

网络计算技术

实验一

专 业： 通信工程

年 级： 2022级

学 号：

姓 名：

2025年 月 日

1. 实验目的：熟悉Linux操作系统的基础概念和常用操作指令，为后续云计算相关实验奠定基础。
2. 实验内容：
   1. 基于“蓝桥云课”网站中的《Linux基础入门》（https://www.lanqiao.cn/courses/1）课程，学习相应的理论内容，并完成该课程对应的实验1~实验17的所有内容将每个部分的实验完成。
   2. 每个部分的实验报告内容包括：基本原理分析、关键命令的解释和实验结果截图和分析。

实验1：

1. 基本原理分析、关键命令的解释
2. 实验结果

实验2：

1. 基本原理分析、关键命令的解释
2. 实验结果

实验3：

1. 基本原理分析、关键命令的解释
2. 实验结果

实验4：

1. 基本原理分析、关键命令的解释

Linux 的目录结构遵循 FHS（Filesystem Hierarchy Standard，文件系统层次结构标准），目的是提供一致性和可维护性，使得不同的 Linux 发行版能够保持相似的文件组织方式。FHS 主要定义了根目录（/）下的各个子目录的用途，并进一步规范了 /usr 和 /var 目录的结构。其核心思想是根据文件的用途、访问频率及用户是否可修改，将系统目录划分为不同的类别，使系统结构更加有序。例如，/bin 存放基本的用户可执行文件，/sbin 存放系统管理员命令，/etc 主要用于存放配置文件，而 /var 目录则存储变化较频繁的数据，如日志文件、缓存等。

关键命令解释：

tree / —— 以树形结构显示根目录下的所有文件和文件夹，方便直观了解 Linux 目录结构。

sudo apt-get update —— 更新软件包索引，确保可以安装最新的软件包。

sudo apt-get install tree —— 安装 tree 命令，使其可以用于查看目录结构。

ls -l / —— 以长格式列出根目录下的文件和目录，查看它们的权限、所有者等信息。

df -h / —— 显示根目录的磁盘使用情况，以人类可读的格式（GB、MB 等）呈现。

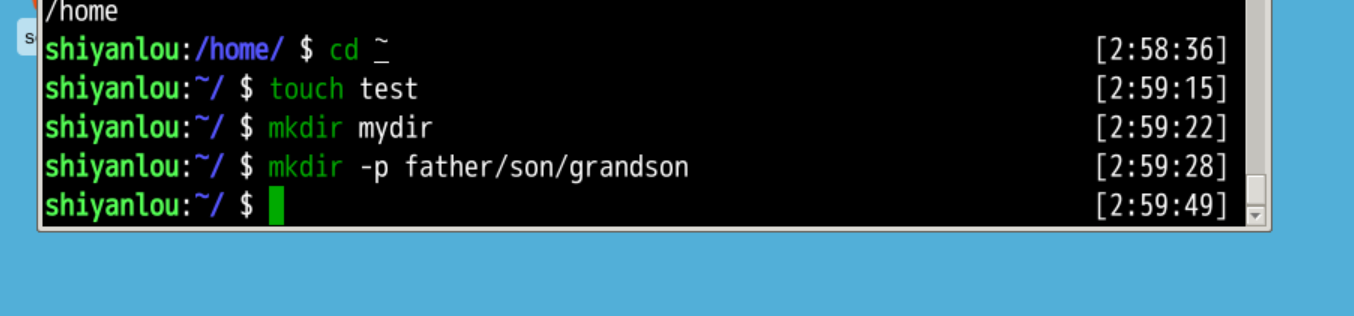
du -sh / —— 统计根目录所占用的磁盘空间。

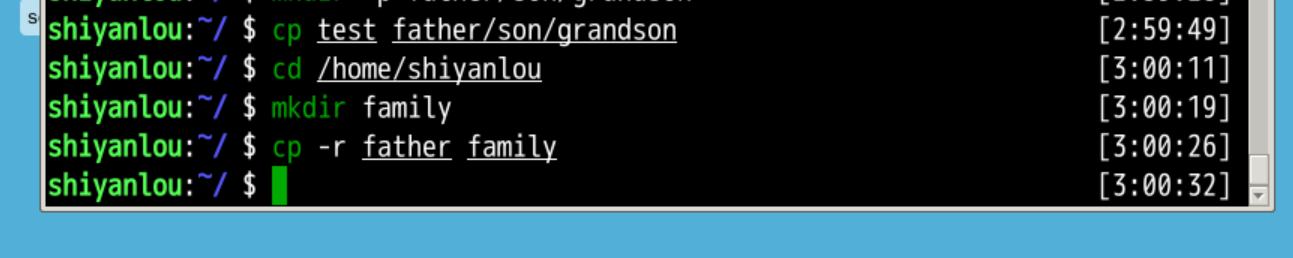
find / -name "filename" —— 在整个系统中搜索名为 filename 的文件。

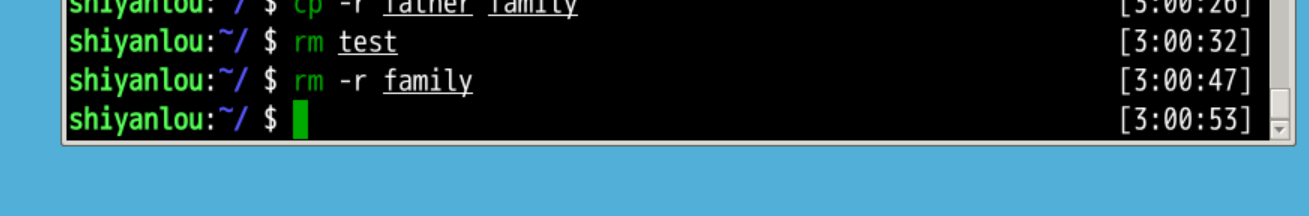
cd /usr/share —— 进入 /usr/share 目录，查看共享资源文件。

1. 实验结果



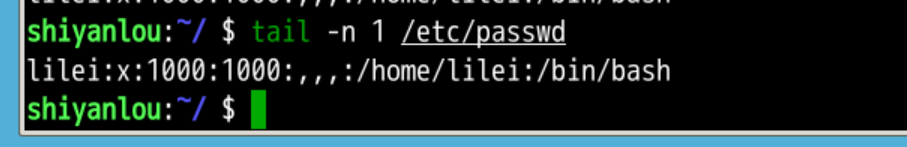


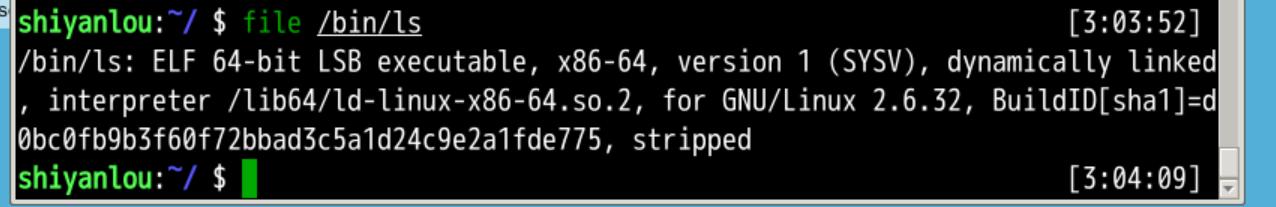












实验5：

1. 基本原理分析、关键命令的解释

环境变量是 Linux 系统中的全局变量，影响着系统和应用程序的行为。常见环境变量的作用包括：

PATH：定义可执行文件的搜索路径。

HOME：指定当前用户的主目录路径。

SHELL：指定当前用户使用的 shell 类型（如 /bin/bash）。

LANG：设置系统的语言环境。

USER：表示当前登录的用户名。

环境变量可以分为以下两类：

全局环境变量（系统级）：适用于所有用户，通常由 /etc/profile、/etc/environment 等文件定义。

局部环境变量（用户级）：仅对当前用户有效，通常由 ~/.bashrc、~/.bash\_profile、~/.profile 等文件定义。

env # 显示所有环境变量

printenv # 列出当前环境变量

echo $PATH # 显示 PATH 变量的值

export VAR\_NAME="value" # 设置环境变量

echo $VAR\_NAME # 显示变量值

unset VAR\_NAME # 删除环境变量

echo 'export VAR\_NAME="value"' >> ~/.bashrc

source ~/.bashrc # 使更改立即生效

echo 'export VAR\_NAME="value"' >> ~/.bashrc

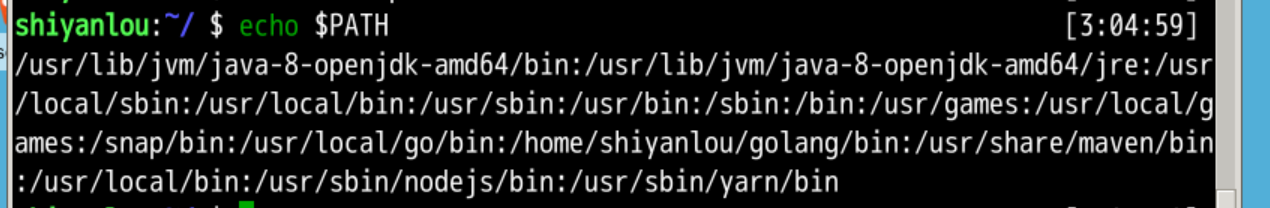
source ~/.bashrc # 使更改立即生效

echo 'export VAR\_NAME="value"' >> ~/.bashrc

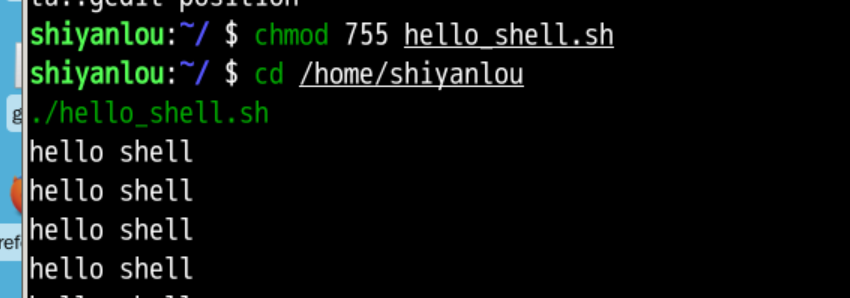
source ~/.bashrc # 使更改立即生效

（2）实验结果









实验6：

1. 基本原理分析、关键命令的解释

zip 是 Linux 中用于压缩文件的工具，常用于创建 .zip 格式的压缩包。

zip archive.zip file1 file2 # 将 file1 和 file2 压缩成 archive.zip

zip -r archive.zip dir/ # 递归压缩整个目录 dir

zip -d archive.zip file1 # 从 archive.zip 中删除 file1

zip -u archive.zip file1 # 更新 archive.zip 中的 file1

unzip archive.zip # 解压 archive.zip 到当前目录

unzip archive.zip -d /path # 解压到指定路径 /path

unzip archive.zip -d /path # 解压到指定路径 /path

unzip -l archive.zip # 列出压缩包中的文件

tar 是 Linux 中用于打包和压缩文件的工具，通常与 gzip 或 bzip2 结合使用，生成 .tar.gz 或 .tar.bz2 格式的压缩包。

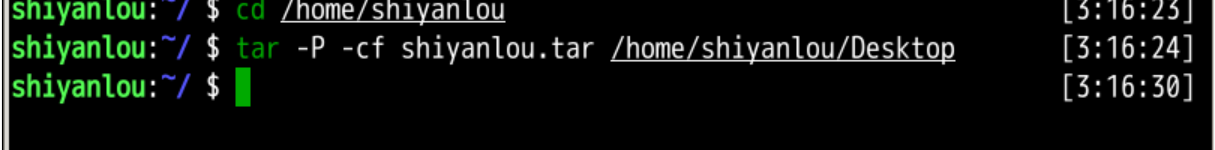
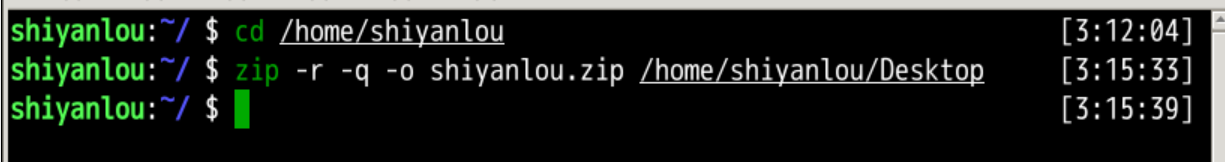
tar -cvf archive.tar file1 file2 # 将 file1 和 file2 归档为 archive.tar

tar -cvf archive.tar dir/ # 归档整个目录 dir

tar -czvf archive.tar.gz dir/ # 归档并使用 gzip 压缩

tar -cjvf archive.tar.bz2 dir/ # 归档并使用 bzip2 压缩

（2）实验结果



实验7：

1. 基本原理分析、关键命令的解释

本实验介绍 Linux 文件系统的基本操作，重点讲解 df、du 和 mount 命令的使用。虽然实验内容较为基础，但掌握这些命令将极大地提高用户在 Linux 系统管理中的效率，并为后续深入学习打下坚实的基础。

df -h # 以人类可读格式（MB、GB）显示磁盘使用情况

df -T # 显示文件系统类型

df -i # 显示 inode 使用情况

du -sh /path/to/dir # 显示目录 /path/to/dir 的总大小

du -h --max-depth=1 # 仅显示当前层级的大小

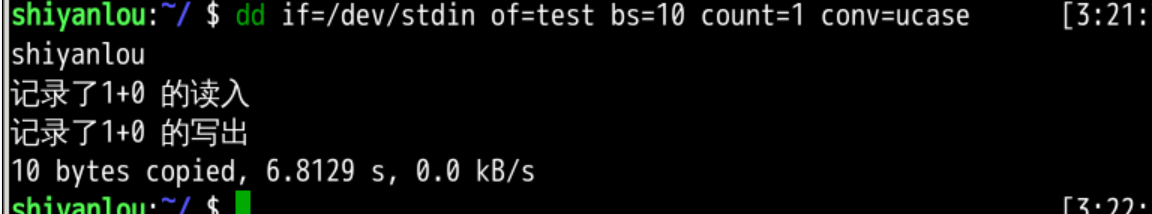
du -ah | sort -rh | head -10 # 显示当前目录中占用空间最大的 10 个文件或文件夹

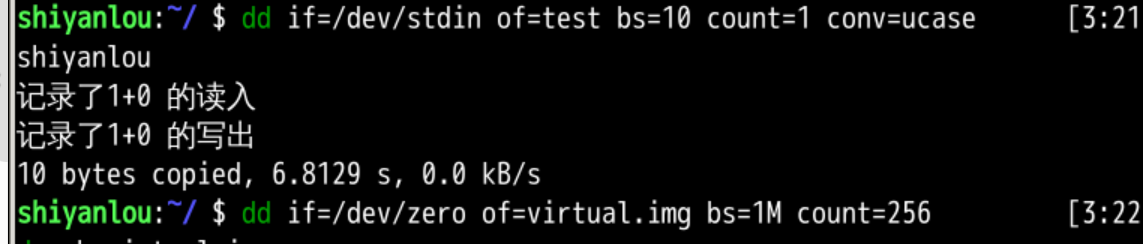
mount # 显示当前所有挂载的文件系统

sudo mount /dev/sdb1 /mnt # 将 /dev/sdb1 挂载到 /mnt 目录

sudo umount /mnt # 卸载 /mnt 目录下的设备

（2）实验结果





实验8：

1. 基本原理分析、关键命令的解释

内建命令（Built-in Commands）：由 Shell 内部提供，执行速度快，无需额外进程。例如：cd、echo、exit、alias。

外部命令（External Commands）：通常是可执行文件，存放在 /bin、/usr/bin 等目录。例如：ls、grep、tar、vim。

type command\_name # 显示命令类型（内建或外部）

help cd # 查看 cd 命令的用法

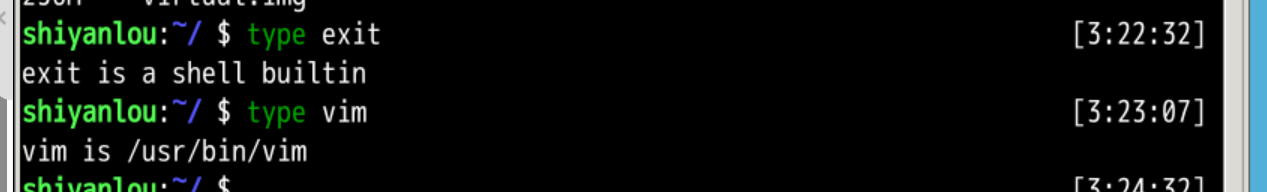
help exit # 查看 exit 命令的帮助信息

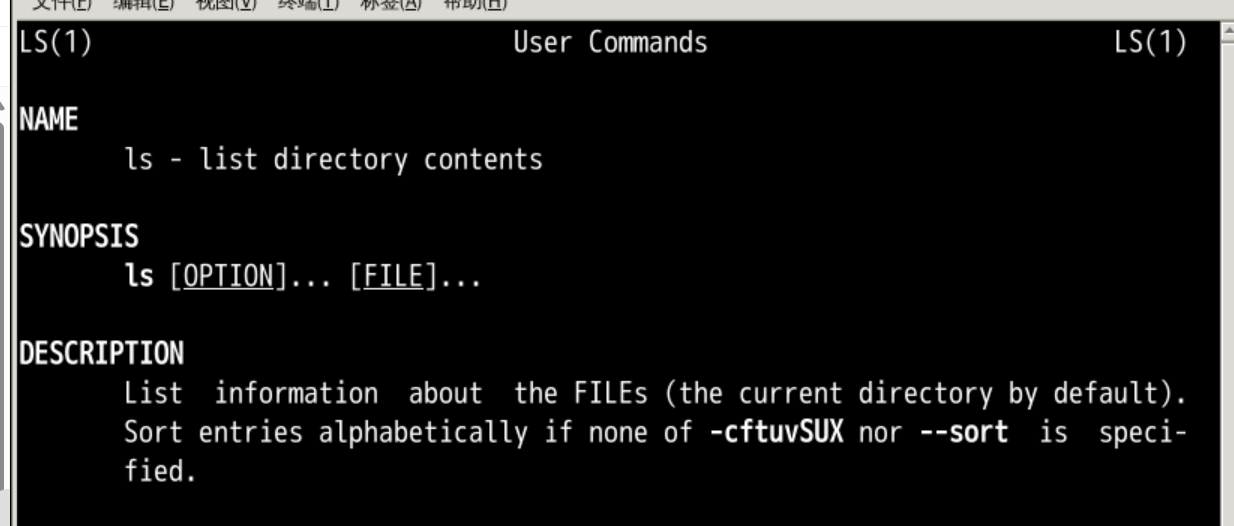
man ls # 查看 ls 命令的手册

man -k keyword # 搜索包含 keyword 相关内容的手册

info ls # 查看 ls 命令的详细信息（2）实验结果

（2）实验结果





实验9：

1. 基本原理分析、关键命令的解释

Linux 系统的计划任务工具 crontab，用于实现定期执行任务，如自动清理 /tmp 目录、定期备份数据库、分析日志等。掌握 crontab 语法后，可高效管理和调度系统任务，提高自动化水平。

\* \* \* \* \* command\_to\_execute

- - - - -

| | | | |

| | | | +---- 星期 (0-7，0 和 7 均表示周日)

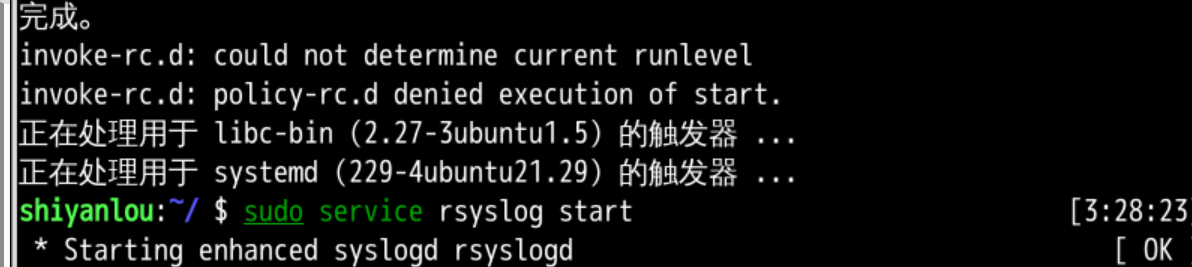
| | | +------ 月份 (1-12)

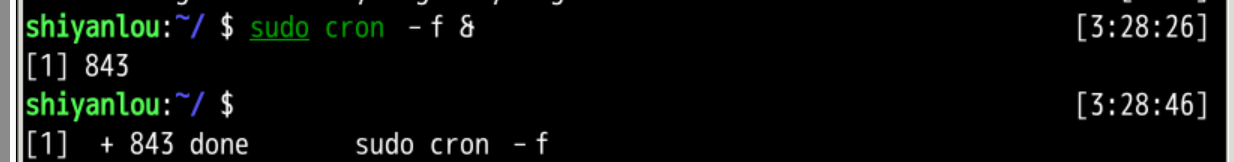
| | +-------- 日期 (1-31)

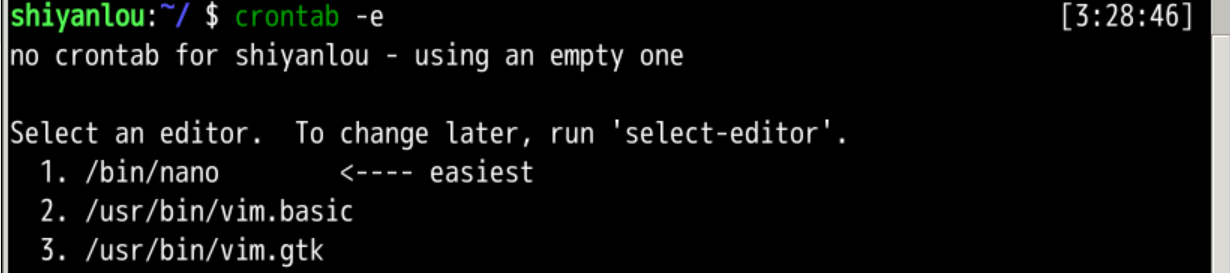
| +---------- 小时 (0-23)

+------------ 分钟 (0-59)

1. 实验结果







实验10：

1. 基本原理分析、关键命令的解释

本实验介绍 Linux 中用于高效数据处理的常用命令，包括 cut、grep、wc 和 sort，以及管道 (|) 的使用。掌握这些命令可以帮助用户快速筛选、统计、排序和分析文本数据，提高 Linux 命令行操作的效率。

cut 用于按列或分隔符提取文本内容，适用于结构化数据处理，如 CSV 文件或日志分析。

cut -c1-5 filename.txt # 提取文件的第 1-5 个字符

cut -d':' -f1 /etc/passwd # 提取 `/etc/passwd` 文件中用户名字段

cut -d':' -f1 /etc/passwd | sort | uniq

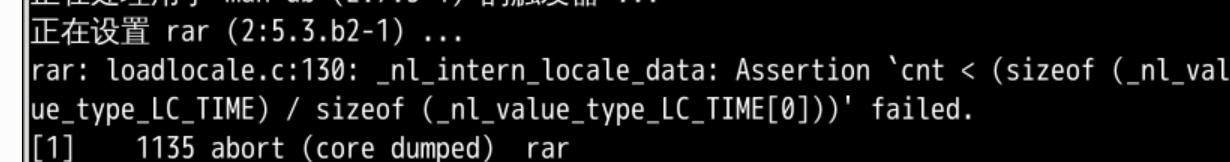
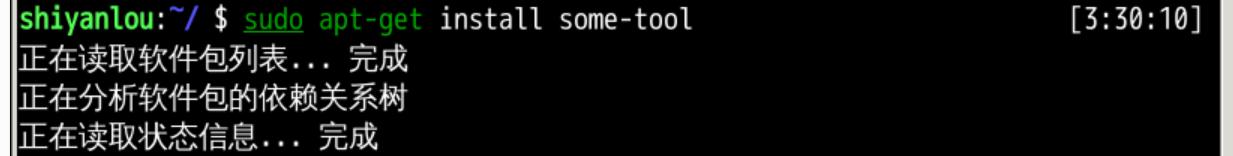
grep 用于在文件或标准输入中搜索匹配的文本，支持正则表达式。

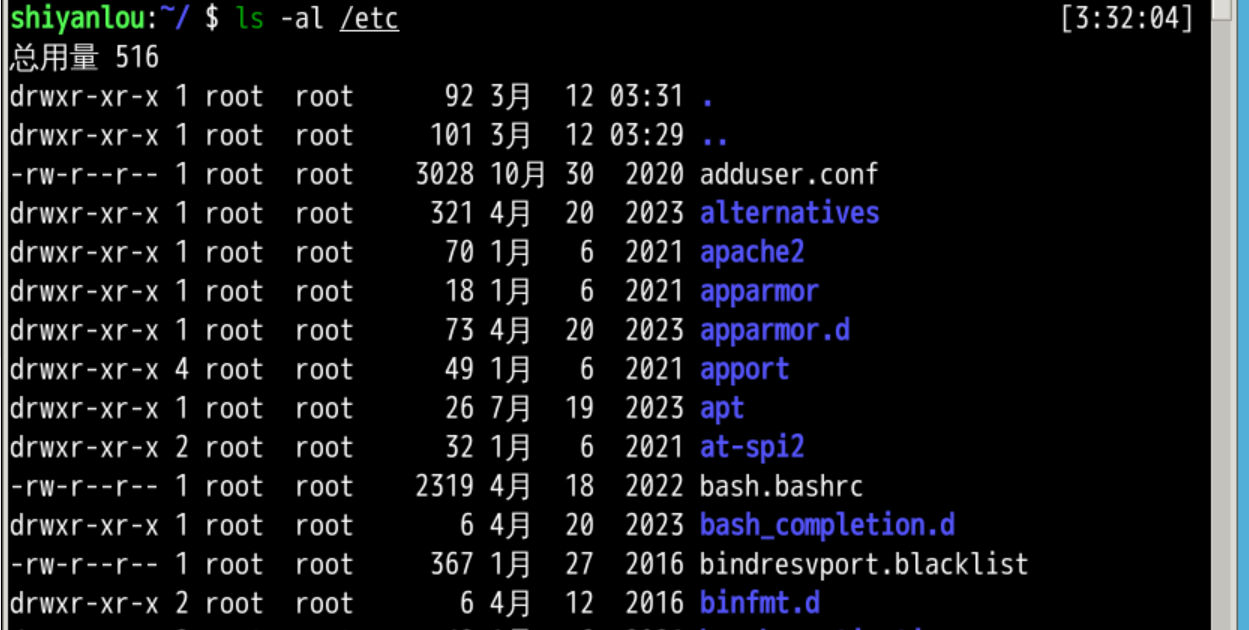
grep "error" logfile.txt # 查找包含 "error" 的行

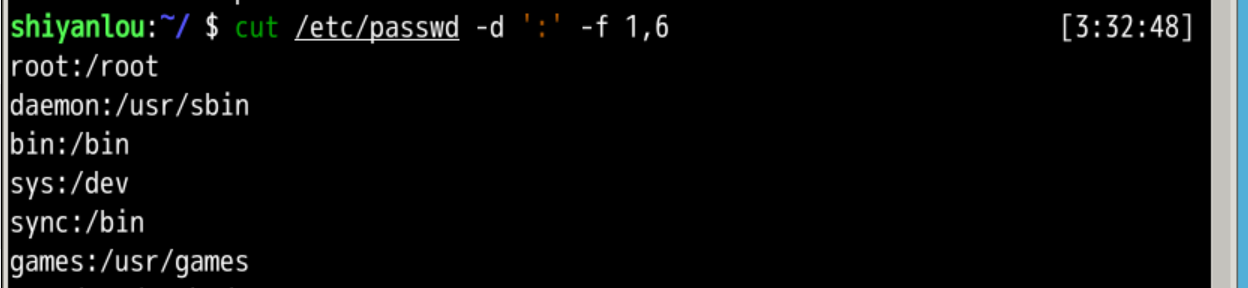
grep -i "warning" logfile.txt

grep -E "fail|error|critical" logfile.txt

（2）实验结果









实验11：

1. 基本原理分析、关键命令的解释

本实验介绍了 Linux 中的文本处理命令 tr、col、join 和 paste，并结合管道 (|) 操作来熟悉这些命令的实际使用。这些命令有助于我们高效地处理和格式化文本数据，特别是在数据清理、文本转换等方面。

tr（translate）命令用于字符的替换、删除或压缩，常用于文本格式化操作。

echo "hello world" | tr 'a-z' 'A-Z' # 将小写字母转换为大写字母

echo "hello 123 world" | tr -d '0-9' # 删除文本中的数字

echo "aaabbbccc" | tr -s 'a-z' # 压缩重复字符

col 命令用于将文本按列对齐，通常用于处理带有换行符或回车符的文本输出，使其格式化为便于阅读的样式。

cat somefile.txt | col -b

cat somefile.txt | col -x

join 命令用于将两个文件按行连接，通常用于按某一列对齐文件。

join file1.txt file2.txt

join -1 2 -2 3 file1.txt file2.txt

paste 命令用于将多个文件的内容按列合并，适用于合并文件的列而非按行合并。

paste file1.txt file2.txt

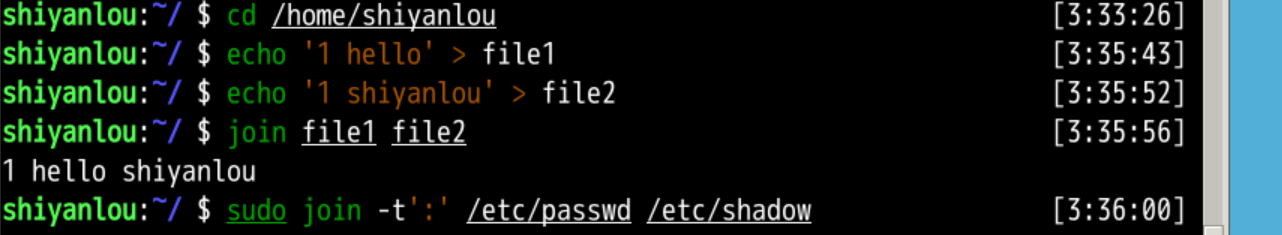
paste -d ',' file1.txt file2.txt

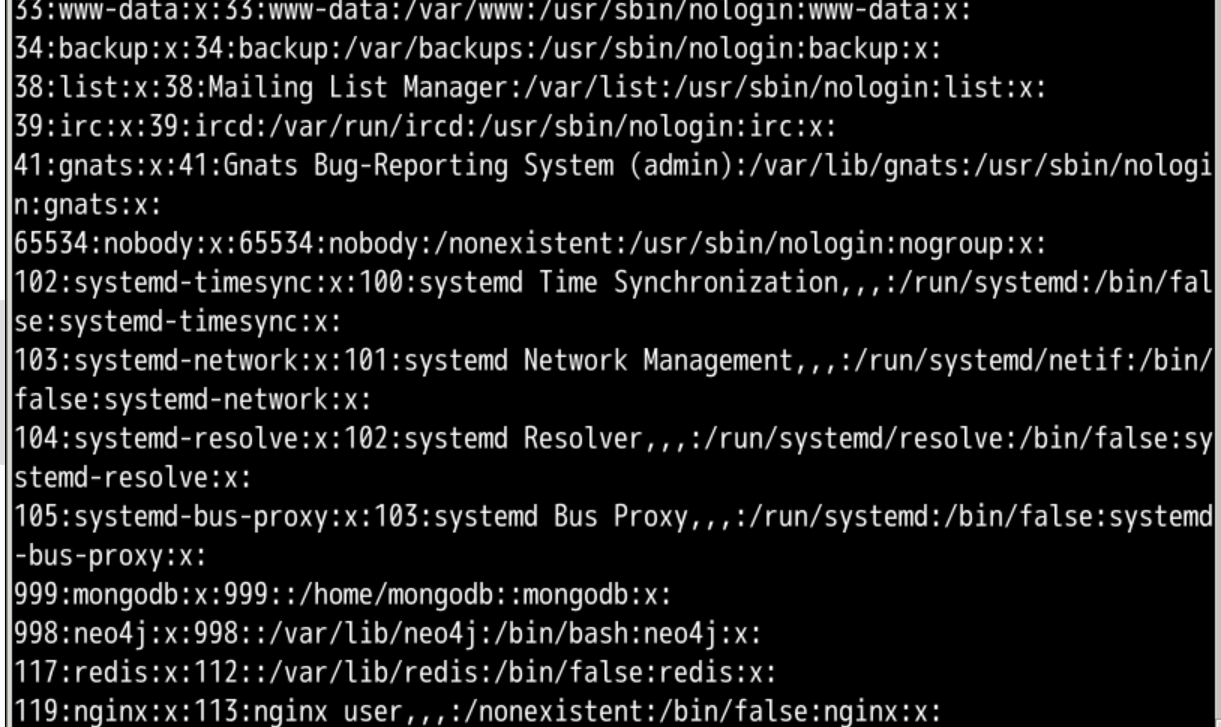
这些命令通常与管道结合使用，可以实现更强大的文本处理功能。以下是一些示例：

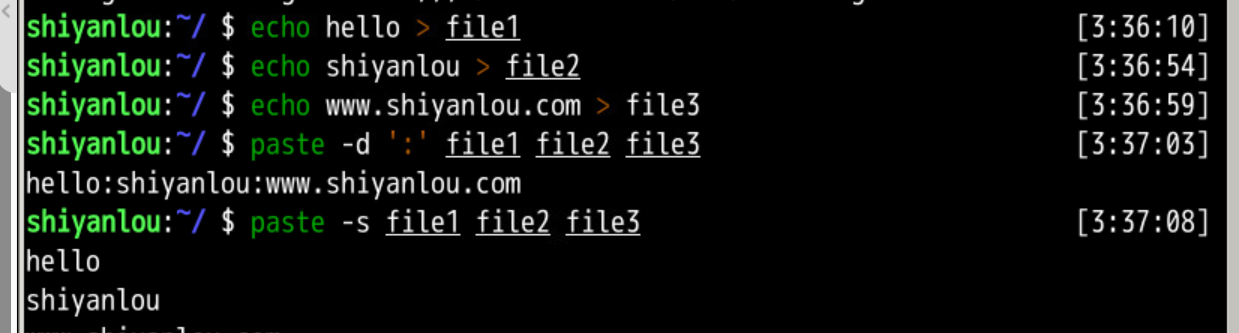
cat file.txt | tr 'a-z' 'A-Z' | sort

join file1.txt file2.txt | col -b

（2）实验结果





实验12：

1. 基本原理分析、关键命令的解释

在 Linux 中，重定向是一个非常重要的概念，通常用于将命令的标准输出或错误输出导向一个文件。你可能已经在前面的课程中多次接触过 > 和 >> 操作符，它们分别用于将标准输出重定向到一个文件或将数据追加到文件中。重定向的本质是将原本输出到标准输出（/dev/stdout）的数据重新定向到文件中，因为标准输出本身也是一个文件。所以我们可以通过改变文件描述符来控制数据的输出位置。本节内容将进一步讲解重定向的使用方法及文件描述符的概念。

2.1.1 输出重定向 (> 和 >>)

将命令输出重定向到文件：  
使用 > 操作符将标准输出重定向到文件中，若文件已存在，会覆盖原有内容；若文件不存在，会创建一个新文件。

echo "Hello, World!" > output.txt # 将输出重定向到 output.txt 文件，覆盖原文件内容

echo "Appended line" >> output.txt # 将新行追加到 output.txt 文件末尾

将错误输出重定向到文件：  
在 Linux 中，标准错误输出使用文件描述符 2。使用 2> 可以将错误输出重定向到指定的文件。

ls non\_existing\_directory 2> error.log # 将错误信息输出到 error.log 文件

ls non\_existing\_directory 2>> error.log # 将错误信息追加到 error.log 文件

将错误追加到文件：  
使用 2>> 可以将标准错误输出追加到文件末尾。

ls non\_existing\_directory 2>> error.log # 将错误信息追加到 error.log 文件

有时需要将标准输出和标准错误同时重定向到同一个文件，使用如下命令：

command > output.txt 2>&1 # 将标准输出和标准错误都重定向到 output.txt 文件

在 Linux 中，每个进程都会打开 3 个标准文件描述符，分别是标准输入、标准输出和标准错误输出。

标准输入 (stdin)：文件描述符 0，用于接收输入数据。

标准输出 (stdout)：文件描述符 1，用于输出数据。

标准错误输出 (stderr)：文件描述符 2，用于输出错误信息。

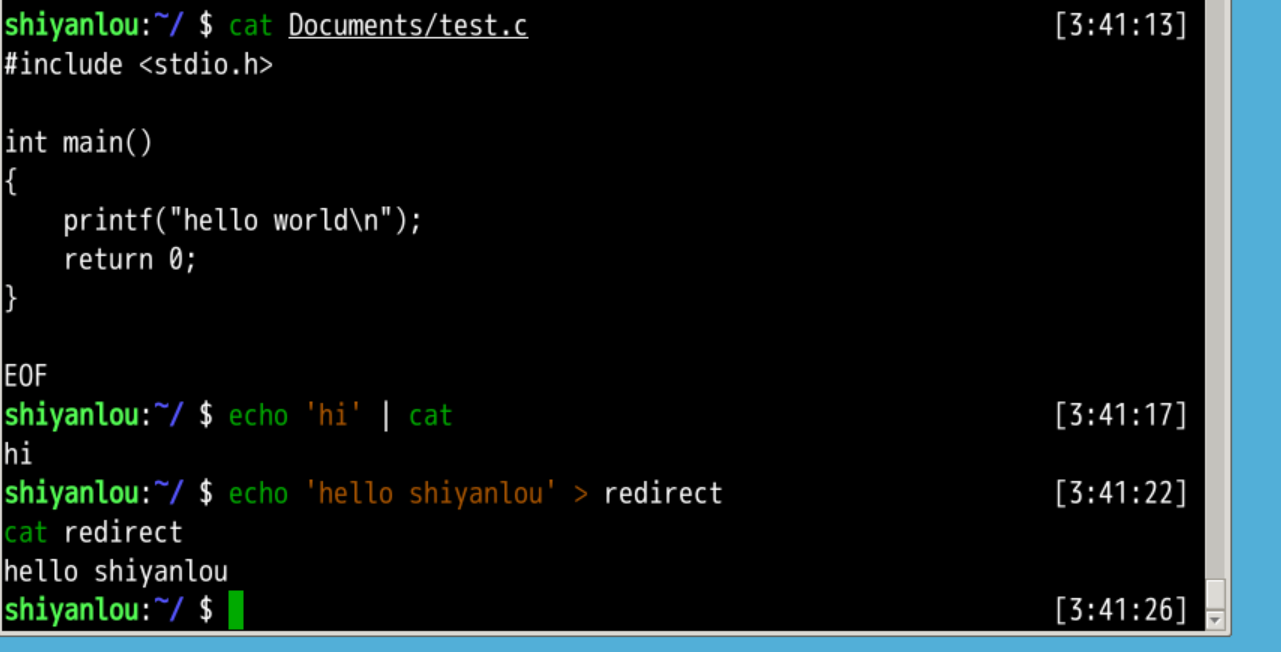
你可以通过修改文件描述符来实现重定向操作。

command 0< input.txt 1> output.txt 2> error.log

command > /dev/null 2>&1 # 丢弃所有标准输出和标准错误

1. 实验结果





实验13：

1. 基本原理分析、关键命令的解释

本实验介绍了三个常用的文本处理命令：grep、sed 和 awk，并探讨了正则表达式在这些命令中的应用。正则表达式作为一种强大的文本匹配工具，广泛用于 grep、sed 和 awk 中，以实现灵活的文本搜索、替换和处理。虽然本节课程的重点是学习这三个命令的基本用法，但我们也将涵盖一些常用的正则表达式符号和语法，使你能够在日常操作中高效处理文本数据。

grep 命令——文本搜索

grep（global regular expression print）是 Linux 中用于文本搜索的命令，它通过正则表达式在文件中查找匹配的行，并输出这些行。

grep "pattern" file.txt # 在 file.txt 文件中查找包含 "pattern" 的行

grep "^Hello" file.txt # 查找以 "Hello" 开头的行

grep "world$" file.txt # 查找以 "world" 结尾的行

grep "a.\*b" file.txt # 查找包含 "a" 和 "b" 之间任意字符的行

grep -i "pattern" file.txt # 不区分大小写查找 "pattern"

sed（stream editor）是一个强大的文本处理工具，常用于对文本进行替换、删除和插入等操作。

sed 's/old/new/' file.txt # 替换 file.txt 中第一次出现的 "old" 为 "new"

sed 's/old/new/g' file.txt # 替换文件中的所有 "old" 为 "new"

使用正则表达式：  
sed 也支持正则表达式，如：

^：匹配行的开头。

$：匹配行的结尾。

\*：匹配前一个字符 0 次或多次。

sed 's/^Hello/Hi/' file.txt # 将以 "Hello" 开头的行替换为 "Hi"

sed 's/.\*error.\*/[ERROR FOUND]/' file.txt # 用 "[ERROR FOUND]" 替换包含 "error" 的行

sed '/pattern/d' file.txt # 删除包含 "pattern" 的行

awk 是一种强大的文本处理工具，适用于格式化输出和进行复杂的文本操作。它基于模式-动作对进行处理，即根据模式匹配的文本执行某个动作。

awk '/pattern/ {print $0}' file.txt # 打印包含 "pattern" 的行

awk -F "," '{print $1, $2}' file.txt # 按逗号分隔字段，并打印前两列

awk '/^abc/ {print $0}' file.txt # 打印以 "abc" 开头的行

1. 实验结果

