《计算机操作系统》实验报告

实验题目：Linux 文件操作二

姓名：严昕宇　　学号：20121802　　实验日期：2023.01.13

实验环境：

实验设备：Lenovo Thinkbook16+ 2022
操作系统：Ubuntu 22.04.1 LTS 64 位

实验目的：

1. 熟悉 Linux 下的基本操作，学会使用各种 Shell 命令去操作 Linux，对 Linux 有一个感性认识
2. 熟悉 Linux 下文件打包、解压的命令及 RPM 包的操作
3. 熟悉 Linux 下文件系统管理命令，以及加载其他分区的方法

实验内容：

1. 文件打包、解压(tar、gzip)
2. 挂载文件系统
3. 文件系统维护命令

一、文件打包、解压(tar、gzip)

操作过程 1：

【操作要求 1】tar：掌握 tar 的参数：zcvfxt
【操作步骤】

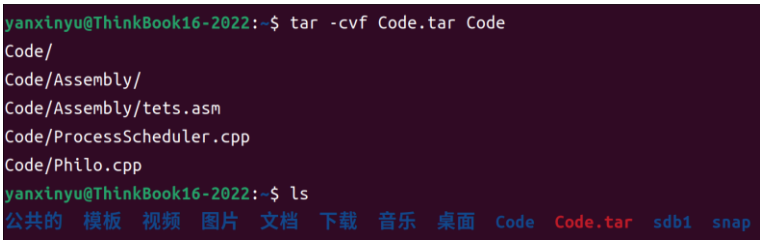
实现以下操作：(目标目录可以选择/boot 或者其他较小的目录，包名自定义)

① 文件不压缩打包，格式：tar -cvf 包名.tar 目标目录
② 文件解包，格式：tar -xvf 包名.tar
③ 文件包测试，格式：tar -tf 包名.tar
④ 文件压缩打包，格式：tar -zcvf 包名.tar.gz 目标目录
⑤ 文件解压缩，格式：tar -zxvf 包名.tar.gz
⑥ 文件压缩包测试，格式：tar -ztf 包名.tar.gz

打包或者解包以后可以用 ls 看下结果。

结果 1：

① **文件不压缩打包，格式：tar -cvf 包名.tar 目标目录**



② 文件解包，格式：tar -xvf 包名.tar

```
yanxinyu@ThinkBook16-2022:~$ tar -xvf Code.tar
Code/
Code/Assembly/
Code/Assembly/tets.asm
Code/ProcessScheduler.cpp
Code/Philo.cpp
yanxinyu@ThinkBook16-2022:~$ ls
公共的  模板  视频  图片  文档  下载  音乐  桌面  Code  Code.tar  sdb1  snap
```

③ 文件包测试，格式：tar -tf 包名.tar

```
yanxinyu@ThinkBook16-2022:~$ tar -tf Code.tar
Code/
Code/Assembly/
Code/Assembly/tets.asm
Code/ProcessScheduler.cpp
Code/Philo.cpp
yanxinyu@ThinkBook16-2022:~$ ls
公共的  模板  视频  图片  文档  下载  音乐  桌面  Code  Code.tar  sdb1  snap
```

④ 文件压缩打包，格式：tar -zcvf 包名.tar.gz 目标目录

```
yanxinyu@ThinkBook16-2022:~$ tar -zcvf Code.tar.gz Code
Code/
Code/Assembly/
Code/Assembly/tets.asm
Code/ProcessScheduler.cpp
Code/Philo.cpp
yanxinyu@ThinkBook16-2022:~$ ls
公共的  视频  文档  音乐  Code      Code.tar.gz  snap
模板    图片  下载  桌面  Code.tar  sdb1
```

⑤ 文件解压缩，格式：tar -zxvf 包名.tar.gz

```
yanxinyu@ThinkBook16-2022:~$ tar -zxvf Code.tar.gz
Code/
Code/Assembly/
Code/Assembly/tets.asm
Code/ProcessScheduler.cpp
Code/Philo.cpp
yanxinyu@ThinkBook16-2022:~$ ls
公共的  视频  文档  音乐  Code      Code.tar.gz  snap
模板    图片  下载  桌面  Code.tar  sdb1
```

⑥ 文件压缩包测试，格式：tar -ztf 包名.tar.gz

```
yanxinyu@ThinkBook16-2022:~$ tar -ztf Code.tar.gz
Code/
Code/Assembly/
Code/Assembly/tets.asm
Code/ProcessScheduler.cpp
Code/Philo.cpp
yanxinyu@ThinkBook16-2022:~$ ls
公共的  视频  文档  音乐  Code      Code.tar.gz  snap
模板    图片  下载  桌面  Code.tar  sdb1
```

操作过程 2：

【操作要求 2】gzip：掌握 gzip 的参数：-v　-d

【操作步骤】

实现以下操作：

① 将当前目录下的每个文件压缩成.gz 文件，gzip *

② 将当前目录下的每个压缩的文件解压，并列出详细信息，gzip -dv *

③ 详细当前目录下的压缩文件的信息，但不进行解压，gzip -l *

④ 递归的压缩目录，gzip -rv test

⑤ 递归的解压目录，gzip -drv test

结果 2：

① 将当前目录下的每个文件压缩成.gz 文件，gzip *

```
yanxinyu@ThinkBook16-2022:~/Code$ gzip *
gzip: Assembly is a directory -- ignored
yanxinyu@ThinkBook16-2022:~/Code$ ls
Assembly  Philo.cpp.gz  ProcessScheduler.cpp.gz
```

② 将当前目录下的每个压缩的文件解压，并列出详细信息，gzip -dv *

```
yanxinyu@ThinkBook16-2022:~/Code$ gzip -dv *
gzip: Assembly is a directory -- ignored
Philo.cpp.gz:    63.8% -- replaced with Philo.cpp
ProcessScheduler.cpp.gz:         72.8% -- replaced with ProcessScheduler.cpp
```

由于 Assembly 为文件夹，因此被忽略。

③ 详细当前目录下的压缩文件的信息，但不进行解压，gzip -l *

```
yanxinyu@ThinkBook16-2022:~/Code$ gzip -l *
gzip: Assembly is a directory -- ignored
         compressed        uncompressed  ratio uncompressed_name
                158                 359  63.8% Philo.cpp
               1824                6551  72.8% ProcessScheduler.cpp
               1982                6910  71.9% (totals)
```

由于 Assembly 为文件夹，因此被忽略。

④ 递归的压缩目录，gzip -rv 目录

```
yanxinyu@ThinkBook16-2022:~/Code$ gzip -rv Assembly
Assembly/tets.asm:       31.7% -- replaced with Assembly/tets.asm.gz
yanxinyu@ThinkBook16-2022:~/Code$ cd Assembly
yanxinyu@ThinkBook16-2022:~/Code/Assembly$ ls
tets.asm.gz
```

⑤ 递归的解压目录，gzip -drv 目录

```
yanxinyu@ThinkBook16-2022:~/Code$ gzip -drv Assembly
Assembly/tets.asm.gz:    31.7% -- replaced with Assembly/tets.asm
yanxinyu@ThinkBook16-2022:~/Code$ cd Assembly
yanxinyu@ThinkBook16-2022:~/Code/Assembly$ ls
tets.asm
```

思考：tar 与 gzip 有何区别、联系？

答：首先要区分两个概念——打包和压缩。

打包是指将一大堆文件或目录什么的变成一个总的文件，压缩则是将一个大的文件通过一些压缩算法变成一个小文件。为什么要区分这两个概念呢？其实这源于 Linux 中的很多压缩程序只能针对一个文件进行压缩，这样当想要压缩一大堆文件时，就需要先借助另它的工具将这一大堆文件先打成一个包，然后再就原来的压缩程序进行压缩。

区别：

- **tar**：tar 是 Linux 下最常用的打包程序(原生不包括压缩功能)。使用 tar 程序打出来的包常称为 tar 包，tar 包文件的命令通常都是以.tar 结尾的。生成 tar 包后，就可用其它的程序来进行压缩。
- **gzip**：gzip 是 gnu/Linux 的一种压缩文件工具，算法是基于 DEFLATE，文件是 gz。

联系

- 在 Linux 里面，为了方便用户在打包解包的同时可以压缩或解压文件，tar 一般和其他没有文件管理的压缩算法文件结合使用，用 tar 打包整个文件目录结构成一个文件，再调用 gz，bzip 等压缩算法压缩成一次。也是 Linux 常见的压缩归档的处理方法。
- 而 gzip 可以和 tar 组合，tar 中使用-z 这个参数来调用 gzip。与 gzip 相对的解压程序是 gunzip。

二、挂载文件系统

操作过程 1：

【操作要求 1】mount 命令，挂载磁盘

【操作步骤】

① 选择“VM”----“setting”并打开，将光标定位在 hard Disk 这一选项，然后点击下方的 Add 按钮,点击 next，执行下一个步骤,根据提示，创建一个虚拟的磁盘，并点击下一步。按照默认的点击下一步即可完成虚拟磁盘的添加。

② 使用“fdisk -l”的命令查看当前系统的分区(如果刚才设置 VMware--setting 的时候运行了系统，则会出现磁盘：没有识别到新的磁盘即 sdb)，解决办法，重启虚拟机：shutdown -r now。如果执行第一步的时候是关闭虚拟机中的系统的，则使用“fdisk -l 命令的时候则会出现新的磁盘 sdb(不过提示未分区

③ 对新建的磁盘进行分区及格式化的工作：输入 fdisk /dev/sdb，终端会提示：Command (m for help)

④ 根据提示输入：n，会出现提示，依次输入 p 和 1 即可。

⑤ 接着便会提示卷的起始地址和结束地址，都保持默认按回车的即可(只分一个区)

⑥ 输入“w”保存并推出，再次使用“fdisk -l”这个命令来查看会发现出现了/dev/sdb1(说明已经完成了分区工作)

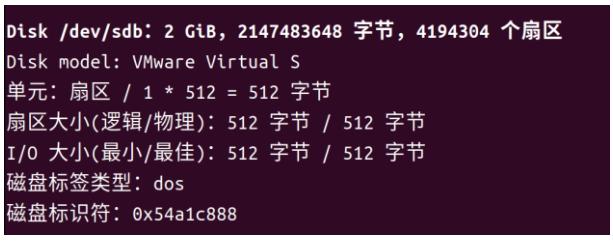
⑦ 对新建的分区进行格式化：输入命令 mkfs -t ext3 /dev/sdb1,格式化成 ext3 的文件系统即可

⑧ 手动挂载磁盘：使用 mount /dev/sdb1 /要挂载的目录(自己自定义)

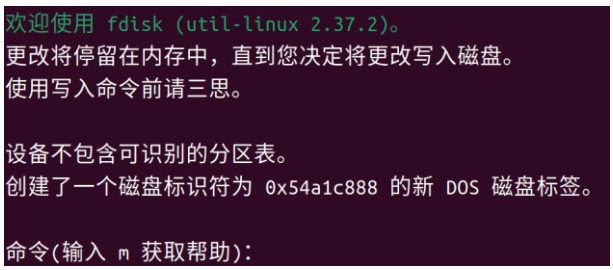
结果 1：

① 选择“VM”----“setting”并打开，将光标定位在 hard Disk 这一选项，然后点击下方的 Add 按钮,点击 next，执行下一个步骤,根据提示，创建一个虚拟的磁盘，并点击下一步。按照默认的点击下一步即可完成虚拟磁盘的添加。

② 使用“fdisk -l”的命令查看当前系统的分区(如果刚才设置 VMware--setting 的时候运行了系统，则会出现磁盘：没有识别到新的磁盘即 sdb)，解决办法，重启虚拟机：shutdown -r now。如果执行第一步的时候是关闭虚拟机中的系统的，则使用“fdisk -l 命令的时候则会出现新的磁盘 sdb(不过提示未分区)



③ 对新建的磁盘进行分区及格式化的工作：输入 fdisk /dev/sdb，终端会提示：Command (m for help)



④ 根据提示输入：n，会出现提示，依次输入 p 和 1 即可。

```
欢迎使用 fdisk (util-linux 2.37.2)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。

设备不包含可识别的分区表。
创建了一个磁盘标识符为 0x54a1c888 的新 DOS 磁盘标签。

命令(输入 m 获取帮助)： n
分区类型
   p   主分区 (0 primary, 0 extended, 4 free)
   e   扩展分区 (逻辑分区容器)
选择 (默认 p)： p
分区号 (1-4, 默认  1): 1
第一个扇区 (2048-4194303, 默认 2048):
```

⑤ 接着便会提示卷的起始地址和结束地址，都保持默认按回车的即可(只分一个区)

```
欢迎使用 fdisk (util-linux 2.37.2)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。

设备不包含可识别的分区表。
创建了一个磁盘标识符为 0x54a1c888 的新 DOS 磁盘标签。

命令(输入 m 获取帮助)： n
分区类型
   p   主分区 (0 primary, 0 extended, 4 free)
   e   扩展分区 (逻辑分区容器)
选择 (默认 p)： p
分区号 (1-4, 默认  1): 1
第一个扇区 (2048-4194303, 默认 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-4194303, 默认 4194303):

创建了一个新分区 1，类型为“Linux”，大小为 2 GiB。

命令(输入 m 获取帮助):
```

⑥ 输入“w”保存并推出，再次使用“fdisk -l”这个命令来查看会发现出现了/dev/sdb1(说明已经完成了分区工作)

```
命令(输入 m 获取帮助)： w
分区表已调整。
将调用 ioctl() 来重新读分区表。
正在同步磁盘。

root@ThinkBook16-2022:/home/yanxinyu# fdisk -l
```

⑦ 对新建的分区进行格式化：输入命令 mkfs -t ext3 /dev/sdb1,格式化成 ext3 的文件系统即可

```
root@ThinkBook16-2022:/home/yanxinyu# mkfs -t ext3 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
创建含有 524032 个块（每块 4k）和 131072 个 inode 的文件系统
文件系统 UUID：278a22ea-c896-4581-8ecf-45465d897d19
超级块的备份存储于下列块：
        32768, 98304, 163840, 229376, 294912

正在分配组表：  完成
正在写入 inode表：  完成
创建日志（8192 个块）：  完成
写入超级块和文件系统账户统计信息：  已完成

root@ThinkBook16-2022:/home/yanxinyu#
```

⑧ 手动挂载磁盘：使用 mount /dev/sdb1 /要挂载的目录(自己自定义)

操作过程 2：

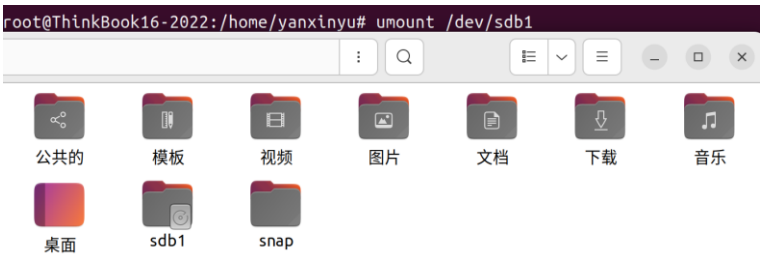
【操作要求 2】umount 命令，卸载文件系统

【操作步骤】

① # umount -v /dev/sda1　　通过设备名卸载

② # umount -v /mnt/mymount/　　通过挂载点卸载

结果 2：



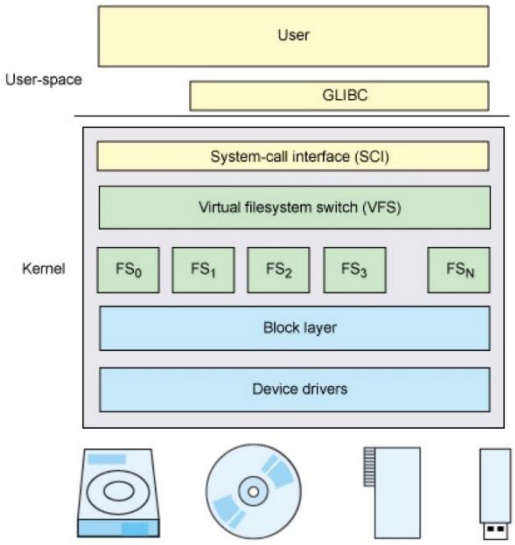
思考：Linux 是如何应对多文件系统？

答：Linux 依靠 VFS(Virtual Filesystem Switch，虚拟文件系统转换，也称虚拟文件系统)应对多文件系统，而且支持各种文件系统之间相互访问。

VFS 是一个内核软件层，在具体的文件系统之上抽象的一层，用来处理与 Posix 文件系统相关的所有调用，表现为能够给各种文件系统提供一个通用的接口，使上层的应用程序能够使用通用的接口访问不同文件系统，同时也为不同文件系统的通信提供了媒介。

VFS 由超级块、inode、dentry、vfsmount 等结构来组成。Linux 系统的 User 使用 GLIBC(POSIX 标准、GUN C 运行时库)作为应用程序的运行时库，然后通过操作系统，将其转换为系统调用 SCI(system-call interface)，SCI 是操作系统内核定义的系统调用接口，这层抽象允许用户程序的 I/O 操作转换为内核的接口调用。VFS 提供了一个抽象层，将 POSIX API 接口与不同存储设备的具体接口实现进行了分离，使得底层的文件系统类型、设备类型对上层应用程序透明。

VFS 在整个 Linux 系统中的架构视图如下：



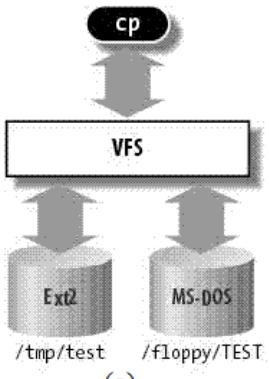
VFS 并不是一种实际的文件系统，它只存在于内存中，不存在任何外存空间，VFS 在系统启动时建立，在系统关闭时消亡。

与此同时，有了 VFS，就能很容易实现不同文件系统之间的数据读写，因为它们对外接口都是一样的，都是 VFS 导出的通用接口。

例如，假设一个用户输入以下 shell 命令：

```
$ cp /floppy/TEST /tmp/test
```

其中/floppy 是 MS-DOS 磁盘的一个安装点，而/tmp 是一个标准的第二扩展文件系统(Ext2, second Extended Filesystom)的目录。正如图(a)所示，VFS 是用户的应用程序与文件系统实现之间的抽象层。因此，cp 程序并不需要知道/floppy/TEST 和 /tmp/test 是什么文件系统类型。相反，cp 程序直接与 VFS 交互，这是通过 Unix 程序设计人员都熟悉的普通系统调用来进行的。cp 的执行代码如图(b)所示：



(a)

```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
       O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

(b)

三、文件系统维护命令

操作过程 1：

【操作要求 1】du 命令，功能：用于查看磁盘使用情况
【操作步骤】
格式：du 目录名

结果 1：

```
yanxinyu@ThinkBook16-2022:~/Code$ du
8       ./Assembly
24      .
yanxinyu@ThinkBook16-2022:~/Code$
```

操作过程 2：

【操作要求 2】df 命令，功能：用于查看磁盘剩余情况
【操作步骤】
格式：df [option]

结果 2：

```
yanxinyu@ThinkBook16-2022:~$ df
文件系统          1K-块     已用     可用 已用% 挂载点
tmpfs             398320     1988   396332    1% /run
/dev/sda3       19946096 11669848 7237708   62% /
tmpfs            1991596        0 1991596    0% /dev/shm
tmpfs               5120        4    5116    1% /run/lock
/dev/sda2         524252     5364  518888    2% /boot/efi
tmpfs             398316     2404  395912    1% /run/user/1000
```

操作过程 3：

【操作要求 3】fsck 命令，功能：扫描文件系统内容检查内部一致性的工具
【操作步骤】
格式：fsck 文件系统，如：fsck / (查根文件系统) (选做)

结果 3：

由 Ubuntu 系统提示可知，不应该用 fsck 检查已挂载的磁盘，此行为很可能会对磁盘造成永久性的伤害。因此，在开始使用 fsck 之前，需要使用命令来卸载磁盘。

而在之前实验中，刚卸载了/dev/sdb1，因此可以利用其进行实验，结果如下：

```
root@ThinkBook16-2022:/home/yanxinyu# fsck /dev/sdb1
fsck，来自 util-linux 2.37.2
e2fsck 1.46.5 (30-Dec-2021)
/dev/sdb1：没有问题，11/131072 文件，17205/524032 块
root@ThinkBook16-2022:/home/yanxinyu#
```

检查结果为没有问题(clean)，表示文件系统正常。

操作过程 4：

【操作要求 4】free 命令，功能：查看系统的物理内存和虚拟内存的使用情况

【操作步骤】

格式：free

结果 4：

```
yanxinyu@ThinkBook16-2022:~$ free
               total        used        free      shared  buff/cache   available
内存:        3983192     1317212     1467008      114864     1198972     2276656
交换:        2191356           0     2191356
```

思考：free 命令显示内容分别代表什么？

答：free 命令用来显示系统内存状态，包括系统物理内存、虚拟内存(swap 交换分区)、共享内存和系统缓存的使用情况，其输出和 top 命令的内存部分非常相似。

第一行显示的是各列的列表头信息，含义如下：

- total：系统总的可用物理内存和交换空间大小
- used：已经被使用的物理内存和交换空间大小
- free：空闲的物理内存和交换空间大小
- shared：被共享使用的物理内存大小
- buff/cache：被 buffer 和 cache 使用的物理内存大小
- available：可以被应用程序使用的物理内存大小

而内存(Mem)一行指的是内存的使用情况；交换(Swap)则指的就是交换空间(swap 分区)的使用情况。

实验体会

本次实验中，我通过实际操作，熟悉了 Linux 的一些基本操作，学会使用各种 Shell 命令去操作 Linux。过程中，我了解到如下 Linux 下文件打包、解压的命令及 RPM 包的操作。初步掌握了 Linux 下文件系统管理命令，以及加载其他分区的方法。

这虽然是操作系统课程的最后一个实验，但就像陈老师在课堂中所提及的那样，现代操作系统内部的真实实现与理论知识的是有巨大鸿沟的，我们都还有非常巨大的空间，去深入一步的学习，了解现代操作系统的魅力。