大
- CPU/内存并行吞吐优先于 UI 界面

2. 超级计算机几乎都跑Linux

- 根据 Top500.org 数据，≈90% + 节点部署 Linux
- 类 Unix 环境在大规模并行计算与调度方面更成熟

常见的生物信息分析软件

1. 本地高性能C/C++工具

- 用途：比对（如Bowtie）、组装（SOAP 系列）、BLAST/BLAT等
- 要求：仅在类nix环境下编译/运行，Windows下运行不便

2. Java通用工具（跨平台）

- 用途：变异调用（GATK）、QC（Picard）、Viewer（IGV）
- 特点：只需.jar便可运行，Linux/Windows/Unix共用

脚本工具及扩展

1. 基础脚本

- 语言：Perl、Python
- 功能：文本查找、数据匹配等轻量操作（无需额外系统依赖）

2. 扩展包依赖

- 常见扩展：Bioperl、Biopython、Bioconductor
- 需依赖底层动态库（如 zlib、libcurl、R、Fortran 库），Windows/macOS 下配置易出兼容问题

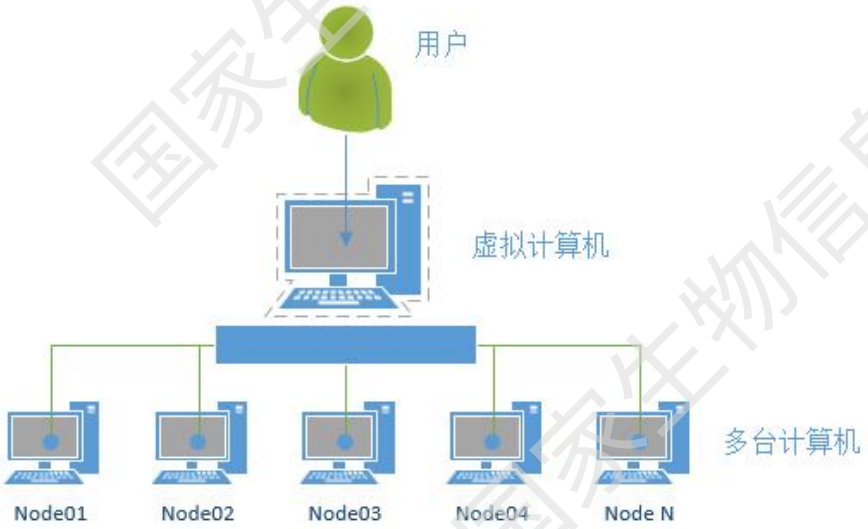
Linux提供高吞吐命令行环境、多核并行支持，建议在Linux下搭建生信分析环境，以获得最佳性能与兼容性

集群的定义及其组成要素

集群是通过多台计算节点+高速网络+统一存储+调度管理组成的整体，旨在提供高性能、可扩展、高可用的计算/存储服务，广泛应用于生物信息学分析计算领域。

集群是将多台互相独立的计算机（也称节点、Node）通过网络连接，在应用层面上看似一台逻辑整体，共同承担工作负载。每个节点既能独立执行任务，也能与其他节点协同完成更大规模的运算或服务请求。

集群基本模式图



节点（Node）：

- **计算节点（Compute Node）**：负责执行具体的计算或存储任务，配置通常包括多核 CPU、大容量内存，以及必要时配备 GPU。
- **管理节点（Master/Head Node）**：负责作业调度、资源管理、用户身份认证、集群监控等关键任务。一般不参与实际的并行计算。

高性能计算集群

用于科学计算、数值模拟、生物信息学分析、人工智能模型训练等，对运算速度和并行度要求极高。

存储集群

将多台存储服务器组成分布式存储系统，实现海量数据的冗余备份与高速访问（如 HDFS、Ceph）。

服务/应用集群

多台应用服务器或 Web 服务器组成负载均衡集群，提高网站或在线服务的可用性和并发访问能力（如 Nginx/HAProxy + 多台后端服务器）。

数据库集群

通过主从复制或分片，将数据库服务分布在多个节点上，兼顾高可用与读写性能提升（如 MySQL 主从、PostgreSQL 集群、MongoDB 副本集或分片集群）。

集群的主要特点

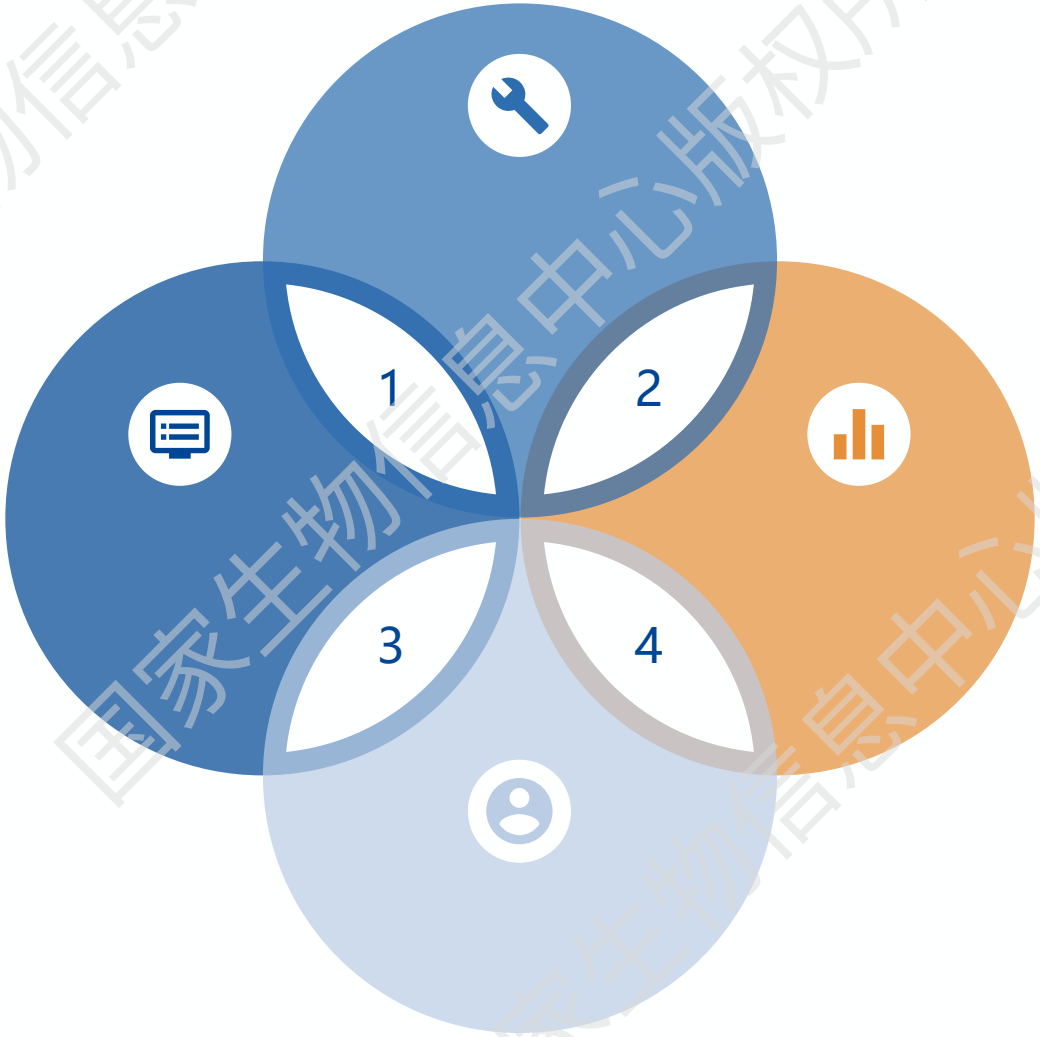
集群通过多节点并行处理实现高性能，支持横向扩展提升可扩展性，具备故障容错机制确保高可用性，并依靠集中调度系统统一管理资源。

高性能与并行处理

将任务拆分为多个子任务，在不同节点并行计算。例如，在生物信息学中，大规模序列比对可以将基因组分段后交由多个节点同时处理，从而显著加快总耗时。

可扩展性高

需要更多计算或存储能力时，只需按需增加节点即可。通过容量扩展（横向扩展，Scale-Out），集群可以从几十节点增长到上千节点，保持相对线性提升的性能。



高可用性

如果某个节点故障，其他节点可以自动接管其工作或数据，避免单点故障导致整个系统不可用。常见做法包括节点冗余、心跳检测、故障转移（Failover）等机制。

统一管理与调度

集群通常配备集中化的调度/管理系统（如 Slurm、PBS、Kubernetes、Hadoop YARN），负责将作业分配到空闲节点、监控节点状态、收集资源利用率等。

集群工作环境架构

集群工作环境通过登录节点提交作业，由调度系统分配到计算节点并访问共享存储，同时利用环境模块管理软件和监控工具保障软件依赖一致与系统健康。

登录节点

- 用户通过 SSH 登录 / 编写脚本 / 调试代码 / 提交作业

作业调度系统

- 接收作业、排队、调度
- 分配资源、监控作业状态

环境模块管理

- 动态加载软件/库版本

调度任务

计算节点池

- 执行并行计算 (MPI、OpenMP 等)
- 读写共享存储、临时计算目录

I/O访问

共享存储/分布式文件系统

- 所有节点统一挂载，用于数据读写与持久化

资源监控/告警

监控与管理工具

- 监控各节点 CPU、内存、网络、存储使用情况

容器/虚拟化层次

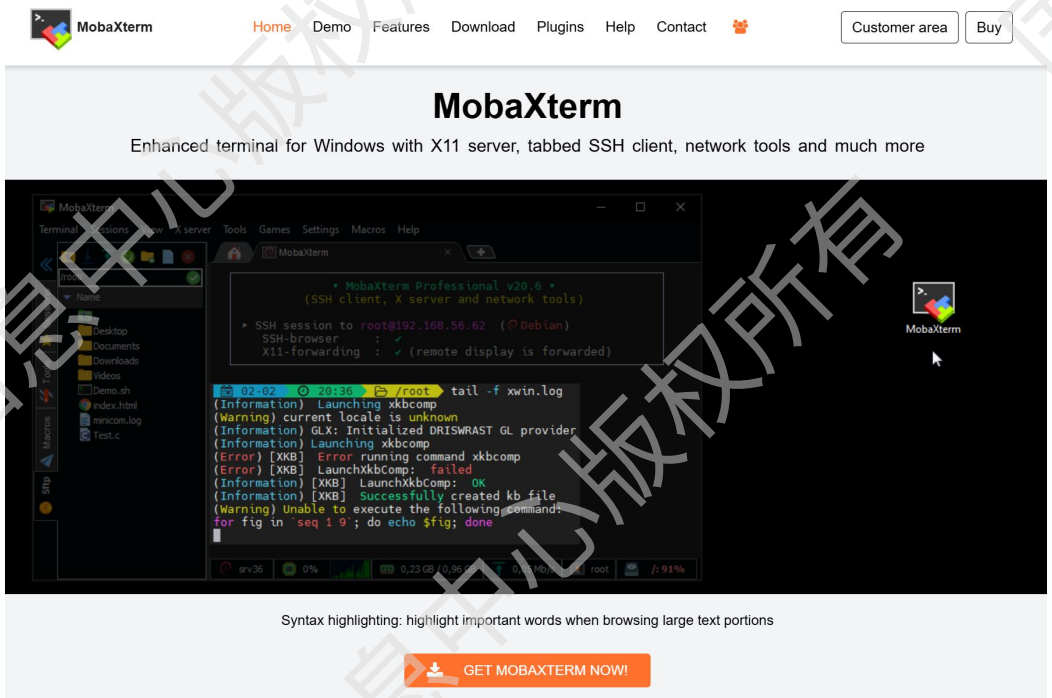
- 打包运行时环境，保证依赖一致、移植性强

国家生物信息中心机房及配套设施：主力计算节点250余台，配套高性能存储容量约9.2PB，总计算内核约11248个，其中国产CPU节点62台，国产CPU总核数3,968个，集群中配备GPU计算节点9台，部署GPU卡共18块。中心配备各档高内存容量节点26台，主力计算节点内存总数约54.5TB。用于生物大数据管理的存储系统总容量约64PB，配套建设磁带库备份系统总容量约30PB。建成了由50余台专用服务器构成的集群网络服务器系统；配有多套结构化数据管理专业服务器并部署了ORACLE、MySQL等数据库管理系统。

SSH远程登录软件推荐

MobaXterm是非常实用的SSH远程登陆软件，可通过该软件登陆集群节点（192.168.121.201），登陆节点仅允许编辑程序、脚本等操作

- 随着Linux在服务器端应用的普及，Linux系统管理越来越依赖于远程。Linux远程登录三种方式telnet，SSH， vnc. 由于SSH在安全性，功能和性能上都比较好，推荐使用SSH。



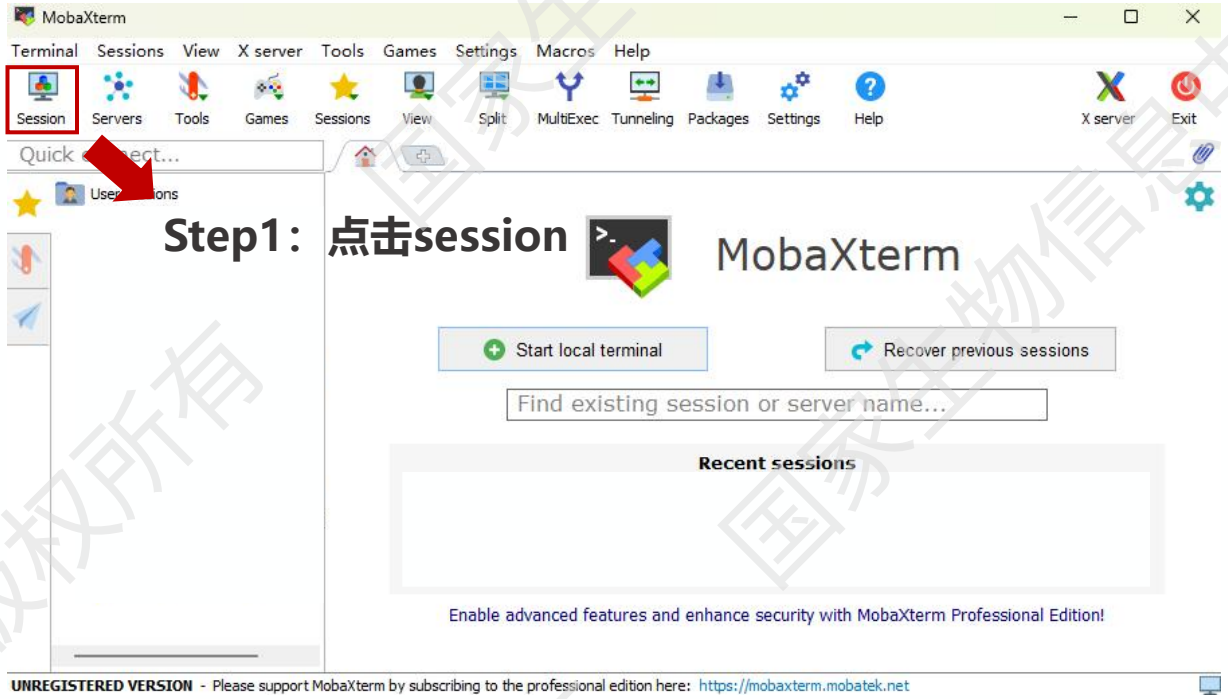
MobaXterm的官方网站：
https://mobaxterm.mobatek.net/

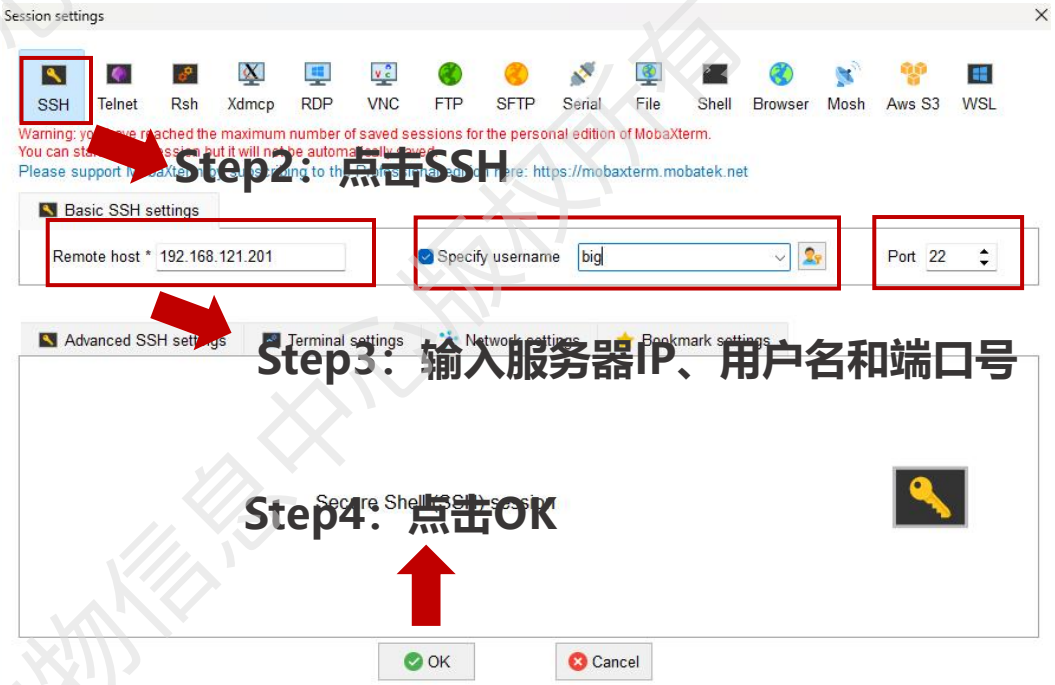
利用MobaXterm软件远程登录节点

通过MobaXterm的Session→SSH界面填写服务器IP、用户名和端口号（默认为22）并点击确认

步骤要点：

- **点击 “Session” → 选择 “SSH”**
- **填写 Remote host（服务器 IP/域名）、Username（远程用户），端口默认 22**





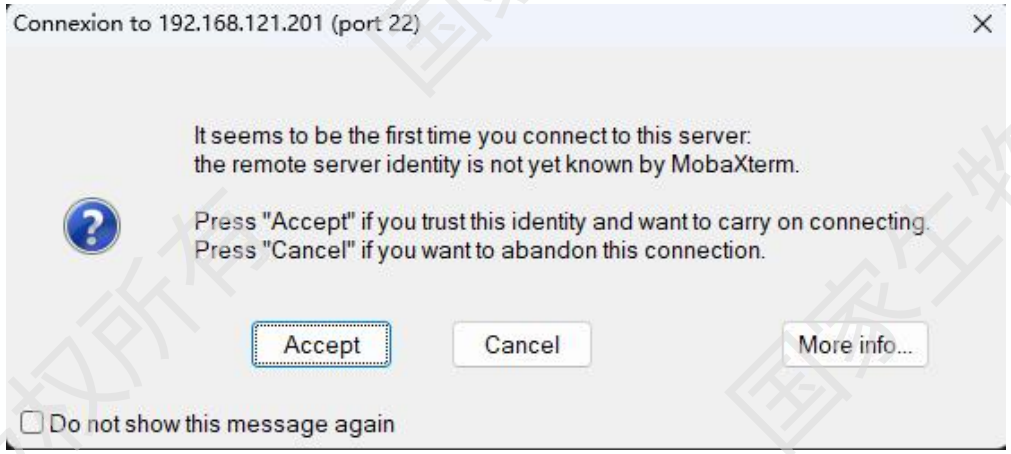
利用MobaXterm软件远程登录节点

弹出终端窗口后输入密码即可连接成功，进入远程终端

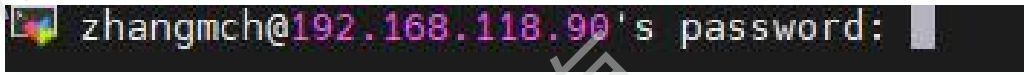
步骤要点：

- **点击 “Session” → 选择 “SSH”**
- **填写 Remote host（服务器 IP/域名）、Username（远程用户），端口默认 22**
- **“OK” → 弹出终端窗口 → 输入密码（或自动密钥认证）**
- **登录成功后，出现 username@hostname:~$ 提示符**

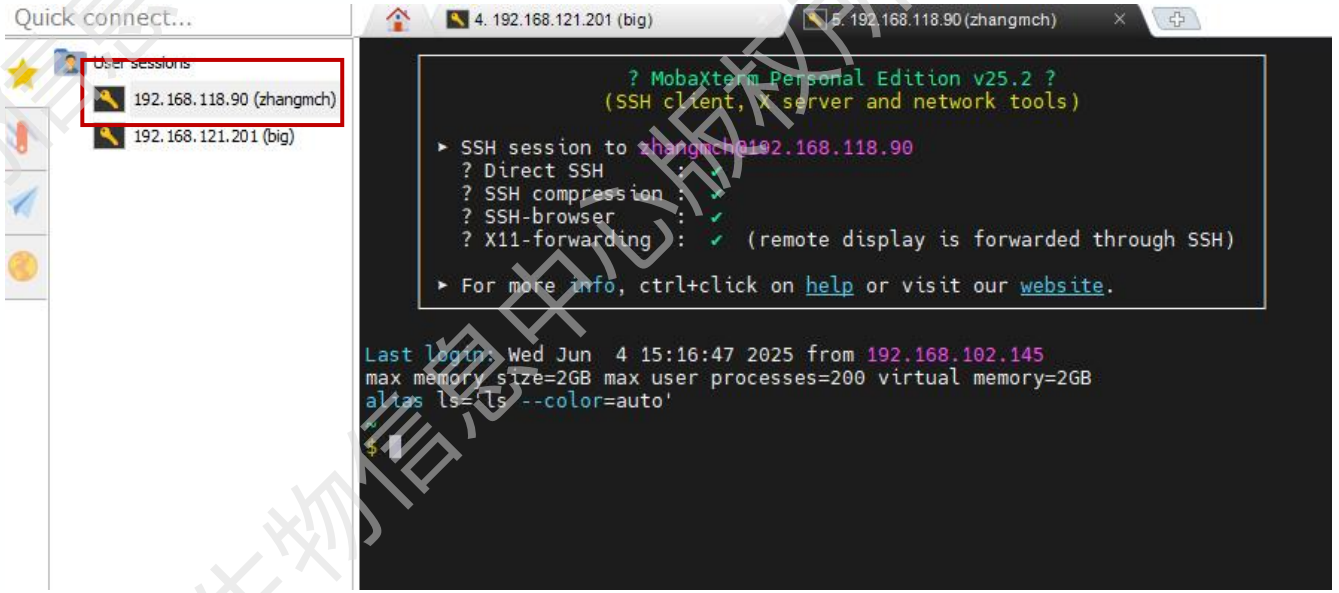
① 如果是第一次远程登陆节点，会弹出是否同意链接的提示框，点击同意



② 在终端窗口中输入对应的密码，就可以正常登陆



③ 远程登陆节点成功，且该服务器已经默认保存，后续登陆仅需双击红框中链接

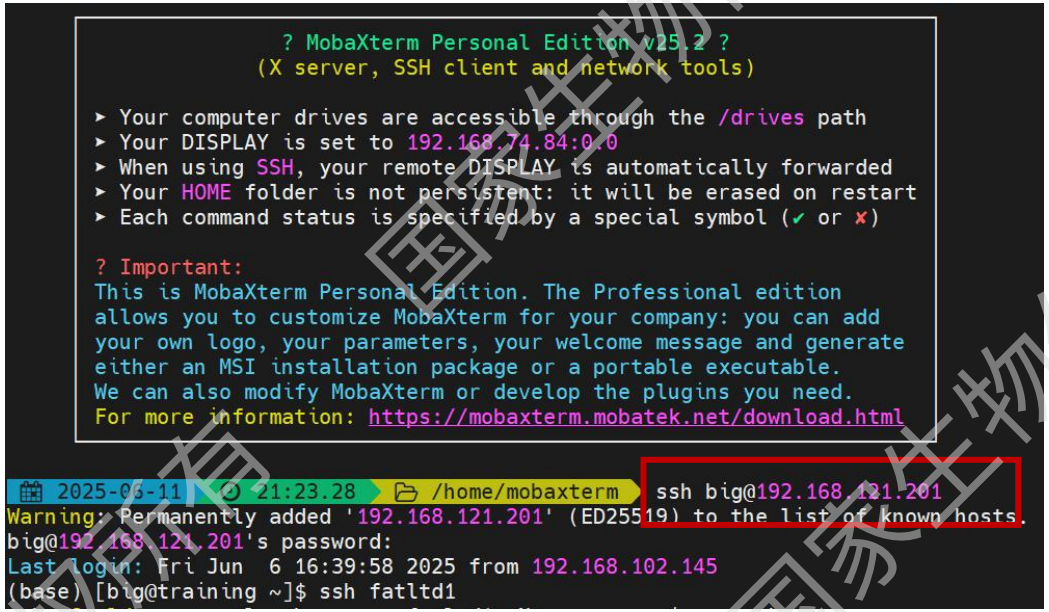


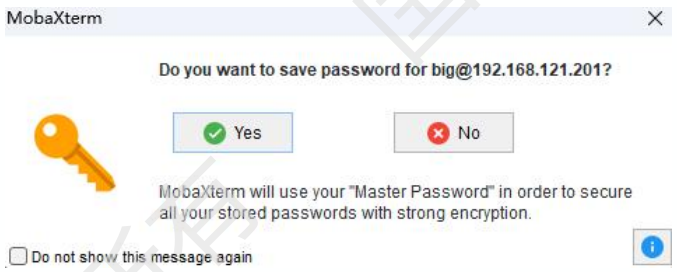
另一种远程登录节点方法及节点切换

输入密码即可连接成功，进入远程终端

步骤要点：

直接在页面中输入ssh 用户名@远程服务器IP，输入密码即可完成远程节点登录





默认询问是否存储该节点登陆密码，按照自己的需求进行选择即可

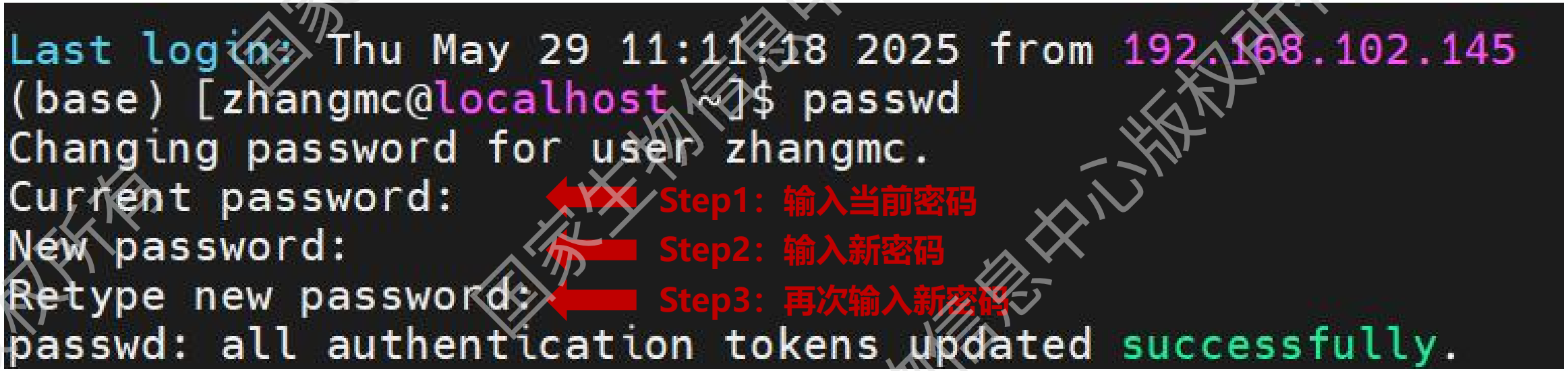
- **切换节点：ssh 用户名@远程服务器IP**
- **退出节点：exit**

密码修改

登录到远端服务器后，执行 passwd 命令，按提示输入当前密码和新密码即可完成修改

步骤要点：

- 执行 “passwd” 命令
- 输入当前密码后按回车
- 依次输入两次新密码，即可完成修改
- 注意：修改成功后，下一次通过 SSH 登录时就要使用新密码。



备注：在shell中运行export LANG=en_US.UTF-8和export LC_ALL=en_US.UTF-8，确保中文字符的正常显示

文件路径表示法

绝对路径以根目录开头，能够唯一定位文件；相对路径基于当前工作目录进行层级跳转，虽简洁但依赖上下文

绝对路径

定义

从系统根目录 / 开始呈现的完整路径，无论当前工作目录在哪里，都能唯一定位目标文件或目录。

特点

以 / 开头；始终指向相同位置；最为可靠。

示例

- /home/zhangmc/project/data.txt
- /etc/passwd

VS

相对路径

定义

基于当前工作目录（pwd）起点表示的路径，不以 / 开头，通过 . 或 .. 来跳转层级。

特点

灵活简洁；依赖于当前所在目录；适合在脚本或命令行中快速导航。

示例

假设当前目录为/home/zhangmc/project/：

- ../config/settings.conf （表示 /home/zhangmc/config/settings.conf）
- ./script/run.sh （表示 /home/zhangmc/project/script/run.sh）

基础命令之pwd

pwd命令全称为print working directory，其功能为在终端中显示当前工作目录的绝对路径

主要作用：

1. **确认当前路径**：在多级目录跳转后，帮助用户快速定位目前所在目录
2. **脚本中使用**：为脚本提供当前目录信息，可结合变量或路径拼接，保证路径操作的正确性

基本用法：

pwd [选项]

- **不带参数**：打印当前目录的绝对路径
- **常见选项：（该部分中的符号链接会在后面进行介绍）**
  - **-L（Logical）**：显示符号链接逻辑路径（默认行为）
  - **-P（Physical）**：显示物理路径，解析所有符号链接后输出真实目录

基本示例：

```
$ pwd -L
/data40T/zhangmch/single-cell
(base) zhangmch@bserver 16:10:52 /data40T/zhangmch/single-cell
```

```
$ tree .
.
├── data
│   ├── 20231129.RData
│   ├── data
│   │   └── sce.output.merge.GSE130001.Rdata
│   └── GSE130001_RAW
│       ├── GSM3729178
│       │   ├── barcodes.tsv.gz
│       │   ├── features.tsv.gz
│       │   └── matrix.mtx.gz
│       └── GSM3729179
│           ├── barcodes.tsv.gz
│           ├── features.tsv.gz
│           └── matrix.mtx.gz
├── GSE130001_RAW.tar
├── GSM3729178
│   ├── barcodes.tsv.gz
│   ├── fatures.tsv.gz
│   └── matrix.mtx.gz
├── GSM3729179
│   ├── barcodes.tsv.gz
│   ├── fatures.tsv.gz
│   └── matrix.mtx.gz
├── scmethbank
│   ├── align.py
│   ├── bis_result.py
│   ├── deduplicate.py
│   ├── QC.py
│   ├── rate.py
│   ├── scBSmap_H.py
│   └── trim.py
└── script
    ├── single-cell.R
    └── single-cell.RData

9 directories, 24 files
(base) zhangmch@bserver 16:06:17 /data40T/zhangmch/single-cell
```

基础命令之ls

ls命令全称为list directory contents，其功能为列出当前目录或指定路径下的文件和子目录

主要作用：

1. **查看目录内容**：快速浏览目录下的文件和文件夹名称
2. **获取详细信息**：结合不同选项，可查看权限、大小、修改时间等属性
3. **脚本及管道**：常与管道 (|) 和重定向结合，用于文件过滤、统计、批处理

基本用法：

ls [选项]

- **不带参数**：列出当前目录下所有文件（不含隐藏文件）
- **常见选项**：
  - **-a（all）**：显示所有文件，包括以 “.” 开头的隐藏文件
  - **-l（long）**：以长格式列出，包含权限、链接数、所有者、所属组、大小等信息
  - **-h（human-readable）**：与 -l 一起使用，将文件大小以可读格式显示
  - **-t（time）**：按修改时间排序，默认最新的排在最前
  - **-r（reverse）**：将输出结果倒序排列，可与其他选项组合
  - **-R（recursive）**：递归显示所有子目录内容
  - **-S**：按文件大小排序，大文件优先
  - **-d**：仅显示目录本身，而不列出其内部内容

基本示例：

（1）列出当前目录下所有文件（不含隐藏文件）

```
$ ls
data  GSE130001_RAW.tar  GSM3729178  GSM3729179  scmethbank  script
(base) zhangmch@bserver 16:18:25 /data40T/zhangmch/single-cell
```

解释：默认输出仅显示非隐藏条目。

（2）显示隐藏文件与长格式详细信息

```
$ ls -al
total 32628
drwxr-xr-x  7 zhangmch baoym_group     4096 Nov 23  2023 .
drwxr-xr-x 17 zhangmch baoym_group     4096 Jun  4 16:06 ..
drwxr-xr-x  4 zhangmch baoym_group     4096 Nov 29  2023 data
-rw-r--r--  1 zhangmch baoym_group 33382400 Nov 23  2023 GSE130001_RAW.tar
drwxr-xr-x  2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729178
drwxr-xr-x  2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729179
drwxr-xr-x  2 zhangmch baoym_group     4096 Nov 23  2023 scmethbank
drwxr-xr-x  2 zhangmch baoym_group     4096 Nov 23  2023 script
(base) zhangmch@bserver 16:20:26 /data40T/zhangmch/single-cell
```

解释：通过 -a 显示隐藏文件（.env），-l 提供文件属性、文件数、拥有者、所属的group、文件大小、建档日期、文件名等详细信息

（3）按修改时间倒序（最新在后）显示

```
$ ls -ltr
total 32620
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 scmethbank
-rw-r--r-- 1 zhangmch baoym_group 33382400 Nov 23  2023 GSE130001_RAW.tar
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729178
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729179
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 script
drwxr-xr-x 4 zhangmch baoym_group     4096 Nov 29  2023 data
(base) zhangmch@bserver 16:22:00 /data40T/zhangmch/single-cell
```

解释：-t 按时间排序，-r 倒序，-l 长格式。

基础命令之cd

cd命令全称为change directory，其功能为切换当前工作目录到指定路径

主要作用：

1. **快速导航文件系统层级**
2. **切换到项目目录、脚本所在目录或父目录，准备后续文件操作**
3. **在脚本中结合 cd 保证后续命令在正确路径下执行**

基本用法：

cd [目标路径]

- **绝对路径**：以 / 开头，直接跳转到指定目录
  - 示例：cd /home/zhangmc/project
- **相对路径**：基于当前工作目录（PWD）跳转
  - 示例：cd src/（进入当前目录下的 src）
- **特殊符号：**
  - **cd ~ 或 cd**：回到用户主目录（等价于 cd /home/用户名）
  - **cd .**：留在当前目录（无实际效果）
  - **cd ..**：切换到父目录（上一级）
  - **cd -**：切换到上一次所在目录

基本示例：

（1）切换到绝对路径

```
$ cd /data40T/zhangmch/
(base) zhangmch@bserver 16:30:16 /data40T/zhangmch
$ pwd
/data40T/zhangmch
(base) zhangmch@bserver 16:30:19 /data40T/zhangmch
```

（2）切换到相对路径　假设当前路径为/data40T/zhangmch

```
$ cd download/
(base) zhangmch@bserver 16:31:21 /data40T/zhangmch/download
$ pwd
/data40T/zhangmch/download
(base) zhangmch@bserver 16:31:23 /data40T/zhangmch/download
```

（3）返回主目录与上一次目录

```
$ cd ~
(base) zhangmch@bserver 16:33:09 ~
$ pwd
/home/zhangmch
(base) zhangmch@bserver 16:33:11 ~
$ cd /var/log
(base) zhangmch@bserver 16:33:15 /var/log
$ cd
/home/zhangmch
(base) zhangmch@bserver 16:33:19 ~
```

（4）连续跳转

```
$ cd ../download/../single-cell/
(base) zhangmch@bserver 16:35:08 /data40T/zhangmch/single-cell
$ pwd
/data40T/zhangmch/single-cell
(base) zhangmch@bserver 16:35:08 /data40T/zhangmch/single-cell
```

基础命令之mkdir

mkdir命令全称为make directory，其功能为在指定路径下创建一个或多个新目录

主要作用：

1. **新建目录：**根据项目需求创建文件夹，用于组织文件和代码
2. **批量创建：**可一次性创建多级目录（结合 -p 选项）
3. **设置权限：**使用 -m 选项为新目录指定初始权限

基本用法：

mkdir [选项] 目录名...

- **常用选项：**
- **-p：（递归创建父目录）**
  - 效果：若上级目录不存在则一并创建，且已存在目录不报错
  - 示例：mkdir -p project/src/utils
- **-m：（为新目录设置权限掩码） -（该部分中的权限掩码会在后面进行介绍）**
  - 效果：直接按指定权限创建目录，无需之后再用 chmod 修改
  - 示例：mkdir -m 755 data
- **-v：（显示创建过程）**
  - 效果：每创建一个目录都会输出提示信息
  - 示例：mkdir -v logs archives

基本示例：

（1）创建单级目录 - 在当前目录下生成名为 reports 的文件夹

```
$ mkdir reports
(base) zhangmch@bserver 16:48:18 /data40T/zhangmch/single-cell
```

（2）递归创建多级目录 - 同时创建 project/src、project/bin、project/docs，若 project 不存在也一并生成

```
$ mkdir -p project/{src,bin,docs}
(base) zhangmch@bserver 16:48:41 /data40T/zhangmch/single-cell
```

（3）指定权限创建目录 - 创建 secret 目录并赋予权限 rwx------

```
$ mkdir -m 700 secret
(base) zhangmch@bserver 16:48:59 /data40T/zhangmch/single-cell
```

（4）批量创建并显示过程 -递归创建三层目录，并在终端输出每个新建目录的路径

```
$ mkdir -pv data/{raw,processed,output}
mkdir: created directory ‘data/raw’
mkdir: created directory ‘data/processed’
mkdir: created directory ‘data/output’
(base) zhangmch@bserver 16:49:16 /data40T/zhangmch/single-cell
$ ls -l
total 32632
drwxr-xr-x 7 zhangmch baoym_group     4096 Jun  4 16:49 data
-rw-r--r-- 1 zhangmch baoym_group 33382400 Nov 23  2023 GSE130001_RAW.tar
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729178
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729179
drwxr-xr-x 5 zhangmch baoym_group     4096 Jun  4 16:48 project
drwxr-xr-x 2 zhangmch baoym_group     4096 Jun  4 16:48 reports
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 scmethbank
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 script
drwx------ 2 zhangmch baoym_group     4096 Jun  4 16:48 secret
```

基础命令之touch

touch命令的功能为在指定路径新建空文件；若文件已存在，则修改其访问时间和修改时间为当前时间

主要作用：

1. **创建空文件：**快速生成占位文件，用于脚本或程序初始化
2. **更新时间戳：**重设已有文件的访问和修改时间，常用于构建脚本判断文件变更与自动化任务触发

基本用法：

touch [选项] 文件名...

- **常用选项：**
- **-a：（仅更新访问时间）**
  - 示例：touch -a logfile.txt
- **-m：（仅更新修改时间）**
  - 示例：touch -m report.csv
- **-c：（不创建新文件，仅当文件存在时才更新时间戳）**
  - 示例：touch -c existing.txt
- **-t [[CC]YY]MMDDhhmm[.ss]：（指定时间戳，按 [[世纪][年]月日时分[.秒] 格式设置文件时间）**
  - 示例：touch -t 202506011230.00 archive.log（将时间设为 2025-06-01 12:30:00）
- **-r REF_FILE：（复制参考文件的时间戳，将目标文件更新为与 REF_FILE 相同的访问/修改时间）**
  - 示例：touch -r source.txt target.txt

基本示例：

（1）创建新文件

如果newfile.txt文件不存在，则创建一个大小为0的空文件

```
$ touch newfile.txt
(base) zhangmch@bserver 16:57:17 /data40T/zhangmch/single-cell
$ ls -l
total 32632
drwxr-xr-x 7 zhangmch baoym_group     4096 Jun  4 16:49 data
-rw-r--r-- 1 zhangmch baoym_group 33382400 Jun  4 16:56 GSE130001_RAW.tar
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729178
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729179
-rw-r--r-- 1 zhangmch baoym_group        0 Jun  4 16:57 newfile.txt
drwxr-xr-x 5 zhangmch baoym_group     4096 Jun  4 16:48 project
drwxr-xr-x 2 zhangmch baoym_group     4096 Jun  4 16:48 reports
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 scmethbank
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 script
drwx------ 2 zhangmch baoym_group     4096 Jun  4 16:48 secret
(base) zhangmch@bserver 16:57:19 /data40T/zhangmch/single-cell
```

如果GSE130001_RAW.tar文件存在，则更新时间戳（Jun 4）

```
$ touch GSE130001_RAW.tar
(base) zhangmch@bserver 16:56:35 /data40T/zhangmch/single-cell
$ ls -l
total 32632
drwxr-xr-x 7 zhangmch baoym_group     4096 Jun  4 16:49 data
-rw-r--r-- 1 zhangmch baoym_group 33382400 Jun  4 16:56 GSE130001_RAW.tar
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729178
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 GSM3729179
drwxr-xr-x 5 zhangmch baoym_group     4096 Jun  4 16:48 project
drwxr-xr-x 2 zhangmch baoym_group     4096 Jun  4 16:48 reports
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 scmethbank
drwxr-xr-x 2 zhangmch baoym_group     4096 Nov 23  2023 script
drwx------ 2 zhangmch baoym_group     4096 Jun  4 16:48 secret
(base) zhangmch@bserver 16:56:38 /data40T/zhangmch/single-cell
```

基础命令之rm

rm命令全称为remove，其功能为删除一个或多个文件和目录。使用时请小心谨慎，避免误删

主要作用：

1. **删除文件：**永久移除指定的文件
2. **删除目录：**结合递归选项，可删除非空目录及其内容
3. **脚本清理：**常用于清理临时文件、清空目标目录等自动化操作

基本用法：

rm [选项] 文件或目录...

- **常用选项：**
- **-f：强制删除（忽略不存在的文件，不显示错误提示）**
  - 示例：rm -f old.txt
- **-i：交互式删除（删除前逐一询问用户确认）**
  - 示例：rm -i data.csv
- **-r：递归删除（删除目录及其所有子文件和子目录）**
  - 示例：rm -r project/backup
- **-v：显示过程（删除每个文件或目录时输出提示信息）**
  - 示例：rm -rv logs/

基本示例：

（1）删除单个文件 – 直接删除newfile.txt

```
$ rm newfile.txt
(base) zhangmch@bserver 17:04:49 /data40T/zhangmch/single-cell
```

（2）强制删除多个文件 – 强制删除temp1.txt和temp2.txt文件

```
$ rm -f temp1.txt temp2.txt
(base) zhangmch@bserver 17:05:17 /data40T/zhangmch/single-cell
```

如果temp1.txt和temp2.txt不存在，则不显示错误信息

（3）递归删除目录

```
$ rm -r project/
(base) zhangmch@bserver 17:06:34 /data40T/zhangmch/single-cell
```

（4）交互式确认删除递归目录

```
$ rm -r -i project/
rm: descend into directory 'project/'? yes
rm: remove directory 'project/src'? yes
rm: remove directory 'project/docs'? yes
rm: remove directory 'project/bin'? yes
rm: remove directory 'project/'? yes
(base) zhangmch@bserver 17:22:40 /data40T/zhangmch/single-cell
```

逐个询问是否删除project下面的所有文件夹及对应文件

本模块小结

Linux 系统与终端基础模块的主要目标是：理解 Linux 系统的核心特性，并掌握常见的终端操作技巧

Linux系统

- Linux操作系统概览及其特点
- Linux系统内核及发行版本
- 绝大部分生物信息软件都是为了Linux系统开发的

远程登录

- SSH远程登录软件推荐
- MobaXterm软件下载与安装
- 如何MobaXterm远程登录节点

集群

- 集群的定义及其组成元素
- 集群的主要特点
- 集群的工作环境架构

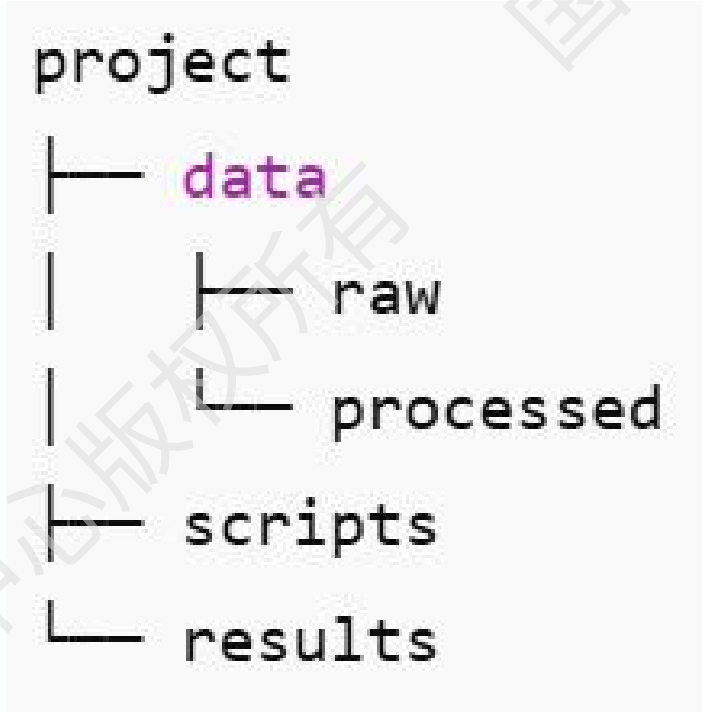
基础命令

- 密码修改
- 文件路径表示法
- pwd、ls、cd、mkdir、touch、rm

实战练习

请在你的主目录 ~/ 下完成以下操作，模拟创建一个科研项目文件夹，并组织数据与代码结构：

1. 创建如右侧目录结构
2. 进入 scripts/ 目录，创建两个脚本文件：preprocess.sh和analyze.sh
3. 进入 data/raw/ 目录，创建两个模拟数据文件：->涉及后面的知识echo，可以暂不实战
    1. sample1.csv，内容为 sample1,data1,data2
    2. sample2.csv，内容为 sample2,data3,data4
4. 进入 results/，创建空白文档 README.txt
5. 误删 analyze.sh，再重新创建该文件
6. 查看当前路径并展示整个 project/ 目录结构



实战要求及答案路径：/home/big/linux/pratical_training/1-Overview_of_the_Linux_System_and_Terminal_Basics

02

文件操作与权限管理

文件操作之cp

cp命令全称为copy，其功能为在文件系统中复制文件或目录。需根据需求选择合适参数，避免误操作

主要作用：

1. **复制文件：**将一个或多个源文件复制到目标文件或目录
2. **复制目录：**结合递归选项，可复制整个目录及其子内容
3. **备份与更新：**内置选项可以实现基于时间或版本的复制备份

基本用法：

cp [选项] 源文件 目标文件或目录

- **常用选项：**
- **-r 或 –R：（递归复制目录及其所有子文件、子目录）**
  - 示例：cp -r src_dir/ dest_dir/
- **-a：（以归档模式复制，相当于 -dR --preserve=all，保留符号链接、权限、时间戳、所有者等）**
  - 示例：cp -a project/ project_backup/
- **-p：（保留源文件的权限、时间戳和所有者信息）**
  - 示例：cp -p file.txt /backup/
- **-v：（显示详细复制过程，输出每个复制操作的源和目标路径）**
  - 示例：cp -v a.txt b.txt

基本示例：

（1）复制单个文件 – 将GSE130001_RAW.tar复制到/data40T/zhangmch

```
$ cp GSE130001_RAW.tar /data40T/zhangmch/
(base) zhangmch@bserver 17:43:18 /data40T/zhangmch/single-cell
```

（2）复制并重命名 - 将GSE130001_RAW.tar复制为temp.tar

```
$ cp GSE130001_RAW.tar temp.tar
(base) zhangmch@bserver 17:44:33 /data40T/zhangmch/single-cell
```

（3）递归复制目录 – 将script目录及其所有内容复制到script_backup

```
$ cp -r script script_backup
(base) zhangmch@bserver 17:45:28 /data40T/zhangmch/single-cell
```

（4）归档模式复制 – 以完整归档模式复制script目录，保留符号链接、权限、时间戳等

```
$ cp -a script script_backup
(base) zhangmch@bserver 17:46:54 /data40T/zhangmch/single-cell
```

文件操作之mv

mv命令全称为move，其功能为移动文件或目录位置，或重命名文件/目录。使用时务必确认目标路径与权限

主要作用：

1. **移动文件或目录：**将源路径下的文件/目录转移到目标路径
2. **重命名：**若目标路径指向同一目录下的新名称，则实现重命名
3. **脚本应用：**批量重构文件结构、整理目录、修改文件名等

基本用法：

mv [选项] 源文件 目标文件或目录

- **常用选项：**
- **-i（interactive）：交互式模式，覆盖前提示确认**
  - 示例：mv -i old.txt /backup/
- **-f（force）：强制移动，不提示，若目标存在则直接覆盖**
  - 示例：mv -f report.pdf /home/user/docs/
- **-n（no-clobber）：不覆盖目标，如果目标已存在则跳过该文件**
  - 示例：mv -n data.csv /data/archive/
- **-v（verbose）：显示详细操作过程，列出每个移动或重命名的源与目标**
  - 示例：mv -v *.log /var/log/old_logs/

基本示例：

（1）移动文件到目录 – 将GSE130001_RAW.tar移动到script相对路径中

```
$ mv GSE130001_RAW.tar script
(base) zhangmch@bserver 17:57:17 /data40T/zhangmch/single-cell
```

（2）批量移动并覆盖（带提示） - 逐个提示是否覆盖目标文件

```
$ mv -i *.tar script/
mv: overwrite 'script/GSE130001_RAW.tar'? yes
(base) zhangmch@bserver 17:59:07 /data40T/zhangmch/single-cell
```

（3）重命名文件 – 在同一目录下将matrix.mtx.gz重命名为matrix_backup.mtx.gz

```
$ mv matrix.mtx.gz matrix_backup.mtx.gz
(base) zhangmch@bserver 18:00:53 /data40T/zhangmch/single-cell/GSM3729178
```

（4）重命名并移动目录 – 将GSM3729178目录移动到新位置并重命名为GSM3729178_new

```
$ mv -v GSM3729178/ /data40T/zhangmch/single-cell/GSM3729178_new
'GSM3729178/' -> '/data40T/zhangmch/single-cell/GSM3729178_new'
```

（5）防止覆盖已有文件 – 如果matrix_new.mtx.gz文件已存在则跳过

```
$ mv -n matrix.mtx.gz matrix_new.mtx.gz
(base) zhangmch@bserver 18:03:46 /data40T/zhangmch/single-cell/GSM3729178
```

文件操作之file

file命令的功能为检测并输出指定文件的类型（文本、二进制、图像、可执行等）

主要作用：

1. **快速识别文件类型**：无需查看扩展名，直接根据文件内容判断
2. **排查脚本与二进制冲突**：确认某个脚本或可执行文件的真实类型
3. **调试与分析**：在编译、打包、传输过程中，验证文件完整性或格式正确性

基本用法：

file [选项] <文件或目录路径>

- **常用选项：**
- **-b（brief）：仅输出类型信息，不带文件名**
  - 示例：file -b report.pdf → PDF document, version 1.5
- **-i（mime）：输出 MIME 类型（如 text/plain; charset=us-ascii)**
  - 示例：file -i image.png → image/png; charset=binary
- **-L（dereference）：跟随符号链接，对链接指向实际文件进行检测**
  - 示例：file -L symlink_name
- **-z：检测压缩文件内部类型，对 .gz、.bz2 等压缩包进行深度分析**
  - 示例：file -z archive.tar.gz → gzip compressed data, was “archive.tar”, from Unix, ...

基本示例：

（1）检测文本文件 – 纯文本文件，编码格式为ASCII

```
$ file align.py
align.py: ASCII text, with CRLF line terminators
(base) zhangmch@bserver 18:18:03 /data40T/zhangmch/single-cell/scmethbank
```

（2）检测可执行文件 – ELF，64位架构，小端自字节序

```
$ file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=3d705971a4c4544545cb78fd890d27bf792af6d4, stripped
(base) zhangmch@bserver 18:17:42 /data40T/zhangmch/single-cell/scmethbank
```

（3）输出简洁类型 – 纯文本文件，编码格式为ASCII

```
$ file -b align.py
ASCII text, with CRLF line terminators
(base) zhangmch@bserver 18:16:29 /data40T/zhangmch/single-cell/scmethbank
```

（4）查看MIME类型 – 纯文本文件，编码格式为US-ASCII

```
$ file -i align.py
align.py: text/plain; charset=us-ascii
(base) zhangmch@bserver 18:16:01 /data40T/zhangmch/single-cell/scmethbank
```

（5）分析压缩包内容 – 经过gzip压缩的文件，原文件名，压缩时的环境

```
$ file -z barcodes.tsv.gz
barcodes.tsv.gz: ASCII text (gzip compressed data, was "GSM3729178_sample1_barcodes.tsv", from Unix, last modified: Tue Apr 16 01:27:44 2019)
(base) zhangmch@bserver 18:14:52 /data40T/zhangmch/single-cell/GSM3729178
```

文件操作之du –sh以及df -h

du（disk usage）用于显示文件或目录的磁盘使用情况

- **-s（summarize）**：只输出汇总结果，不列出子目录详细信息
- **-h（human-readable）**：以可读性更高的单位（K、M、G）显示大小

主要作用：

1. **快速查看单个目录或文件占用磁盘总量**
2. **适用于排查磁盘空间使用热点，确认某个目录或文件是否过大**
3. **可在脚本中结合其他命令，定期统计空间并生成报表**

基本用法： **du –sh <目录或文件路径>**

（1）查看当前目录总大小 – 输出表示目录下所有文件和子目录共占用491M空间

```
$ du -sh script/
491M    script/
(base) zhangmch@bserver 18:26:43 /data40T/zhangmch/single-cell
```

（2）查看指定子目录大小 – 输出表示该目录总占用约40KB空间

```
$ du -sh /data40T/zhangmch/single-cell/scmethbank/
40K     /data40T/zhangmch/single-cell/scmethbank/
(base) zhangmch@bserver 18:27:31 /data40T/zhangmch/single-cell
```

（3）查看单个压缩包大小 – 输出表示barcodes.tsv.gz文件占用约16KB空间

```
$ du -sh barcodes.tsv.gz
16K     barcodes.tsv.gz
(base) zhangmch@bserver 18:28:30 /data40T/zhangmch/single-cell/data/GSE130001_RAW/GSM3729178
```

df（disk free）用于显示文件系统的磁盘使用情况

- **-h（human-readable）**：以可读性更高的单位（K、M、G）显示容量

主要作用：

1. **查看磁盘空间总量与可用量**：快速了解各挂载点的已用、可用和总容量
2. **排查磁盘空间告警**：定位空间紧张的分区，以便及时清理或扩容
3. **系统运维常用**：适合在脚本中定时监控并发送告警

基本用法： **df –h <目录文件或挂载点>**

- **不带参数**：列出所有已挂载的文件系统及其使用情况
- **带路径参数**：仅显示该路径所在文件系统的使用情况

（1）查看所有挂载点容量（文件系统大小、已用/可用空间、可用比例、挂载点目录）

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       197G  186G  1.7G 100% /
devtmpfs        252G     0  252G   0% /dev
tmpfs           252G  8.1G  244G   4% /dev/shm
tmpfs           252G  987M  251G   1% /run
tmpfs           252G     0  252G   0% /sys/fs/cgroup
/dev/sda6        99G   61M   94G   1% /bkup
/dev/sdb        220G   61M  209G   1% /ssd
/dev/sda2       477M  197M  251M  44% /boot
/dev/sda4       4.1T  1.3T  2.7T  32% /data
tmpfs            51G  4.0K   51G   1% /run/user/42
/dev/sdc         37T   28T  6.7T  81% /data40T
(base) zhangmch@bserver 10:57:25 /data40T/zhangmch/single-cell
```

（2）查看指定文件所在文件系统（仅显示）

```
$ df -h /data40T/zhangmch/
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc         37T   28T  6.7T  81% /data40T
(base) zhangmch@bserver 10:59:10 /data40T/zhangmch/single-cell
```

文件搜索之find

find命令的功能为在文件系统中递归搜索符合条件的文件和目录。find实时精准但速度相对较慢

主要作用：

1. **精确定位文件：**按名称、类型、大小、时间等多种条件查找目标
2. **批量处理：**结合 -exec、xargs 实现对查找到的文件批量操作
3. **脚本自动化：**常用于备份、日志清理、权限审计等自动化任务

基本用法：

find [起始目录] [筛选条件] [处理动作]

- **常用选项：**
- **起始目录：指定要从哪个目录开始递归查找，例如 .（当前目录）、/home/user 等。**
- **筛选条件：根据文件名、类型、大小、修改时间、权限等进行过滤，常用的条件有：**
  - -name <模式>：按文件名匹配（支持通配符），如 -name "*.txt"
  - -type <f|d>：按类型匹配，f 表示文件，d 表示目录
  - -size <+|−><大小>：按文件大小匹配，+10M 表示大于 10MB，-1G 表示小于 1GB
  - -mtime <±天数>：按修改时间匹配，-mtime +7 表示 7 天前修改的文件
  - -perm <权限>：按权限匹配，例如 -perm 644
- **处理动作（可选）：对匹配到的文件或目录执行操作，常用的有：**
  - -print：打印匹配路径（大多数系统默认会打印）
  - -exec <命令> {} \;：对每个匹配项执行指定命令，{} 代表当前文件路径，最后要以 \; 结尾
  - -delete：直接删除匹配到的文件或空目录（必须谨慎使用）

基本示例：

（1）按名称查找当前目录及子目录下所有.tsv.gz文件

```
$ find . -name "*.tsv.gz"
./GSM3729178/fatures.tsv.gz
./GSM3729178/barcodes.tsv.gz
./GSM3729179/fatures.tsv.gz
./GSM3729179/barcodes.tsv.gz
./data/GSE130001_RAW/GSM3729178/features.tsv.gz
./data/GSE130001_RAW/GSM3729178/barcodes.tsv.gz
./data/GSE130001_RAW/GSM3729179/features.tsv.gz
./data/GSE130001_RAW/GSM3729179/barcodes.tsv.gz
(base) zhangmch@bserver 11:08:01 /data40T/zhangmch/single-cell
```

（2）查找/data40T/zhangmch/single-cell/data下所有目录

```
$ find /data40T/zhangmch/single-cell/data -type d
/data40T/zhangmch/single-cell/data
/data40T/zhangmch/single-cell/data/raw
/data40T/zhangmch/single-cell/data/data
/data40T/zhangmch/single-cell/data/GSE130001_RAW
/data40T/zhangmch/single-cell/data/GSE130001_RAW/GSM3729178
/data40T/zhangmch/single-cell/data/GSE130001_RAW/GSM3729179
/data40T/zhangmch/single-cell/data/output
/data40T/zhangmch/single-cell/data/processed
(base) zhangmch@bserver 11:39:05 /data40T/zhangmch/single-cell
```

（3）查看/data40T/zhangmch/single-cell/data下大于100MB的文件

```
$ find /data40T/zhangmch/single-cell/data -type f -size +100M
/data40T/zhangmch/single-cell/data/20231129.RData
/data40T/zhangmch/single-cell/data/data/sce.output.merge.GSE130001.Rdata
(base) zhangmch@bserver 11:40:16 /data40T/zhangmch/single-cell
```

（4）查找/data40T/zhangmch/single-cell下30天前修改过的文件

```
$ find /data40T/zhangmch/single-cell -type f -mtime +30
/data40T/zhangmch/single-cell/GSM3729178/fatures.tsv.gz
/data40T/zhangmch/single-cell/GSM3729178/matrix_new.mtx.gz
/data40T/zhangmch/single-cell/GSM3729178/barcodes.tsv.gz
```

文件搜索之locate

locate命令的功能为基于预先建立的文件名索引数据库，快速查找文件路径。locate速度快但索引有延迟

主要作用：

1. **超高速搜索：**不逐目录遍历，而是直接查询索引数据库，速度极快
2. **适合全系统文件查找：**命中率高，常用于查找路径或文件名已知但不确定目录的位置
3. **与 find 补充：**find 实时遍历可精确匹配条件；locate 更适合按名称模糊快速检索

基本用法：

locate [选项] <搜索模式>

- **常用选项：**
- **-i：忽略大小写匹配**
  - 例：locate -i README（同时匹配 README.md、readme.txt 等不同大小写组合）
- **-r <正则表达式>：使用正则表达式过滤结果**
  - 例：locate -r '\.conf$'（匹配所有以 .conf 结尾的文件路径）
- **-n <数量>：限制输出行数，只显示前 N 条结果**
  - 例：locate -n 5 backup.tar（仅列出前 5 条匹配 backup.tar 的路径）
- **-c：只显示匹配到的条目数量，不列出具体路径**
  - 例：locate -c /usr/bin/python（输出数据库中包含 /usr/bin/python 的条目总数）
- **--existing：仅显示当前仍存在于文件系统中的路径，忽略已删除但未更新索引的条目**
  - 例：locate --existing old.log（只列出实际存在的 old.log 文件）

基本示例：

（1）快速模糊查找 – 列出数据库中所有以.tar.gz结尾的文件路径

```
$ locate *.tar.gz
/libiconv-1.15.tar.gz
/data/home_backup/xingpq/v2.0.8-beta.tar.gz
/data/home_backup/xingpq/anaconda2/bin/Mozilla-CA.tar.gz
/data/home_backup/xingpq/anaconda2/lib/python2.7/site-packages/dateutil/zoneinfo/dateutil-zoneinfo.tar.gz
/data/home_backup/xingpq/anaconda2/pkgs/entrez-direct-11.0-pl526_1/bin/Mozilla-CA.tar.gz
```

（2）忽略大小写查找 – 匹配config.yaml、CONFIG.yaml等

```
$ locate -i config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/_ipyw_jlab_nb_ext_conf-0.1.0-py27_0/info/recipe/conda_build_config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/_libgcc_mutex-0.1-main/info/recipe/conda_build_config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/alabaster-0.7.12-py27_0/info/recipe/conda_build_config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/anaconda-2019.10-py27_0/info/recipe/conda_build_config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/anaconda-client-1.7.2-py27_0/info/recipe/conda_build_config.yaml
/data/home_backup/xingpq/anaconda2/pkgs/anaconda-navigator-1.9.7-py27_0/info/recipe/conda_build_config.yaml
```

（3）限制结果数量并显示 – 只列出前5条包含docker-compose.yml的路径

```
$ locate -n 5 docker-compose.yml
/data40T/lirui/miniconda3/envs/kingfisher/share/rubygems/gems/net-ssh-7.2.0/docker-compose.yml
/data40T/lirui/miniconda3/pkgs/aspera-cli-4.14.0-hdfd78af_1/share/rubygems/gems/net-ssh-7.2.0/docker-compose.yml
/data40T/peib/biosoft/miniconda3-u/envs/tseq/share/rubygems/gems/net-ssh-7.2.0/docker-compose.yml
/data40T/peib/biosoft/miniconda3-u/pkgs/aspera-cli-4.14.0-hdfd78af_1/share/rubygems/gems/net-ssh-7.2.0/docker-compose.yml
/data40T/wus/Cancer/DIGGER/DIGGER-master/docker-compose.yml
(base) zhangmch@bserver 11:55:33 /data40T/zhangmch/single-cell
```

（4）统计匹配数量 – 输出索引中包含/etc/ssh的条目总数

```
$ locate -c /etc/ssh
25
(base) zhangmch@bserver 15:30:59 /data40T/zhangmch/single-cell
```

注意事项：

① **索引延迟：**locate查询依赖数据库，若刚新建或删除文件，需要先运行sudo updatedb或等待定期更新，才能检索到最新结果。

② **权限限制：**数据库仅包含更新时对当前用户可见的文件路径；某些受限目录下文件可能不会被索引，所以结果不一定完整。

文件搜索之which

which命令的功能为在 PATH 环境变量指定的搜索路径中查找并显示可执行程序的位置

主要作用：

1. **快速定位命令路径：**确定正在调用的可执行文件实际位于哪个目录
2. **排查环境冲突：**当系统中存在同名脚本或程序时，验证到底使用了哪一个版本
3. **调试与脚本编写：**确保脚本中引用的是正确的二进制文件，而非错误或别名版本

基本用法：

which [选项] <命令名>

- **常用选项：**
- **-a：显示所有匹配该名称的可执行文件路径，而不仅限于第一个**
  - 示例：which -a python
  - # 输出示例：
  - # /usr/bin/python
  - # /usr/local/bin/python
- **--tty-only：仅当标准输出指向终端时才输出结果，否则静默（常用于脚本）**
- **--skip-dot：在搜索时忽略当前目录 .，即使 . 在 PATH 中；避免优先匹配当前目录下的可执行文件**
- **--read-alias、--read-functions（针对 bash）：含义：命令不仅搜索磁盘上的可执行，还检测 shell 别名和函数，显示对应的定义位置**

基本示例：

（1）定位普通命令 – 系统调用pwd的路径为/usr/bin/pwd

```
$ which pwd
/usr/bin/pwd
(base) zhangmch@bserver 15:38:29 /data40T/zhangmch/single-cell
```

（2）查看所有匹配路径 – 系统会尝试依次调用这些路径下的python

```
$ which -a python
/data40T/zhangmch/software/conda/no/bin/python
/data40T/zhangmch/software/conda/no/bin/python
/data40T/zhangmch/software/conda/no/bin/python
/data40T/zhangmch/software/conda/no/bin/python
/usr/bin/python
(base) zhangmch@bserver 15:40:37 /data40T/zhangmch/single-cell
```

（3）跳过当前目录搜索 – 假设当前目录下存在ls脚本而PATH中也包含.

```
$ which --skip-dot ls
alias ls='ls --color=auto'
        /usr/bin/ls
(base) zhangmch@bserver 15:43:17 /data40T/zhangmch/single-cell
```

（4）查看别名或函数定义

```
$ rm -rf ls
(base) zhangmch@bserver 15:43:37 /data40T/zhangmch/single-cell
$ alias lal='ls -al'
(base) zhangmch@bserver 15:43:48 /data40T/zhangmch/single-cell
$ which lal
alias lal='ls -al'
        /usr/bin/ls
(base) zhangmch@bserver 15:43:51 /data40T/zhangmch/single-cell
```

文件搜索命令比较

虽然find、locate 和 which 命令都具备文件搜索功能，但各自适用的场景有所不同，应根据具体需求选择最合适的命令以提高搜索效率

特征	find	locate	which
搜索方式	实时遍历	索引数据库搜索	环境变量$PATH中查找
搜索范围	全部文件属性	文件名或路径片段	仅限可执行程序
速度	慢（实时遍历）	快（基于索引数据库）	极快（路径变量查找）
实时性	实时精准	非实时（需定期更新）	实时（路径固定）
灵活性	非常高	中（简单模糊匹配）	低（只适用命令路径）
常用场景	精确、多条件搜索	快速粗略定位文件	命令或脚本路径确认

1. 实际场景举例说明

场景需求	推荐命令
实时查找昨天修改的文件	find (-mtime)
快速定位某个文件路径	locate
快速确认git命令执行路径	which
查找所有大于1G的视频文件	find (-size)
检查系统中所有Python脚本位置	locate (*.py)

2. 使用建议与注意事项

主要作用：

1. **find实时精准，适合日常精细文件管理；**
2. **locate快速粗略，日常定位文件高效便捷，但需定期更新数据库；**
3. **which明确专用于查找可执行命令路径，排查环境冲突时极为有效。**

权限解释

Linux系统的权限机制用于控制用户对文件和目录的访问与操作，分为读(r)、写(w)、执行(x)三种权限，分别赋予给文件所有者、所属用户组和其他用户

1. 权限表示方法

Linux权限采用三个类别，每个类别都有3个权限位，总共9位。

- **类别：**
  - 用户（Owner）
  - 用户组（Group）
  - 其他人（Others）
- **权限类型：**
  - 读权限（r）
  - 写权限（w）
  - 执行权限（x）

符号	文件（File）含义	目录（Directory）含义
r	可读取文件内容	可列出目录中的文件名
w	可修改文件内容	可在目录中创建、删除、重命名文件
x	可执行文件 - 程序/脚本	可进入目录（如：cd进入该目录）

2. 权限表示实例

命令ls –l输出示例为例来解释权限：

```
$ ls -l
total 6834148
-rw-r--r-- 1 zhangmch baoym_group 6998142667 Nov 29  2023 20231129.RData
drwxr-xr-x 2 zhangmch baoym_group       4096 Nov 29  2023 data
drwxr-xr-x 4 zhangmch baoym_group       4096 Nov 23  2023 GSE130001_RAW
drwxr-xr-x 2 zhangmch baoym_group       4096 Jun  4 16:49 output
drwxr-xr-x 2 zhangmch baoym_group       4096 Jun  4 16:49 processed
drwxr-xr-x 2 zhangmch baoym_group       4096 Jun  4 16:49 raw
(base) zhangmch@bserver 16:22:27 /data40T/zhangmch/single-cell/data
```

- \- 普通文件
- d 目录
- l 链接文件

rw- | r-- | r--

读写 只读 只读

权限组合	二进制	八进制
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

权限管理之chmod

chmod（change mode）的主要作用为修改文件或目录的访问权限（读、写、执行）

常用权限表示方法：

- **数字模式（推荐使用）**

数字	对应权限	含义
4	r	读权限
2	w	写权限
1	x	执行权限

举例说明：

chmod 644 file.txt	文件	rw-r--r--
chmod 755 script.sh	脚本	rwxr-xr-x
chmod 700 secret	目录/文件	rwx------

- **符号模式**
  - **+：增加权限**　　**-：去除权限**　　**=：设定权限**

举例说明：

chmod u+x script.sh	所有者增加执行权限
chmod g+w data.txt	用户组增加写权限
chmod o-r confidential	其他用户去除读权限
chmod a=r file.txt	所有人权限设为只读

典型案例演示：

（1）给脚本增加执行权限

```
$ chmod +x 20231129.RData
(base) zhangmch@bserver 16:48:59 /data40T/zhangmch/single-cell/data
```

（2）批量修改权限为755

```
$ chmod -R 755 ./*
(base) zhangmch@bserver 16:49:39 /data40T/zhangmch/single-cell/data
```

（3）只允许所有者访问

```
$ chmod 600 ./*
(base) zhangmch@bserver 16:50:24 /data40T/zhangmch/single-cell/data
```

（4）去掉其他人的所有权限

```
$ chmod o-rwx ./*
(base) zhangmch@bserver 16:51:06 /data40T/zhangmch/single-cell/data
```

使用注意事项：

1. **递归修改**：使用 -R 参数对目录及所有子文件递归修改
2. **权限确认**：修改后建议通过 ls -l 验证权限是否符合预期
3. **安全考量**：
   ① 避免给关键文件目录赋予过于宽松的权限，如 777
   ② 对私钥和敏感配置文件应严格限制权限 (600 或 700)

权限管理之chown

chown（change mode）的主要作用为修改文件或目录的所有者和用户组

基本用法：

chown [选项] 用户[:组] 文件或目录

- **用户和组之间用冒号:隔开。**
- **如果只修改所有者，省略:组；只修改组，则用:组。**

常用选项：

选项	含义	示例
-R	递归修改目录及子目录	chown -R user:group dir
-v	显示详细的修改过程	chown -v user file.txt
-c	显示修改成功的项目	chown -c user file.txt

基本示例：

（1）修改文件所有者为用户zhangmc

```
$ chown zhangmc 20231129.RData
chown: changing ownership of '20231129.RData': Operation not permitted
(base) zhangmch@bserver 17:06:02 /data40T/zhangmch/single-cell/data
```

（2）同时修改所有者和组

```
$ chown zhangmc:bioinfo script.sh
chown: invalid group: 'zhangmc:bioinfo'
(base) zhangmch@bserver 17:06:43 /data40T/zhangmch/single-cell/data
```

（3）仅修改用户组

```
$ chown :bioinfo data.txt
chown: invalid group: ':bioinfo'
(base) zhangmch@bserver 17:07:11 /data40T/zhangmch/single-cell/data
```

（4）递归修改目录所有权

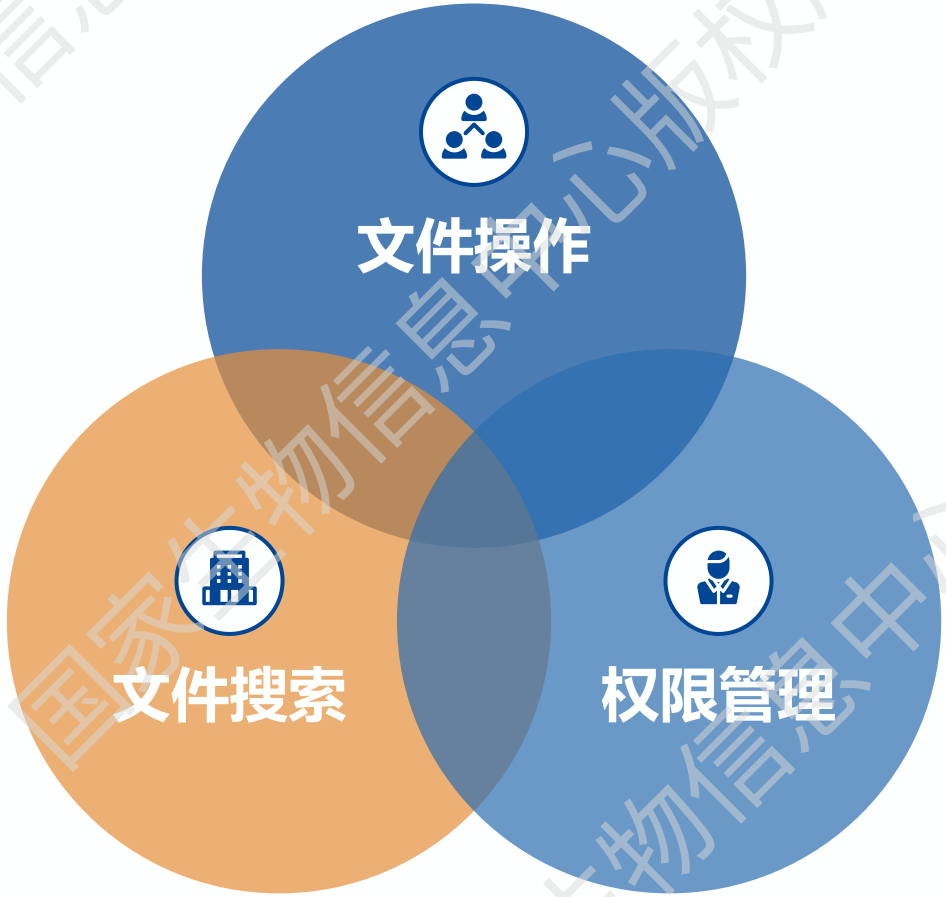
```
$ chown -R zhangmc:bioinfo ~/project
chown: invalid group: 'zhangmc:bioinfo'
(base) zhangmch@bserver 17:07:45 /data40T/zhangmch/single-cell/data
```

使用注意事项：

- **权限要求：**
  - **普通用户只能将文件的组修改为自己所属的其他组（需为组成员）。**
  - **修改文件所有者通常需要管理员权限 (sudo)。**
- **谨慎使用-R选项：**
  - **递归修改影响所有子目录和文件，需谨慎防止误操作。**
- **安全考量：**
  - **合理设置所有者和组，避免权限混乱造成的安全风险。**

本模块小结

文件管理与权限控制模块的主要目标是：掌握文件的复制、移动、查找与权限管理



1. 文件操作

- cp
- mv
- file
- du –sh
- df –h
- rm – 上个模块介绍

2. 文件搜索

- find
- locate
- which

3. 权限管理

- chmod
- chown
- 权限解释（rwx）
- ls –l – 上个模块介绍

实战练习

假设你需要为一个“生物信息学项目”创建初始目录结构，并在其中创建一些脚本和数据文件，然后对这些文件设置合理的权限，最后通过搜索和查看命令来验证。

1. **项目目录结构创建**
    1. 在用户的 home 目录下创建一个名为 bioinfo_project 的项目顶层目录，并在其中新建 scripts、data、results 三个子目录。
    2. 进入 scripts 目录，创建两个示例脚本文件：align_reads.sh 和 analyze_variants.py。
    3. 进入 data 目录，创建两个示例数据文件：sample1.fastq 和 sample2.fastq。
    4. 检查整个项目目录结构，验证上述文件和目录是否正确创建。
2. **文件复制、移动与删除**
    1. 将 scripts 目录及其所有内容复制到一个名为 backup 的新目录下。
    2. 在 backup/scripts 目录中，将 align_reads.sh 重命名为 align_reads_v1.sh。
    3. 将 data/sample2.fastq 移动到 results 目录下。
    4. 删除 results 目录中的临时文件 temp.txt（假设该文件可能存在，也可能不存在）。
    5. 查看 data 目录和 results 目录当前的磁盘占用情况。
3. **文件搜索**
    1. 在整个 bioinfo_project 目录中，查找所有以 .sh 结尾的文件。
    2. 在系统中查找 python3 可执行文件的路径。
    3. 更新系统的 locate 索引数据库后，使用 locate 检索 sample1.fastq 的路径。

实战练习

假设你需要为一个“生物信息学项目”创建初始目录结构，并在其中创建一些脚本和数据文件，然后对这些文件设置合理的权限，最后通过搜索和查看命令来验证。

4. **权限管理**
    1. 查看 scripts 目录下所有文件的当前权限和归属关系。
    2. 为所有以 .sh 结尾的脚本文件添加可执行权限。
    3. 假设系统中已有名为 bioinfo 的用户组，将 data 目录及其所有内容的归属更改为所属用户为当前用户、所属组为 bioinfo。
    4. 为 data 目录及其所有文件赋予“组写”权限。
    5. 切换到另一个用户（比如 guest），验证该用户是否可以在data目录中创建新文件（前提是该用户属于bioinfo组，否则会提示权限不足）。
5. **复盘与检查**
    1. 输出 bioinfo_project 目录的完整文件结构，以确认所有文件和子目录均在预期位置。
    2. 检查当前挂载点的磁盘空间，并确认 bioinfo_project 目录所在分区是否有足够空间。
    3. 使用 locate 验证 sample3.fastq 文件是否已被系统索引（注意需要先运行 updatedb）。
    4. 在 scripts 目录中运行 align_reads.sh 脚本（可以提前向脚本中添加简单的 echo 输出以便验证执行）。

实战要求及答案路径：/home/big/linux/pratical_training/2-File_Management_and_Permission_Control_Module

03

文本处理与内容提取

文件内容查看之head

head的主要作用是显示文本文件开头多行内容，默认输出前10行，常用于快速查看日志或数据文件的头部概况

基本用法：

head [选项] [文件...]

常用选项：

选项	含义	示例
-n <行数>	指定输出前 <行数> 行	head -n 5 logfile.txt
-c <字节数>	指定输出前 <字节数> 字节	head -c 100 data.csv
-q	静默模式，不输出文件名前缀	head -q file1 file2
-v	输出文件名前缀（多文件时可区分来源）	head -v file1 file2

注意事项：

- **对于非常大的文件，仅输出头部避免全量加载造成延迟**
- **如果需要查看尾部，应使用 tail 命令**
- **若想分页查看，可结合管道：**
  - **head -n 50 largefile.txt | less （这部分内容会在模块4中涉及）**

基本示例：

（1）默认查看前10行 – 默认不加-n时，head等同于head –n 10

```
$ head align.py
import os
import sys


if __name__ == '__main__':
        bin_path = "bismark"
        output_path = sys.argv[2]
        input_path = sys.argv[1]
        contraction_path = "/data40T/zongwt/genome/hg19_lambda/"
        params = "--bowtie2 --non_directional -p 4 -unmapped --bam --quiet"
(base) zhangmch@bserver 17:41:03 /data40T/zhangmch/single-cell/scmethbank
```

（2）查看前128字节 - -c适用于查看二进制文件或固定字节段

```
$ head -c 128 single-cell.R
#单细胞scRNA_Seq代码
library(Seurat)
library(mindr)
#上面的包在/home/zhangmch/R4.3.0-package/mindr_1.3.2.tar.gz中
lib(base) zhangmch@bserver 17:40:26 /data40T/zhangmch/single-cell/script
```

（3）同时查看多个文件的前3行 – 多文件场景配合-v/-q可控制是否显示文件名标签

```
$ head -n 3 align.py QC.py bis_result.py
==> align.py <==
import os
import sys

==> QC.py <==
import os
import sys


==> bis_result.py <==
import os
import sys

(base) zhangmch@bserver 17:41:40 /data40T/zhangmch/single-cell/scmethbank
```

文件内容查看之tail

tail的主要作用是显示文本文件的最后几行，特别适用于查看日志文件或实时更新文件的末尾内容

基本用法：

tail [选项] [文件...]

常用选项：

选项	含义	示例
-n <行数>	显示文件的最后 <行数> 行	tail -n 20 logfile.txt
-f	实时监视文件内容，显示新追加的内容	tail -f /var/log/syslog
-c <字节数>	显示文件的最后 <字节数> 字节	tail -c 100 file.bin
-q	静默模式，多个文件时不显示文件名	tail -q file1.txt file2.txt
-v	显示文件名（多个文件时默认显示）	tail –v file1.txt file2.txt

使用场景：

- **查看日志文件末尾：实时查看系统或应用的日志文件（如 syslog, access.log）**
- **调试脚本输出：监视脚本输出的动态变化，及时捕捉错误或数据更新**
- **处理大文件：对于大文件或持续更新的文件，tail 可节省内存资源，避免一次性加载整个文件**

基本示例：

（1）显示文件最后十行（默认输出最后10行 - 查看日志文件的最新记录）

```
$ tail /p300s/methbank_baoym/zhangmch/scWGBS/SRR20307773_1.fastq
+SRR20307773.29083194 29083194 length=150
AAFFFJJJJJJJJF#JJJJJJJJJJJJJJJFJJJJJJJJJ#JJJFJJJJJJFJJJJJJJJJJFJ<FJFJJJJJJJJJJJJJJJJAFJJJJJJJ7A--777
@SRR20307773.29083195 29083195 length=150
TTTTTTAGTGTATTNTTTTAGATTTGGTAGTAAATTAGTTNTTAGGTTGTTTAGGATAGTTTTGGTTGTGATAATTGAAGAAATTTTGTGTATAGGTAT
+SRR20307773.29083195 29083195 length=150
AAFFFJJJJJJJJF#JJJJJJJJJJJJJJJFJJJJJJJJ#JJJJJJJJJAJJJJFJJJJ<JFFAJFJJJJJJJJFJA7FAFFJJJJJJJJJJFJJ<J
@SRR20307773.29083196 29083196 length=150
CCTAACAAAACTATNCAAATAATTACCATATATAACCAAANAAAAAAACACTACTCTAATAAAACTAACTATAAATAATTCTCAAACAACTACACATAT
+SRR20307773.29083196 29083196 length=150
AAAFFJJJJJJFJJ#JJJJJJAJJAJJFJFJJJJJJJJFF#JJJ<JJJFFJJFFFJFFFJJJAJJJJAJFFFFJJJFF<-FJJAFFFFAFJFJFJ--AJ
/p300s/methbank_baoym/zhangmch/scWGBS
```

（2）实时查看文件更新（用于查看持续更新的日志文件）

```
Bismark completed in 0d 9h 20m 22s

====================
Bismark run complete
====================

Waiting for data... (interrupt to abort)
```

（3）实时监控并输出特定字节内容

```
$ tail -c 200 nohup.out
context:       0.9%
C methylated in CHH context:     1.1%
C methylated in unknown context (CN or CHN):     4.9%


Bismark completed in 0d 9h 20m 22s

====================
Bismark run complete
====================
```

文件内容查看之less

less是一个强大的文本查看器，用于分页显示文件内容，允许前后翻页和快速搜索，常用于查看大文件

基本用法：

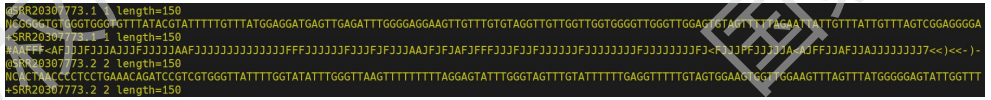
less [选项] [文件...]

常用选项：（按q退出less查看模式）

选项	含义	示例
-n	不显示行号	less -n file.txt
-N	显示行号	less -N file.txt
-S	不自动换行，超长行将直接显示为一行	less -S file.txt
-X	不清除屏幕，退出时保留显示内容	less -X file.txt
-F	如果文件内容可以在一页内显示，直接显示文件内容	less -F file.txt
-I	忽略大小写进行搜索	less -I file.txt
/pattern	搜索模式：在文件中向下搜索匹配的文本	/error（查找文件中的error）
?pattern	向上搜索模式：查找文本的反向匹配	?warning（查找反向的warning）

基本示例：

（1）查看文件内容并分页显示



- 分页显示日志文件，支持前后翻页

（2）查看文件内容并显示行号

- 以行号显示文件内容，方便定位行数

（3）不自动换行，显示超长行

- 查看csv文件时，防止自动换行，长行数据不会被折叠

（4）搜索文件内容 - SRR

- 搜索并高亮显示文件中的SRR

文件内容查看之cat

cat命令用于连接文件并输出内容到标准输出，主要用于查看小文件的内容，或者将多个文件的内容合并输出

基本用法：

cat [选项] [文件...]

常用选项：

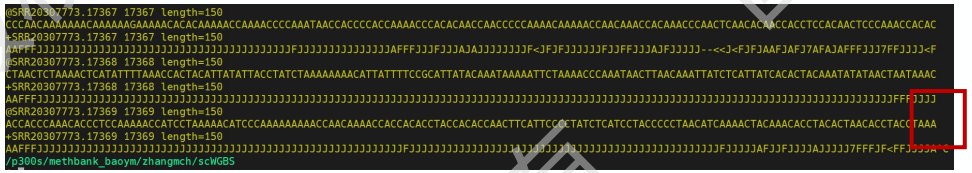
功能	示例	说明
显示文件内容	cat file.txt	显示文件 file.txt 的全部内容
合并多个文件	cat file1.txt file2.txt	合并 file1.txt 和 file2.txt 并输出
输出到另一个文件	cat file1.txt > file2.txt	将 file1.txt 的内容输出到 file2.txt
追加内容到文件末尾	cat file1.txt >> file2.txt	将 file1.txt 的内容追加到 file2.txt
显示行号	cat -n file.txt	在输出内容前显示每一行的行号
显示非打印字符	cat –v file.txt	显示文件中的不可见字符（如换行符）

注意事项：

- **性能问题：对于较大的文件，cat会一次性输出文件全部内容，消耗大量内存，应考虑使用less等分页命令**
- **输出重定向：**
  - **使用 > 时会覆盖目标文件内容，请谨慎操作**
  - **使用>>时会追加内容**

基本示例：

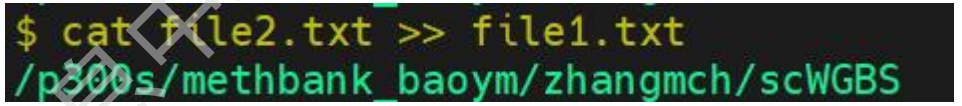
（1）显示文件内容 – cat SRR20307773_1.fastq文件会直接输出文件file.txt的内容



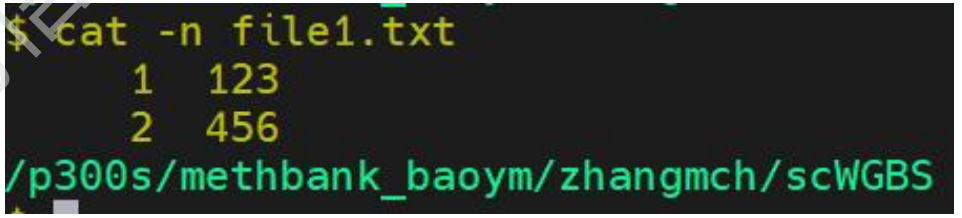
（2）重定向文件内容到另一个文件 – cat file1.txt > file2.txt

将file1.txt的内容写入到file2.txt中；如果file2.txt已经存在，则覆盖它

（3）追加内容到文件末尾 – 假设file1.txt为123，file2.txt为456，则cat file2.txt >> file1.txt的内容就是456 回车 123



（4）显示带行号的内容 – cat –n file.txt



文件内容查看之grep

grep支持在文本文件中根据指定模式（字符串或正则表达式）搜索并打印匹配行，是非常强大的文本搜索工具

基本用法：

grep [选项] PATTERN [文件...]

常用选项：

选项	含义	示例
-i	忽略大小写匹配	grep -i "error" logfile.txt
-v	反向匹配，只显示不包含模式的行	grep -v DEBUG logfile.txt
-n	显示匹配行的行号	grep -n "TODO" script.sh
-r/-R	递归搜索目录下所有文件	grep -R "main()" src/
-c	只输出匹配行的计数	grep -c "WARNING" syslog
-l	只列出包含匹配的文件名	grep -l "password" *.conf

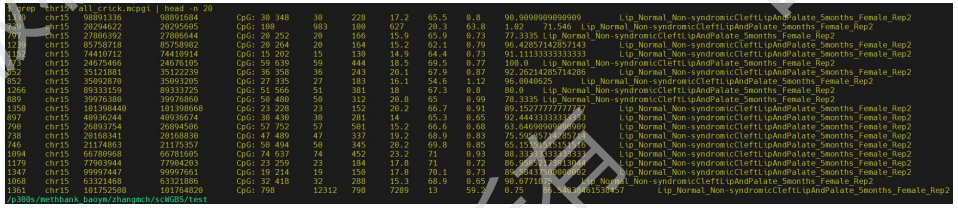
注意事项：

- **正则语法差异：默认使用基本正则，需 -E 才能启用扩展正则。**
- **大文件性能：对于超大文件或二进制文件，建议先用 head/tail 限制范围，或使用 fgrep（grep -F）进行固定字符串匹配。**
- **文件名匹配：-l 列出文件名时，不会输出具体内容。**
- **高亮支持：部分终端需加 --color=always 才能正确高亮。**

基本示例：

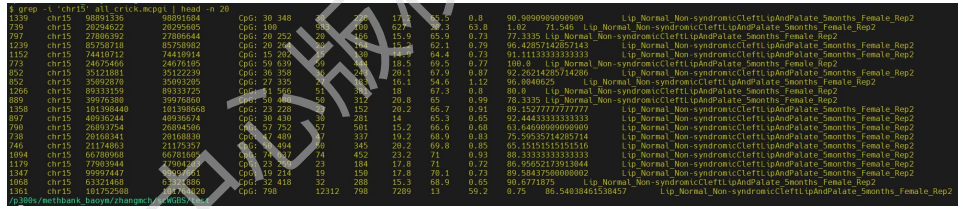
（1）基本搜索：输出所有包含“chr15”的行

grep 'chr15' all_crick.mcpgi | head -n 20



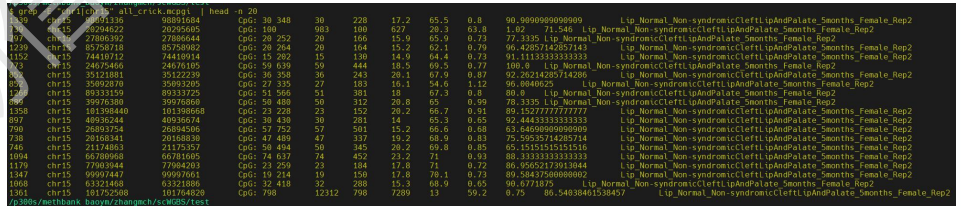
（2）忽略大小写搜索：不考虑大小写的情况下匹配chr15字符

grep -i 'chr15' all_crick.mcpgi | head -n 20



（2）扩展正则，同时匹配chr1和chr15字符

grep -E "chr1|chr15" all_crick.mcpgi | head -n 20



文件属性统计之wc、sort和uniq

01. wc

统计文件的行数、单词数、字符数

选项	含义	示例
-l	统计行数	wc -l file.txt
-w	统计单词数	wc -w file.txt
-c	统计字节数	wc -c file.txt
-m	统计字符数	wc -m file.txt

最常用的是wc -l来统计行数

02. sort

按字母、数字或其他顺序
对文本文件中的内容进行排序

选项	含义	示例
-n	按数字排序	sort -n numbers.txt
-r	逆序排序	sort -r file.txt
-k	按指定列排序	sort -k 2 file.txt
-u	去除重复行	sort -u file.txt

最常用的是sort -k对比对文件排序

03. uniq

去除文件中重复的行，
只保留唯一的行

选项	含义	示例
-c	统计每一行出现的次数	uniq -c file.txt
-d	只显示重复出现的行	uniq -d file.txt
-u	只显示不重复的行	uniq -u file.txt

最常用的是uniq -c去重显示行次数

文件属性统计之wc、sort和uniq

01. wc

（1）查看文件行数：wc –l file.txt

```
$ wc -l SRR20307773_1.fastq
116332784 SRR20307773_1.fastq
/p300s/methbank_baoym/zhangmch/scWGBS
```

（2）查看文件单词数：wc –w file.txt

```
$ wc -w SRR20307773_1.fastq
232665568 SRR20307773_1.fastq
/p300s/methbank_baoym/zhangmch/scWGBS
```

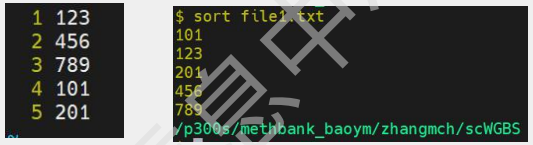
（3）查看文件字节数：wc –c file.txt

```
$ wc -c SRR20307773_1.fastq
11181669244 SRR20307773_1.fastq
/p300s/methbank_baoym/zhangmch/scWGBS
```

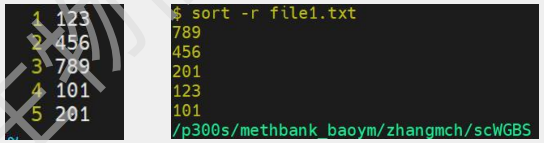
02. sort

（1）按照字母顺序排序：sort file.txt

（2）按照数字大小排序：sort –n numbers.txt

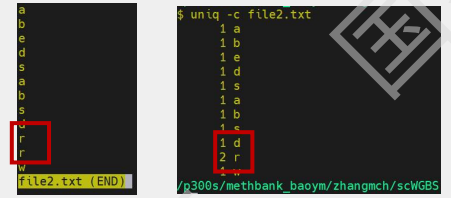


（3）逆序排序：sort –r file.txt



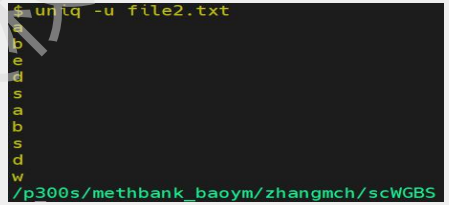
03. uniq

（1）去重并显示每行出现的次数 – uniq -c file2.txt



（2）显示重复的行 – uniq –d file2.txt

（3）显示唯一的行 – uniq –u file2.txt



必须是连续出现的行才可以将次数进行累计

命令组合使用：

- **结合 sort 和 uniq 去除文件中的重复行并排序 -> sort file.txt | uniq**
- **统计文件中每个单词的出现次数 -> sort file.txt | uniq –c**
- **备注：管道符会在下一个章节中讲解**

文件字段处理之cut

cut用于从文本行中提取指定的字段或字符范围，常用于拆分和筛选以定界符分割的数据列

基本用法：

cut [选项] [文件...]

常用选项：

选项	含义	示例
-f <字段列表>	按字段号提取，以 - 或 , 分隔多个字段	cut -f 1,3 file.tsv
-d <分隔符>	指定字段分隔符（默认是制表符 \t）	cut -d',' -f2 file.csv
-c <字符范围>	按字符位置提取，如 1-5、-4、3-	cut -c1-10 file.txt
--complement	取反选项，输出不在指定字段/字符范围内的内容	cut -d: --complement -f1 /etc/passwd

注意事项：

- **默认分隔符：若不指定 -d，cut 会将制表符视作分隔符，其他分隔符需显式指定**
- **字段越界：请求的字段或字符范围超出行长度，则只输出现有部分，不报错**
- **兼容性：GNU cut 与 BSD cut 在处理 -c 范围表达式时行为略有差异，请根据环境测试**
  - **GNU cut在绝大多数实现中按字节截取，而BSD cut是字符感知的，会正确按字符分隔。**

基本示例：

```
name,age,city,score
Alice,30,New York,85
Bob,25,Los Angeles,90
Charlie,28,Chicago,88
David,35,Houston,92
```

（1）提取第一列和第四列

cut -d',' -f1,4 data.csv

```
$ cut -d',' -f1,4 data.csv
name,score
Alice,85
Bob,90
Charlie,88
David,92
/p300s/methbank_baoym/zhangmch/scWGBS
```

（2）提取每行前五个字符

cut -c1-5 data.csv

```
$ cut -c1-5 data.csv
name,
Alice
Bob,2
Charl
David
/p300s/methbank_baoym/zhangmch/scWGBS
```

文件字段处理之paste

paste命令是将多个文件的对应行并排合并，以指定分隔符合并成一行输出，常用于将列文件“粘贴”在一起

基本用法：

paste [选项] [文件...]

常用选项：

选项	含义	示例
-d <分隔符列表>	指定字段分隔符序列，默认使用制表符 (\t)	paste -d',' file1.txt file2.txt
-s	将单个文件的所有行“串联”并排输出，而非行对行合并	paste -s file.txt
-z	对空行也输出分隔符（GNU 特殊选项，不是所有实现均支持）	paste -z file1.txt file2.txt

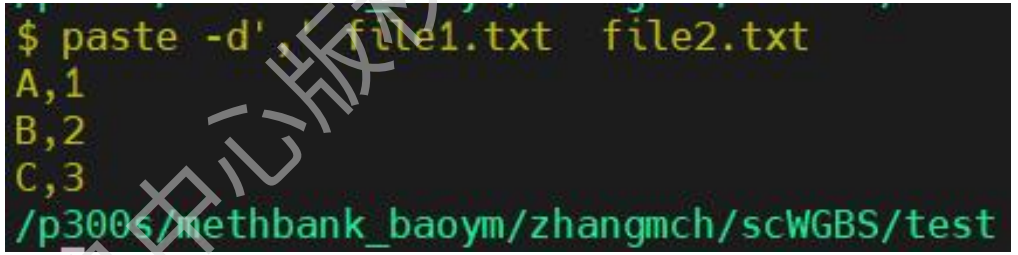
注意事项：

- **行数不一致：当文件行数不一致时，paste 会对短文件用空字段补齐，不报错。**
- **分隔符循环：-d 后提供多个字符时，paste 会循环使用这些分隔符。**
- **兼容性：大多数系统支持基本选项，-z 仅在 GNU paste 中可用。**

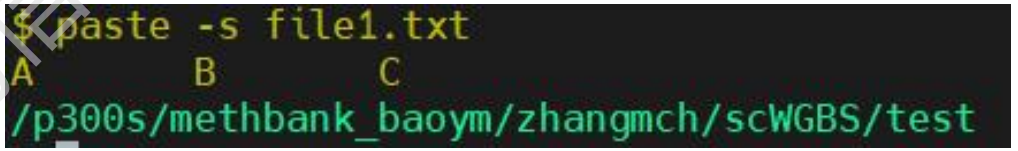
基本示例：

（1）并排合并两列文件：paste file1.txt file2.txt

（2）使用逗号分隔：paste -d',' file1.txt file2.txt



（3）串联单文件所有行：paste –s file1.txt



文件字段处理之tr

tr（translate）用于替换、删除或压缩标准输入中的字符，常用于清洗和转换文本流

基本用法：

tail [选项] [文件...]

常用选项：

选项	含义	示例
-d	删除所有出现在源字符集合中的字符	tr -d '\r' < file.txt
-s	压缩 - 将重复出现的源字符集合中的字符合并为一个	tr -s ' ' < file.txt
-c	补集 - 作用于不在源字符集合中的所有字符	tr -c '[:alnum:]' '\n' < data.txt
-t	截断 - 当目标集合比源集合短时，截断源集合	通常与-s联合使用（GNU）

字符集合语法：

- **简单列表：abc 表示字符 a、b、c**
- **范围：a-z 表示所有小写字母，A-Z 表示所有大写字母**
- **POSIX 类：**
  - **[:upper:] 大写字母　　[:lower:] 小写字母**
  - **[:digit:] 数字　　[:space:] 空白字符（空格、制表符、换行等）**

基本示例：

（1）将小写字母转换为大写

```
$ echo 'hello world' | tr 'a-z' 'A-Z'
HELLO WORLD
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

（2）删除回车符，尤其适用于Windows->Unix转换

```
$ tr -d '\r' < windows.txt > unix.txt
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

（3）压缩连续空格为单个空格

```
$ echo 'A   B     C' | tr -s ' '
A B C
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

（4）将非字母数字字符替换为换行

```
$ tr -c '[:alnum:]' '\n' < data.txt
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

注意事项：

- **字符集匹配：确保引用字符集合时加上引号，避免 shell 扩展**
- **目标长度：当目标集合短于源集合时，需要加 -t 或注意截断行为**

文件字段处理之awk

awk是一门强大的文本处理语言，用于按字段解析、筛选、格式化和计算文本数据

基本用法：

awk 'pattern { action }' [选项] [文件]

- **pattern：匹配条件（如行号、正则表达式、字段值等）**
- **action：对匹配行执行的操作（如打印、算术运算、内置函数等）**
- **默认分隔符：空格或制表符；可用-F指定其他分隔符**

常用选项：

选项	含义	示例
-F fs	设置输入字段分隔符 fs（如逗号、冒号等）	awk -F',' '{...}' file.csv
-v var=val	在脚本外定义变量 var，值为 val	awk -v OFS=',' '{...}' file.txt
BEGIN{}	在读取文件前先执行一次的动作	awk 'BEGIN{print "Start"} ...' file
END{}	在处理完所有行后执行一次的动作	awk '... END{print "Done"}' file

基本示例：

（1）打印每行第二列：awk '{ print $2 }' all_crick.mcpgi

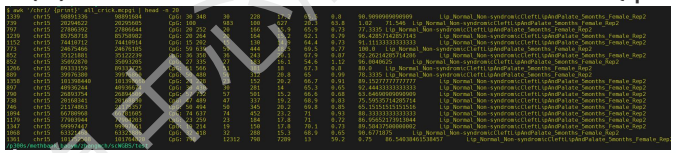
```
$ awk '{print $2}' all_crick.mcpgi | head -n 5
chrom
chr15
chr15
chr15
chr15
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

（2）按逗号分隔并打印第一与第三列：

awk -v OFS=',' '{print $1, $3}' all_crick.mcpgi

```
$ awk -v OFS=',' '{print $1, $3}' all_crick.mcpgi | head -n 5
#bin,chromStart
1339,98891336
739,20294622
797,27806392
1239,85758718
/p300s/methbank_baoym/zhangmch/scWGBS/test
```

（3）只打印包含“chr1”的行：awk '/error/ { print }' logfile.txt



（4）累加并求和：

awk '{ sum += $2 } END { print "Total score:", sum }' scores.txt

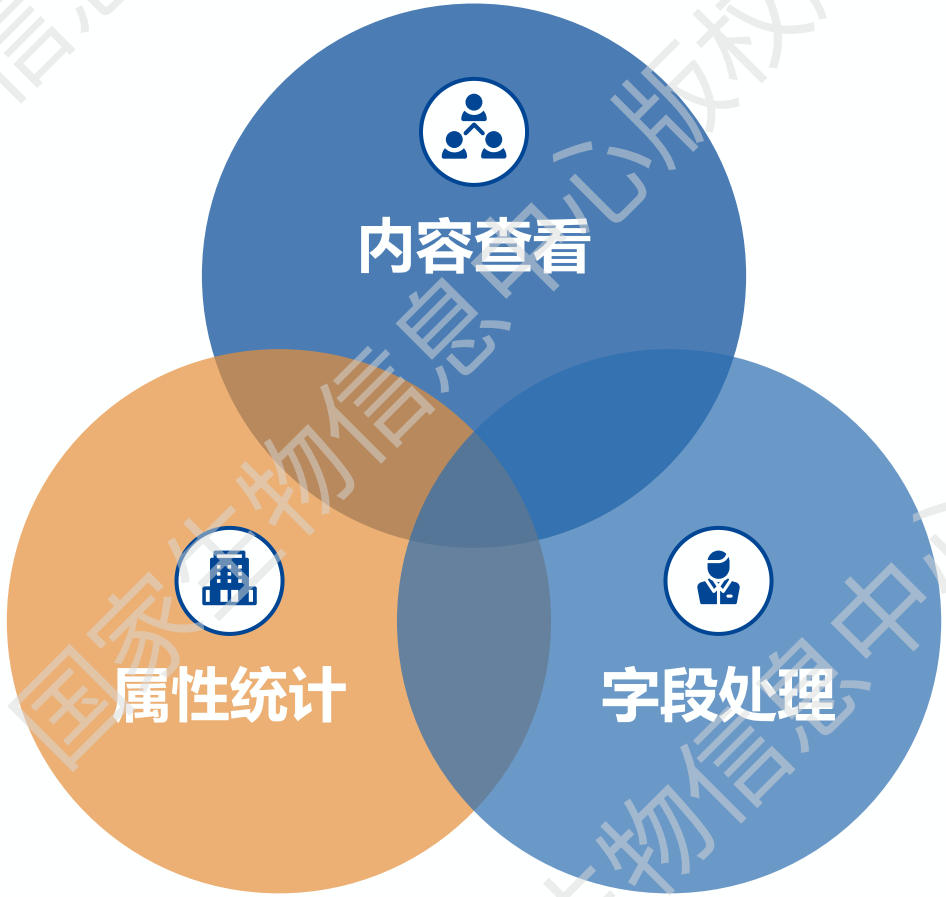
- **字段下标：列号从 1 开始，$0 代表整行文本。**
- **分隔符兼容：处理逗号或其他定界符时务必使用 -F 并设置 OFS**
- **性能考虑：对超大文件可用 gawk 的 –bignum 或脚本分段处理**
- **便捷脚本：可将复杂脚本写入文件，使用 awk -f script.awk file 调用**

本模块小结

文本处理与内容提取模块的主要目标是掌握科研常见文本文件的查看与预处理

1. 内容查看

- head
- tail
- less
- cat
- grep



2. 属性统计

- wc
- sort
- uniq

3. 字段处理

- cut
- awk
- tr
- paste

实战练习

假设你收到两份科研常见文本文件（放置再~/workspace/目录下），一份是多行实验日志experiment.log（格式为<时间戳><级别><消息>），一份是逗号分隔值文件samples.csv（格式为SampleID,Value,Group）

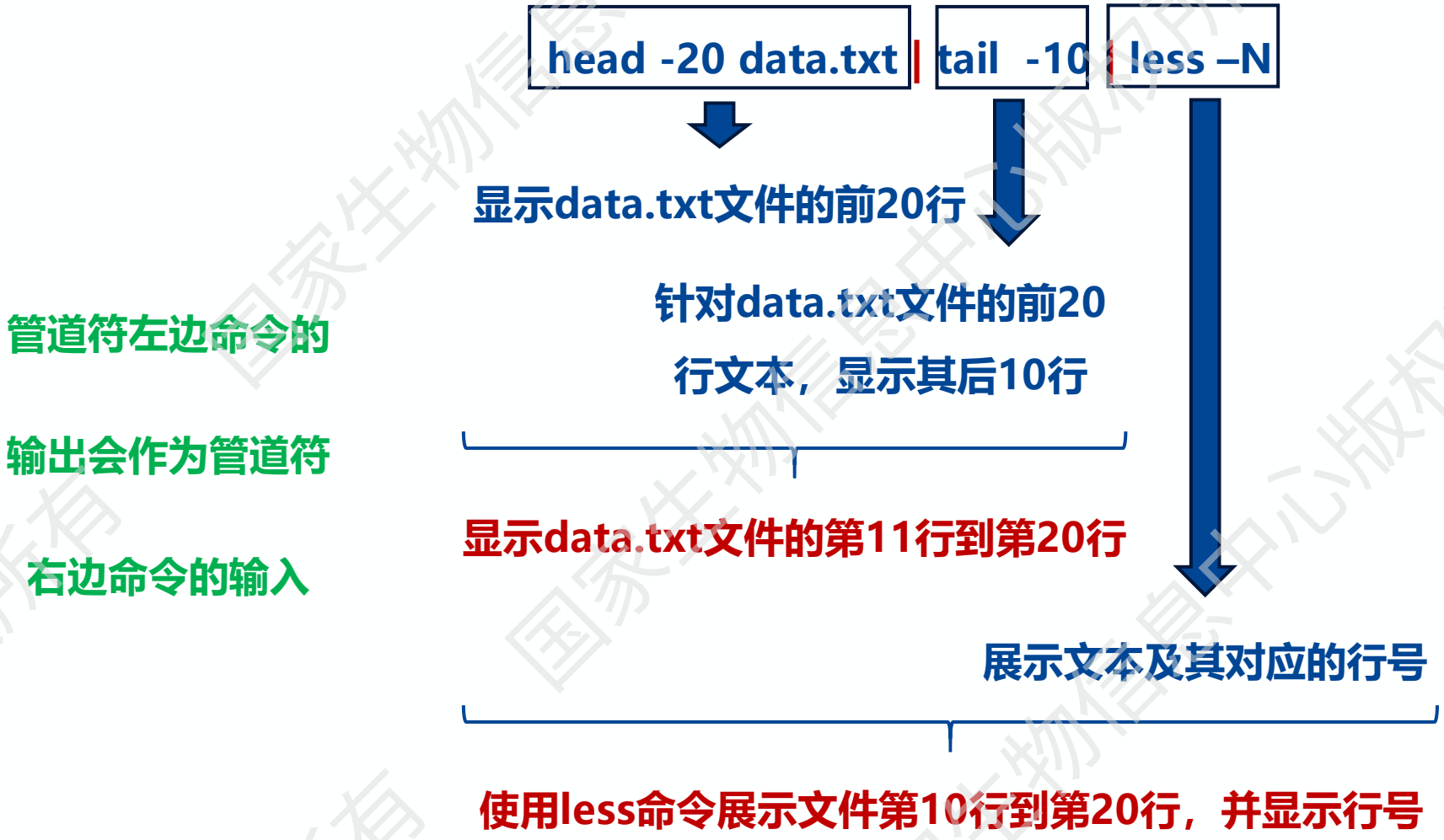
1. **内容查看**
    1. 查看 experiment.log 的前 5 行。
    2. 查看 experiment.log 的最后 5 行。
    3. 用 less 打开 experiment.log，并在其中搜索 “ERROR”。
    4. 合并 samples.csv 和 controls.csv（假设同目录下有一个结构相同的 controls.csv），只预览前 8 行。
2. **属性统计**
    1. 统计 experiment.log 的总行数。
    2. 统计 experiment.log 中每种日志级别（第二列）的出现次数。
    3. 统计 samples.csv 中每个组别（第三列）的样本数。
3. **字段处理**
    1. 从 experiment.log 中提取时间戳（第 1 列）和消息（第 3 列及以后），预览前 5 行。
    2. 使用 awk 从 samples.csv 过滤出 Value > 50 的样本，并以“SampleID Value”格式输出。
    3. 将 samples.csv 的逗号分隔替换为制表符，生成 samples.tsv，并预览前 5 行。
    4. 将上一步得到的 samples.tsv 三列分别拆分到 col1.txt、col2.txt、col3.txt，然后用 paste 合并并预览前 5 行。

实战要求及答案路径：/home/big/linux/pratical_training/3-Text_Processing_and_Content_Extraction_Module

04 重定向控制与管道操作

Linux神器之管道符 |

管道符 | 将左侧命令的输出流连接到右侧命令的输入流，相当于在内存中开辟一个缓冲区（匿名管道），自动完成读写同步



标准流与文件描述符

标准输入 (fd 0)、标准输出 (fd 1) 和标准错误 (fd 2) 分别对应命令的输入、正常输出和错误输出，并可通过重定向灵活地将它们导向文件或其他流

基本重定向符号

重定向通过符号（如 >, >>, <, 2> 等）将命令的标准输入、标准输出或标准错误流指向或合并到文件或其他流，以实现数据流的灵活控制

符号	含义	示例
>	将 stdout （fd2）重定向到文件，覆盖写入	ls > files.txt
>>	将 stdout（fd2）重定向到文件，追加写入	echo "log" >> app.log
<	将文件内容重定向到 stdin（fd0）	sort < unsorted.txt
2>	将 stderr （fd2）重定向到文件，覆盖写入	grep 'foo' nonexist.txt 2> error.log
2>>	将 stderr （fd2）重定向到文件，追加写入	make 2>> build_errors.log
&> 或 > file 2>&1	同时将 stdout 和 stderr 重定向到同一文件	cmd &> all.log 或 cmd > all.log 2>&1
|	将前一个命令的 stdout（fd2）连接到后一个命令的 stdin（fd1）	cat file.txt | grep "error"

常见案例

01 覆盖 VS 追加

① 覆盖：echo "first" > demo.txt　　# demo.txt: "first\n"

② 追加：echo "second" >> demo.txt　　# demo.txt: "first\n second\n"

02 错误与正常输出分别记录

sh./run_analysis.sh > out.log 2> err.log

① > 代表正常输出记录到out.log　　2> 代表错误输出记录到err.log

03 合并错误与正常输出

./run_analysis.sh > all.log 2>&1

2>&1代表正常输出记录和错误输出记录均被重定向到了all.log文件中

04 将管道和重定向组合

grep "WARN" /var/log/syslog | tee warnings.txt | wc -l > count.txt

tee会将前一个命令接收到的stdout输出到文件，同时继续传递给下一个命令

05 多级fd重定向（难点）

cmd 2>&1 >> log.txt

2>&1：将stderr重定向到stdout，但目前的stdout仍然是终端，所以stderr被输出到了终端；>>：然后stdout被追加写入到log.txt文件中

本模块小结

重定向控制与管道操作模块的主要目标是掌握数据流向控制与命令组合处理

1 输入输出重定向

- **>**

  将stdout重定向到文件，覆盖写入

- **>>**

  将stdout重定向到文件，追加写入

- **<**

  将文件内容重定向为stdin

2 管道符

- **定义**

  将前一个命令的输出作为后一个命令的输入

- **多个命令串联处理数据**

  cat file.txt | grep cancer | wc –l

  ① cat file.txt 输出全文

  ② grep cancer 筛选包含关键词行

  ③ wc -l 统计匹配行数

3 要点说明

- **> / >>控制输出去向**
- **<控制输入来源**
- **| 将命令链接成“数据处理管道”**
- **常用于日志分析、数据筛选、自动化脚本等场景**

实战练习

你是一名系统管理员，需要从服务器日志sample.log中提取关键信息并生成报告文件，请完成如下任务

1. 提取日志中包含关键词 “ERROR” 的所有行
2. 追加提取 “WARN” 日志，不覆盖前面的文件
3. 统计包含 “ERROR” 和 “WARN” 的行总数，输出到 count.txt
4. 模拟一个错误命令，捕获错误信息保存到 err.log
5. 提取 “CRITICAL” 日志，保存到 critical.log 并显示数量
6. 进阶挑战：
    1. 使用 sort + uniq -c 对错误类型进行频率分析
    2. 使用 < 读取文件
    3. 使用 2>&1 >> combined.log 合并所有输出信息

实战要求及答案路径：/home/big/linux/pratical_training/4-Redirection_Control_and_Pipeline_Operations_Module

05 批量处理与自动化脚本

脚本语法

重复的命令可以结构化，脚本就是让计算机为你重复劳动

我们学会了用一条命令处理一个文件，

那如果有100个文件怎么办?

file1.txt ~ file100.txt，每一个都需要提取”ERROR”行并存储

```
grep "ERROR" file1.txt > out1.txt
grep "ERROR" file2.txt > out2.txt
grep "ERROR" file3.txt > out3.txt
...
```

能否只写一次命令，就让它对所有文件自动完成?

目标：将重复的命令提取为结构性代码

脚本声明行 1. 必须放在脚本文件的第一行 2. 告诉系统使用bash执行器执行本脚本

```
#!/bin/bash

echo "hello, world!"

# 使用 for 循环处理 1 到 100 号文件
for i in {1..100}
do
  # 定义输入和输出文件名
  infile="file$i.txt"
  outfile="out$i.txt"

  # 提取包含 ERROR 的行并保存到对应输出文件
  grep "ERROR" "$infile" > "$outfile"
done

extract_errors.sh (END)
```

输出命令echo

1.将内容打印到终端

2. 提示、调试、输出变量内容

注释语法 #

这是一个注释，不会被执行

变量定义与引用

变量的定义、引用、拼接与嵌套

变量是脚本的容器，定义用“=”，引用用“$”，变量是bash自动化的第一步

```
#!/bin/bash

# 脚本功能演示：变量定义、引用与拼接

# 变量定义（注意等号两边不能有空格）
name="Alice"
greeting="Hello"
user="Bob"

# 正确的变量引用方式
echo $name          # 输出：Alice
echo "$name"        # 推荐写法，防止空格或特殊字符错误

# 变量嵌套 + 字符串拼接
echo "$greeting, $user!"  # 输出：Hello, Bob!

# 进一步拼接演示：拼接为文件名或路径
filename="${user}_report.txt"
echo "生成的文件名是：$filename"
variable_demo.sh (END)
```

变量拼接与嵌套：将变量与其他字符串（或变量）直接连接，从而形成新的字符串值

基本语法：

name="Alice"

filename="${name}_report.txt"

echo "生成的文件名是：$filename"

为什么要加大括号?

bash在处理变量时，会将$var当作变量名，直到遇到第一个不能作为变量名的字符为止。

filename="$name_report.txt “

Bash会误以为变量名为name_report

条件判断之if、else

if 条件判断是 Bash 控制流程的核心，配合 else/elif 实现逻辑分支与自动决策

基本语法结构

① if [条件]
then
cmd1
else
cmd2
fi

② if [条件]; then
cmd1
else
cmd2
fi

fi是if的结束标准，必须写！

基本判断条件 [条件]

判断类型	示例	含义
字符串是否相等	["$a" = "$b"]	判断两个字符串是否相等
字符串非空	[-n "$a"]	判断字符串不为空
文件是否存在	[-e "file.txt"]	判断文件是否存在
是否是目录	[-d "/etc"]	判断路径是否是一个目录
整数比较	["$a" -gt 10]	判断变量 a 是否大于 10

1. **字符串和文件判断用 = / -e，整数判断用 -eq / -gt / -lt 等**
2. **推荐用""包括变量，防止变量有空或有空格时报错**
3. **fi不可遗漏，它是fi语句块的结束标志**
4. **条件左右必须要有空格**

条件判断之if、else

if 条件判断是 Bash 控制流程的核心，配合 else/elif 实现逻辑分支与自动决策

01. 判断文件是否存在

```
#!/bin/bash
file="data.txt"
if [ -e  “$file" ]; then
  echo "$file 存在，开始处理..."
else
  echo "$file 不存在，终止脚本。"
fi
```

02. 判断用户输入

```
#!/bin/bash
read -p "请输入一个数字：" num
if [ "$num" -gt 10 ]; then
  echo "数字大于 10"
else
  echo "数字小于或等于 10"
fi
```

1. 脚本运行时终端输出提示文字
2. 用户在终端输入内容并按回车
3. bash将输入赋值给变量num

03. elif：多个条件判断

```
#!/bin/bash
if [ "$score" -ge 90 ]; then
  echo "优秀"
elif [ "$score" -ge 60 ]; then
  echo "及格"
else
  echo "不及格"
fi
```

循环语句之for

for 循环语句用于依次遍历一组值或范围，对每个值执行指定命令，实现批量处理与自动化操作

01. 遍历字符串列表

```
for fruit in apple banana cherry
do
  echo "I like $fruit"
done
```

```
I like apple
I like banana
I like cherry
```

02. 遍历文件（通配符）

```
for file in *.txt
do
  echo "Processing $file"
done
```

```
Processing *.txt
遍历当前目录下所有 .txt 文件
```

循环语句之for

for 循环语句用于依次遍历一组值或范围，对每个值执行指定命令，实现批量处理与自动化操作

03. 处理命名模式的文件

```
for i in {1..5}
do
  echo "文件名：file$i.txt"
done
```

```
文件名：file1.txt
文件名：file2.txt
...
```

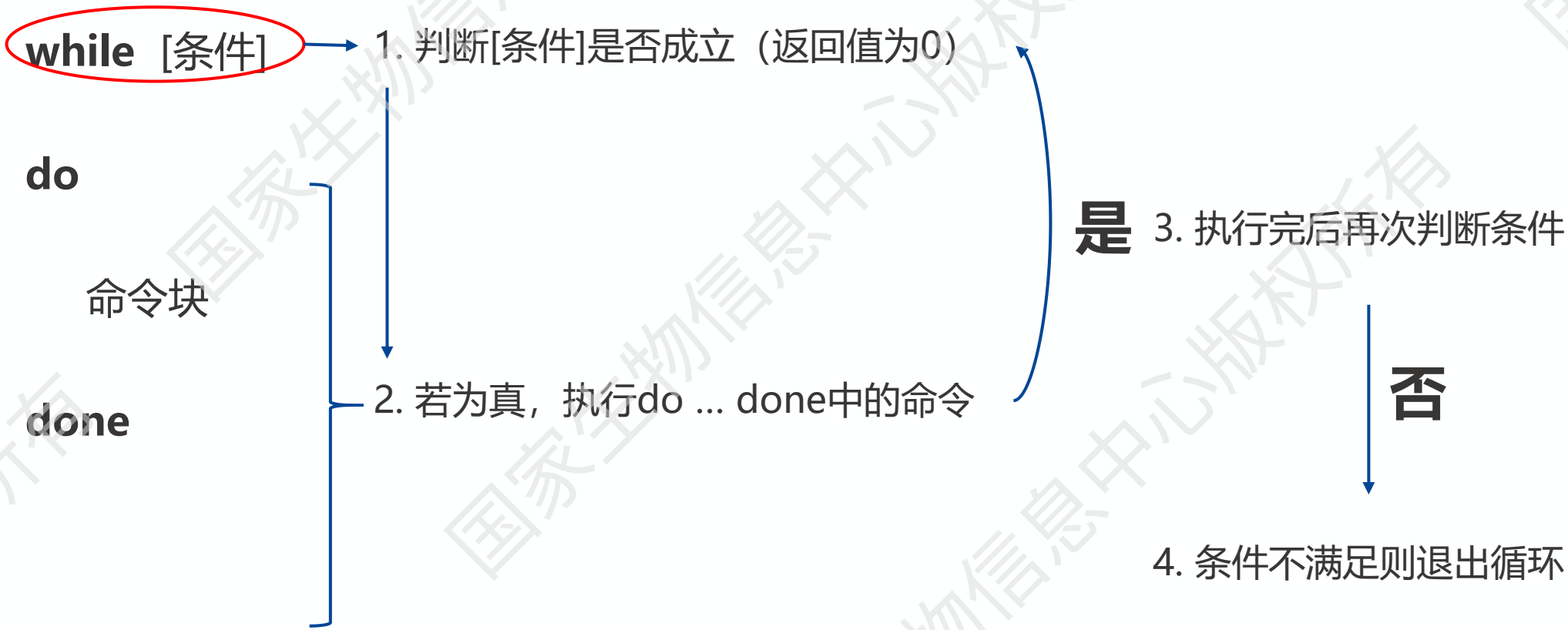
04. 数值循环（类似C语言结构）

```
for (( i=1; i<=5; i++ ))
do
  echo "数字是：$i"
done
```

```
数字是：1
...
数字是：5
```

循环语句之while

while 循环用于在满足条件时持续执行命令，是 Bash 中最灵活的控制结构之一



循环语句之while

while 循环用于在满足条件时持续执行命令，是 Bash 中最灵活的控制结构之一

打印数字1到5

```
i=1
while [ $i -le 5 ]
do
  echo "当前数字是：$i"
  ((i++))  # 或 i=$((i+1))
done
```

```
当前数字是：1
当前数字是：2
当前数字是：3
当前数字是：4
当前数字是：5
```

逐行读取文件内容

```
while read line
do
  echo "读取到：$line"
done < file.txt
```

```
读取到：apple
读取到：banana
读取到：cherry
```

比较点	for 循环	while 循环
遍历结构	遍历固定集合（列表、文件名）	条件控制，适合不确定次数或动态判断
常见用途	文件批处理、固定次数任务	实时监控、等待事件、数据读取
循环控制方式	遍历对象决定循环次数	条件逻辑决定循环是否继续

文本编辑器

vi编辑器是所有Unix及Linux系统下标准的编辑器，它的强大不逊色于任何最新的文本编辑器

- vi

  语法：vi [文件]

  如果文件已存在，会打开这个文件，如果不存在会新建一个文件。

- 三种模式：

  一般模式，用vi打开文件直接进入一般模式

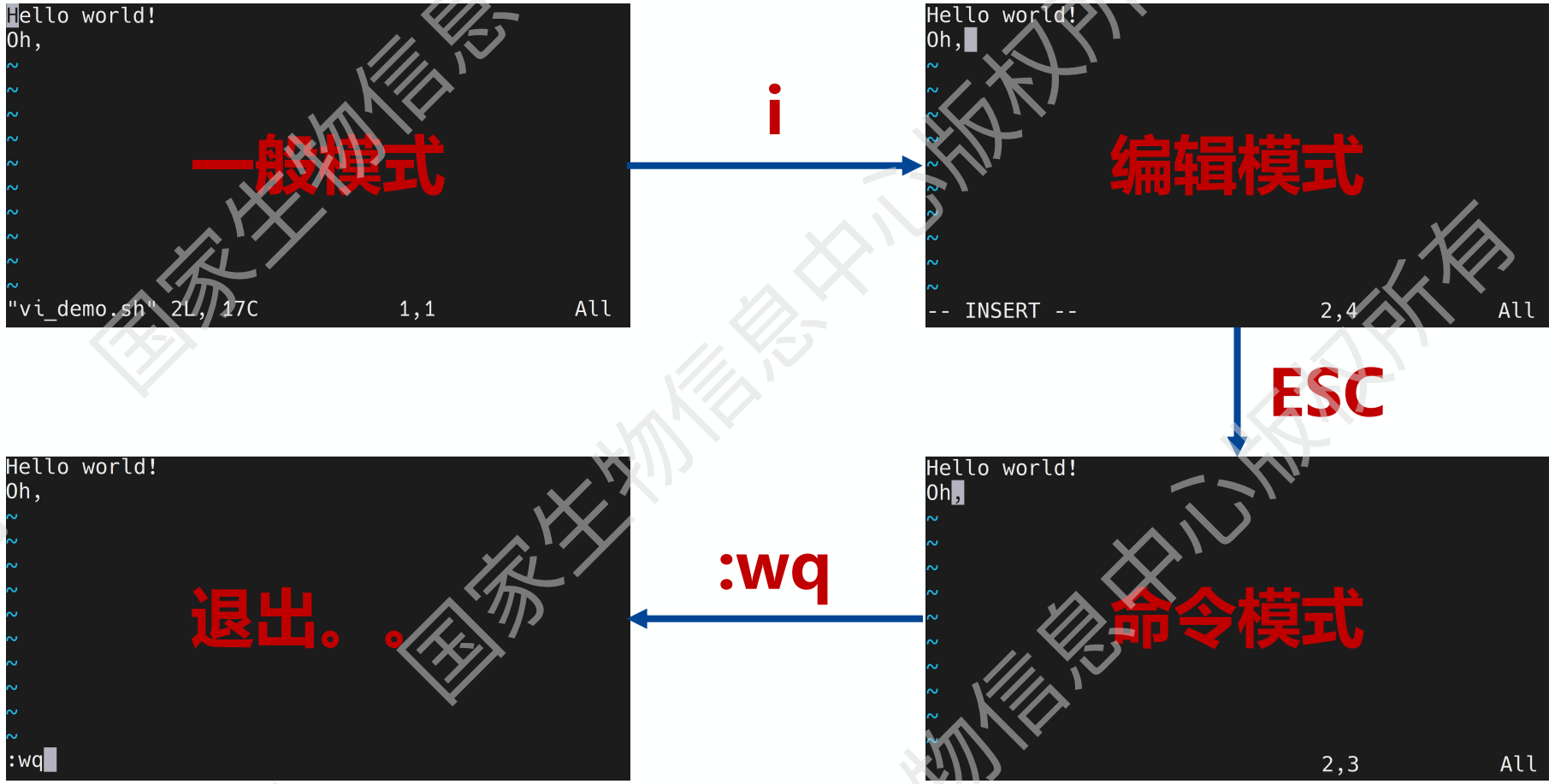
  编辑模式，用于编辑文件内容，输入“i、 o、 a ”

  命令模式，用于输入命令，输入 “:、 /、？”

操作	操作说明
G	移动到文件末尾
dd	删除光标所在的行
：set number	显示行号
i	在光标位置插入字符
Esc	退出编辑界面
yy	复制光标所在行
#yw	复制光标所在位置的#个字符
p	粘贴
:wq	保存退出
:q!	强制退出

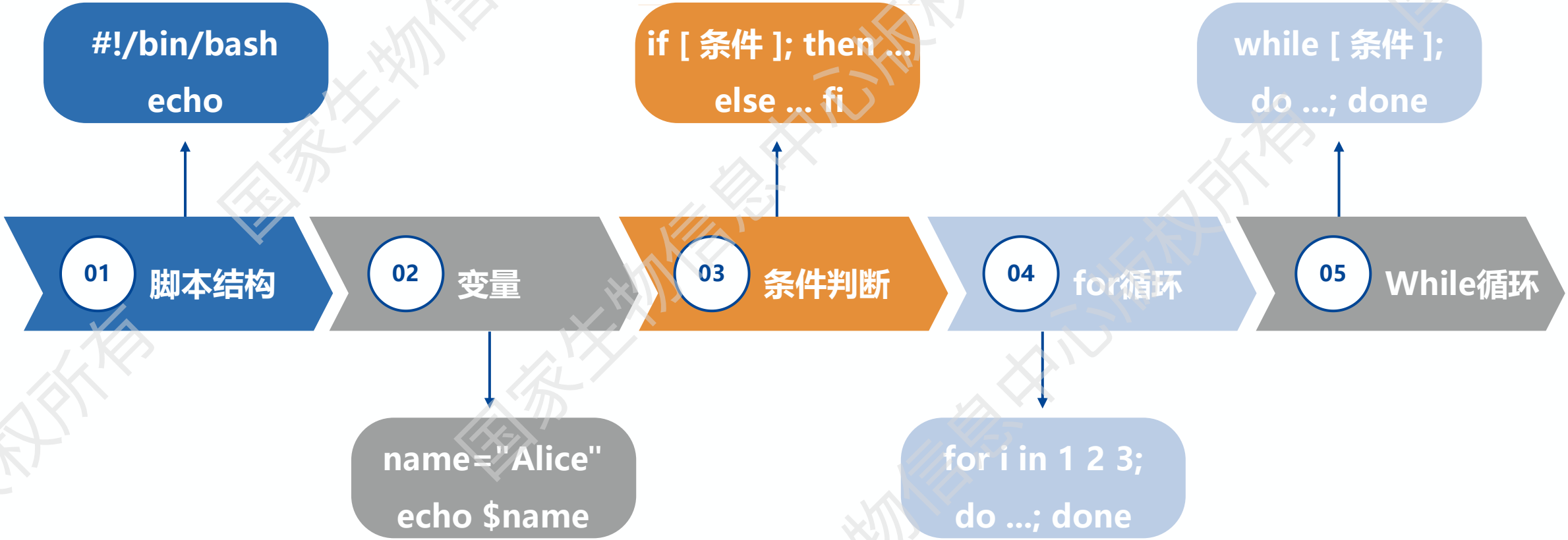
文本编辑器

vi编辑器是所有Unix及Linux系统下标准的编辑器，它的强大不逊色于任何最新的文本编辑器



本模块小结

批量处理与自动化脚本模块的主要目标是掌握shell脚本基本结构与变量使用，Shell脚本是Linux自动化的钥匙，从变量、判断到循环，每一部分都是高效运维的基础。



实战练习

你需要编写一个脚本count_lines.sh，该脚本将：

1. 遍历当前目录下所有以 .log 结尾的日志文件；
2. 对每个文件统计其行数；
3. 将每个文件的统计结果输出到终端；
4. 把汇总结果保存到一个文件 summary.txt 中；
5. 若某个文件不存在，输出警告信息到 error.log。

实战要求及答案路径：/home/big/linux/pratical_training/5-Batch_Processing_and_Automation_Scripts_Module

06

远程连接与进阶操作

文件远程传输

Linux文件远程传输方式丰富，scp支持快速复制远程，rsync高效同步大批量文件

scp

安全、简单、类 cp 语法

```
# 复制文件到远程主机
scp file.txt user@remote:/home/user/
# 从远程主机复制文件
scp user@remote:/home/user/file.txt .
# 递归复制整个目录
scp -r mydir/ user@remote:/data/backup/
```

适合传输小文件或目录

- -r：递归复制整个目录
- -v：显示详细传输过程
- -c：启用压缩传输，适用于大文件

&

- -v：显示详细过程
- -z：开启压缩，提高传输效率
- -a：归档模式（保留权限、时间戳）

适合大文件/批量同步，支持断点续传

rsync

更高效、更智能

```
# 将本地文件同步到远程主机
rsync -avz file.txt user@remote:/home/user/
# 将整个目录同步（可省流量）
rsync -avz mydir/ user@remote:/backup/mydir/
# 反向同步（远程到本地）
rsync -avz user@remote:/data/project/ ./project/
```

文件远程传输客户端FileZilla

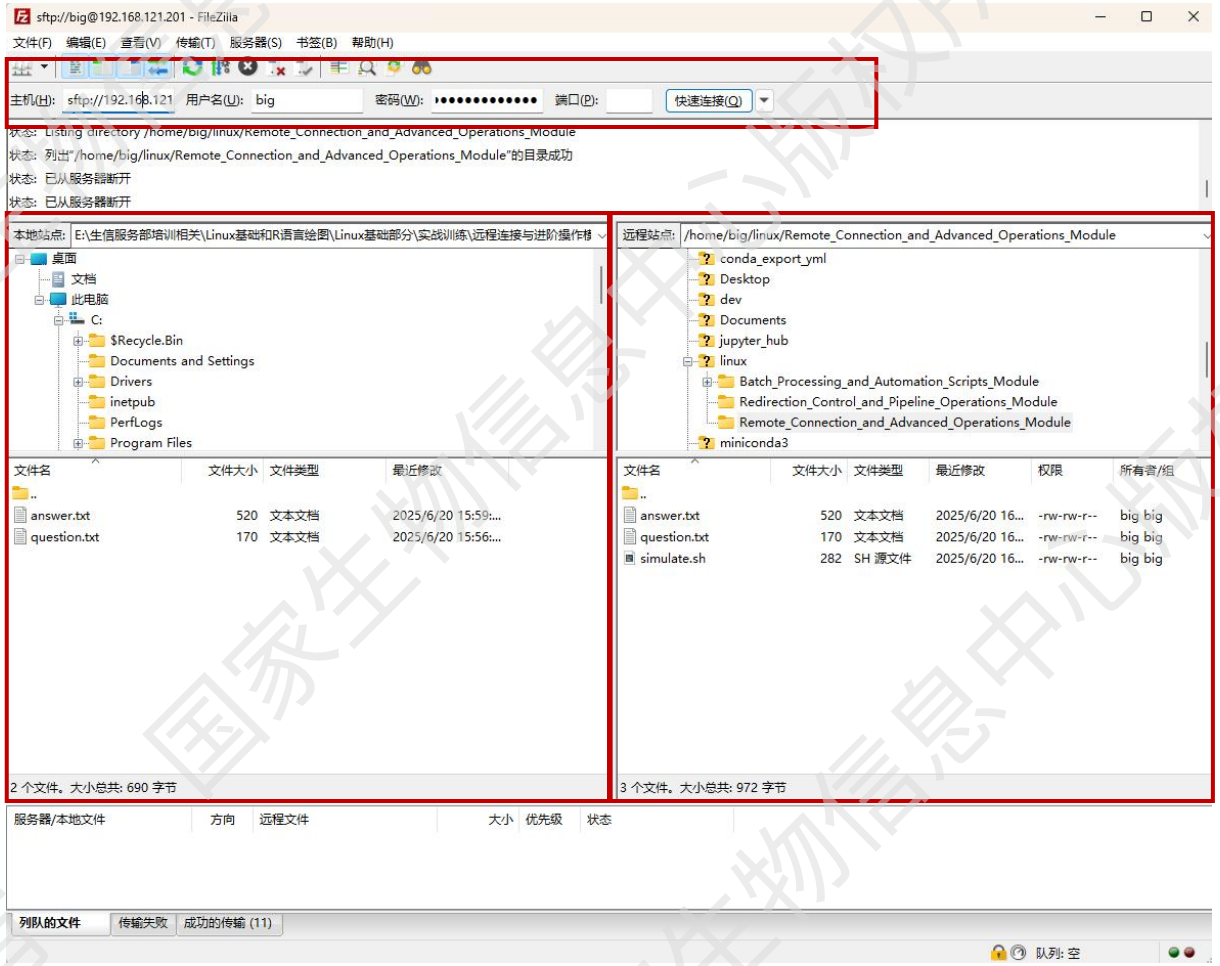
Filezilla是一款免费开源、跨平台的图形界面文件传输客户端，适用于从本地计算机与远程服务器之间安全地上传、下载和管理文件

主机：192.168.121.201

用户名：用户账号

密码：用户密码

端口：22



本地计算机

远程服务器

后台运行命令之& - 快速将命令放入后台

&的功能主要是实现本地后台运行

command [args] &

表示将命令放入后台执行

执行后立即返回终端，不阻塞当前会话

输出仍然会显示在当前终端（除非你重定向）

后台运行命令之nohup - 不挂断运行

nohup 用于防止命令因用户退出、断开 SSH 或关闭终端而被杀死

nohup command [args] **> output.log** **2>&1** **&**

- **nohup**：忽略HUP（挂断）信号
- **> output.log**：将标准输出重定向到文件
- **2>&1**：将标准错误也重定向到同一文件
- **&**：后台运行

功能项	&	nohup + &
后台运行	√	√
SSH断开后继续	会中断	任务继续
默认输出	当前终端	nohup.out（或你自定义日志文件）
推荐使用场景	简单测试、临时任务	长时间运行、稳定性要求高的任务

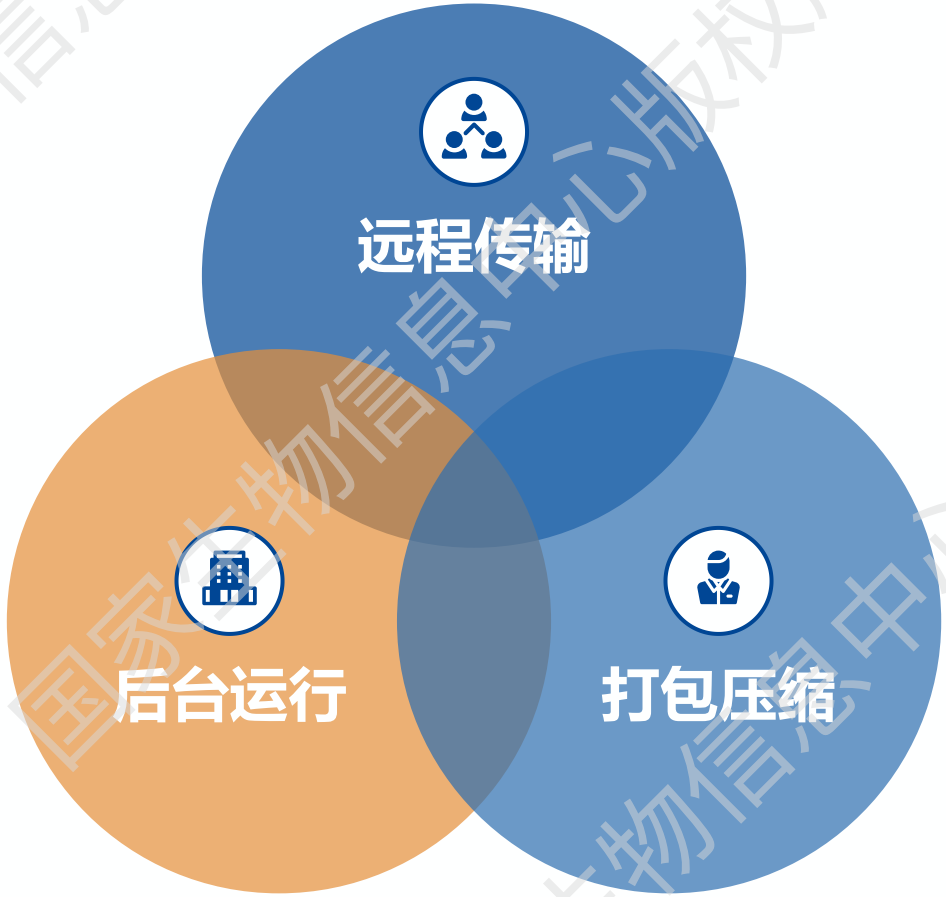
文件打包与压缩

常见的打包与压缩工具有四类：.tar负责归档，.gz用于压缩，而.tar.gz和.zip实现打包与压缩的一体化处理

格式	.tar	.gz	.tar.gz	.zip
简介	打包格式，不是压缩	单个文件压缩	先打包，再压缩	先打包，再压缩
适用范围	常用于将多个文件/目录打包成一个归档文件，便于传输与统一管理	体积小，解压快，常用于大文件压缩传输	适用于多个文件/目录的统一压缩与分发	更通用于 Windows 和 macOS 平台，可压缩多个文件
常用命令	① **打包多个文件为一个 .tar 文件** tar -cvf archive.tar file1 dir/ ② **解包 .tar 文件** tar -xvf archive.tar	① **压缩一个文件为 .gz** gzip file.txt ② **解包 .gz 文件** gunzip file.txt.gz	① **打包并压缩** tar -czvf archive.tar.gz file1 dir/ ② **解压并解包** tar -xzvf archive.tar.gz	① **压缩** zip archive.zip file1 dir/ ② **解压并解包** unzip archive.zip

本模块小结

批量处理与自动化脚本模块的主要目标是：了解常用科研服务器操作技巧



1. 文件远程传输

scp

rsync

Filezilla软件

2. 后台运行命令

nohup – 快速将命令放入后台

& - 不挂断运行

3. 文件打包与压缩

.tar

.gz

.tar.gz

.zip

实战练习

你正在远程科研服务器上完成一个模拟实验，需要你：

1. 使用 nohup 启动运行任务脚本simulate.sh，模拟长时间任务；
2. 将实验生成的多个结果文件打包为 .tar.gz；
3. 将该压缩包传输回本地或指定远程主机保存；
4. 额外任务：使用 rsync 将整个实验目录同步到远程备份服务器。

实战要求及答案路径：/home/big/linux/pratical_training/6-Remote_Connection_and_Advanced_Operations_Module

07 综合实战与流程演练

实战练习

某课题组每天会产生多个原始实验日志文件（以 .exp.log 结尾），文件中包含 DATA, ERROR, META 等信息。

你需要编写一个脚本完成以下任务：

1. 自动识别当日的所有 .exp.log 文件；
2. 提取每个文件中的关键数据（如 DATA 行）并统一汇总；
3. 检查是否存在 ERROR 行，并生成错误报告；
4. 所有输出和步骤记录在日志文件中；
5. 打包处理结果并传输到远程服务器备份；
6. 全过程支持后台运行、不依赖 SSH 连接；

```
project/
├── raw_logs/                ← 原始日志目录
├── processed/               ← 提取结果目录
│   ├── data_summary.txt
│   ├── error_report.txt
├── logs/                    ← 脚本执行日志
│   └── run_YYYYMMDD.log
├── scripts/
│   └── run_pipeline.sh      ← 主执行脚本
├── archive/
│   └── backup_YYYYMMDD.tar.gz
```

所需文件路径：/home/big/linux/pratical_training/Simulate_Reality_Pipeline

THANK YOU