****

**Unix/Linux系统**

|  |  |
| --- | --- |
| 学生姓名 | 张子洋 |
| 学 号 | 808221223 |
| 专业班级 | 计科2203班 |
| 指导教师 | 胡小龙 |
| 学 院 | 计算机学院 |
| 完成时间 | 2024.12.10 |

计算机学院

2024年7月

**实验一 Linux的安装（课后）**

**1、实验目的**

* 1. 了解硬盘分区的概念和方法；
  2. 掌握硬盘的分区规划；
  3. 掌握Linux操作系统的安装和配置过程。

**2、实验设备**

一台pc机、RedHat Linux 7.2以上版本、**VMware Workstation v5.5**

**3、实验原理**

Linux可以以多种方式安装在PC机上： (1)独立分区安装、 (2)DOS分区安装和 (3)虚拟机VMWare下安装。鉴于VMware下安装对原来系统影响较小且不影响本实验目的，因此采用VMWare下安装方式。

**4、实验步逐**

(1) 在Windows XP下安装VMware 5.5

1. 配置虚拟机
2. 启动虚拟机
3. 启动Linux安装过程
4. 安装过程配置
5. 安装后配置
6. 第1次启动 VMWare下Linux操作系统

**5、实验记录**

1. 记录详细安装过程

安装VMware 5.5



导入相应光盘映像文件



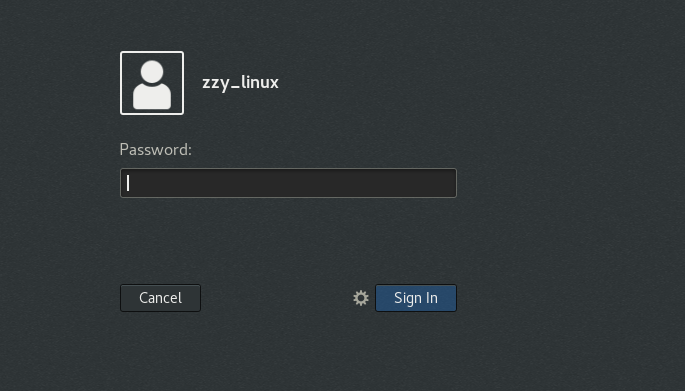
输入用户信息

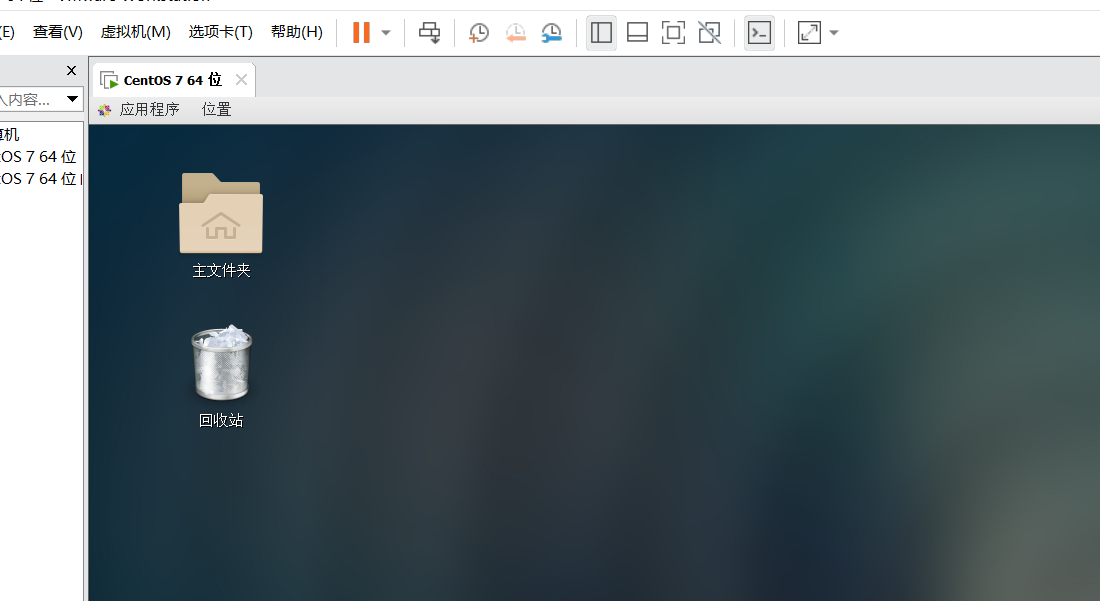


启动虚拟机



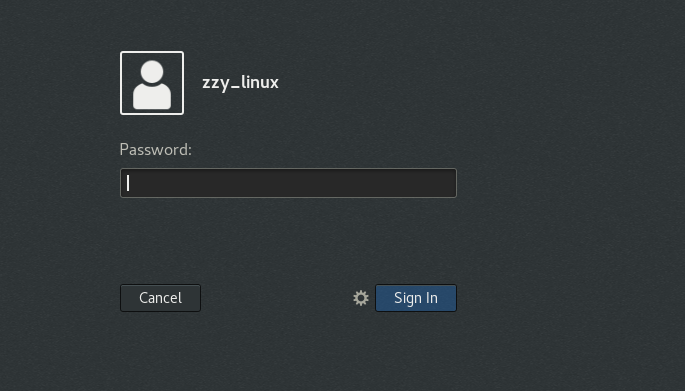
用户登录





1. 安装过程中出现的问题及其解决措施

设置完用户名，总是显示密码错误，登陆失败进不去，多次尝试重建虚拟机，发现需要在这个界面点击该按钮，进入系统用户创建界面创建用户。



**实验2 Linux基本操作**

**1、实验目的**

(1)复习Linux基本命令。

(2)掌握常用Linux命令

**2、实验内容**

**建议在自己安装的Linux系统、华为云平台、OpenEuler等系统下操作使用。**

(1)练习命令行模式下的常用命令：

man:命令帮助

ALT-Fx:虚终端切换

常用命令:cat、Ls、ps、chmod、kill、ln、cp、mv、rm、cd、pwd、mkdir、chown、who、w、wc、whoami、date、uname、find、grep等

(2) 编辑器vi的使用：使用vi建立并修改一个文本文件

(3) 档案管理

rpm、gzip、tar等命令

(4) 网络命令

telnet、ftp、talk

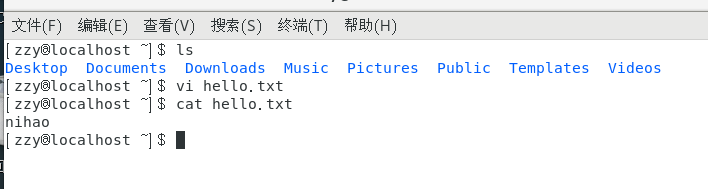
**3、实验记录**

(1)在实验过程中，注意记录实验现象

man：命令帮助

用法：man [命令名]

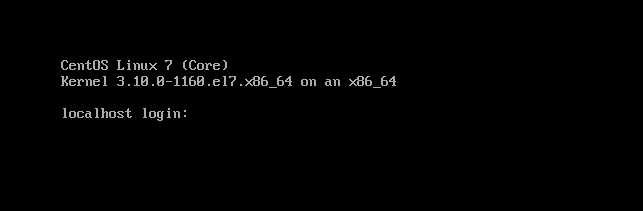
示例：man ls

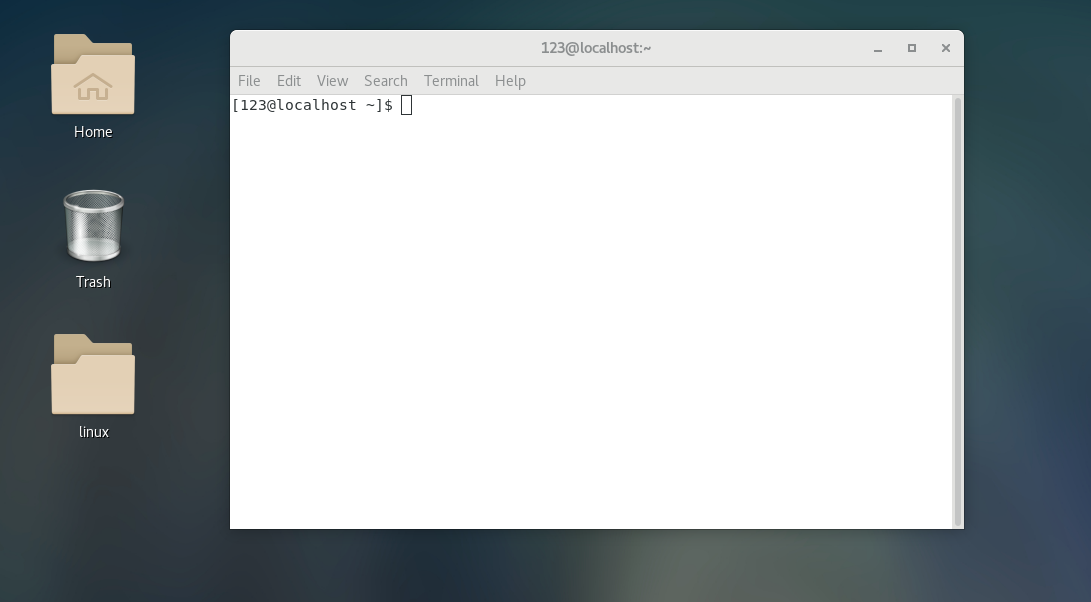


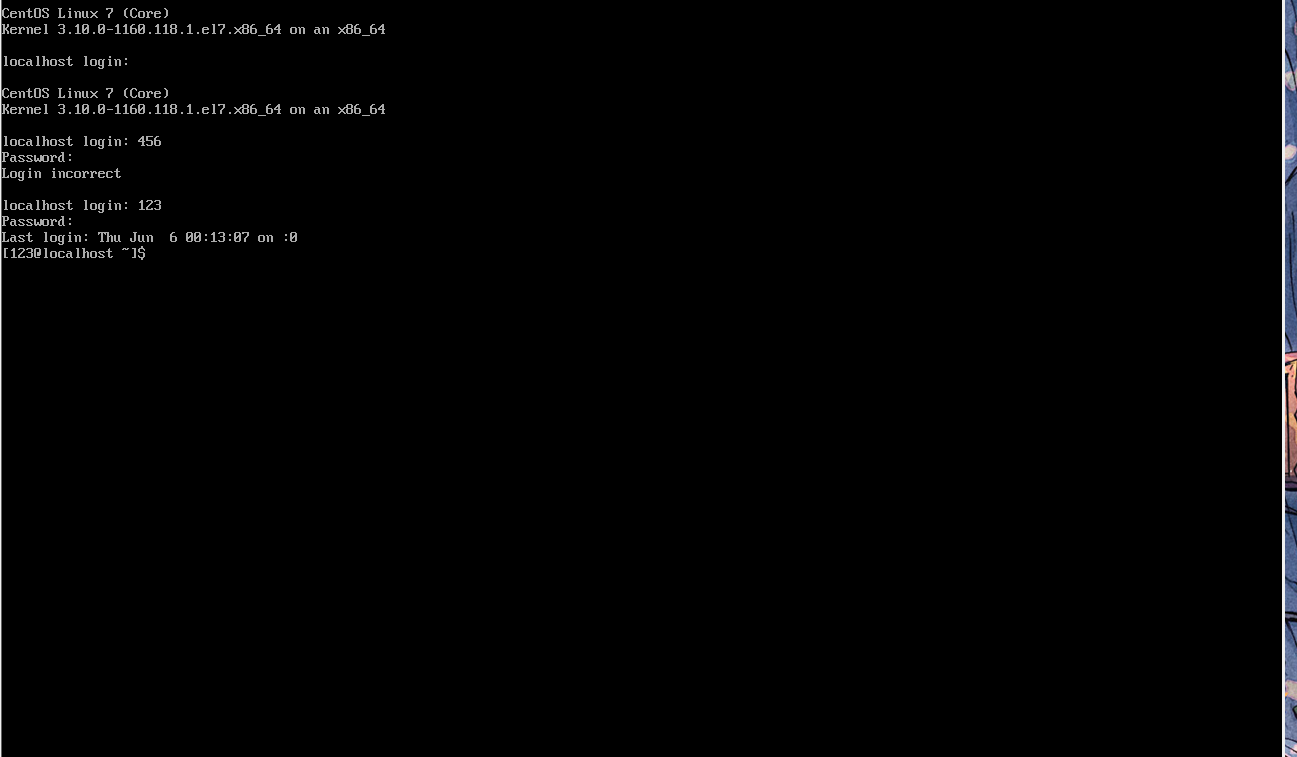
ALT-Fx:虚终端切换

用法：按 Alt-F1 至 Alt-F6 切换到不同虚拟终端。

实验现象：切换到不同终端后，每个终端运行独立的会话。按 Alt-F1 返回第一个终端。







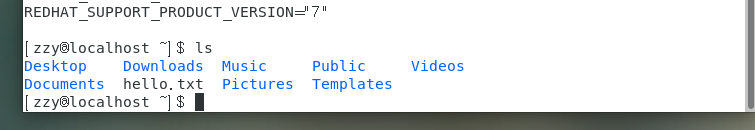
常用命令:cat、Ls、ps、chmod、kill、ln、cp、mv、rm、cd、pwd、mkdir、chown、who、w、wc、whoami、date、uname、find、grep等

| 命令 | 功能 | 示例 |
| --- | --- | --- |
| cat | 查看文件内容 | cat /etc/os-release 显示系统版本信息 |
| ls | 列出目录内容 | ls -lh 显示当前目录文件，格式为人类可读 |
| ps | 查看进程状态 | ps aux |
| chmod | 修改文件权限 | chmod 644 file.txt 设置文件权限为 644 |
| kill | 终止进程 | kill -9 1234 强制终止进程 ID 为 1234 |
| ln | 创建链接 | ln -s /etc/os-release mylink 创建符号链接 |
| cp | 复制文件 | cp file1.txt /tmp/ 复制文件到 /tmp 目录 |
| mv | 移动或重命名文件 | mv oldname.txt newname.txt 重命名文件 |
| rm | 删除文件 | rm -i file.txt 删除文件前提示确认 |
| cd | 切换目录 | cd /var/log 切换到 /var/log |
| pwd | 显示当前工作目录 | pwd 显示当前所在路径 |
| mkdir | 创建目录 | mkdir newdir 创建名为 newdir 的目录 |
| chown | 修改文件的所有者 | chown user:group file.txt 将文件所有者改为 user，组改为 group |
| who | 查看当前登录用户 | who 显示登录的用户及终端 |
| w | 查看当前登录用户及运行情况 | w 显示登录用户及负载情况 |
| wc | 统计文件的行数、单词数、字符数 | wc -l file.txt 显示文件行数 |
| whoami | 显示当前用户名称 | whoami |
| date | 显示或设置系统日期时间 | date “+%Y-%m-%d %H:%M:%S” |
| uname | 显示系统内核及版本信息 | uname -a |
| find | 查找文件 | find / -name “\*.log” 搜索以 .log 结尾的文件 |
| grep | 搜索文本 | grep “error” /var/log/syslog 搜索日志中的 “error” 关键字 |

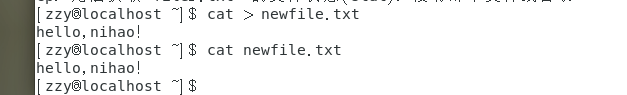
**查看文件内容：**cat cat /etc/os-release



**列出文件：**ls ls -lh /home



cat > newfile.txt 创建新文件，并输入文件内容 按ctrl+d可退出编辑状态



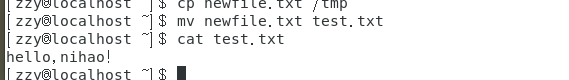
**复制文件：**cp cp file1.txt /tmp/

将 file1.txt 复制到 /tmp 目录。



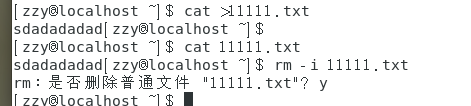
**移动或重命名：**mv mv file1.txt file2.txt

将 file1.txt 重命名为 file2.txt。



**删除文件：**rm rm -i file2.txt

删除文件，-i 参数会提示确认。



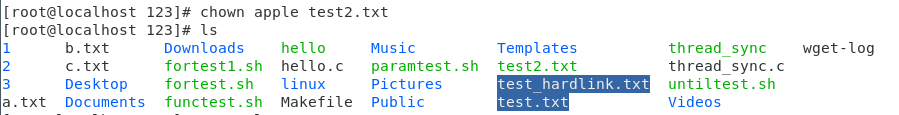
**修改权限：**chmod

**chown**：改变文件或目录的所有者（owner），所属组（group）

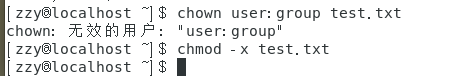
改变文件或目录的所有者chown new\_owner filename

查询所有用户：（后面的命令创建的用户apple和banana）

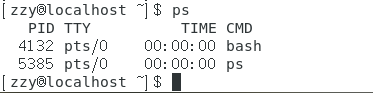


将test2.txt 转移到apple用户下，hown 命令不会移动文件或改变它的位置，它只更改文件的元数据，使用 chown 命令改变一个文件的所有权时，这个文件仍然保留在它原来的目录中。

**更改所有者：**chown chmod permissions filename，其中permissions是新权限的表示，filename是文件或目录的名称。

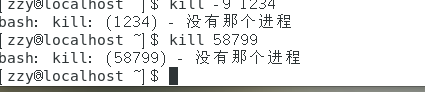


**ps：显示当前进程**



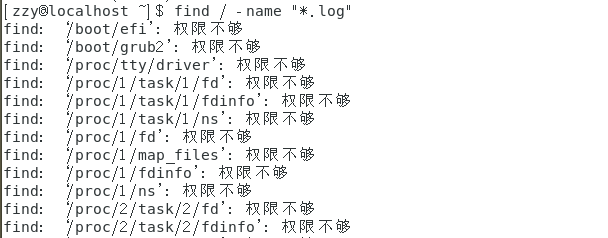
**终止进程：**kill kill -9 1234

强制终止进程 ID 为 1234 的进程。

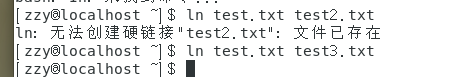


**查找文件：**find find / -name "\*.log"

在根目录下查找所有 .log 文件。



**ln**：创建链接，主要有硬链接和符号链接两种类型

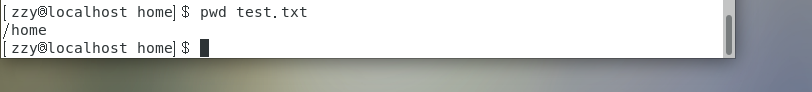


**cd**：用于改变当前工作目录

**切换到指定目录**cd /path/to/directory, 这将切换当前工作目录到指定的绝对路径 /path/to/directory。



**pwd**: 显示当前工作目录的完整路径

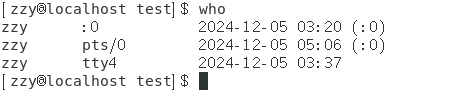


**mkdir**: 创建新目录

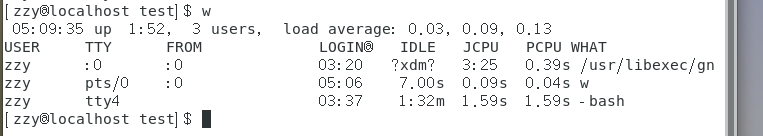
mkdir directory\_name这将在当前工作目录下创建一个名为 directory\_name 的新目录，可创建多个目录



**who**：显示当前登录的用户：

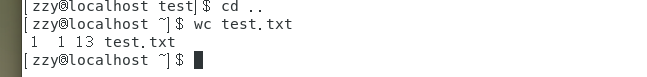


**w**：显示当前已登录用户的信息, 包括他们正在执行的命令和他们的登录时间



**wc**: 计算文件中的行数、单词数、字节数和字符数

可输入 -l 只计算行数，-w只计算单词数，-c只计算字节数，-m只计算字符数，直接输入则显示行数，单词数，字节数



**whoami**：显示当前执行命令用户的用户名



**uname**：显示系统信息，输入-a显示所有系统信息



**date**：**显示当前日期和时间**

**若要求格式化输出时间可输入：** date +"%Y-%m-%d %H:%M:%S"



**grep**：用于搜索文件中的文本模式



（2）编辑器vi的使用：使用vi建立并修改一个文本文件

**1. 创建或打开文件**：

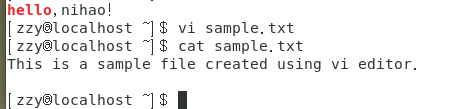
vi example.txt

**2. 进入插入模式**： 按 i 开始输入内容。保存并退出：

**3. 保存并退出**：

* 按 ESC 键退出插入模式。
* 输入 :wq 保存并退出。
* 如果不保存，输入 :q! 强制退出。



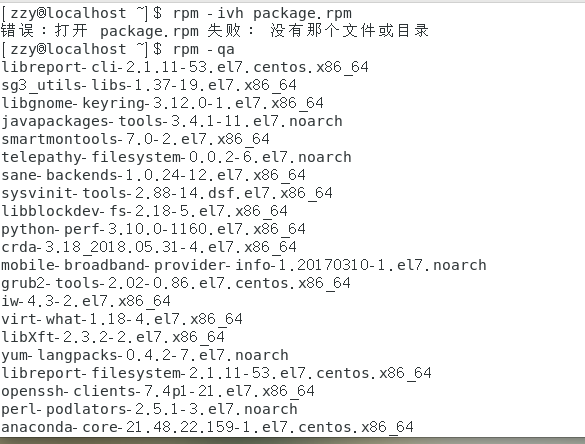


(3) 档案管理

rpm、gzip、tar等命令

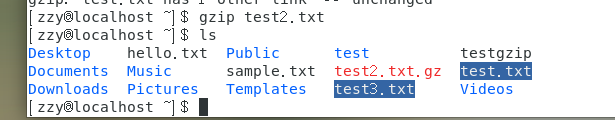
**1. 安装 RPM 包**： rpm -ivh package.rpm

安装过程中显示进度条，成功后可以用 rpm -qa 检查是否安装。

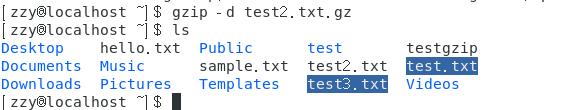


**2. 压缩与解压**：

**gzip 压缩**：

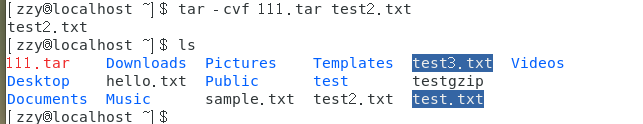


解压



**3. tar**：归档

tar -cvf archive\_name.tar directory\_or\_file：创建一个名为 archive\_name.tar 的归档文件，其中 directory\_or\_file 是要归档的目录或文件。

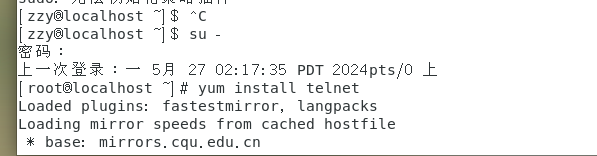


(4) 网络命令

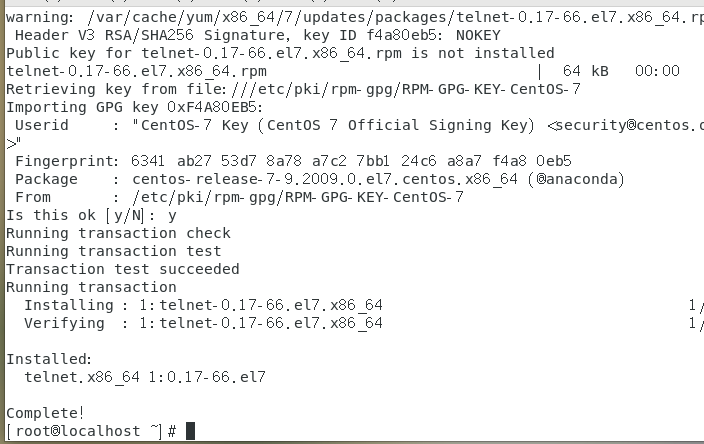
**telnet**：

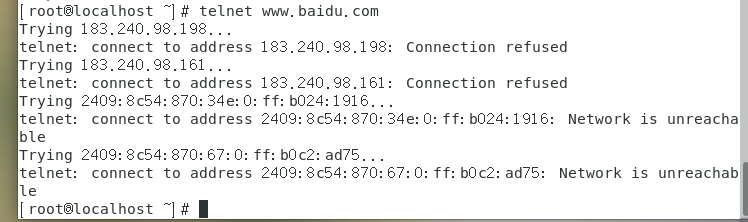
输入telnet相关命令显示没有这种命令，centos7默认情况下不安装telnet命令，可选择手动安装。

**切换到 root 用户**： 输入 root 用户的密码。成功登录后，提示符会变为 #，表示你已切换为 root 用户。



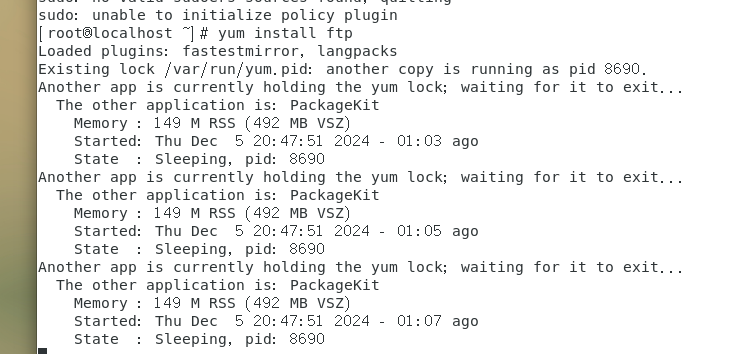
下载



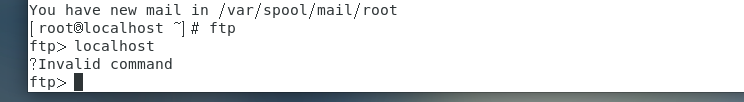


**ftp**：是一种网络协议，用于在网络上的计算机之间传输文件。它允许用户上传和下载文件，以及执行其他文件管理操作，如重命名、移动和删除文件

这个我们默认没有安装，仍然需要手动下载

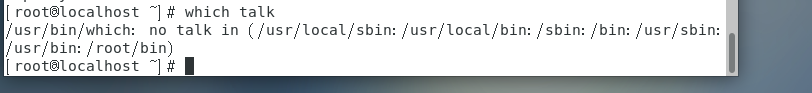


输入ftp即可开始连接，输入bye退出连接。



**Talk**

这个我们默认没有安装，仍然需要手动下载



**（2）心得体会**

#### ****1. 命令行常用命令****

在实验过程中，我练习了多种 Linux 命令，包括文件操作、权限管理、进程管理等。最常用的命令有：

**文件管理命令：** cat、ls、cp、mv、rm、mkdir 等。通过这些命令，我可以方便地查看、复制、移动和删除文件。特别是 ls 命令，在不同的参数下（如 ls -l、ls -a 等）展示了不同的信息，帮助我更好地了解文件夹内容。

**权限管理命令：** chmod、chown、chgrp。通过这些命令，我掌握了如何修改文件或目录的权限、拥有者和用户组。比如，通过 chmod 755 来设置文件的可执行权限。

**进程管理命令：** ps、kill。这些命令让我了解了如何查看当前系统中运行的进程，并通过 kill 来结束特定的进程，避免系统过载。

**系统信息命令：** uname、whoami、date。这些命令帮助我快速获取当前系统的信息，如操作系统类型、当前登录的用户名、系统时间等。

#### ****2. 编辑器 vi 的使用****

vi 编辑器的使用是实验中一个重要的部分。尽管最初我对命令行编辑器不太熟悉，但经过一段时间的练习，我逐渐掌握了它的基本操作。

**基本操作：** 进入 vi 后，我学习了如何进入插入模式、退出插入模式以及保存退出文件。例如，使用 i 进入插入模式，输入内容后按 ESC 键退出，再通过 :wq 保存并退出文件。对于不需要保存的操作，使用 :q! 强制退出。

**文本编辑：** 使用 yy 复制一行，dd 删除一行，p 粘贴，u 撤销等命令，使得文本编辑变得更加高效。此外，使用 / 查找文本功能也极大提高了查找和定位内容的速度。

#### ****3. 档案管理命令****

在进行文件压缩与解压缩的实验时，我使用了 gzip、tar 和 rpm 等命令。

**tar：** tar 是一个非常有用的命令，可以打包多个文件成一个文件，方便管理。通过 tar -cvf 创建归档文件，tar -xvf 解压缩文件，我熟悉了该命令的用法，能够轻松处理大批量文件的打包和解压。

**gzip：** 用于压缩文件，通过 gzip 和 gunzip 来压缩和解压文件。它能够减小文件的大小，节省存储空间，非常适合用来处理日志文件或大量数据文件。

**rpm：** rpm 命令用于管理系统中的安装包。通过 rpm -ivh 安装包和 rpm -e 卸载包，我对 CentOS 系统的包管理有了更深的理解。

#### ****4. 网络命令****

在实验中，我还使用了 telnet、ftp 和 talk 命令进行网络相关的操作。

**telnet：** telnet 用于远程连接到主机，虽然不加密数据传输，但作为基础的网络连接工具，它帮助我理解了如何通过命令行进行远程管理。不过，现在通常推荐使用 SSH 来替代 telnet，以提高安全性。

**ftp：** ftp 用于文件传输，我学习了如何上传和下载文件，掌握了基本的文件管理操作。ftp 在传输大文件时非常有用，但同样由于缺乏加密，现在更推荐使用 SFTP。

**talk：** talk 是一个较为古老的即时通讯工具，允许两台计算机之间的用户进行实时聊天。尽管功能简单，但对于老旧系统或者需要简易沟通的场景仍然有效。

**实验3 Linux系统管理**

**建议在自己安装的Linux系统、华为云平台、OpenEuler等系统下操作使用。**

**1、实验目的**

(1)掌握Linux基本系统管理命令

(2)掌握基本管理方法

(3)初步了解Linux中与系统管理相关的有关配置文件

**2、实验内容**

(1) 用户管理

adduser、useradd、passwd等

观察：/etc/passwd文件

(2) 文件系统管理

df、du命令使用

mount、umount命令使用：安装usb盘或者光盘，查看分析/etc/fstab文件

(3) 存储器

free：了解存储器使用情况

(4) 关机

halt、shutdown –h、init 0

reboot、shutdown -r now

并比较它们之间的差别

(5) 网络配置

netconfig的使用

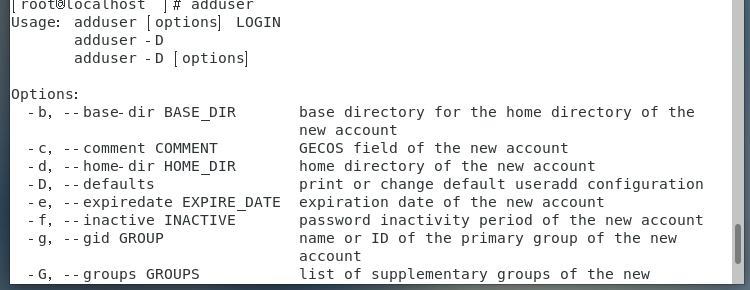
（6）启动配置

/etc/inittab：查看、修改并观察其响应

**3、实验记录**

(1) 用户管理

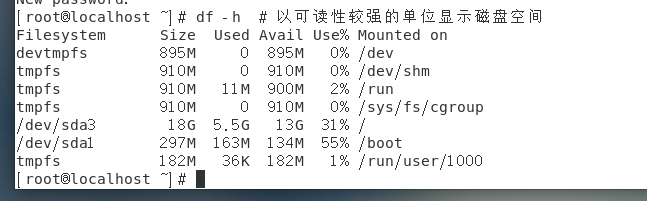
输入adduser，我们有很多选项，我们根据需求进行管理员的添加。



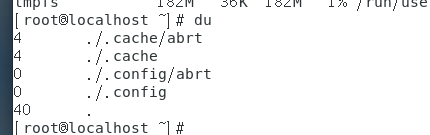
这个是修改我们的密码



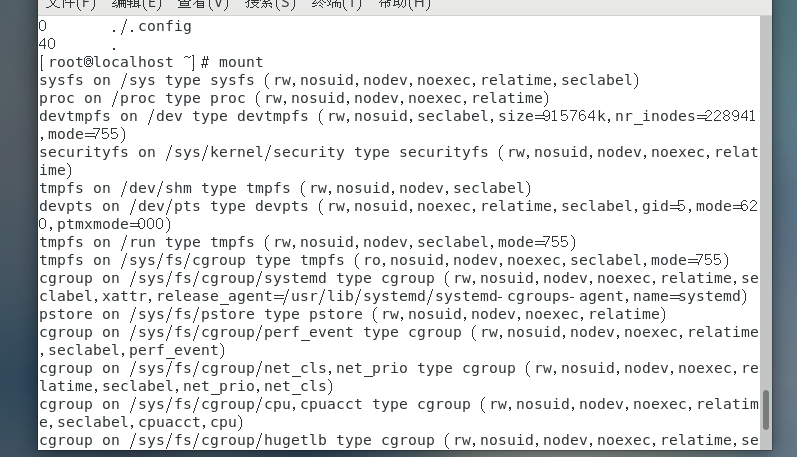
(2) 文件系统管理

df 用于显示文件系统的磁盘空间使用情况，它帮助我们了解系统中各个分区的剩余空间。

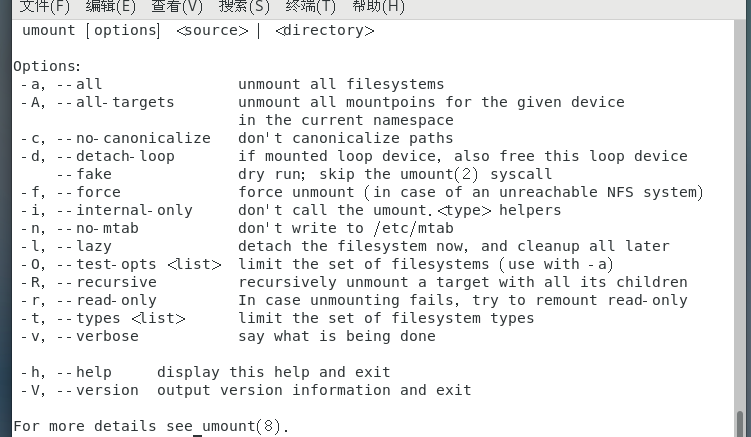
du 用于查看目录或文件占用的磁盘空间。可以结合 -h 参数显示为易于理解的单位



mount 用于挂载文件系统，通常是将外部存储设备（如 USB 或光盘）挂载到指定的目录。

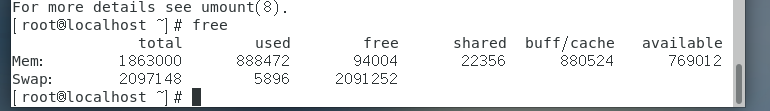


umount 用于卸载已挂载的设备。



(3) 存储器

free：了解存储器使用情况，用于显示内存使用情况，包括物理内存、交换空间等的使用情况。



(4) 关机

halt：立即停止系统，直接关闭电源。通常会触发硬件关闭。

shutdown -h：优雅关机，执行关机操作并安全地停止所有进程，参数 -h 表示关机并关掉电源。

**init 0**：通过改变运行级别来关闭系统，0 表示系统关机。

**reboot：**直接重启系统，类似于重置系统。

**shutdown -r now：**重启系统，-r 表示重启。

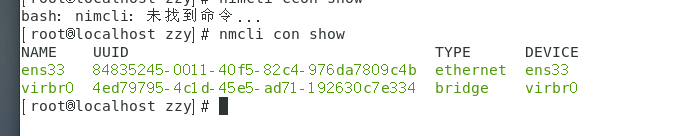


(5) 网络配置

netconfig的使用，需要手动安装。

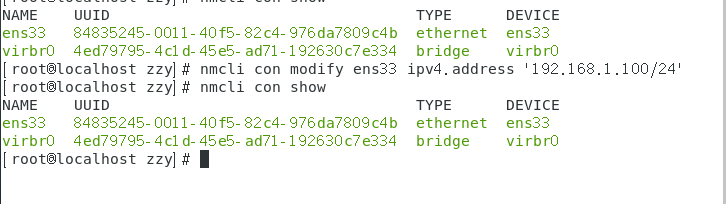


Centos7 虚拟机可使用nmcli 命令行工具来管理网络配置



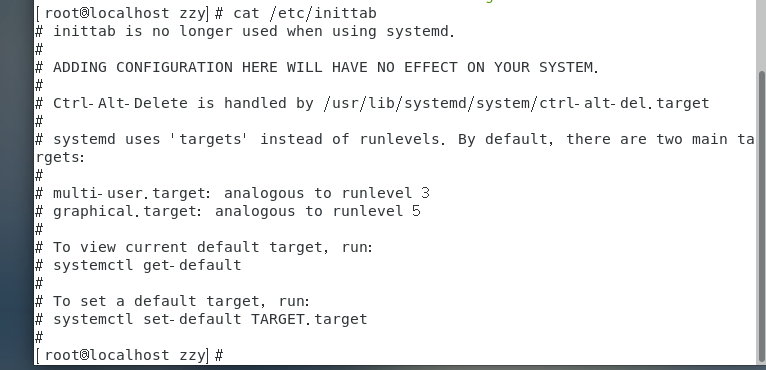
创建一个新的网络连接

nmcli con modify ens33 ipv4.address '192.168.1.100/24'

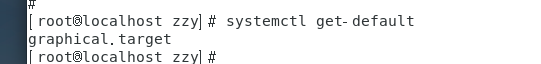


（6）启动配置

使用的是较新版本的 Linux 系统（如 CentOS 7 或 Ubuntu 16.04 及以上版本），这些系统使用 systemd 作为初始化系统，/etc/inittab 文件不再使用。



输入命令systemctl get-default显示系统当前使用的目标，可知启动配置不再通过/etc/inittab文件来管理。相反，systemd使用自己的配置文件来管理



编辑服务配置文件，可以使用文本编辑器。例如，要编辑sshd.service服务，可以使用：

sudo nano /usr/lib/systemd/system/sshd.service

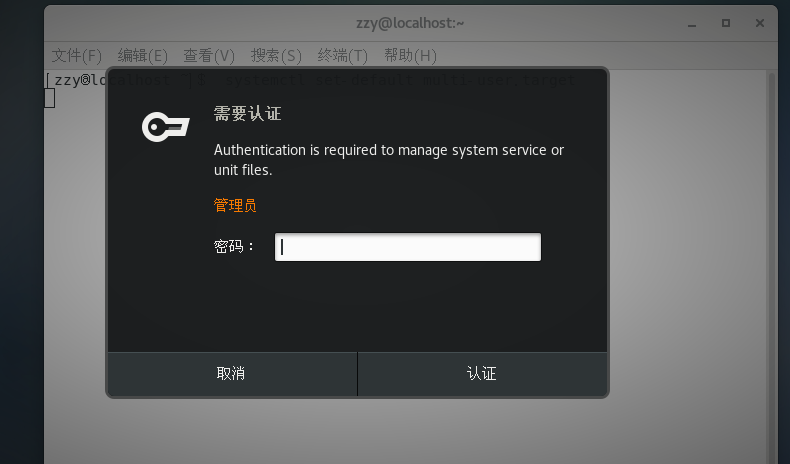


编辑完文件后，需要重新加载或重新启动服务或目标以应用更改。

例如，要重新加载sshd.service服务，可以使用：

systemctl set-default multi-user.target # 类似于运行级别 3

systemctl set-default graphical.target # 类似于运行级别 5



**4、实验体会**

#### ****1. 用户管理****

* **命令：** adduser、useradd、passwd
* **观察：** /etc/passwd 文件

**体会：** 在 Linux 中，用户管理是一个非常重要的任务，尤其是在多用户环境下。通过 useradd 或 adduser 命令创建新用户后，可以使用 passwd 命令为用户设置密码。新用户的信息被存储在 /etc/passwd 文件中，每个用户在该文件中占有一行，其中包含用户名、用户ID（UID）、用户组ID（GID）、用户信息、主目录路径、登录Shell等信息。

* adduser 和 useradd 都用于添加用户。adduser 更为用户友好，通常会引导用户输入相关信息；而 useradd 是一个较为底层的命令。
* passwd 用于设置或更改用户密码。它会提示你输入新密码并验证。

#### ****2. 文件系统管理****

* **命令：** df、du、mount、umount
* **分析：** /etc/fstab 文件

**体会：**

df 用于显示文件系统的磁盘空间使用情况，它帮助我们了解系统中各个分区的剩余空间。

du 用于查看目录或文件占用的磁盘空间。可以结合 -h 参数显示为易于理解的单位。

mount 用于挂载文件系统，通常是将外部存储设备（如 USB 或光盘）挂载到指定的目录。

umount 用于卸载已挂载的设备。

/etc/fstab 文件记录了所有文件系统的挂载信息。在这个文件中，我们可以查看系统启动时会自动挂载哪些文件系统，也可以手动修改此文件来调整挂载设置。

#### ****3. 存储器****

* **命令：** free

**体会：** free 命令用于显示内存使用情况，包括物理内存、交换空间等的使用情况。常用的 free -h 命令可以以易读的方式显示内存和交换空间的使用情况（单位为 MB 或 GB）。这是监控系统内存的重要工具之一，帮助我们了解当前系统内存的压力。

#### ****4. 关机****

* **命令：** halt、shutdown -h、init 0、reboot、shutdown -r now

**体会：** 这些命令都用于关机或重启，但它们之间存在一些细微的区别：

halt 和 shutdown -h 都会关闭系统，但 shutdown 命令更安全，会通知其他用户系统即将关机，允许系统做好关机准备。init 0 等价于 shutdown -h，而 reboot 则重启系统。

#### ****5. 网络配置****

* **命令：** netconfig

**体会：** netconfig 是一种较为简单的网络配置工具，可以用来设置主机名、IP 地址、网关、DNS 等网络配置。在 CentOS 或其他基于 Red Hat 的系统中，netconfig 提供了一个简单的命令行界面，方便用户进行网络配置。

通过运行 netconfig，用户可以选择配置静态 IP 或动态 IP。它适用于一些基础的网络配置，但在较复杂的网络环境下，可能需要手动编辑网络配置文件（如 /etc/sysconfig/network-scripts/ 下的配置文件）。

#### ****6. 启动配置****

* **文件：** /etc/inittab

**体会：** /etc/inittab 文件是系统初始化的重要配置文件，用于指定系统启动时的运行级别和启动过程。通过修改该文件，可以调整系统在启动时执行的操作，如设置默认的运行级别、自动启动服务等。

**实验4 Linux Shell程序设计**

**建议在自己安装的Linux系统、华为云平台、OpenEuler等系统下操作使用。**

**1、实验目的**

(1)掌握Linux shell程序运行方法

(2)掌握Linux Shell程序基本语法

(3)了解Linux环境变量

(3)编写简单Linux shell程序

**2、实验内容**

（1）查看/etc/.profile文件：相当于DOS下autoexe.bat

（2）通配符“\*”、“?”、“[]”的使用

$ls [a-c]\* 和 $ls [a,m,t]\*命令

(3)重定向和管道的使用

ls | more

cat > test.txt

(4) 变量

$lookup=/usr/mydir

$echo $lookup

$export lookup：让进程使用

(5) 编辑并运行以下程序（shell程序控制结构）

1 fortest

#!/bin/bash

for a in x y z

do

echo now a=$a

done

2 fortest1

#!/bin/bash

for a

do

echo now a=$a

done

3 functest

#!/bin/bash

setup()

{

echo setup…

}

do\_date()

{

date

}

chgdir()

{

cd $1

}

do\_date

setup

chgdir

4 paramtest

#!/bin/bash

echo filename:$0

echo arguments:$\*

echo number arg:$#

echo arg2:$2

shift

echo number arg:$#

echo arg2:$2

set hello,everone

echo args:$\*

echo arg2:$2

5 untiltest

#!/bin/bash

number=0

until (test $number –gt 5)

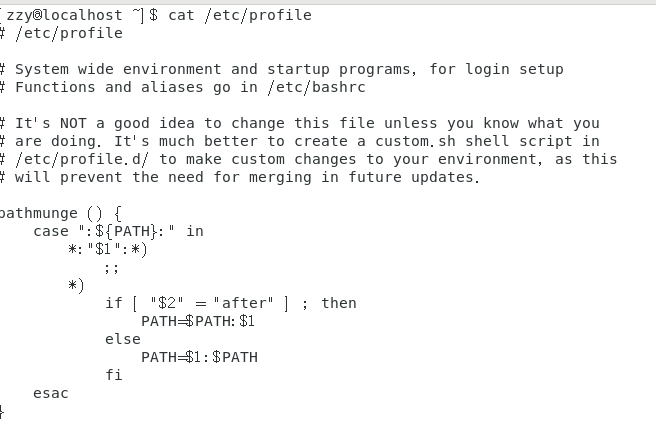
do

echo “ $number”

number=`expr $number + 1`

done

1. **实验要求**
2. 查看/etc/.profile文件：相当于DOS下autoexe.bat

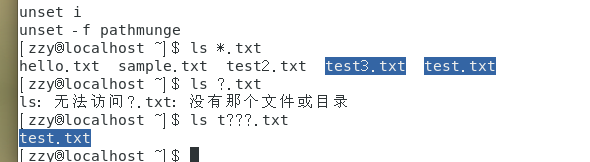


1. 通配符“\*”、“?”、“[]”的使用

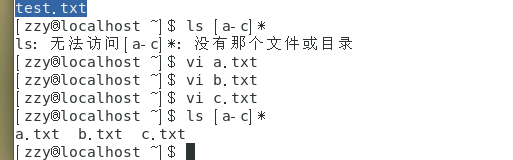
\*: 匹配任意数量的字符

**?**：匹配任意单个字符

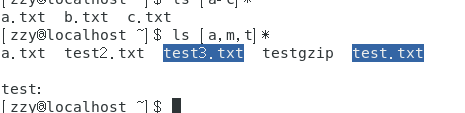
当输入？？？时，可匹配到test.txt，说明？只能匹配单个字符



ls [a-c]\* # 列出以 a、b、c 开头的文件或目录



ls [a,m,t]\* # 列出以 a、m、t 开头的文件或目录



1. 重定向和管道的使用

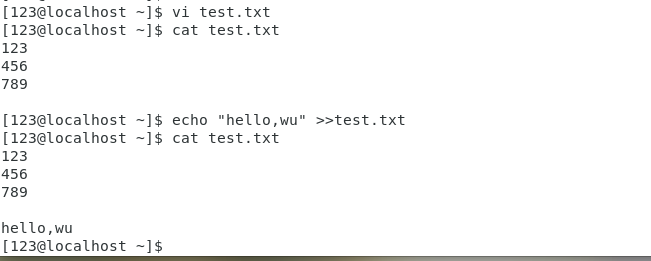
**输出重定向**：

>：将输出重定向到文件，如果文件已存在，会被覆盖。

>>：将输出追加到文件末尾，而不是覆盖它。

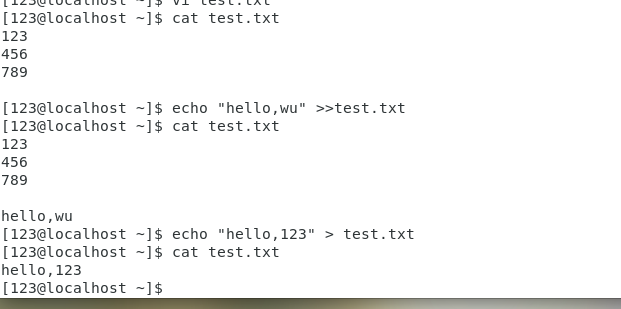
2>：将错误输出重定向到文件。

&>：将标准输出和错误输出都重定向到同一个文件。



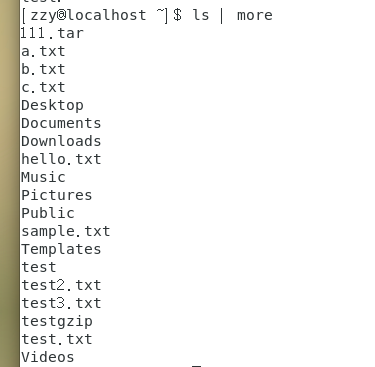
**输入重定向：**

<：从文件读取输入。



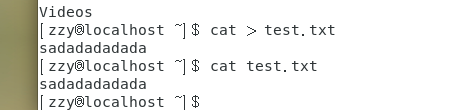
**其中：**

· **ls | more：**将 ls 输出通过管道传递给 more，分页显示。



**管道**（|）允许你将一个命令的输出作为另一个命令的输入。这是通过将前一个命令的标准输出连接到下一个命令的标准输入来实现的。

· **cat > test.txt：**输入内容并保存到 test.txt 文件中（Ctrl+D 结束输入）。



(4) 变量

$lookup=/usr/mydir

$echo $lookup

$export lookup：让进程使用



(5) 编辑并运行以下程序（shell程序控制结构）

1 fortest

#!/bin/bash

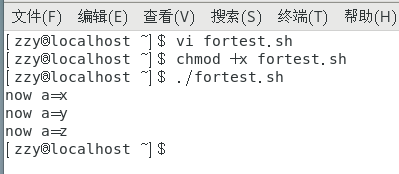
for a in x y z

do

echo now a=$a

Done

用vi创建fortest.sh,赋予执行权限，运行代码



2 fortest1

#!/bin/bash

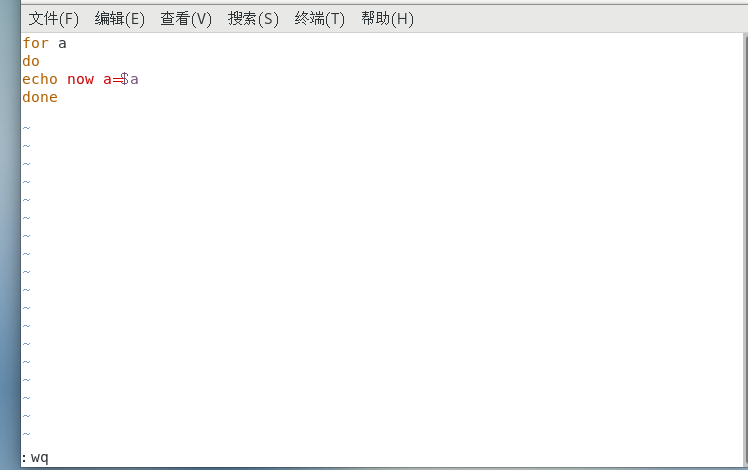
for a

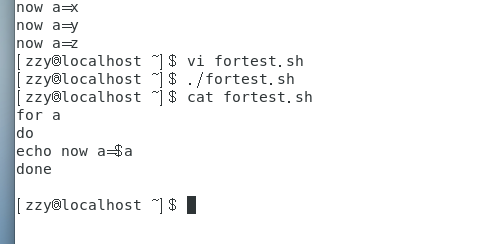
do

echo now a=$a

Done

脚本缺少迭代的元素，for循环不能正确执行，因此没有输出





3 functest

#!/bin/bash

setup()

{

echo setup…

}

do\_date()

{

date

}

chgdir()

{

cd $1

}

do\_date

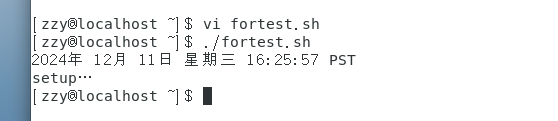
setup

Chgdir

定义了三个函数：setup、do\_date和chgdir，并调用了它们

* setup 函数：打印字符串 “setup…” 到控制台。
* do\_date 函数：调用 date 命令，打印当前日期和时间。
* chgdir 函数：接受一个参数（一个目录路径），并尝试改变当前工作目录到该路径。

发现显示错误，是chgdir(){cd $1} 这个函数输入的参数与定义的输入参数格式不同，导致失败



4 paramtest

#!/bin/bash

echo filename:$0

echo arguments:$\*

echo number arg:$#

echo arg2:$2

shift

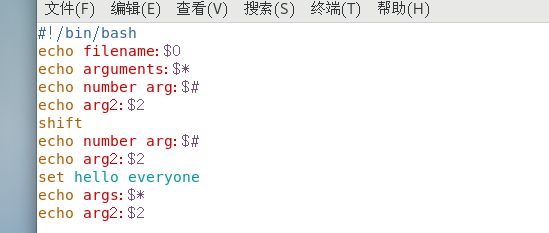
echo number arg:$#

echo arg2:$2

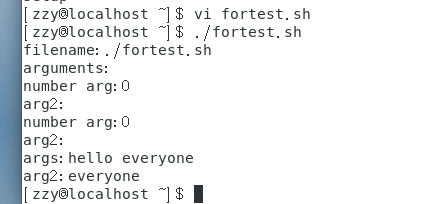
set hello,everone

echo args:$\*

echo arg2:$2



* echo filename:$0：打印脚本名称。
* echo arguments:$\*：打印所有传递给脚本的参数。
* echo number arg:$#：打印传递给脚本的参数数量。
* echo arg2:$2：打印第二个参数（参数从0开始计数，$0 是脚本名称）。
* shift：移除位置参数列表中的第一个参数，其余参数向左移动。
* 再次打印参数数量和第二个参数，此时因为 shift，原来的第二个参数变成了新的第一个参数。
* set hello everyone：重新设置位置参数为 “hello” 和 “everyone”。
* 再次打印所有参数和第二个参数。



5 untiltest

#!/bin/bash

number=0

until (test $number –gt 5)

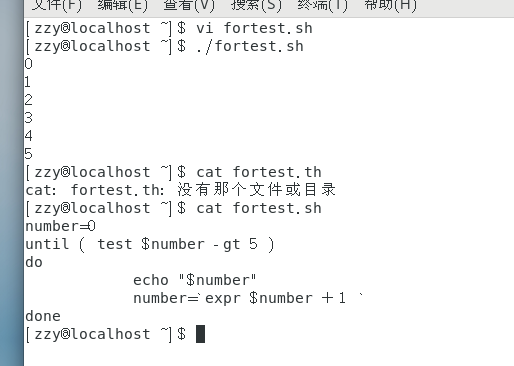
do

echo “ $number”

number=`expr $number + 1`

Done

* 初始化变量 number 为 0。
* until 循环：只要 test $number -gt 5 的条件不成立（即 number 小于或等于 5），就执行循环体内的代码。
* 在循环体内，打印当前的 number 值，并将其增加 1。
* 当 number 大于 5 时，循环结束。



**实验5 Linux 高级程序设计**

**建议在自己安装的Linux系统、华为云平台、OpenEuler等系统下操作使用。**

**1、实验目的**

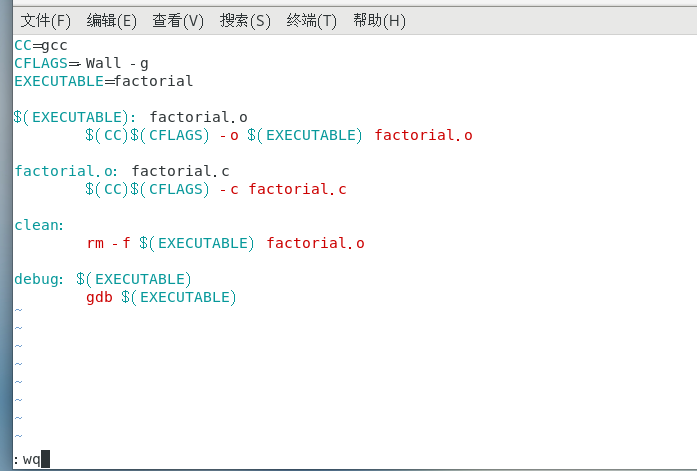
(1)了解Linux操作系统下应用程序开发流程

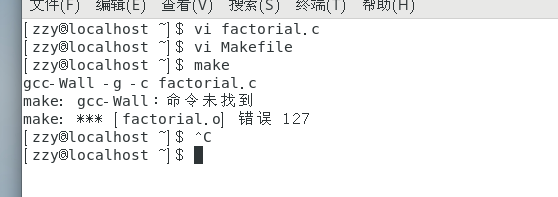
(2)掌握gun工具链的使用

(3)了解Linux高级编程技巧（例如IPC机制、系统调用等）

**2、实验内容**

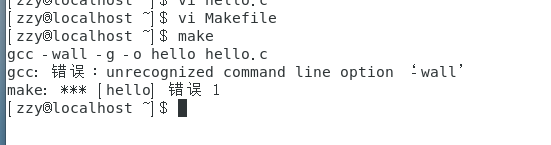
1. 编写一个简单的C语言程序，编写Makefile文件。了解编译过程，并用gdb进行调试。



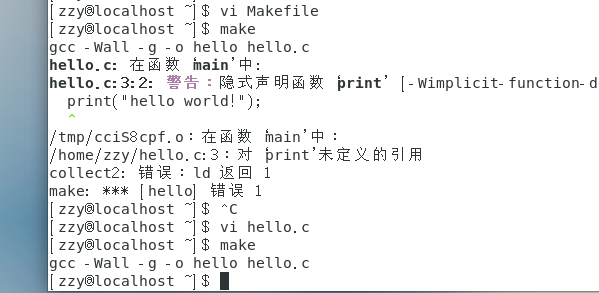


这里显示出错，错误信息表明在Makefile中，编译命令的格式有误。

我们修改一下代码



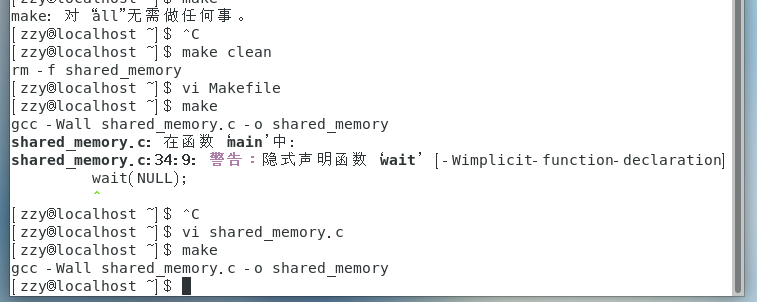
改了以后，还是有问题，这个是gcc 编译器不识别 -wall 选项。正确的选项应该是 -Wall（注意是大写的 W

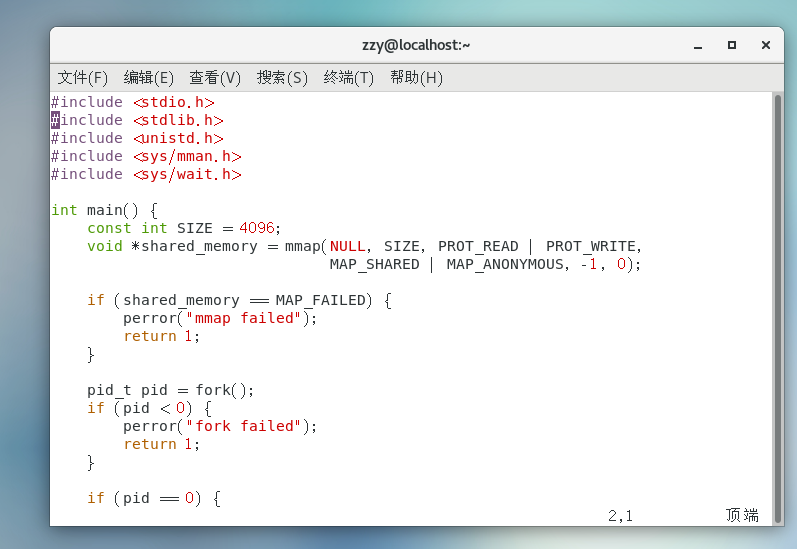


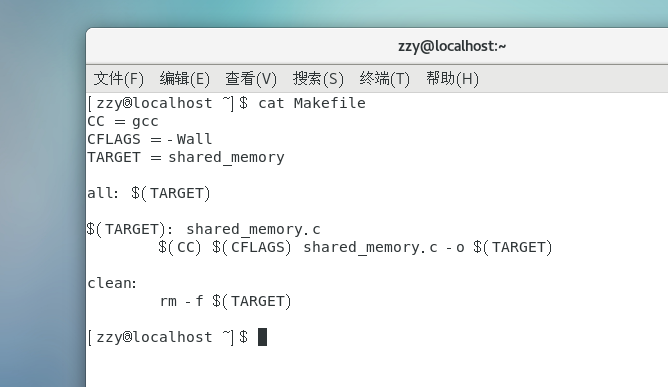
1. 以下任选其一：
   1. 编写一个多进程通信程序，采用Message Queue或shared Memory或者Maped File机制进行通信
   2. 编写一个多线程程序(pthread)，实现2程同步互斥

第一题

采用 **共享内存 (Shared Memory)** 机制实现父子进程间的通信。







int main() {

const int SIZE = 4096;

void \*shared\_memory = mmap(NULL, SIZE, PROT\_READ | PROT\_WRITE,

MAP\_SHARED | MAP\_ANONYMOUS, -1, 0);

* 定义共享内存的大小为4096字节。
* 使用mmap函数创建一块共享内存。参数说明：
* NULL：让系统选择共享内存的地址。
* SIZE：共享内存的大小。
* PROT\_READ | PROT\_WRITE：共享内存可读可写。
* MAP\_SHARED：共享内存可以被多个进程共享。
* MAP\_ANONYMOUS：不与任何文件关联。
* -1和0：因为使用了MAP\_ANONYMOUS，这两个参数无效。

**子进程（pid == 0）：**

* 从共享内存中读取字符串并打印。
* 使用exit(0)正常退出。

**父进程：**

* 向共享内存写入字符串"Hello from parent process!"。
* 使用wait(NULL)等待子进程结束。
* 打印"Parent process done"。

父进程写入数据到共享内存，子进程从共享内存中读取数据。通过fork创建子进程，并通过wait确保子进程先执行。最后，父进程释放共享内存并结束程序。

**错误信息：**

警告信息表明 wait 函数被隐式声明。wait 是一个系统调用，用于等待子进程结束，但其定义需要包含合适的头文件。

添加头文件即可！**#include <sys/wait.h>**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/mman.h>

#include <sys/wait.h>

int main() {

const int SIZE = 4096;

void \*shared\_memory = mmap(NULL, SIZE, PROT\_READ | PROT\_WRITE,

MAP\_SHARED | MAP\_ANONYMOUS, -1, 0);

if (shared\_memory == MAP\_FAILED) {

perror("mmap failed");

return 1;

}

pid\_t pid = fork();

if (pid < 0) {

perror("fork failed");

return 1;

}

if (pid == 0) {

printf("Child process read: %s\n", (char \*)shared\_memory);

exit(0);

} else {

sprintf((char \*)shared\_memory, "Hello from parent process!");

wait(NULL);

printf("Parent process done\n");

}

munmap(shared\_memory, SIZE);

return 0;

}

**3、实验要求**

* 1. 写出源程序，并编译运行
  2. 详细记录程序调试及运行结果

**实验6 Linux内核**

**1、实验目的**

(1)了解Linux操作系内核

(2)掌握内核编译和内核升级方法

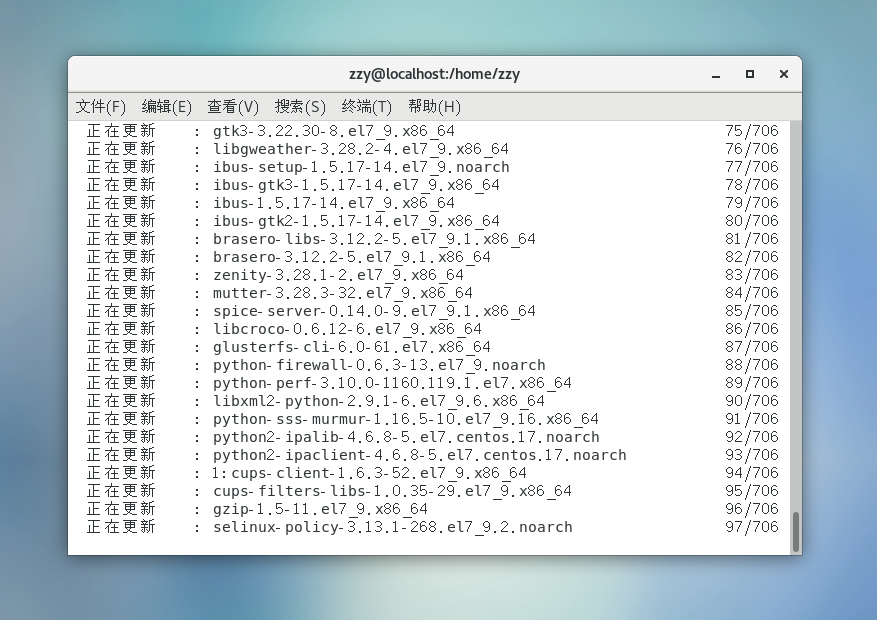
(3)了解有关内核编程

**2、实验内容**

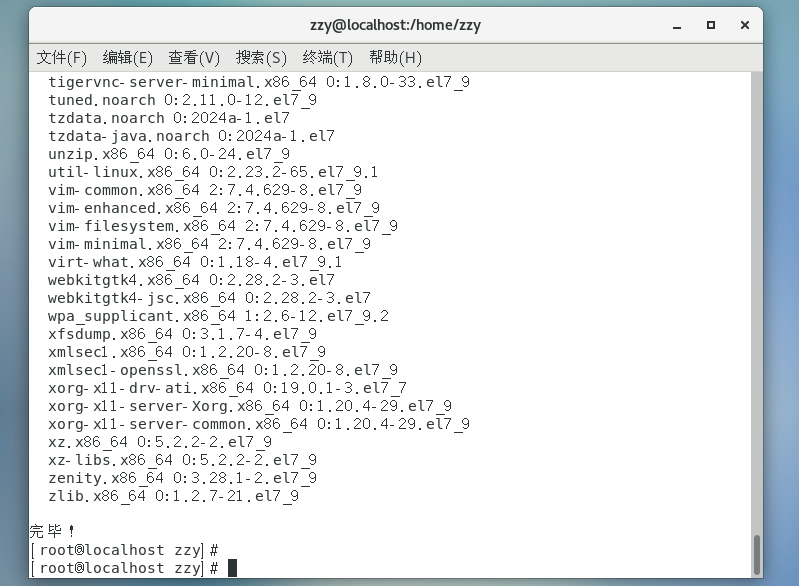
1. 内核配置和编译（课后完成）
2. 更新系统

输入su 进入管理者模式

输入 sudo yum update

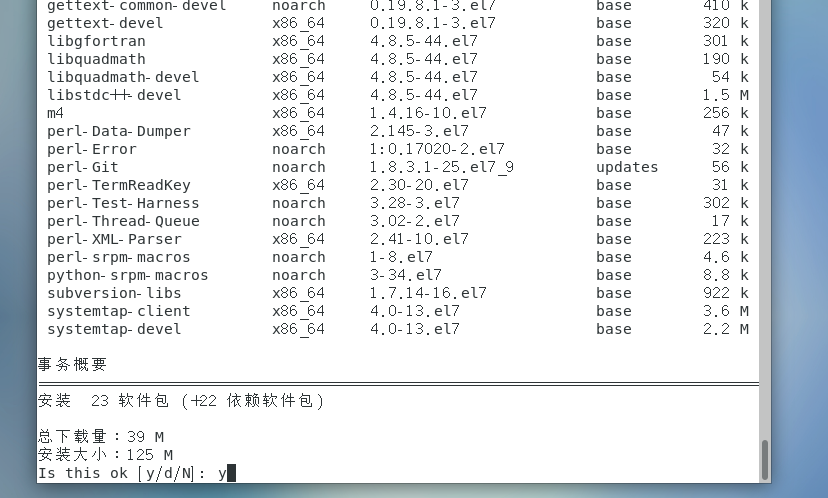


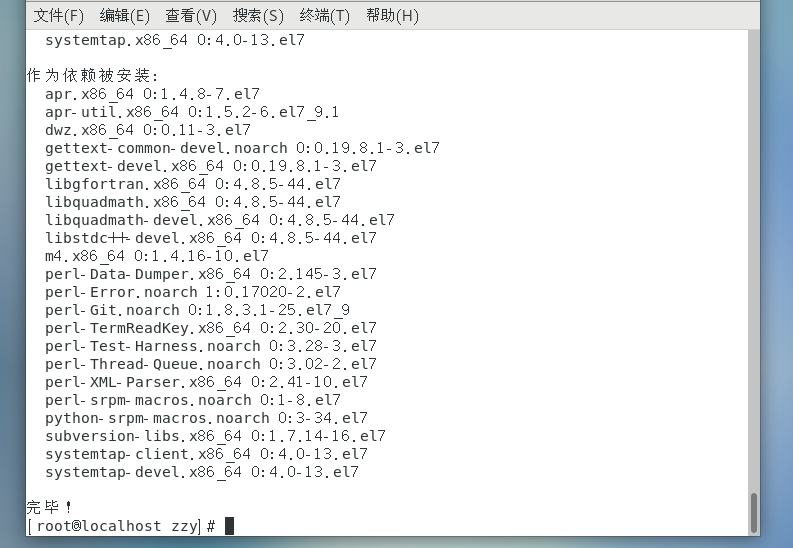
完成



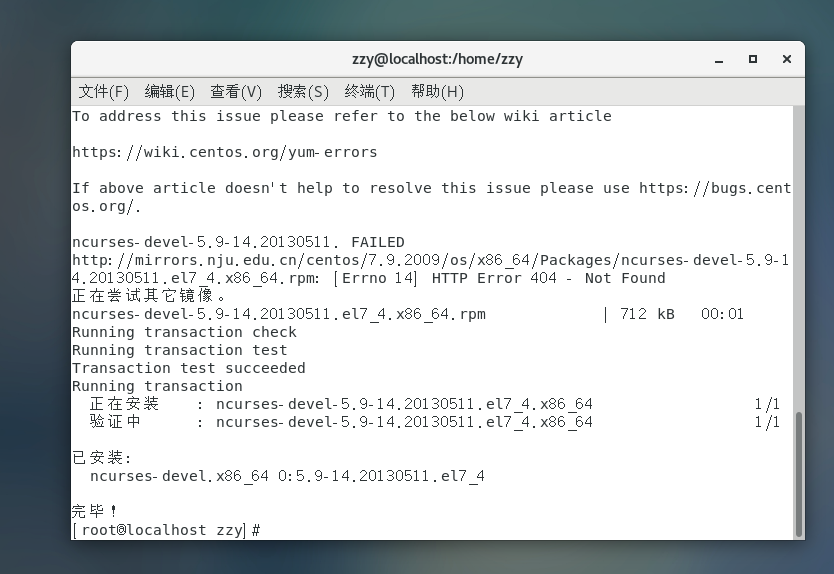
② 安装编译内核所需的依赖包

sudo yum groupinstall "Development Tools"

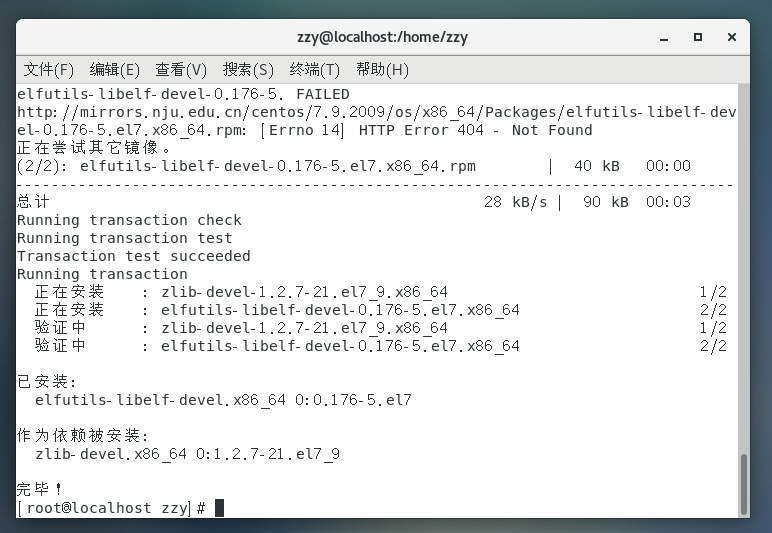




yum install ncurses-devel



yum install elfutils-libelf-devel

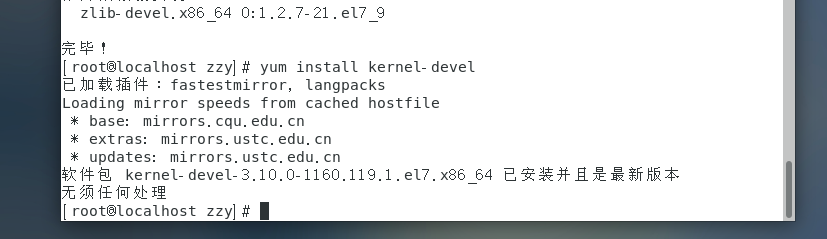


③ 获取内核源码

#输入命令获取内核源码包

sudo yum install kernel-devel

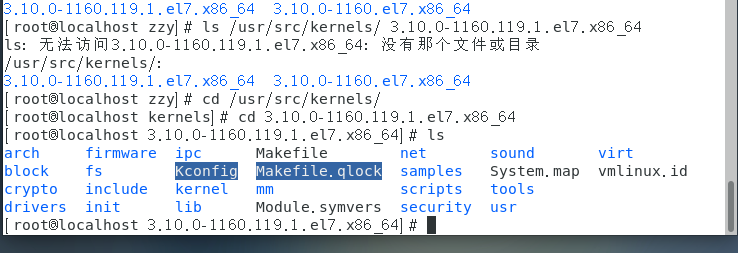
获取最新内核

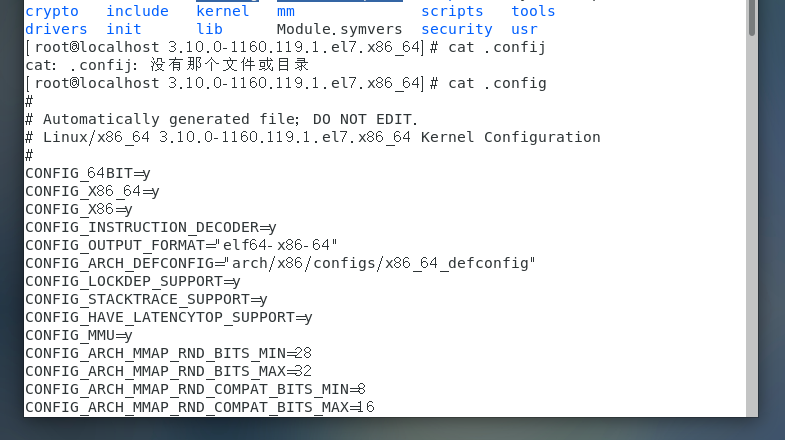


查看内核源码;

ls /usr/src/kernels/

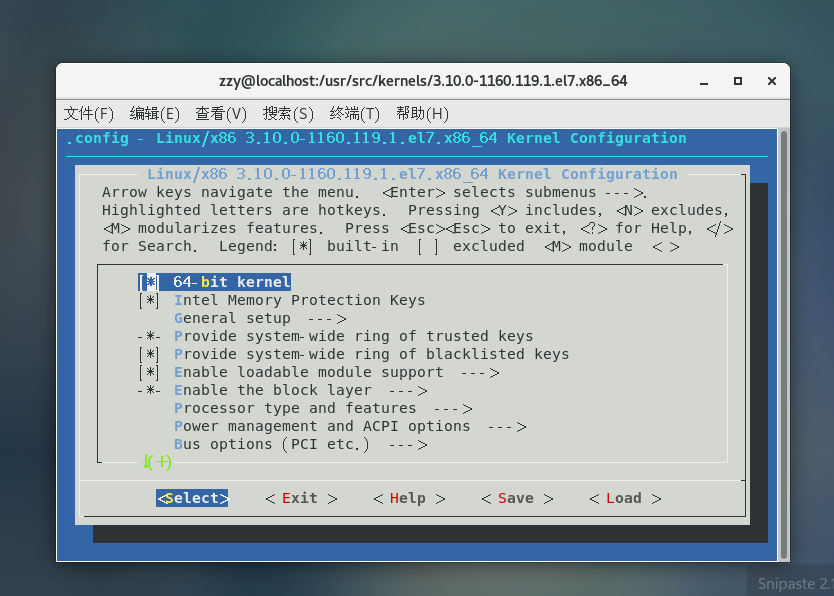
输入内核版本号进入其中一个内核目录



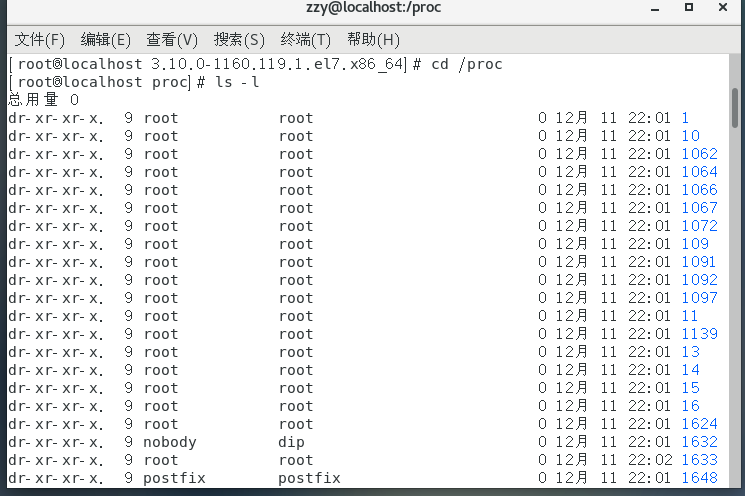


图形化地查看内核配置：

make menuconfig

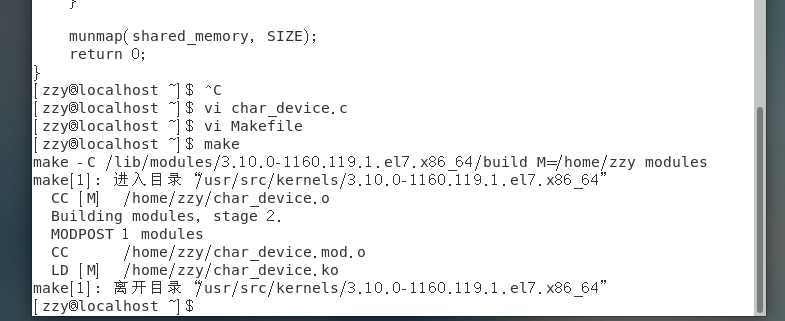


1. 察看/proc内存文件系统下的文件，写出每个目录内容



|  |  |
| --- | --- |
| 文件/目录 | 说明 |
| cpuinfo | 显示 CPU 的详细信息，例如型号、内核数、频率等。 |
| meminfo | 显示内存的详细使用情况，例如总量、已用、剩余等。 |
| uptime | 显示系统启动以来的时间和空闲时间。 |
| loadavg | 显示系统的平均负载。 |
| pid/ (数字目录) | 每个运行中的进程对应一个目录，包含该进程的详细信息，例如状态、内存使用、文件描述符等。 |
| filesystems | 显示系统支持的文件系统类型。 |
| modules | 列出已加载的内核模块及其状态。 |
| net/ | 网络相关信息，例如接口、连接状态等。 |
| devices | 显示设备文件的映射。 |
| version | 显示内核版本信息。 |

(3) 编写一个简单的字符虚拟设备程序（以模块方式编译、加载、使用）



#include <linux/module.h>

#include <linux/fs.h>

#include <linux/uaccess.h>

#define DEVICE\_NAME "char\_device"

static int major\_number;

static char message[256] = {0};

static int size\_of\_message;

static int dev\_open(struct inode \*inodep, struct file \*filep) {

printk(KERN\_INFO "char\_device: Device opened\n");

return 0;

}

static int dev\_release(struct inode \*inodep, struct file \*filep) {

printk(KERN\_INFO "char\_device: Device closed\n");

return 0;

}

static ssize\_t dev\_read(struct file \*filep, char \*buffer, size\_t len, loff\_t \*offset) {

int error\_count = copy\_to\_user(buffer, message, size\_of\_message);

if (error\_count == 0) {

printk(KERN\_INFO "char\_device: Sent %d characters to user\n", size\_of\_message);

return (size\_of\_message = 0);

} else {

printk(KERN\_INFO "char\_device: Failed to send %d characters to user\n", error\_count);

return -EFAULT;

}

}

static ssize\_t dev\_write(struct file \*filep, const char \*buffer, size\_t len, loff\_t \*offset) {

sprintf(message, "%s", buffer);

size\_of\_message = strlen(message);

printk(KERN\_INFO "char\_device: Received %zu characters from user\n", len);

return len;

}

static struct file\_operations fops = {

.open = dev\_open,

.read = dev\_read,

.write = dev\_write,

.release = dev\_release,

};

static int \_\_init char\_device\_init(void) {

major\_number = register\_chrdev(0, DEVICE\_NAME, &fops);

if (major\_number < 0) {

printk(KERN\_ALERT "char\_device failed to register a major number\n");

return major\_number;

}

printk(KERN\_INFO "char\_device: Registered with major number %d\n", major\_number);

return 0;

}

static void \_\_exit char\_device\_exit(void) {

unregister\_chrdev(major\_number, DEVICE\_NAME);

printk(KERN\_INFO "char\_device: Unregistered device\n");

}

module\_init(char\_device\_init);

module\_exit(char\_device\_exit);

MODULE\_LICENSE("GPL");

MODULE\_DESCRIPTION("A simple Linux char device");

MODULE\_AUTHOR("Your Name");

Makefile代码:

obj-m += char\_device.o

all:

make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:

make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

加载模块：

insmod char\_dev.ko



检查模块是否加载：

lsmod | grep char\_dev

可以看到模块正确加载



创建设备文件:

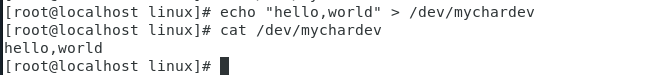
sudo mknod /dev/mychardev c 242 0

创建一个名为 mychardev 的字符设备文件，并且你的设备驱动程序使用的主设备号是 242，次设备号是 0。



测试输入:

echo "hello, world" > /dev/mychardev



1. 内核模块
   1. 内核模块

内核模块（Kernel Module）是用于Linux内核的动态可加载代码块，它们可以在运行时被加载或卸载，而不需要重新启动系统.

内核模块的加载：sudo insmod 模块文件路径.ko

内核模块的卸载：sudo rmmod 模块名称

内核模块的查看：lsmod

编写内核模块的基本步骤：

①编写模块代码: 使用C语言编写内核模块代码，通常遵循内核编程的规范。

②编译模块: 使用 make 命令和适当的内核配置文件（通常是 .config 文件）来编译模块。

③安装模块: 将编译好的模块文件（通常是 .ko 文件）复制到 /lib/modules/内核版本/kernel/ 目录下。

④加载模块: 使用上述提到的 insmod命令来加载模块。

**3、实验要求**

* + 1. 给出源程序
    2. 记录调试过程和运行结果