東北大學秦皇島分校

学院	计算机与通信工程学院
专业	计算机科学与技术
班级号	200523
学号	202012143
姓名	熊舟桐

Linux 基本命令

实验环境

本地 Linux 版本：Manjaro

```
Linux northboat-nhx0dbde 6.1.12-1-MANJARO #1 SMP PREEMPT_DYNAMIC Tue Feb 14 21:59:10 UTC 2023 x86_64 GNU/Linux
```

ssh 版本

```
OpenSSH_9.2p1, OpenSSL 3.0.8 7 Feb 2023
```

目标机版本：Debian

```
Linux VM-0-17-debian 5.10.0-19-amd64 #1 SMP Debian 5.10.149-2 (2022-10-21) x86_64 GNU/Linux
```

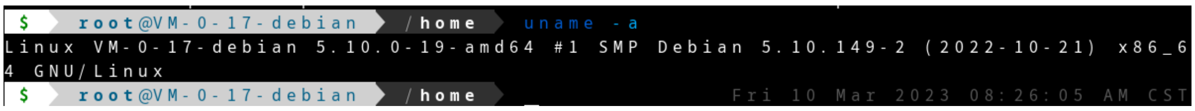
实验内容

ssh 连接 Linux

在`manjaro`上连接`debian`服务器

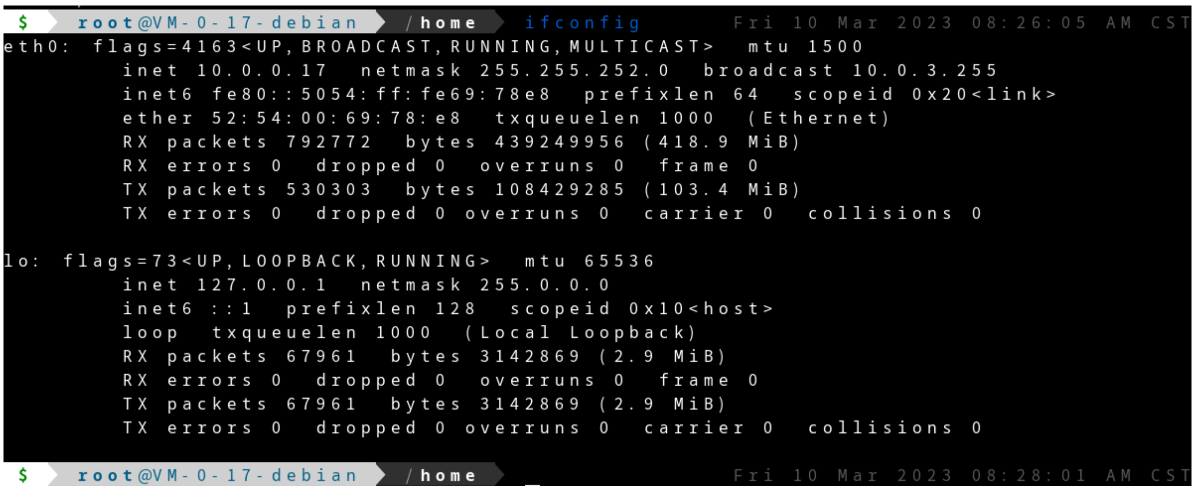
```
ssh root@43.163.218.127
```

查看主机基本信息



```
$  root@VM-0-17-debian  /home  uname -a
Linux VM-0-17-debian 5.10.0-19-amd64 #1 SMP Debian 5.10.149-2 (2022-10-21) x86_64 GNU/Linux
$  root@VM-0-17-debian  /home                              Fri 10 Mar 2023 08:26:05 AM CST
```

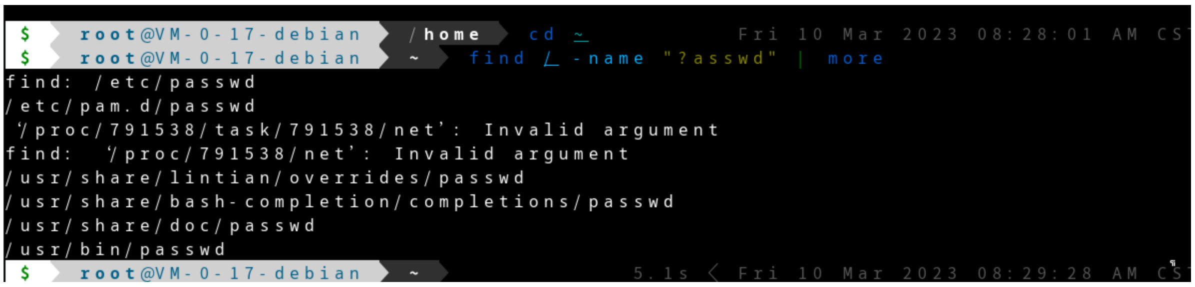
查看网卡信息



```
$  root@VM-0-17-debian  /home  ifconfig                   Fri 10 Mar 2023 08:26:05 AM CST
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.17  netmask 255.255.252.0  broadcast 10.0.3.255
        inet6 fe80::5054:ff:fe69:78e8  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:69:78:e8  txqueuelen 1000  (Ethernet)
        RX packets 792772  bytes 439249956 (418.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 530303  bytes 108429285 (103.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 67961  bytes 3142869 (2.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 67961  bytes 3142869 (2.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

$  root@VM-0-17-debian  /home                              Fri 10 Mar 2023 08:28:01 AM CST
```

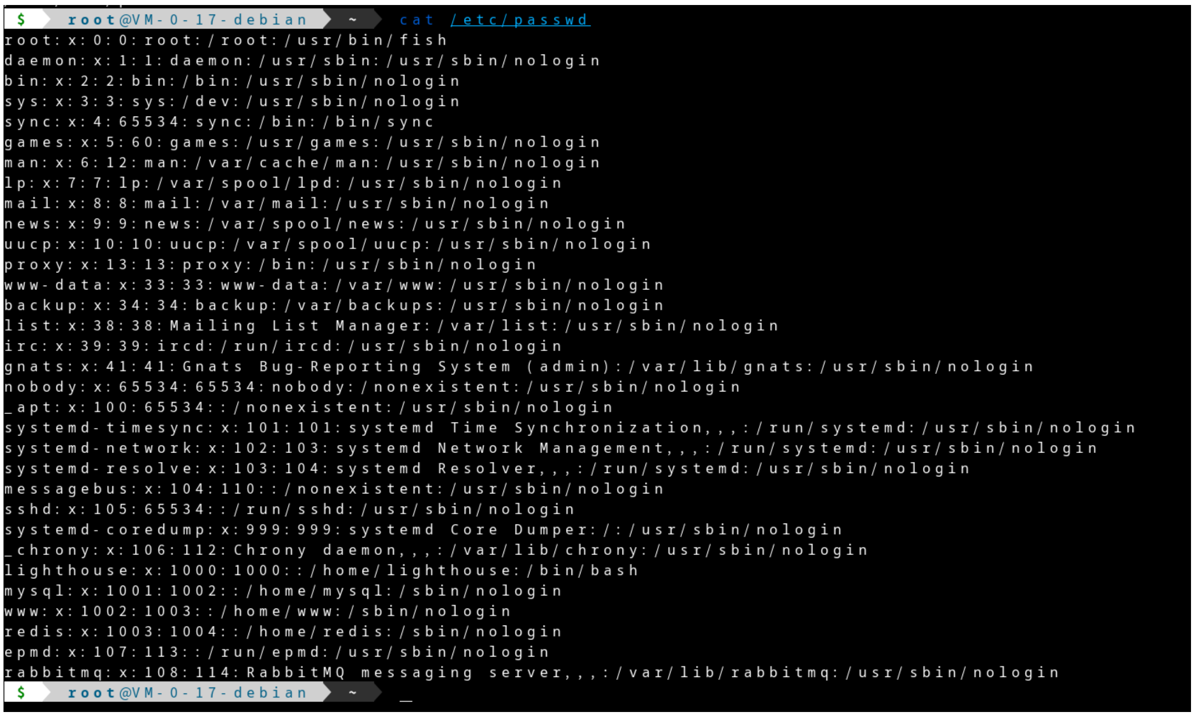
文件管理命令

搜索文件



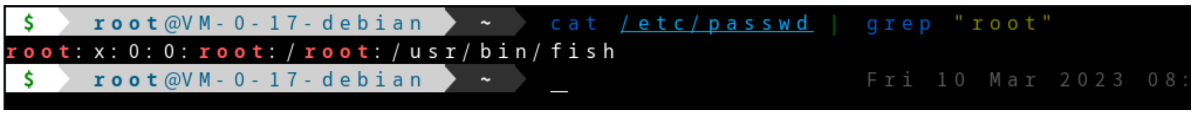
```
$  root@VM-0-17-debian  /home  cd ~                        Fri 10 Mar 2023 08:28:01 AM CS
$  root@VM-0-17-debian  ~  find / -name "?asswd" | more
find: /etc/passwd
/etc/pam.d/passwd
'/proc/791538/task/791538/net': Invalid argument
find: '/proc/791538/net': Invalid argument
/usr/share/lintian/overrides/passwd
/usr/share/bash-completion/completions/passwd
/usr/share/doc/passwd
/usr/bin/passwd
$  root@VM-0-17-debian  ~                           5.1s ❮ Fri 10 Mar 2023 08:29:28 AM CS
```

查看文件内容



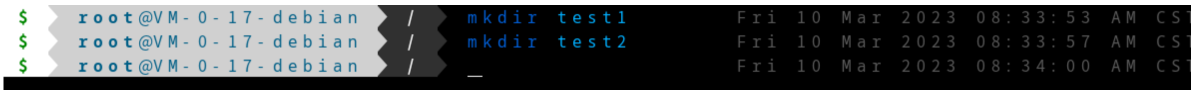
```
$  root@VM-0-17-debian  ~  cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/fish
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
_chrony:x:106:112:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
lighthouse:x:1000:1000::/home/lighthouse:/bin/bash
mysql:x:1001:1002::/home/mysql:/sbin/nologin
www:x:1002:1003::/home/www:/sbin/nologin
redis:x:1003:1004::/home/redis:/sbin/nologin
epmd:x:107:113::/run/epmd:/usr/sbin/nologin
rabbitmq:x:108:114:RabbitMQ messaging server,,,:/var/lib/rabbitmq:/usr/sbin/nologin
$  root@VM-0-17-debian  ~  _
```

通过管道过滤查找关键字



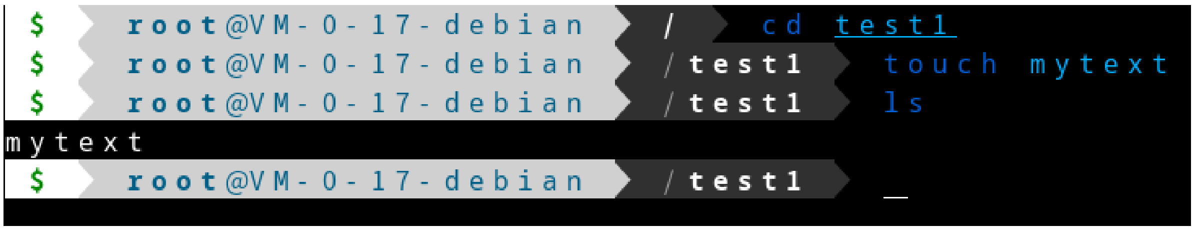
```
$  root@VM-0-17-debian  ~  cat /etc/passwd | grep "root"
root:x:0:0:root:/root:/usr/bin/fish
$  root@VM-0-17-debian  ~  _                    Fri 10 Mar 2023 08:
```

创建目录



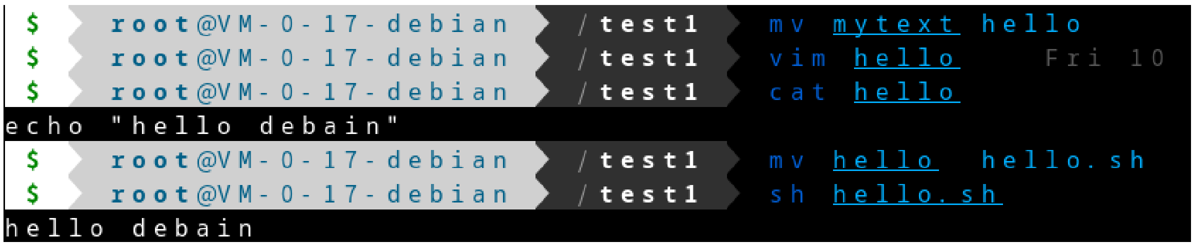
```
$  root@VM-0-17-debian  /  mkdir test1              Fri 10 Mar 2023 08:33:53 AM CS
$  root@VM-0-17-debian  /  mkdir test2              Fri 10 Mar 2023 08:33:57 AM CS
$  root@VM-0-17-debian  /  _                        Fri 10 Mar 2023 08:34:00 AM CS
```

创建文本文件



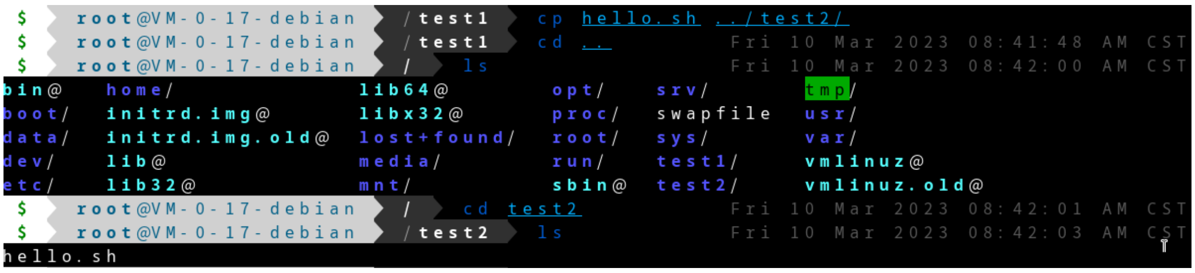
```
$  root@VM-0-17-debian  /  cd test1
$  root@VM-0-17-debian  /test1  touch mytext
$  root@VM-0-17-debian  /test1  ls
mytext
$  root@VM-0-17-debian  /test1  _
```

编辑文件



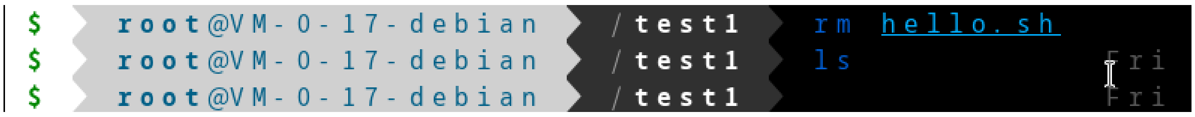
```
$  root@VM-0-17-debian  /test1  mv mytext hello
$  root@VM-0-17-debian  /test1  vim hello      Fri 10
$  root@VM-0-17-debian  /test1  cat hello
echo "hello debain"
$  root@VM-0-17-debian  /test1  mv hello hello.sh
$  root@VM-0-17-debian  /test1  sh hello.sh
hello debain
```

复制文件



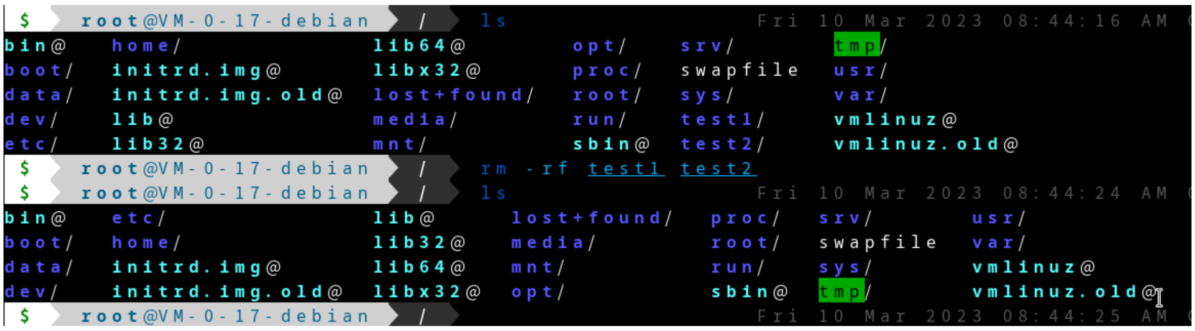
```
$   root@VM-0-17-debian   /test1   cp hello.sh ../test2/
$   root@VM-0-17-debian   /test1   cd ..                  Fri 10 Mar 2023 08:41:48 AM CST
$   root@VM-0-17-debian   /   ls                          Fri 10 Mar 2023 08:42:00 AM CST
bin@    home/            lib64@        opt/    srv/        tmp/
boot/   initrd.img@      libx32@       proc/   swapfile    usr/
data/   initrd.img.old@  lost+found/   root/   sys/        var/
dev/    lib@             media/        run/    test1/      vmlinuz@
etc/    lib32@           mnt/          sbin@   test2/      vmlinuz.old@
$   root@VM-0-17-debian   /   cd test2                    Fri 10 Mar 2023 08:42:01 AM CST
$   root@VM-0-17-debian   /test2   ls                     Fri 10 Mar 2023 08:42:03 AM CST
hello.sh
```

删除文件



```
$   root@VM-0-17-debian   /test1   rm hello.sh
$   root@VM-0-17-debian   /test1   ls                  Fri
$   root@VM-0-17-debian   /test1                       Fri
```

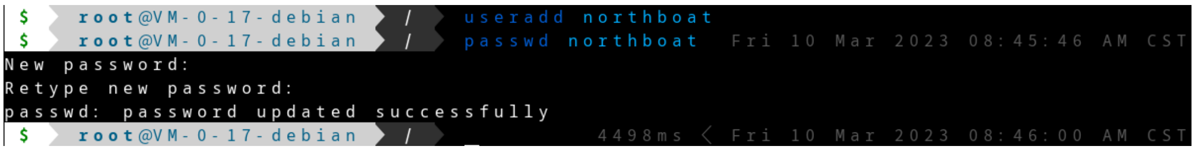
删除目录



```
$   root@VM-0-17-debian   /   ls                        Fri 10 Mar 2023 08:44:16 AM
bin@    home/            lib64@        opt/    srv/        tmp/
boot/   initrd.img@      libx32@       proc/   swapfile    usr/
data/   initrd.img.old@  lost+found/   root/   sys/        var/
dev/    lib@             media/        run/    test1/      vmlinuz@
etc/    lib32@           mnt/          sbin@   test2/      vmlinuz.old@
$   root@VM-0-17-debian   /   rm -rf test1 test2
$   root@VM-0-17-debian   /   ls                        Fri 10 Mar 2023 08:44:24 AM
bin@    etc/             lib@      lost+found/   proc/   srv/       usr/
boot/   home/            lib32@    media/        root/   swapfile   var/
data/   initrd.img@      lib64@    mnt/          run/    sys/       vmlinuz@
dev/    initrd.img.old@  libx32@   opt/          sbin@   tmp/       vmlinuz.old@
$   root@VM-0-17-debian   /                             Fri 10 Mar 2023 08:44:25 AM
```

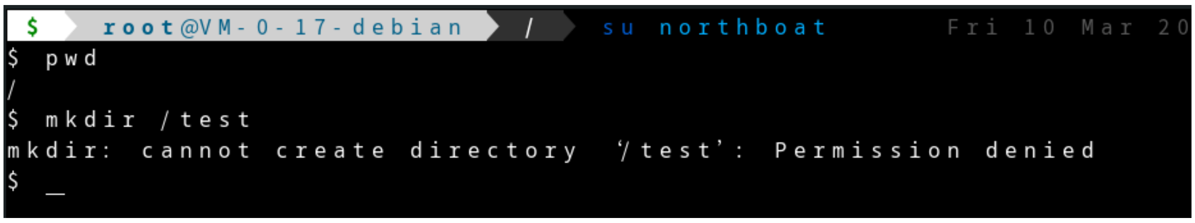
用户管理

新建用户

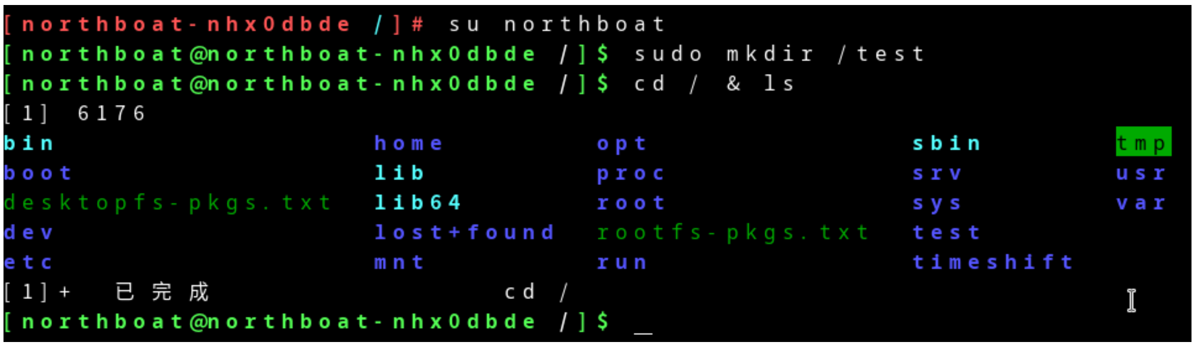


```
$   root@VM-0-17-debian   /   useradd northboat
$   root@VM-0-17-debian   /   passwd northboat   Fri 10 Mar 2023 08:45:46 AM CST
New password:
Retype new password:
passwd: password updated successfully
$   root@VM-0-17-debian   /   _          4498ms ‹ Fri 10 Mar 2023 08:46:00 AM CST
```

切换并测试用户

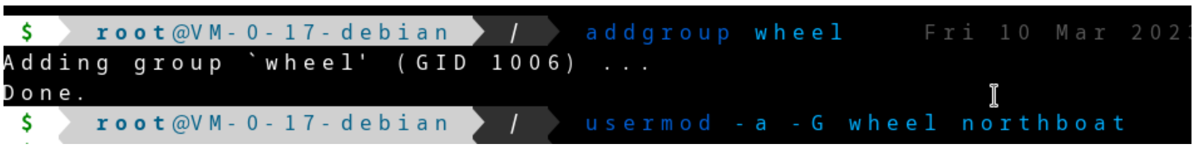


```
$   root@VM-0-17-debian   /   su northboat        Fri 10 Mar 20
$ pwd
/
$ mkdir /test
mkdir: cannot create directory '/test': Permission denied
$ _
```



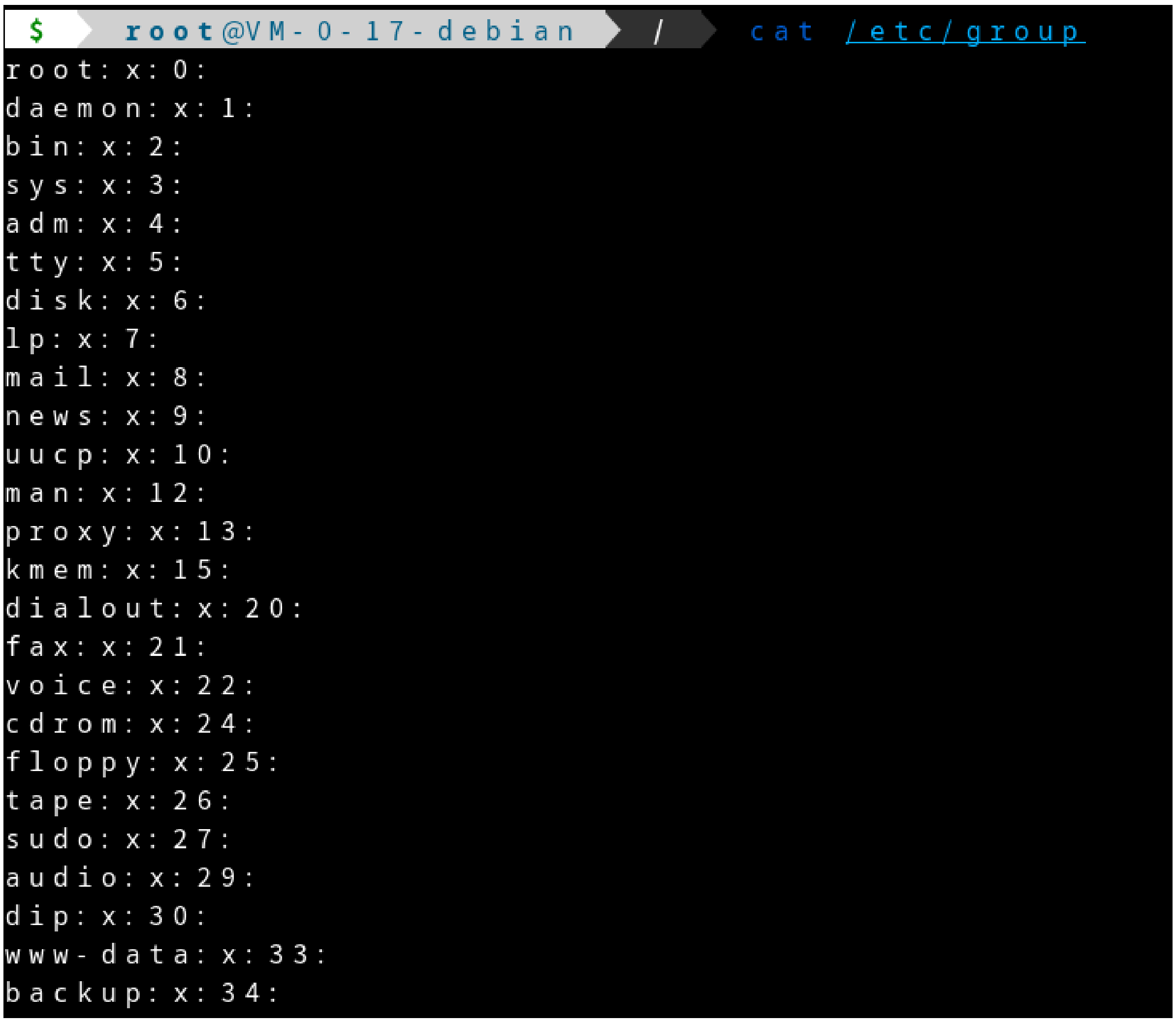
```
[northboat-nhx0dbde /]# su northboat
[northboat@northboat-nhx0dbde /]$ sudo mkdir /test
[northboat@northboat-nhx0dbde /]$ cd / & ls
[1] 6176
bin                    home         opt               sbin        tmp
boot                   lib          proc              srv         usr
desktopfs-pkgs.txt     lib64        root              sys         var
dev                    lost+found   rootfs-pkgs.txt   test
etc                    mnt          run               timeshift
[1]+  已完成               cd /
[northboat@northboat-nhx0dbde /]$ _
```

修改用户权限



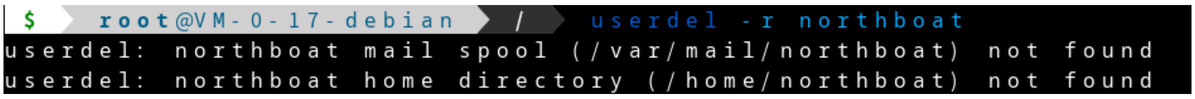
```
$   root@VM-0-17-debian   /   addgroup wheel       Fri 10 Mar 202
Adding group `wheel' (GID 1006) ...
Done.
$   root@VM-0-17-debian   /   usermod -a -G wheel northboat
```

查看用户组



```
$  root@VM-0-17-debian  /  cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:
floppy:x:25:
tape:x:26:
sudo:x:27:
audio:x:29:
dip:x:30:
www-data:x:33:
backup:x:34:
```

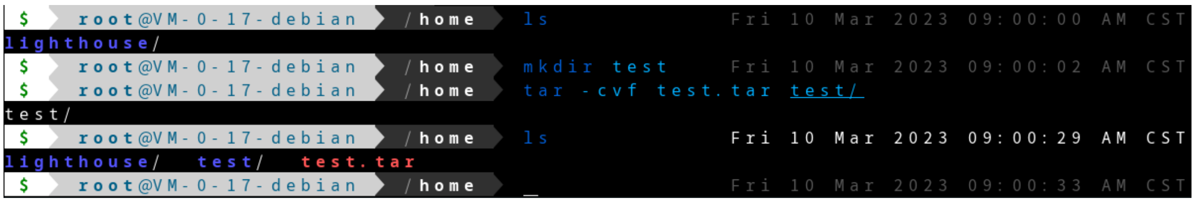
删除用户



```
$  root@VM-0-17-debian  /  userdel -r northboat
userdel: northboat mail spool (/var/mail/northboat) not found
userdel: northboat home directory (/home/northboat) not found
```

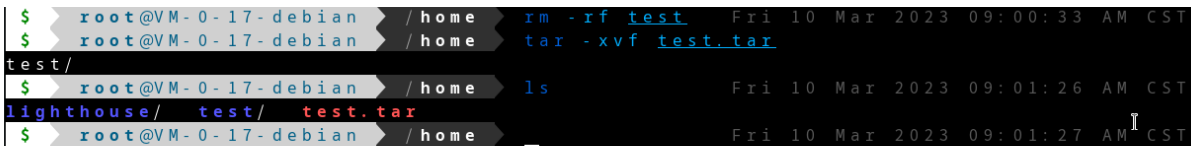
文件解压缩

压缩文件`.tar`



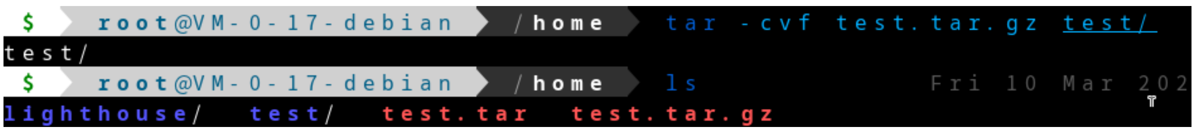
```
$  root@VM-0-17-debian  /home  ls                      Fri 10 Mar 2023 09:00:00 AM CST
lighthouse/
$  root@VM-0-17-debian  /home  mkdir test              Fri 10 Mar 2023 09:00:02 AM CST
$  root@VM-0-17-debian  /home  tar -cvf test.tar test/
test/
$  root@VM-0-17-debian  /home  ls                      Fri 10 Mar 2023 09:00:29 AM CST
lighthouse/  test/  test.tar
$  root@VM-0-17-debian  /home  _                       Fri 10 Mar 2023 09:00:33 AM CST
```

解压文件



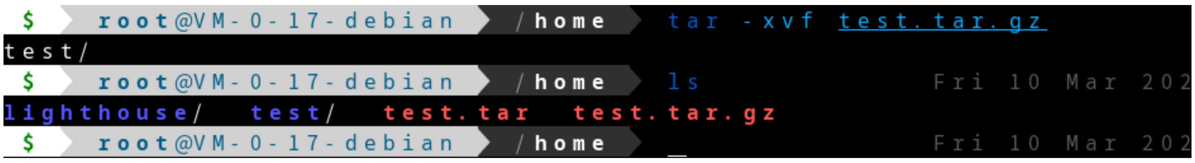
```
$  root@VM-0-17-debian  /home  rm -rf test             Fri 10 Mar 2023 09:00:33 AM CST
$  root@VM-0-17-debian  /home  tar -xvf test.tar
test/
$  root@VM-0-17-debian  /home  ls                      Fri 10 Mar 2023 09:01:26 AM CST
lighthouse/  test/  test.tar
$  root@VM-0-17-debian  /home  _                       Fri 10 Mar 2023 09:01:27 AM CST
```

压缩文件`.tar.gz`



```
$  root@VM-0-17-debian  /home  tar -cvf test.tar.gz test/
test/
$  root@VM-0-17-debian  /home  ls                  Fri 10 Mar 202
lighthouse/  test/  test.tar  test.tar.gz
```

解压文件


```
$  root@VM-0-17-debian  /home  tar -xvf test.tar.gz
test/
$  root@VM-0-17-debian  /home  ls                              Fri 10 Mar 202
lighthouse/  test/  test.tar  test.tar.gz
$  root@VM-0-17-debian  /home  _                               Fri 10 Mar 202
```

实验总结

debain 默认没有 wheel 组，在加入用户进 wheel 组时会报错：group wheel does not exist

需要新增组

```
groupadd wheel
```

再将用户加入组

```
usermod -a -G wheel northboat
```

删除组

```
groupdel wheel
```

通过查看组 `cat /etc/group` 发现存在 `root` 组，将用户加入 `root` 组

```
usermod -a -G root northboat
```

Linux 系统管理

实现环境

Linux 版本：Manjaro

```
Linux northboat-nhx0dbde 6.1.12-1-MANJARO #1 SMP PREEMPT_DYNAMIC Tue Feb 14
21:59:10 UTC 2023 x86_64 GNU/Linux
```

本地 Shell

实验内容

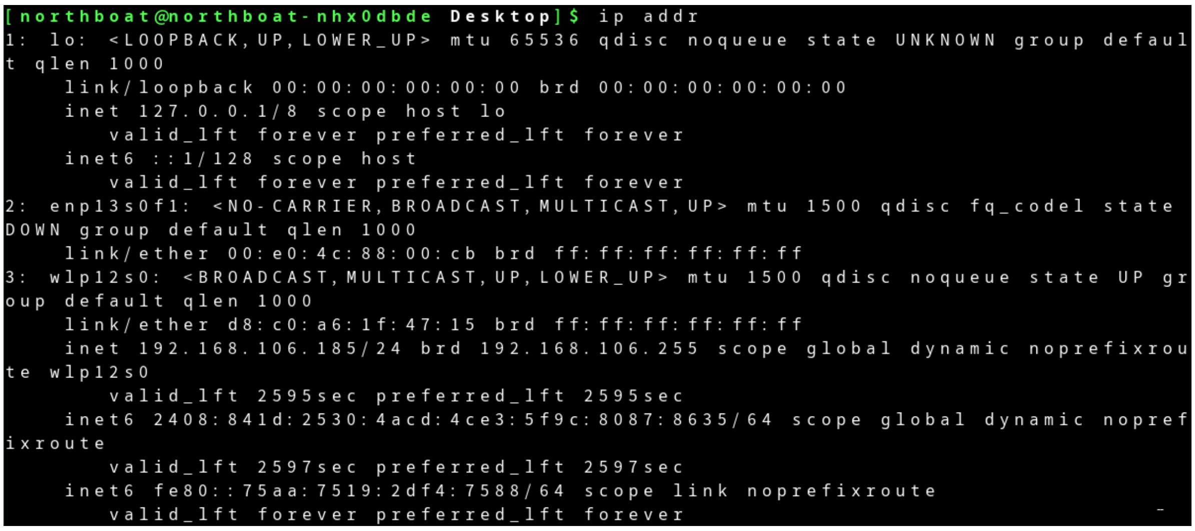
网络管理

> 设置静态 IP，manjaro 下，使用 netctl 实现

下载 netctl

```
yay -S netctl
```

查看网卡信息


```
[northboat@northboat-nhx0dbde Desktop]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp13s0f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state
DOWN group default qlen 1000
    link/ether 00:e0:4c:88:00:cb brd ff:ff:ff:ff:ff:ff
3: wlp12s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gr
oup default qlen 1000
    link/ether d8:c0:a6:1f:47:15 brd ff:ff:ff:ff:ff:ff
    inet 192.168.106.185/24 brd 192.168.106.255 scope global dynamic noprefixrou
te wlp12s0
       valid_lft 2595sec preferred_lft 2595sec
    inet6 2408:841d:2530:4acd:4ce3:5f9c:8087:8635/64 scope global dynamic nopref
ixroute
       valid_lft 2597sec preferred_lft 2597sec
    inet6 fe80::75aa:7519:2df4:7588/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

得知网卡名称`wlp12s0`

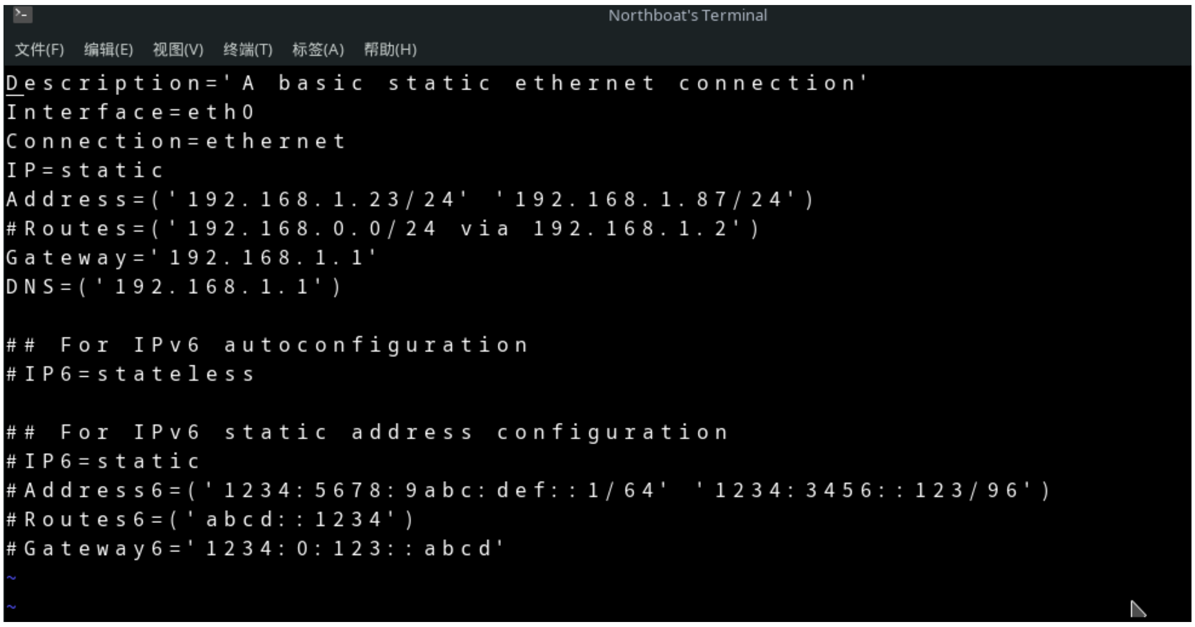
终止网络服务

```
sudo systemctl stop NetworkManager
sudo systemctl disable NetworkManager
```

复制`netctl`默认配置文件

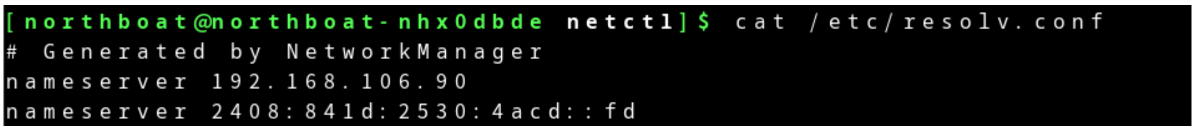
```
sudo cp /etc/netctl/examples/ethernet-static /etc/netctl/enp13s0f1
```

编辑文件`wlp12s0`



```
Description='A basic static ethernet connection'
Interface=eth0
Connection=ethernet
IP=static
Address=('192.168.1.23/24' '192.168.1.87/24')
#Routes=('192.168.0.0/24 via 192.168.1.2')
Gateway='192.168.1.1'
DNS=('192.168.1.1')

## For IPv6 autoconfiguration
#IP6=stateless

## For IPv6 static address configuration
#IP6=static
#Address6=('1234:5678:9abc:def::1/64' '1234:3456::123/96')
#Routes6=('abcd::1234')
#Gateway6='1234:0:123::abcd'
~
~
```

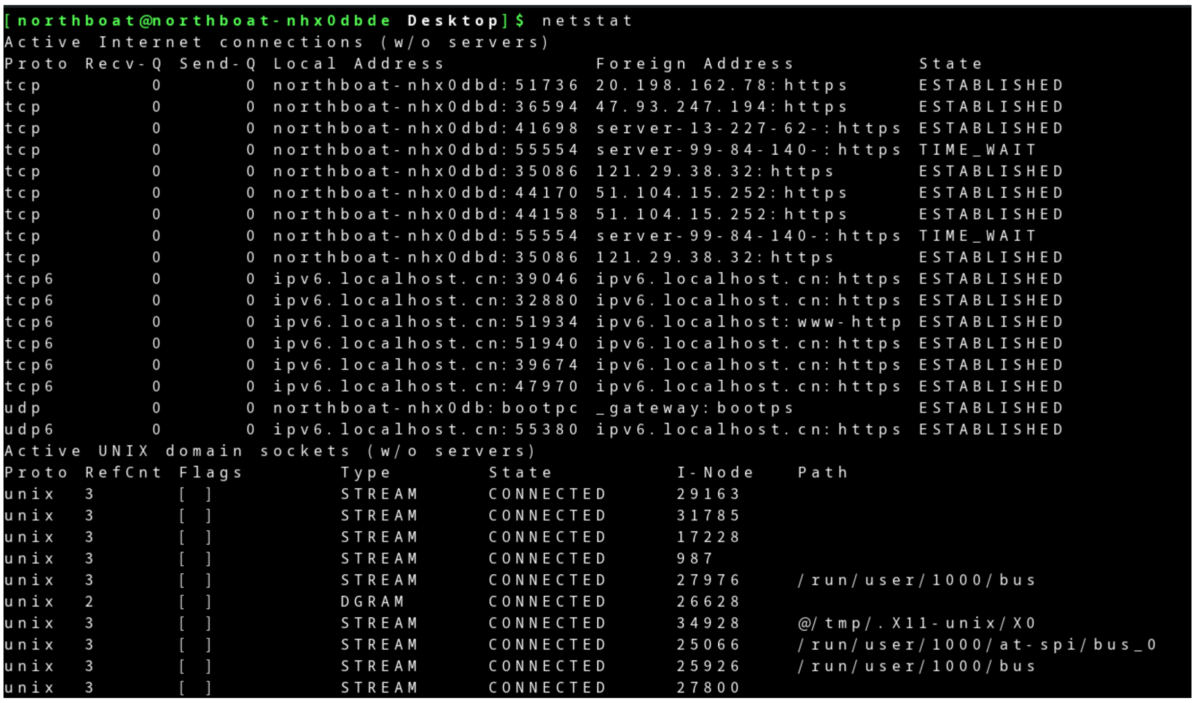
配置 DNS 解析


```
[northboat@northboat-nhx0dbde netctl]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.106.90
nameserver 2408:841d:2530:4acd::fd
```

重启网络服务

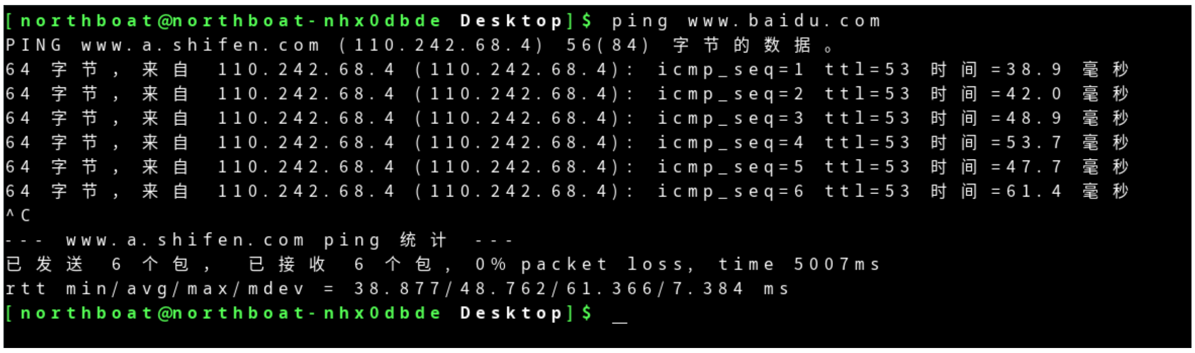
```
sudo systemctl start NetworkManager
sudo systemctl enable NetworkManager
```

查看网络连接状态



```
[northboat@northboat-nhx0dbde Desktop]$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 northboat-nhx0dbd:51736 20.198.162.78:https     ESTABLISHED
tcp        0      0 northboat-nhx0dbd:36594 47.93.247.194:https     ESTABLISHED
tcp        0      0 northboat-nhx0dbd:41698 server-13-227-62-:https ESTABLISHED
tcp        0      0 northboat-nhx0dbd:55554 server-99-84-140-:https TIME_WAIT
tcp        0      0 northboat-nhx0dbd:35086 121.29.38.32:https      ESTABLISHED
tcp        0      0 northboat-nhx0dbd:44170 51.104.15.252:https     ESTABLISHED
tcp        0      0 northboat-nhx0dbd:44158 51.104.15.252:https     ESTABLISHED
tcp        0      0 northboat-nhx0dbd:55554 server-99-84-140-:https TIME_WAIT
tcp        0      0 northboat-nhx0dbd:35086 121.29.38.32:https      ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:39046 ipv6.localhost.cn:https ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:32880 ipv6.localhost.cn:https ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:51934 ipv6.localhost:www-http ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:51940 ipv6.localhost.cn:https ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:39674 ipv6.localhost.cn:https ESTABLISHED
tcp6       0      0 ipv6.localhost.cn:47970 ipv6.localhost.cn:https ESTABLISHED
udp        0      0 northboat-nhx0db:bootpc _gateway:bootps         ESTABLISHED
udp6       0      0 ipv6.localhost.cn:55380 ipv6.localhost.cn:https ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     29163
unix  3      [ ]         STREAM     CONNECTED     31785
unix  3      [ ]         STREAM     CONNECTED     17228
unix  3      [ ]         STREAM     CONNECTED     987
unix  3      [ ]         STREAM     CONNECTED     27976    /run/user/1000/bus
unix  2      [ ]         DGRAM      CONNECTED     26628
unix  3      [ ]         STREAM     CONNECTED     34928    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     25066    /run/user/1000/at-spi/bus_0
unix  3      [ ]         STREAM     CONNECTED     25926    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     27800
```

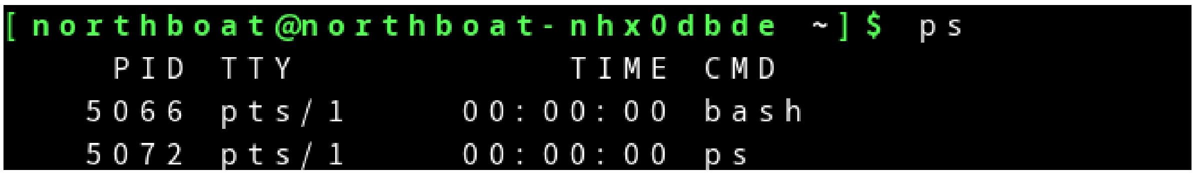
ping 通



```
[northboat@northboat-nhx0dbde Desktop]$ ping www.baidu.com
PING www.a.shifen.com (110.242.68.4) 56(84) 字节的数据。
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=1 ttl=53 时间=38.9 毫秒
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=2 ttl=53 时间=42.0 毫秒
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=3 ttl=53 时间=48.9 毫秒
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=4 ttl=53 时间=53.7 毫秒
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=5 ttl=53 时间=47.7 毫秒
64 字节，来自 110.242.68.4 (110.242.68.4): icmp_seq=6 ttl=53 时间=61.4 毫秒
^C
--- www.a.shifen.com ping 统计 ---
已发送 6 个包， 已接收 6 个包, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 38.877/48.762/61.366/7.384 ms
[northboat@northboat-nhx0dbde Desktop]$ _
```

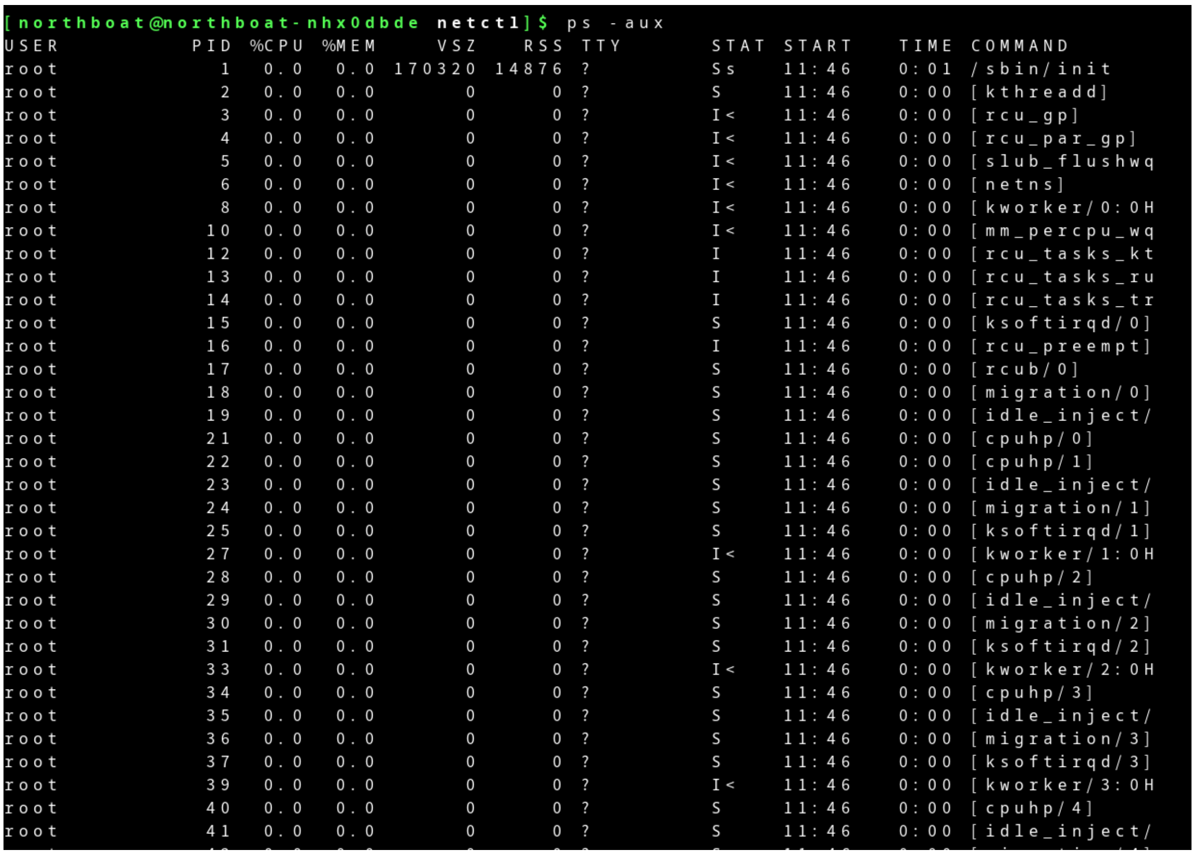
进程管理

`ps`命令查看进程



```
[northboat@northboat-nhx0dbde ~]$ ps
    PID TTY          TIME CMD
   5066 pts/1    00:00:00 bash
   5072 pts/1    00:00:00 ps
```

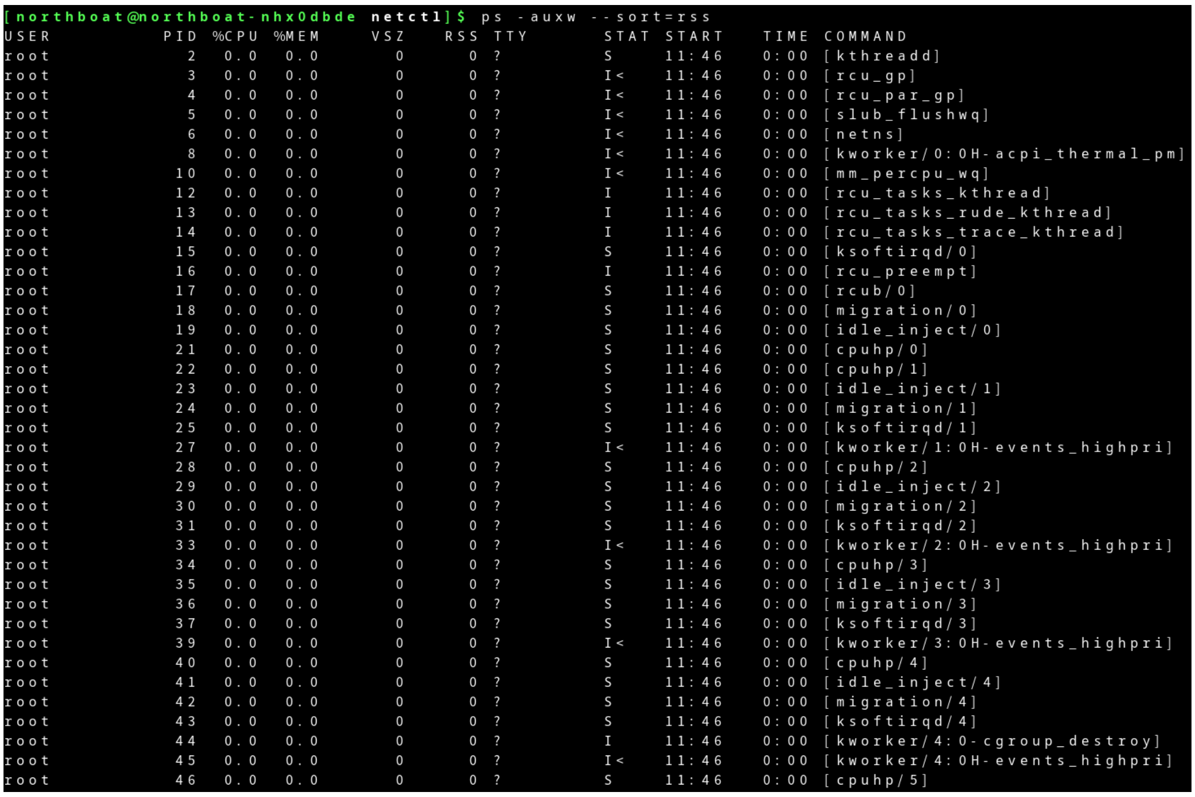
查看所有用户所有进程信息



```
[northboat@northboat-nhx0dbde netctl]$ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 170320 14876 ?        Ss   11:46   0:01 /sbin/init
root           2  0.0  0.0      0     0 ?        S    11:46   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   11:46   0:00 [slub_flushwq
root           6  0.0  0.0      0     0 ?        I<   11:46   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/0:0H
root          10  0.0  0.0      0     0 ?        I<   11:46   0:00 [mm_percpu_wq
root          12  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_kt
root          13  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_ru
root          14  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_tr
root          15  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_preempt]
root          17  0.0  0.0      0     0 ?        S    11:46   0:00 [rcub/0]
root          18  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/
root          21  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/0]
root          22  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/1]
root          23  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/
root          24  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/1]
root          25  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/1]
root          27  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/1:0H
root          28  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/2]
root          29  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/
root          30  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/2]
root          31  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/2]
root          33  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/2:0H
root          34  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/3]
root          35  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/
root          36  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/3]
root          37  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/3]
root          39  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/3:0H
root          40  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/4]
root          41  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/
```

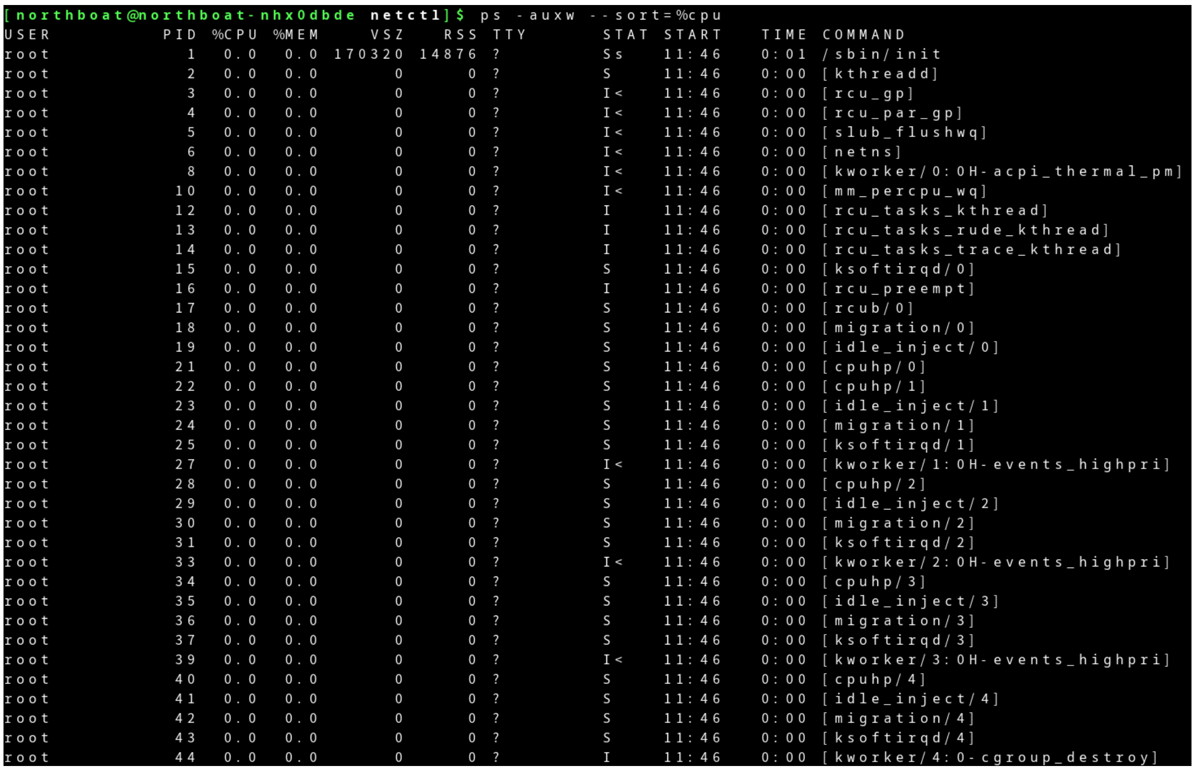
进程信息排序

- 按内存占用



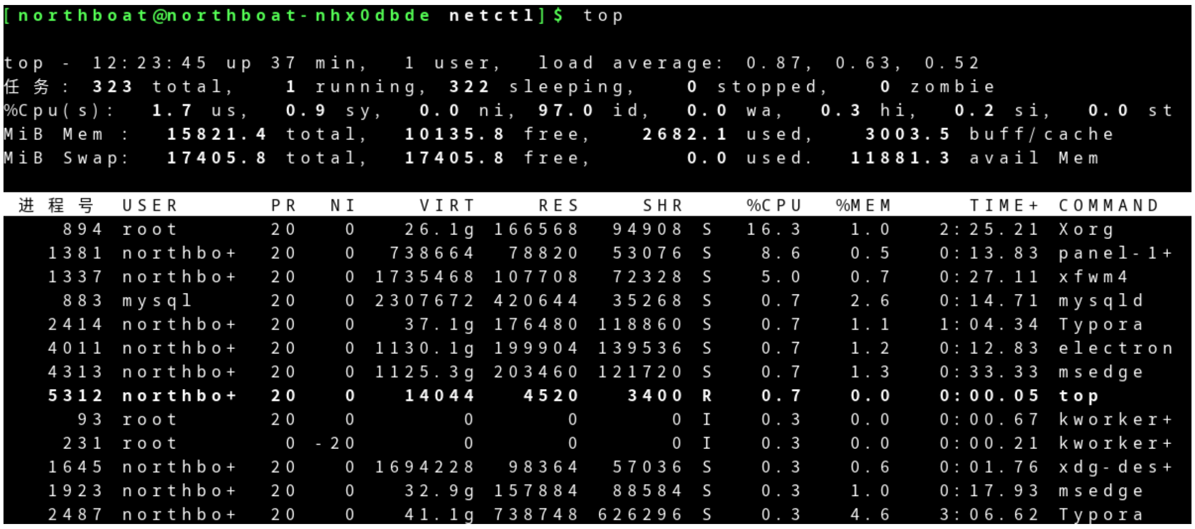
```
[northboat@northboat-nhx0dbde netctl]$ ps -auxw --sort=rss
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           2  0.0  0.0      0     0 ?        S    11:46   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   11:46   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   11:46   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/0:0H-acpi_thermal_pm]
root          10  0.0  0.0      0     0 ?        I<   11:46   0:00 [mm_percpu_wq]
root          12  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_kthread]
root          13  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_rude_kthread]
root          14  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_trace_kthread]
root          15  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_preempt]
root          17  0.0  0.0      0     0 ?        S    11:46   0:00 [rcub/0]
root          18  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/0]
root          21  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/0]
root          22  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/1]
root          23  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/1]
root          24  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/1]
root          25  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/1]
root          27  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/1:0H-events_highpri]
root          28  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/2]
root          29  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/2]
root          30  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/2]
root          31  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/2]
root          33  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/2:0H-events_highpri]
root          34  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/3]
root          35  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/3]
root          36  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/3]
root          37  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/3]
root          39  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/3:0H-events_highpri]
root          40  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/4]
root          41  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/4]
root          42  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/4]
root          43  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/4]
root          44  0.0  0.0      0     0 ?        I    11:46   0:00 [kworker/4:0-cgroup_destroy]
root          45  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/4:0H-events_highpri]
root          46  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/5]
```

- 按 CPU 占用



```
[northboat@northboat-nhx0dbde netctl]$ ps -auxw --sort=%cpu
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 170320 14876 ?        Ss   11:46   0:01 /sbin/init
root           2  0.0  0.0      0     0 ?        S    11:46   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   11:46   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   11:46   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   11:46   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/0:0H-acpi_thermal_pm]
root          10  0.0  0.0      0     0 ?        I<   11:46   0:00 [mm_percpu_wq]
root          12  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_kthread]
root          13  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_rude_kthread]
root          14  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_tasks_trace_kthread]
root          15  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        I    11:46   0:00 [rcu_preempt]
root          17  0.0  0.0      0     0 ?        S    11:46   0:00 [rcub/0]
root          18  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/0]
root          21  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/0]
root          22  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/1]
root          23  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/1]
root          24  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/1]
root          25  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/1]
root          27  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/1:0H-events_highpri]
root          28  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/2]
root          29  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/2]
root          30  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/2]
root          31  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/2]
root          33  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/2:0H-events_highpri]
root          34  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/3]
root          35  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/3]
root          36  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/3]
root          37  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/3]
root          39  0.0  0.0      0     0 ?        I<   11:46   0:00 [kworker/3:0H-events_highpri]
root          40  0.0  0.0      0     0 ?        S    11:46   0:00 [cpuhp/4]
root          41  0.0  0.0      0     0 ?        S    11:46   0:00 [idle_inject/4]
root          42  0.0  0.0      0     0 ?        S    11:46   0:00 [migration/4]
root          43  0.0  0.0      0     0 ?        S    11:46   0:00 [ksoftirqd/4]
root          44  0.0  0.0      0     0 ?        I    11:46   0:00 [kworker/4:0-cgroup_destroy]
```

动态查看进程信息



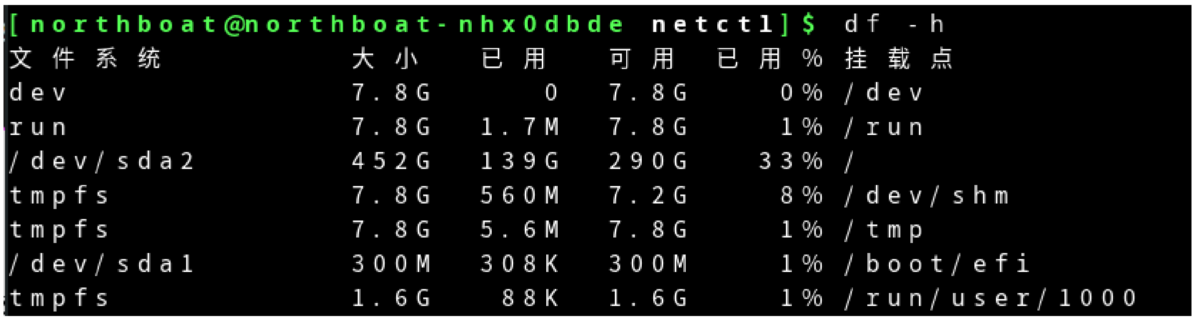
```
[northboat@northboat-nhx0dbde netctl]$ top

top - 12:23:45 up 37 min,  1 user,  load average: 0.87, 0.63, 0.52
任务: 323 total,   1 running, 322 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.7 us,  0.9 sy,  0.0 ni, 97.0 id,  0.0 wa,  0.3 hi,  0.2 si,  0.0 st
MiB Mem :  15821.4 total,  10135.8 free,   2682.1 used,   3003.5 buff/cache
MiB Swap:  17405.8 total,  17405.8 free,      0.0 used.  11881.3 avail Mem

 进程号 USER      PR  NI    VIRT    RES    SHR    %CPU  %MEM     TIME+ COMMAND
    894 root      20   0   26.1g 166568  94908 S  16.3   1.0   2:25.21 Xorg
   1381 northbo+  20   0  738664  78820  53076 S   8.6   0.5   0:13.83 panel-1+
   1337 northbo+  20   0 1735468 107708  72328 S   5.0   0.7   0:27.11 xfwm4
    883 mysql     20   0 2307672 420644  35268 S   0.7   2.6   0:14.71 mysqld
   2414 northbo+  20   0   37.1g 176480 118860 S   0.7   1.1   1:04.34 Typora
   4011 northbo+  20   0 1130.1g 199904 139536 S   0.7   1.2   0:12.83 electron
   4313 northbo+  20   0 1125.3g 203460 121720 S   0.7   1.3   0:33.33 msedge
   5312 northbo+  20   0   14044   4520   3400 R   0.7   0.0   0:00.05 top
     93 root      20   0       0      0      0 I   0.3   0.0   0:00.67 kworker+
    231 root       0 -20       0      0      0 I   0.3   0.0   0:00.21 kworker+
   1645 northbo+  20   0 1694228  98364  57036 S   0.3   0.6   0:01.76 xdg-des+
   1923 northbo+  20   0   32.9g 157884  88584 S   0.3   1.0   0:17.93 msedge
   2487 northbo+  20   0   41.1g 738748 626296 S   0.3   4.6   3:06.62 Typora
```

终止进程

```
# 根据 pid 杀死进程
kill -9 pid

# 根据进程名查找 pid
pgrep -f name

# 根据进程名杀死进程
pkill -f name
```

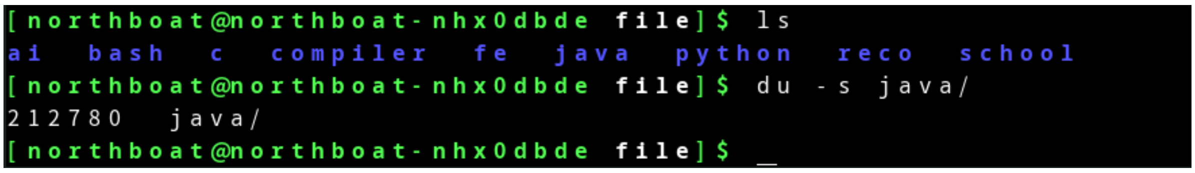
磁盘管理

查看已挂载磁盘总容量、已使用、剩余容量

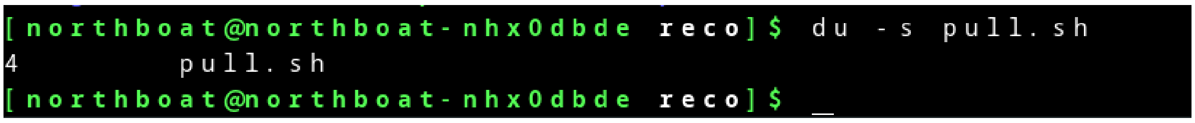


```
[northboat@northboat-nhx0dbde netctl]$ df -h
文件系统          大小   已用   可用  已用% 挂载点
dev               7.8G     0   7.8G    0% /dev
run               7.8G  1.7M   7.8G    1% /run
/dev/sda2         452G  139G   290G   33% /
tmpfs             7.8G  560M   7.2G    8% /dev/shm
tmpfs             7.8G  5.6M   7.8G    1% /tmp
/dev/sda1         300M  308K   300M    1% /boot/efi
tmpfs             1.6G   88K   1.6G    1% /run/user/1000
```

查看目录或文件所占空间



```
[northboat@northboat-nhx0dbde file]$ ls
ai  bash  c  compiler  fe  java  python  reco  school
[northboat@northboat-nhx0dbde file]$ du -s java/
212780  java/
[northboat@northboat-nhx0dbde file]$ _
```



```
[northboat@northboat-nhx0dbde reco]$ du -s pull.sh
4       pull.sh
[northboat@northboat-nhx0dbde reco]$ _
```

实验总结

对于个人用户，修改静态 IP 便于在局域网内访问机器，之前使用系统提供的配置文件对静态 IP 进行过修改，但每次重启或重新联网后都会重置该 IP，后采用`netctl`对静态 IP 进行统一管理，解决问题

Linux 服务器配置

实现环境

centos7

```
Linux VM-0-17-centos 3.10.0-1160.88.1.el7.x86_64 #1 SMP Tue Mar 7 15:41:52 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

ssh

实验内容

下载 Nginx 服务器

通过 wget 在 nginx 官网下载

```
wget http://nginx.org/download/nginx-1.17.6.tar.gz
```

安装必要依赖

```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
```

创建目录

```
mkdir /usr/local/nginx
```

解压 nginx 压缩包

```
tar -zxvf nginx-1.17.6.tar.gz -C /usr/local/nginx
```

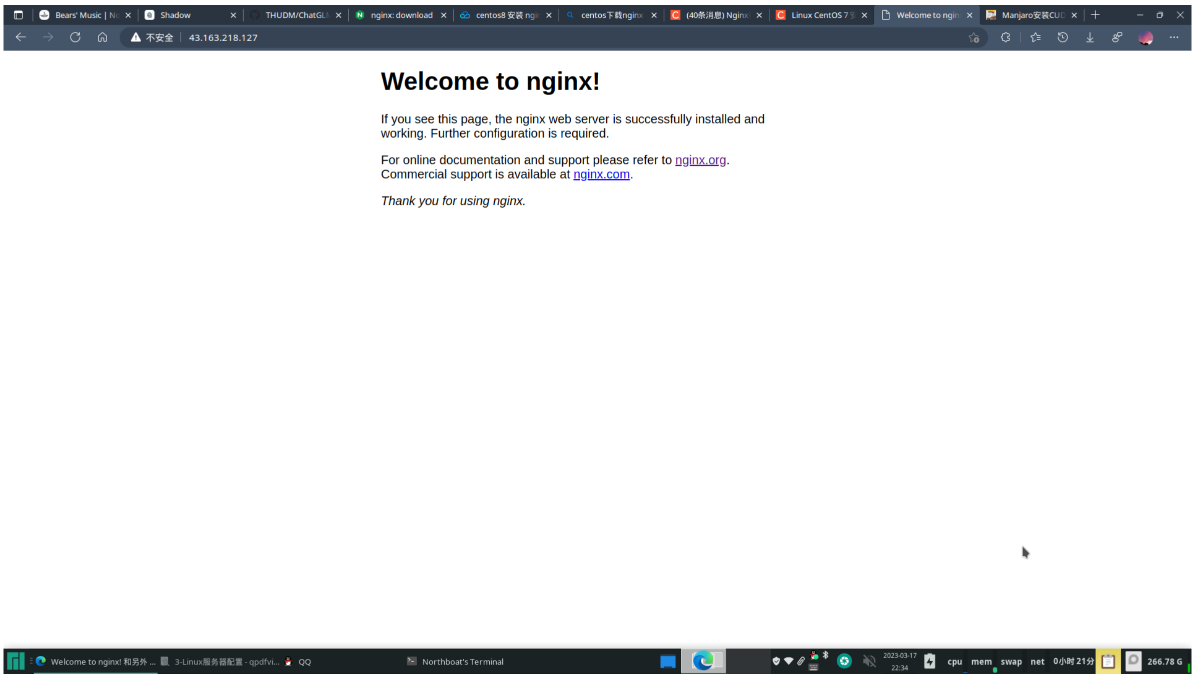
编译 nginx

```
cd /usr/local/nginx/nginx-1.17.6
./configure
make
make install
```

启动 nginx

```
cd /usr/local/nginx
./nginx
```

查看启动情况，浏览器进入`http://43.163.218.127/`



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

下载 MariaDB

通过 yum 安装

```
yum install mariadb-server
```

启动 mariadb

```
systemctl start mariadb  # 开启服务
systemctl enable mariadb  # 设置为开机自启动服务
```

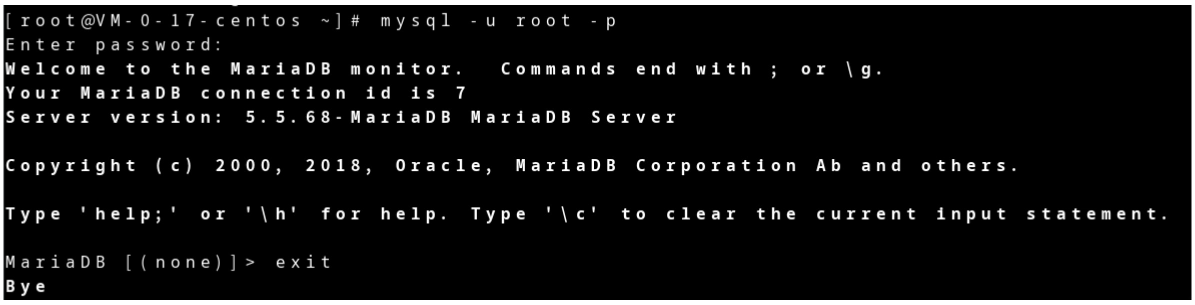
数据库配置

```
mysql_secure_installation
```

```
Enter current password for root (enter for none):  # 输入数据库超级管理员root的密码
(注意不是系统root的密码)，第一次进入还没有设置密码则直接回车
```

```
Set root password? [Y/n]  # 设置密码，y

New password:  # 新密码
Re-enter new password:  # 再次输入密码

Remove anonymous users? [Y/n]  # 移除匿名用户， y

Disallow root login remotely? [Y/n]  # 拒绝root远程登录，n，不管y/n，都会拒绝root远程登录

Remove test database and access to it? [Y/n]  # 删除test数据库，y：删除。n：不删除，数据库中会有一个test数据库，一般不需要

Reload privilege tables now? [Y/n]  # 重新加载权限表，y。或者重启服务也许
```

登录



```
[root@VM-0-17-centos ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 7
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> exit
Bye
```

下载 Redis

wget 下载

```
wget https://github.com/redis/redis/archive/redis-7.0.9.tar.gz
```

解压

```
tar -zvxf redis-7.0.9.tar.gz -C /usr/local/redis
```

编译

```
cd /usr/local/redis/redis/redis-7.0.9
make
```

安装

```
make PREFIX=/usr/local/redis install
```

复制默认配置文件

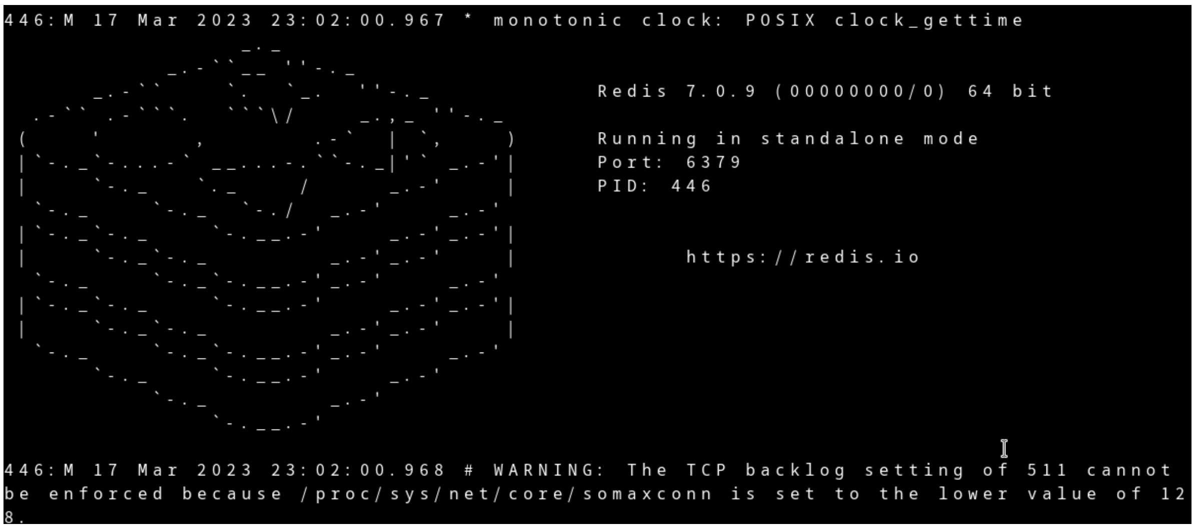
```
cp redis.conf ../bin
```

设置 redis.conf

```
requirepass 123456 # 设置密码
daemonize yes # 允许后台运行
bind 0.0.0.0 # 允许远程访问
```

启动

```
cd /usr/local/redis/bin
./redis-server redis.conf
```



```
446:M 17 Mar 2023 23:02:00.967 * monotonic clock: POSIX clock_gettime
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 7.0.9 (00000000/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 446
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           https://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

446:M 17 Mar 2023 23:02:00.968 # WARNING: The TCP backlog setting of 511 cannot
be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 12
8.
```

安装 OpenJDK17

wget 下载最新的 jdk17

```
wget https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.tar.gz
```

解压

```
tar xf jdk-17_linux-x64_bin.tar.gz
```

移动位置

```
mv jdk-17.0.6/ /usr/lib/jvm/jdk-17.0.6
```

修改环境配置

```
vim /etc/profile
```

添加以下内容

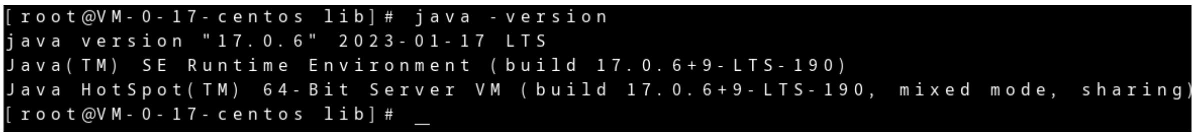
```
export JAVA_HOME=/usr/lib/jvm/jdk-17.0.6
export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

重新加载配置

```
source /etc/profile
```

测试安装

```
java -version
```



```
[root@VM-0-17-centos lib]# java -version
java version "17.0.6" 2023-01-17 LTS
Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)
[root@VM-0-17-centos lib]# _
```

安装 RabbitMQ

安装 Erlang 环境，yum 下载

安装依赖

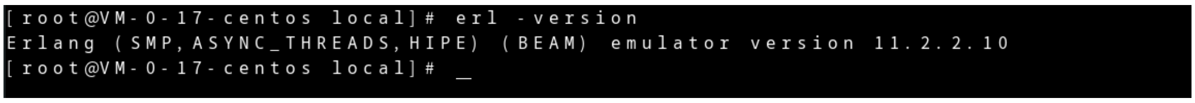
```
curl -s
https://packagecloud.io/install/repositories/rabbitmq/erlang/script.rpm.sh | sudo
 bash
```

下载 erlang

```
yum install -y erlang
```

测试安装

```
erl -version
```



```
[root@VM-0-17-centos local]# erl -version
Erlang (SMP,ASYNC_THREADS,HIPE) (BEAM) emulator version 11.2.2.10
[root@VM-0-17-centos local]# _
```

安装 RabbitMQ

导入 key

```
rpm --import https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey
rpm --import https://packagecloud.io/gpg.key
```

安装依赖

```
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-
server/script.rpm.sh | sudo bash
```

wget 下载 rabbitmq

```
wget https://github.com/rabbitmq/rabbitmq-
server/releases/download/v3.8.5/rabbitmq-server-3.8.5-1.el7.noarch.rpm
```

直接安装将报错

```
rpm -ivh rabbitmq-server-3.8.5-1.el7.noarch.rpm

warning: rabbitmq-server-3.8.5-1.el7.noarch.rpm: Header V4 RSA/SHA256 Signature,
key ID 6026dfca: NOKEY
error: Failed dependencies:
    socat is needed by rabbitmq-server-3.8.5-1.el7.noarch
```

导入 key

```
rpm --import https://www.rabbitmq.com/rabbitmq-release-signing-key.asc
```

安装 socat

```
yum -y install epel-release
yum -y install socat
```

重新安装

```
rpm -ivh rabbitmq-server-3.8.5-1.el7.noarch.rpm
```

启用 rabbitmq 插件

```
rabbitmq-plugins enable rabbitmq_management
```

启动 rabbitmq

```
systemctl start rabbitmq-server
```

创建用户

```
rabbitmqctl add_user admin 011026
```

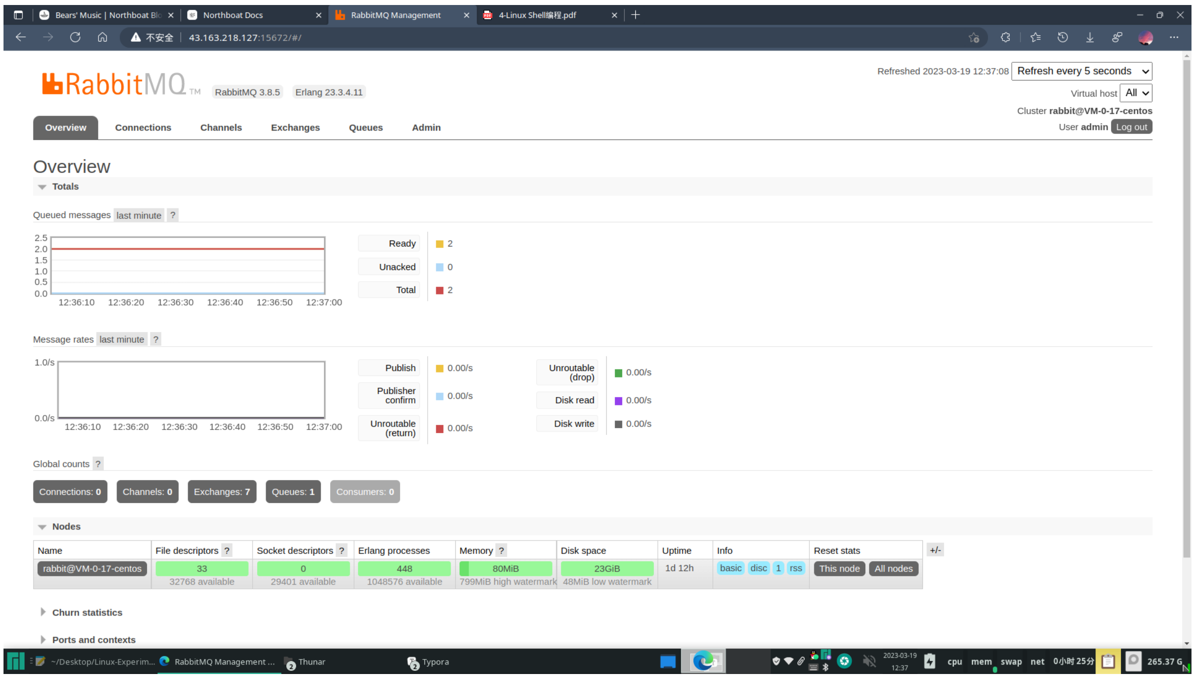
设置超级管理员权限

```
rabbitmqctl set_user_tags admin administrator
```

重启 rabbitmq

```
systemctl restart rabbitmq-server
```

查看可视化界面： `43.163.218.127:15672`



43.163.218.127:15672/#/

RabbitMQ 3.8.5 Erlang 23.3.4.11

Refreshed 2023-03-19 12:37:08 Refresh every 5 seconds

Virtual host All

Cluster rabbit@VM-0-17-centos

User admin Log out

Overview

Totals

Queued messages last minute

Ready	🟨 2
Unacked	🟦 0
Total	🟥 2

Message rates last minute

Publish	🟨 0.00/s	Unroutable (drop)	🟩 0.00/s
Publisher confirm	🟦 0.00/s	Disk read	🟪 0.00/s
Unroutable (return)	🟥 0.00/s	Disk write	⬛ 0.00/s

Global counts

Connections: 0 Channels: 0 Exchanges: 7 Queues: 1 Consumers: 0

Nodes

Name	File descriptors	Socket descriptors	Erlang processes	Memory	Disk space	Uptime	Info	Reset stats
rabbit@VM-0-17-centos	33 32768 available	0 29401 available	448 1048576 available	80MiB 799MiB high watermark	23GiB 48MiB low watermark	1d 12h	basic disc 1 rss	This node All nodes

Churn statistics

Ports and contexts

设置`virtual host`为`/`，默认为`ALL`

服务器使用

使用 ftp 工具上传文件

- 一个前端网页
- 一个 jar 包

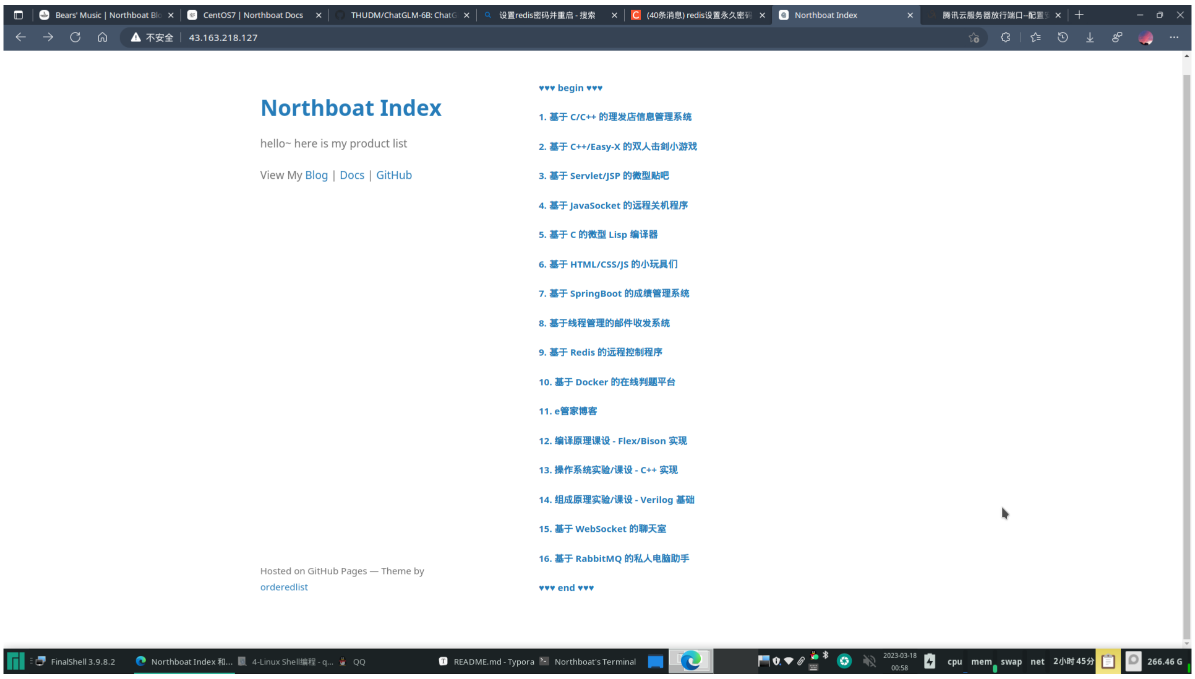
将 nginx 目录下 html 文件夹内容替换为上传的`index.html`，并将资源放在相应目录下

配置 nginx.conf 文件，设置端口及负载均衡

启动 jar 包

```
nohup java -jar Shadow-0.0.1-SNAPSHOT.jar &
```

访问`43.163.218.127:80`



Northboat Index

hello~ here is my product list

View My Blog | Docs | GitHub

Hosted on GitHub Pages — Theme by orderedlist

♥♥♥ begin ♥♥♥

1. 基于 C/C++ 的理发店信息管理系统
2. 基于 C++/Easy-X 的双人击剑小游戏
3. 基于 Servlet/JSP 的微型贴吧
4. 基于 JavaSocket 的远程关机程序
5. 基于 C 的微型 Lisp 编译器
6. 基于 HTML/CSS/JS 的小玩具们
7. 基于 SpringBoot 的成绩管理系统
8. 基于线程管理的邮件收发系统
9. 基于 Redis 的远程控制程序
10. 基于 Docker 的在线判题平台
11. e管家博客
12. 编译原理课设 - Flex/Bison 实现
13. 操作系统实验/课设 - C++ 实现
14. 组成原理实验/课设 - Verilog 基础
15. 基于 WebSocket 的聊天室
16. 基于 RabbitMQ 的私人电脑助手

♥♥♥ end ♥♥♥

实验总结

通过部署安装`mysql、redis、rabbitmq、nginx`实现服务器环境搭建，成功跑通两个 Java 网页服务，使我对服务器的部署流程更加熟练

Linux Shell 编程

实现环境

manjaro 本地 shell，内核版本

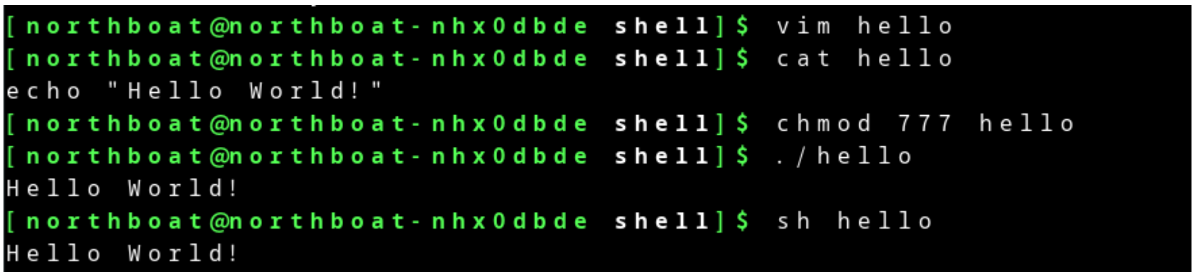
```
Linux northboat-nhx0dbde 6.1.12-1-MANJARO #1 SMP PREEMPT_DYNAMIC Tue Feb 14
21:59:10 UTC 2023 x86_64 GNU/Linux
```

实验内容

第一个 Shell 脚本

hello.sh

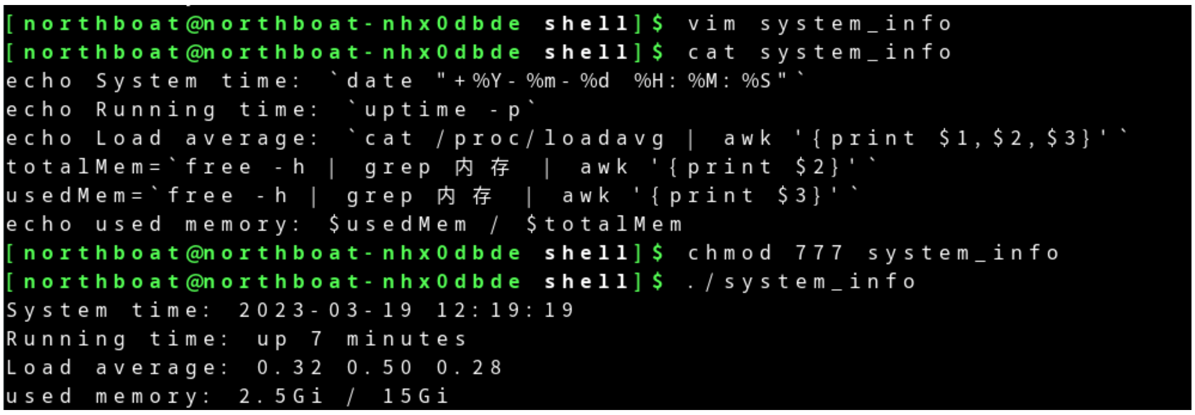
```
echo "Hello World!"
```



```
[northboat@northboat-nhx0dbde shell]$ vim hello
[northboat@northboat-nhx0dbde shell]$ cat hello
echo "Hello World!"
[northboat@northboat-nhx0dbde shell]$ chmod 777 hello
[northboat@northboat-nhx0dbde shell]$ ./hello
Hello World!
[northboat@northboat-nhx0dbde shell]$ sh hello
Hello World!
```

利用脚本获取系统信息

```
echo System time: `date "+%Y-%m-%d %H:%M:%S"`
echo Running time: `uptime -p`
echo Load average: `cat /proc/loadavg | awk '{print $1,$2,$3}'`
totalMem=`free -h | grep 内存 | awk '{print $2}'`
usedMem=`free -h | grep 内存 | awk '{print $3}'`
echo used memory: $usedMem / $totalMem
```



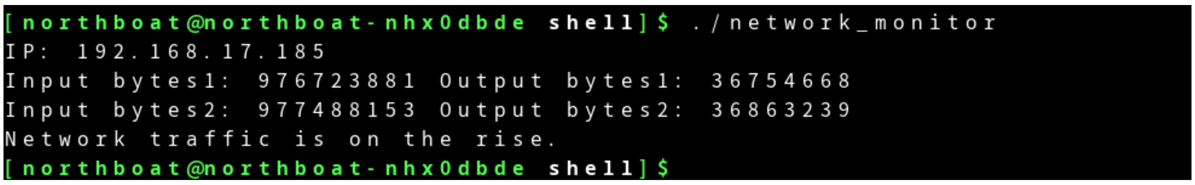
```
[northboat@northboat-nhx0dbde shell]$ vim system_info
[northboat@northboat-nhx0dbde shell]$ cat system_info
echo System time: `date "+%Y-%m-%d %H:%M:%S"`
echo Running time: `uptime -p`
echo Load average: `cat /proc/loadavg | awk '{print $1,$2,$3}'`
totalMem=`free -h | grep 内存 | awk '{print $2}'`
usedMem=`free -h | grep 内存 | awk '{print $3}'`
echo used memory: $usedMem / $totalMem
[northboat@northboat-nhx0dbde shell]$ chmod 777 system_info
[northboat@northboat-nhx0dbde shell]$ ./system_info
System time: 2023-03-19 12:19:19
Running time: up 7 minutes
Load average: 0.32 0.50 0.28
used memory: 2.5Gi / 15Gi
```

获取网卡信息

network_monitor.sh

```
echo IP: `ifconfig wlp12s0 | grep -w inet | awk '{print $2}'`

# get receive bytes 10 seconds ago
inputBytes1=`cat /proc/net/dev | grep wlp12s0 | awk -F: '{print $2}' | awk
'{print $1}'`

# get transmit bytes 10 seconds ago
outputBytes1=`cat /proc/net/dev | grep wlp12s0 | awk -F: '{print $2}' | awk
'{print $9}'`

echo Input bytes1: $inputBytes1 Output bytes1: $outputBytes1

sleep 10

# get receive bytes 10s later
```

```
inputBytes2=`cat /proc/net/dev | grep wlp12s0 | awk -F: '{print $2}'|awk '{print
$1}'`

# get transmit bytes 10s later
outputBytes2=`cat /proc/net/dev | grep wlp12s0 | awk -F: '{print $2}'|awk
'{print $9}'`

echo Input bytes2: $inputBytes2 Output bytes2: $outputBytes2

# evaluate the network
if [ $inputBytes1 -le $inputBytes2 ]
    then
    echo Network traffic is on the rise.
    else
    echo Network traffic is on the falling.
fi
```

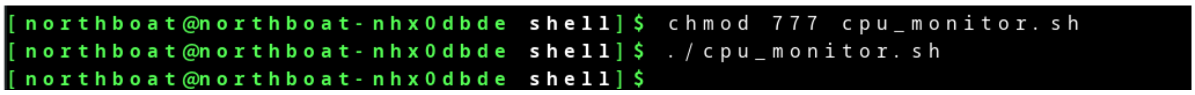


```
[northboat@northboat-nhx0dbde shell]$ ./network_monitor
IP: 192.168.17.185
Input bytes1: 976723881 Output bytes1: 36754668
Input bytes2: 977488153 Output bytes2: 36863239
Network traffic is on the rise.
[northboat@northboat-nhx0dbde shell]$
```

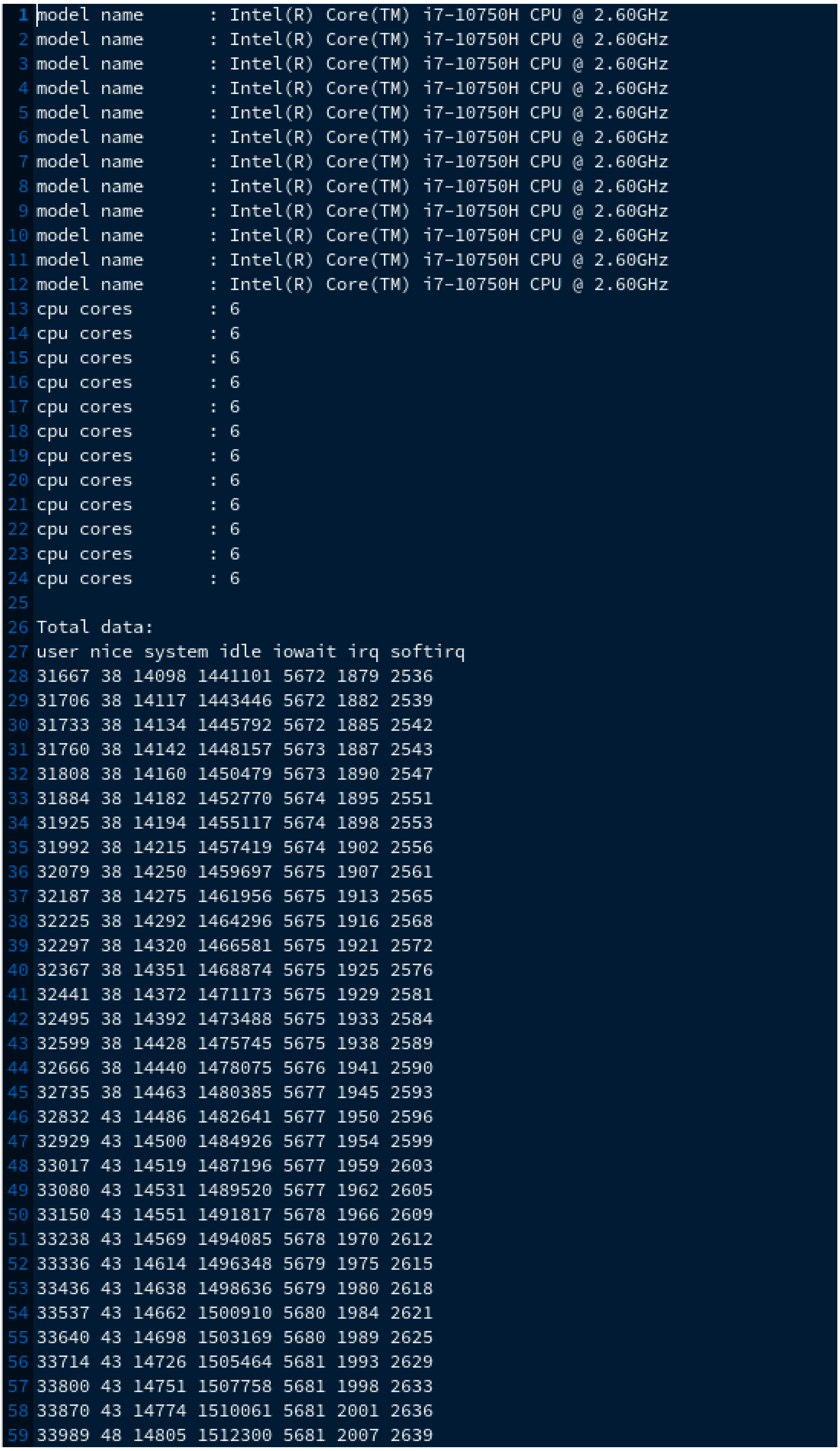
监控 CPU 负载

cpu_monitor.sh

```
#Function: monitor load average of cpu, and write to file
if [ -f cpu_monitor.txt ]
    then
    touch cpu_monitor.txt
fi

# modify file permission
if [ -w cpu_monitor.txt ]
    then
    chmod 755 cpu_monitor.txt
fi

# write cpu infomation
cat /proc/cpuinfo | grep "model name" > cpu_monitor.txt
cat /proc/cpuinfo | grep "cpu cores" >> cpu_monitor.txt

echo " " >> cpu_monitor.txt
echo Total data: >> cpu_monitor.txt
echo user nice system idle iowait irq softirq >> cpu_monitor.txt

#write cpu infomation every 2s
for ((i=0;i<=50;i++))
    do
    cat /proc/stat | grep 'cpu ' | awk '{print $2" "$3" "$4" "$5" "$6" "$7"
"$8}' >> cpu_monitor.txt
    sleep 2
done
```



```
[northboat@northboat-nhx0dbde shell]$ chmod 777 cpu_monitor.sh
[northboat@northboat-nhx0dbde shell]$ ./cpu_monitor.sh
[northboat@northboat-nhx0dbde shell]$
```



```
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
model name      : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6
cpu cores       : 6

Total data:
user nice system idle iowait irq softirq
31667 38 14098 1441101 5672 1879 2536
31706 38 14117 1443446 5672 1882 2539
31733 38 14134 1445792 5672 1885 2542
31760 38 14142 1448157 5673 1887 2543
31808 38 14160 1450479 5673 1890 2547
31884 38 14182 1452770 5674 1895 2551
31925 38 14194 1455117 5674 1898 2553
31992 38 14215 1457419 5674 1902 2556
32079 38 14250 1459697 5675 1907 2561
32187 38 14275 1461956 5675 1913 2565
32225 38 14292 1464296 5675 1916 2568
32297 38 14320 1466581 5675 1921 2572
32367 38 14351 1468874 5675 1925 2576
32441 38 14372 1471173 5675 1929 2581
32495 38 14392 1473488 5675 1933 2584
32599 38 14428 1475745 5675 1938 2589
32666 38 14440 1478075 5676 1941 2590
32735 38 14463 1480385 5677 1945 2593
32832 43 14486 1482641 5677 1950 2596
32929 43 14500 1484926 5677 1954 2599
33017 43 14519 1487196 5677 1959 2603
33080 43 14531 1489520 5677 1962 2605
33150 43 14551 1491817 5678 1966 2609
33238 43 14569 1494085 5678 1970 2612
33336 43 14614 1496348 5679 1975 2615
33436 43 14638 1498636 5679 1980 2618
33537 43 14662 1500910 5680 1984 2621
33640 43 14698 1503169 5680 1989 2625
33714 43 14726 1505464 5681 1993 2629
33800 43 14751 1507758 5681 1998 2633
33870 43 14774 1510061 5681 2001 2636
33989 48 14805 1512300 5681 2007 2639
```

实验总结

注意添加空格，命令后一定要有，加参数后一定要有，否则报错很难找，通过本次实验，大大加强了我的 Shell 编程能力，对父子进程有一定的了解，并且在排错过程中增强了心态

Linux 内核编译

实验环境

Vmware 虚拟机，Ubuntu16，为排除权限问题，本次实验命令均在`root`用户下执行

```
Linux ubuntu 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

实验内容

工具及环境准备

手动下载 Busybox，`apt`安装 QEMU 等工具

环境配置

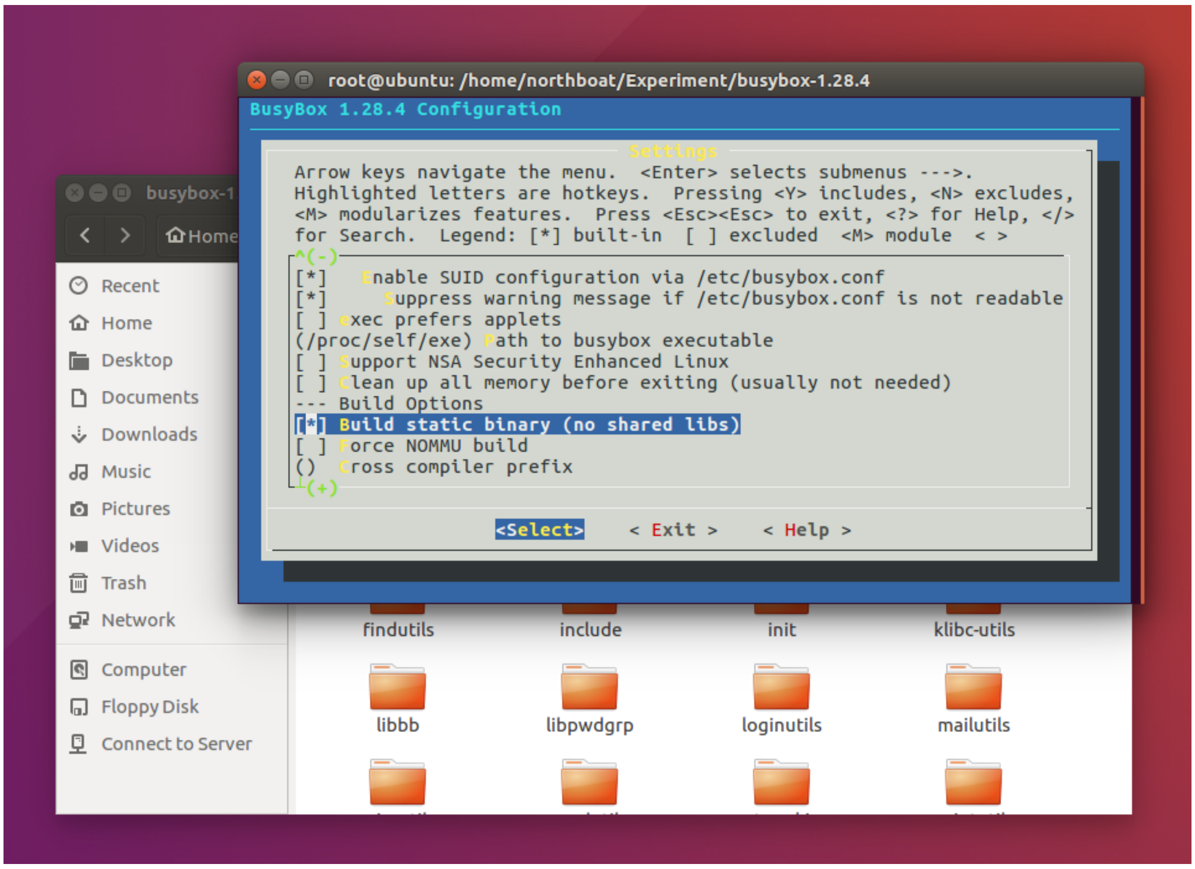
```
apt-get install gcc qemu qemu-system-arm gcc-arm-linux-gnueabi libncurses5-dev build-essential flex bison bc
```

编译最小文件系统

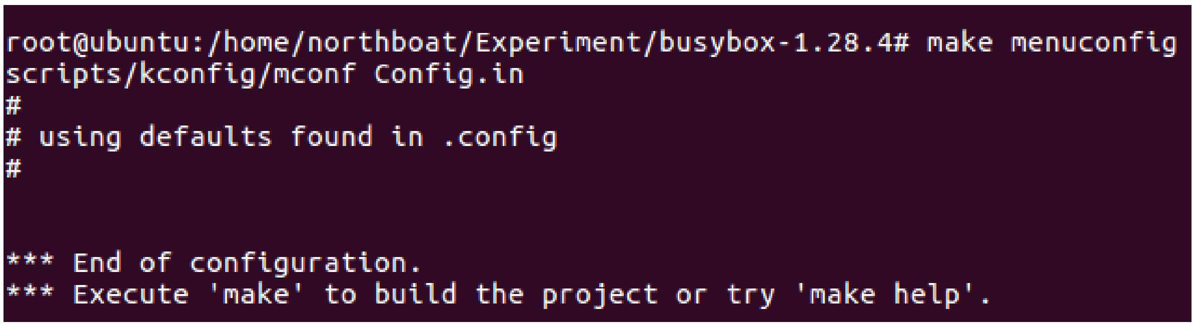
解压 busybox 至根目录，编译配置文件

```
tar -jxvf busybox-1.28.4.tar.bz2
cd /busybox-1.28.4
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
make menuconfig
```

在图形化界面进行内核配置：`settings - Build Options - [*]Build static binary(no shared libs)`



```
root@ubuntu: /home/northboat/Experiment/busybox-1.28.4
BusyBox 1.28.4 Configuration
                                 Settings
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
  ^(-)
  [*]   Enable SUID configuration via /etc/busybox.conf
  [*]     Suppress warning message if /etc/busybox.conf is not readable
  [ ] exec prefers applets
  (/proc/self/exe) Path to busybox executable
  [ ] Support NSA Security Enhanced Linux
  [ ] Clean up all memory before exiting (usually not needed)
  --- Build Options
  [*] Build static binary (no shared libs)
  [ ] Force NOMMU build
  ()  Cross compiler prefix
  (+)

              <Select>    < Exit >    < Help >
```

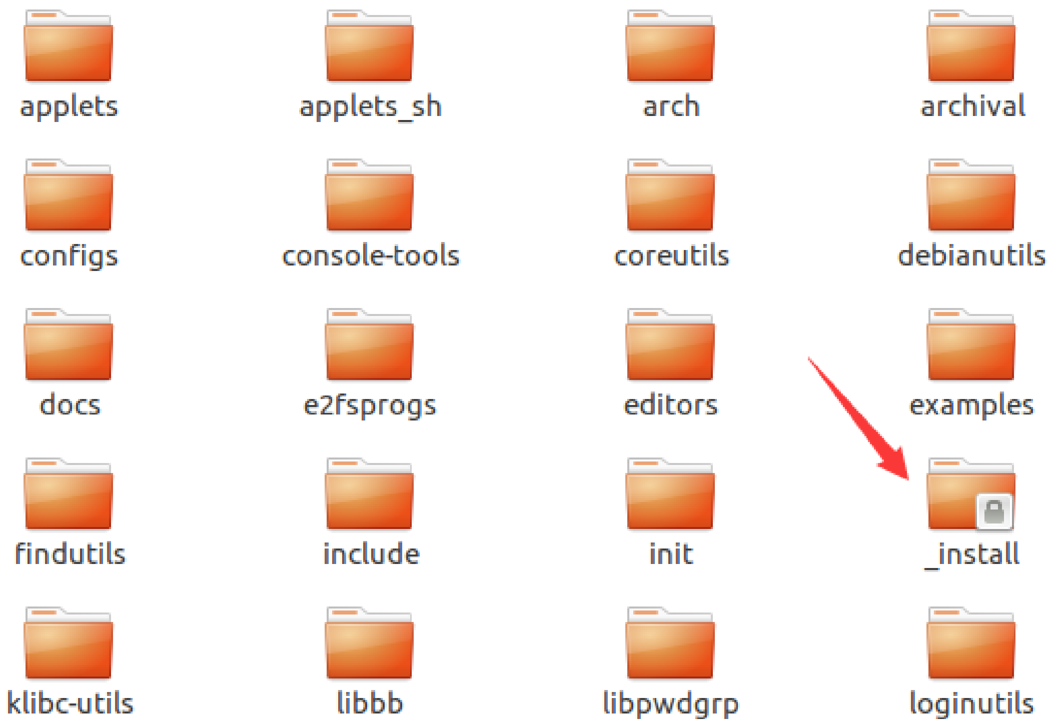


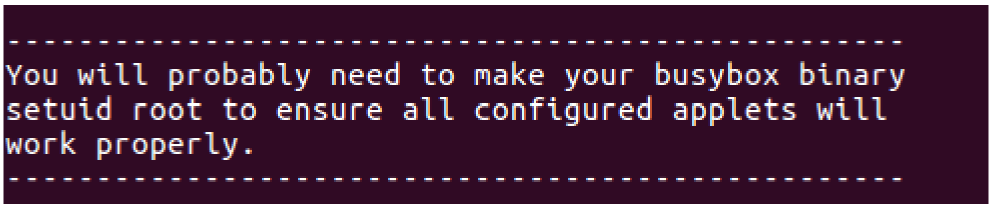
```
root@ubuntu:/home/northboat/Experiment/busybox-1.28.4# make menuconfig
scripts/kconfig/mconf Config.in
#
# using defaults found in .config
#


*** End of configuration.
*** Execute 'make' to build the project or try 'make help'.
```

配置完成后，编译文件系统

```
make install
```

完成后会在目录下生成`_install`目录





```
--------------------------------------------------
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.
--------------------------------------------------
```

编译内核

解压`Linux5.1`内核文件包，将`_install`拷入内核包的根目录，在`_install`下创建以下目录

```
mkdir etc
mkdir dev
mkdir mnt
mkdir -p etc/init.d
```

在`_install/etc/init.d`中创建文件`rcs`

```
mkdir -p /proc
mkdir -p /tmp
mkdir -p /sys
mkdir -p /mnt
/bin/mount -a
mkdir -p /dev/pts
mount -t devpts devpts /dev/pts
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
```

修改该文件权限

```
chmod 755 rcS
```

在`_install/etc`下创建文件`fstab`

```
proc /proc proc defaults 0 0
tmpfs /tmp tmpfs defaults 0 0
sysfs /sys sysfs defaults 0 0
tempfs /dev tmpfs defaults 0 0
debugfs /sys/kernel/debug debugfs defaults 0 0
```

在`_install/etc`下创建文件`inittab`

```
::sysinit:/etc/init.d/rcS
::respawn:-/bin/sh
::askfirst:-/bin/sh
::ctrlaltdel:/bin/umount -a -r
```

在`_install/dev`下创建设备节点

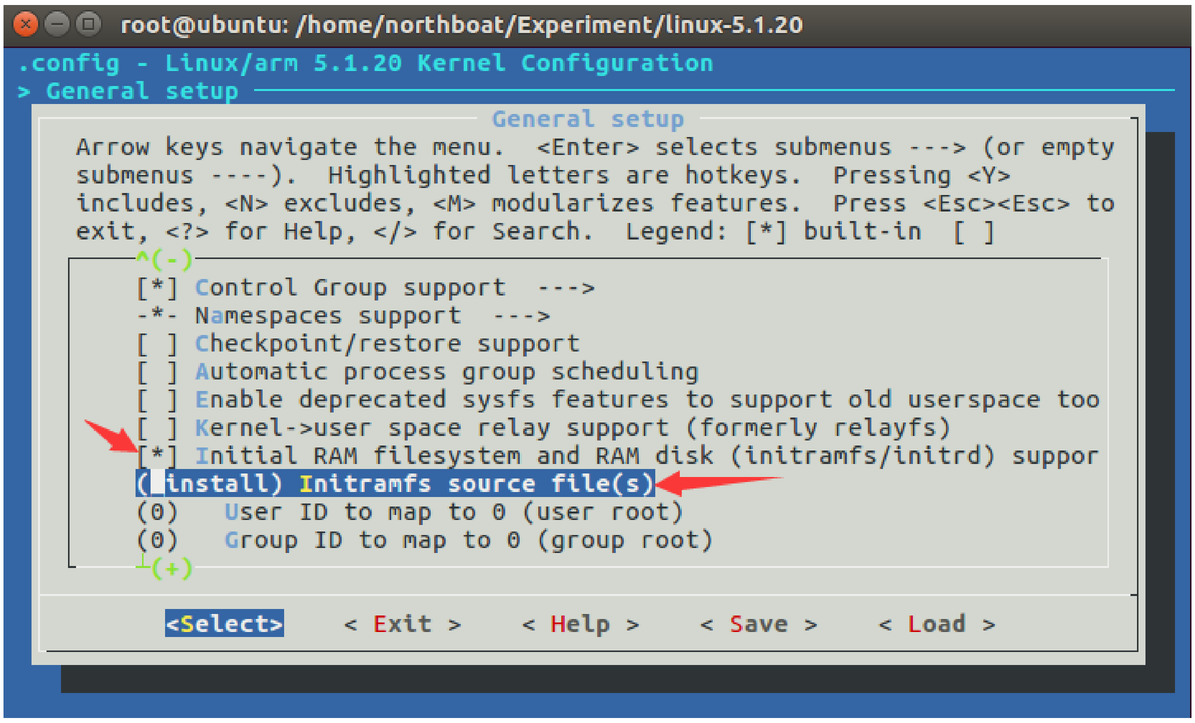
```
mknod console c 5 1
mknod null c 1 3
```

完成设置后，在内核根目录中编译内核配置

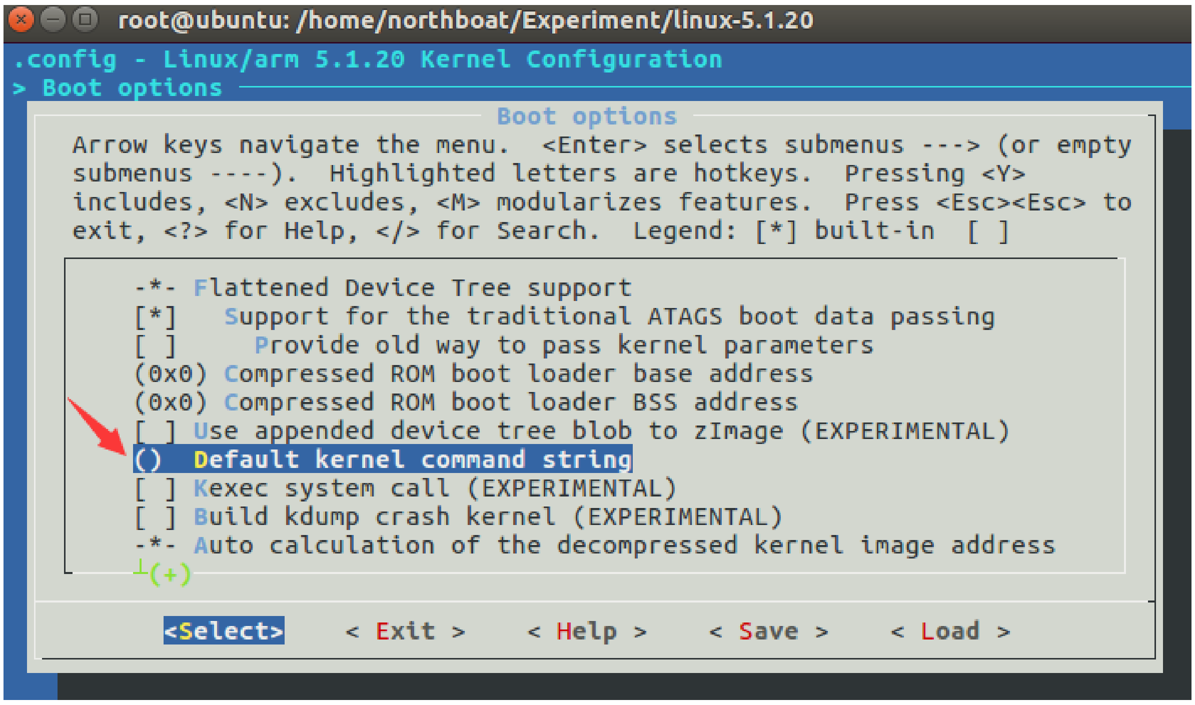
```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
make vexpress_defconfig
make menuconfig
```

完成以下设置

将`_install`填入`Initramfs source file`：位于`General setup - [*]Initial RAM filesystem and RAM disk (initramfs/initrd) support - (_install)Initramfs source file(s)`

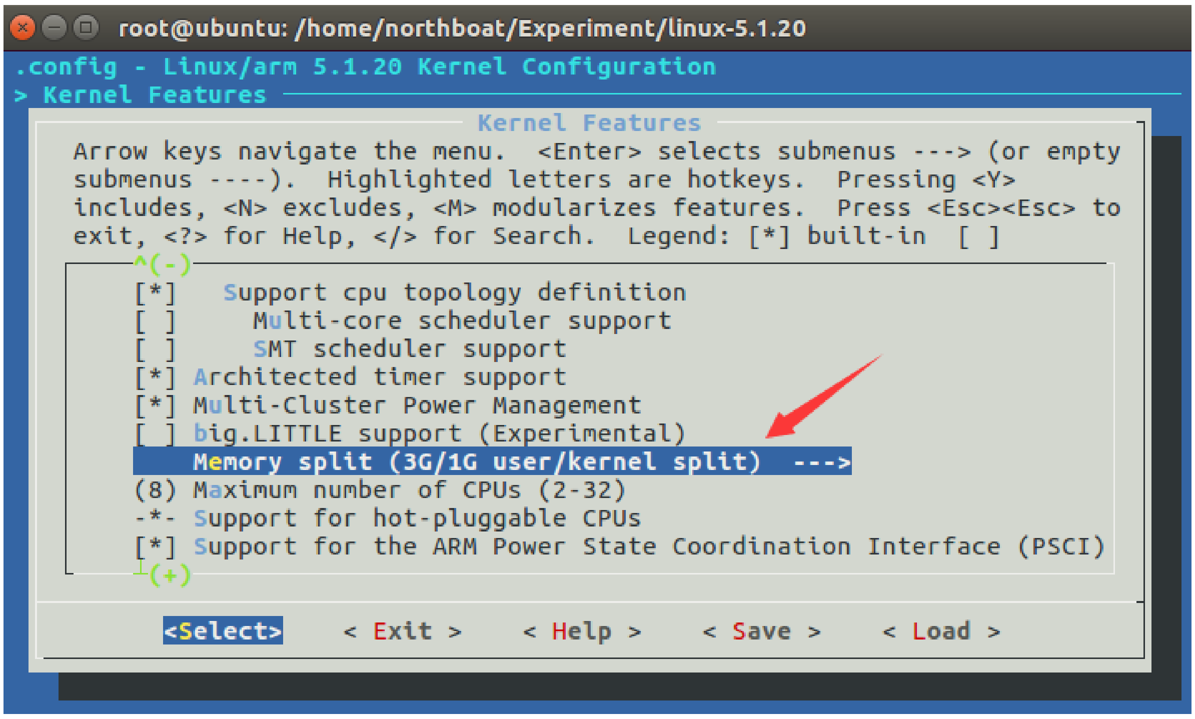

```
root@ubuntu: /home/northboat/Experiment/linux-5.1.20
.config - Linux/arm 5.1.20 Kernel Configuration
> General setup
                          General setup
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
        ^(-)
        [*] Control Group support  --->
        -*- Namespaces support  --->
        [ ] Checkpoint/restore support
        [ ] Automatic process group scheduling
        [ ] Enable deprecated sysfs features to support old userspace too
        [ ] Kernel->user space relay support (formerly relayfs)
        [*] Initial RAM filesystem and RAM disk (initramfs/initrd) suppor
        (_install) Initramfs source file(s)
        (0)   User ID to map to 0 (user root)
        (0)   Group ID to map to 0 (group root)
        ↓(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

清空`Default kernel command string`：位于`Boot option - Default kernel command string`

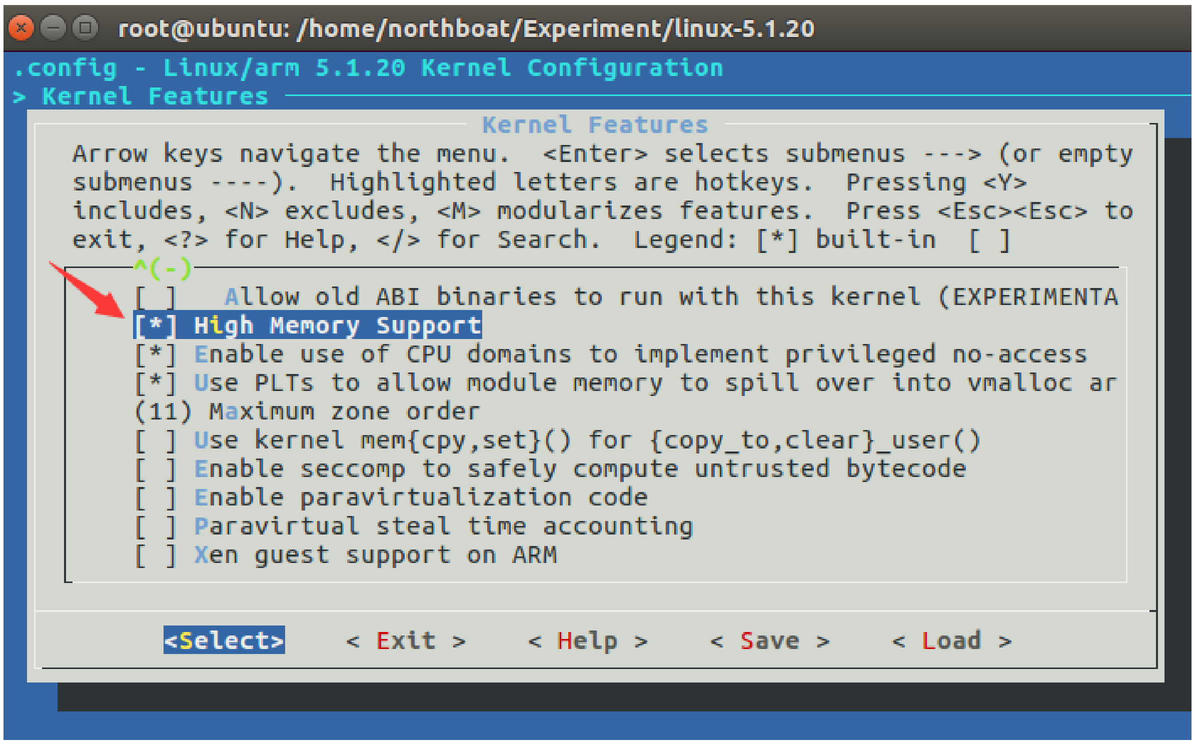


```
root@ubuntu: /home/northboat/Experiment/linux-5.1.20
.config - Linux/arm 5.1.20 Kernel Configuration
> Boot options
                              Boot options
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      -*- Flattened Device Tree support
      [*]   Support for the traditional ATAGS boot data passing
      [ ]     Provide old way to pass kernel parameters
      (0x0) Compressed ROM boot loader base address
      (0x0) Compressed ROM boot loader BSS address
      [ ] Use appended device tree blob to zImage (EXPERIMENTAL)
      ()  Default kernel command string
      [ ] Kexec system call (EXPERIMENTAL)
      [ ] Build kdump crash kernel (EXPERIMENTAL)
      -*- Auto calculation of the decompressed kernel image address
      └(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

配置`memory split`并打开高内存支持：`Kernel features - Memory split(3G/1G user/kernel) & [*] High Memory Support`



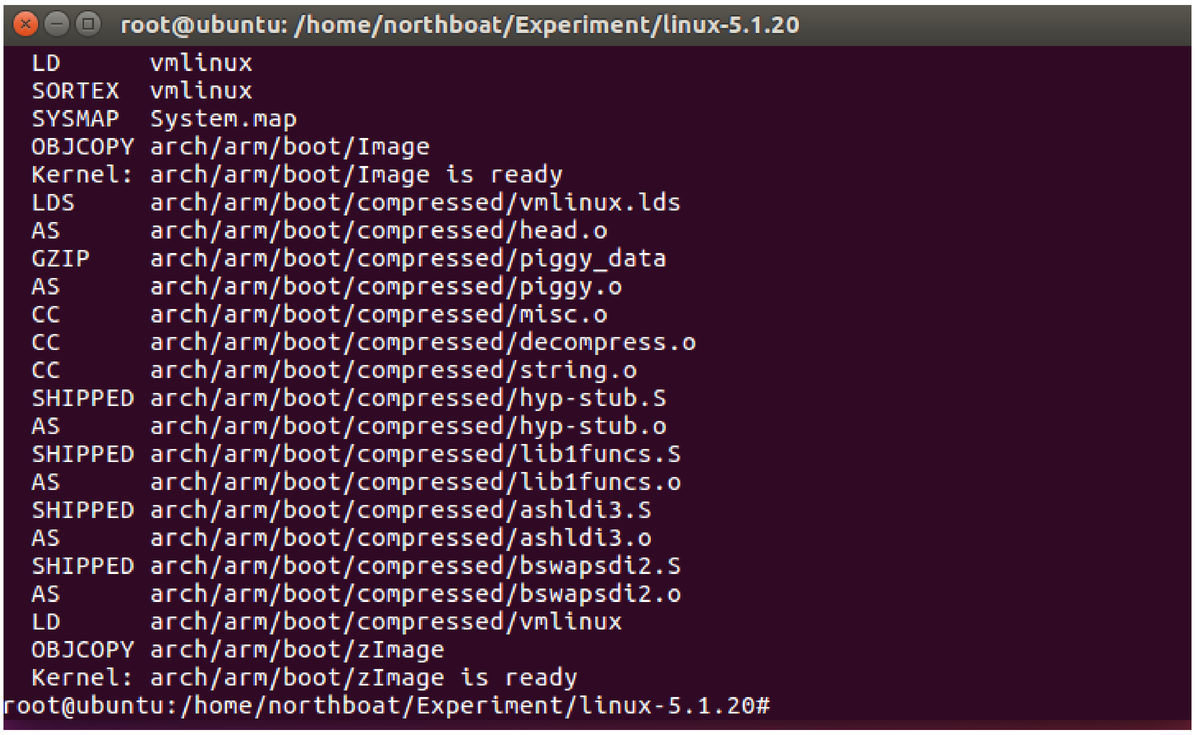
```
root@ubuntu: /home/northboat/Experiment/linux-5.1.20
.config - Linux/arm 5.1.20 Kernel Configuration
> Kernel Features
                              Kernel Features
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
      ^(-)
      [*]   Support cpu topology definition
      [ ]     Multi-core scheduler support
      [ ]     SMT scheduler support
      [*] Architected timer support
      [*] Multi-Cluster Power Management
      [ ] big.LITTLE support (Experimental)
          Memory split (3G/1G user/kernel split)  --->
      (8) Maximum number of CPUs (2-32)
      -*- Support for hot-pluggable CPUs
      [*] Support for the ARM Power State Coordination Interface (PSCI)
      └(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```



```
.config - Linux/arm 5.1.20 Kernel Configuration
> Kernel Features
                         Kernel Features
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
      ^(-)
      [ ]   Allow old ABI binaries to run with this kernel (EXPERIMENTA
      [*] High Memory Support
      [*] Enable use of CPU domains to implement privileged no-access
      [*] Use PLTs to allow module memory to spill over into vmalloc ar
      (11) Maximum zone order
      [ ] Use kernel mem{cpy,set}() for {copy_to,clear}_user()
      [ ] Enable seccomp to safely compute untrusted bytecode
      [ ] Enable paravirtualization code
      [ ] Paravirtual steal time accounting
      [ ] Xen guest support on ARM

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

编译内核

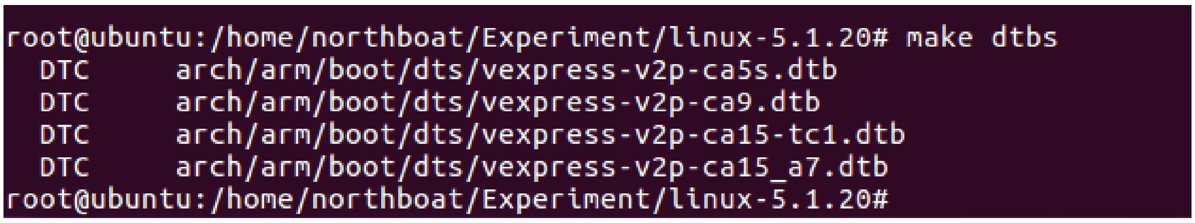
```
make bzImage ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
```



```
  LD      vmlinux
  SORTEX  vmlinux
  SYSMAP  System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  LDS     arch/arm/boot/compressed/vmlinux.lds
  AS      arch/arm/boot/compressed/head.o
  GZIP    arch/arm/boot/compressed/piggy_data
  AS      arch/arm/boot/compressed/piggy.o
  CC      arch/arm/boot/compressed/misc.o
  CC      arch/arm/boot/compressed/decompress.o
  CC      arch/arm/boot/compressed/string.o
  SHIPPED arch/arm/boot/compressed/hyp-stub.S
  AS      arch/arm/boot/compressed/hyp-stub.o
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  AS      arch/arm/boot/compressed/lib1funcs.o
  SHIPPED arch/arm/boot/compressed/ashldi3.S
  AS      arch/arm/boot/compressed/ashldi3.o
  SHIPPED arch/arm/boot/compressed/bswapsdi2.S
  AS      arch/arm/boot/compressed/bswapsdi2.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/northboat/Experiment/linux-5.1.20#
```

编译生成 dtb 文件

```
make dtbs
```



```
root@ubuntu:/home/northboat/Experiment/linux-5.1.20# make dtbs
  DTC     arch/arm/boot/dts/vexpress-v2p-ca5s.dtb
  DTC     arch/arm/boot/dts/vexpress-v2p-ca9.dtb
  DTC     arch/arm/boot/dts/vexpress-v2p-ca15-tc1.dtb
  DTC     arch/arm/boot/dts/vexpress-v2p-ca15_a7.dtb
root@ubuntu:/home/northboat/Experiment/linux-5.1.20#
```

运行 QEMU

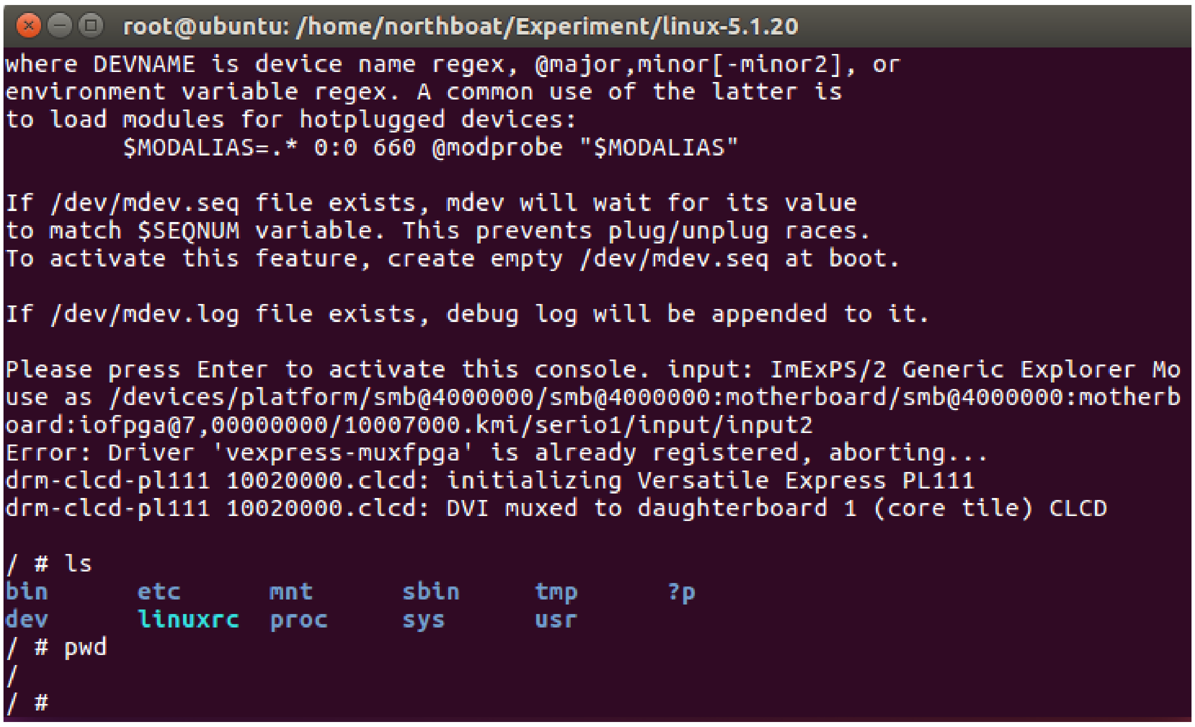
在编译好的 linux 内核根目录下执行

```
qemu-system-arm -M vexpress-a9 -m 256M -kernel arch/arm/boot/zImage -append
"rdinit=/linuxrc console=ttyAMA0 loglevel=8" -dtb arch/arm/boot/dts/vexpress-
v2p-ca9.dtb -nographic
```

以上命令中参数含义如下

- -M：指定硬件芯片框架
- -m：指定运行内存大小
- -kernel：指定运行的内核镜像
- -dtb：指定具体芯片的配置信息
- -nographic：指定不使用图形界面

成功进入内核命令行



root@ubuntu: /home/northboat/Experiment/linux-5.1.20

```
where DEVNAME is device name regex, @major,minor[-minor2], or
environment variable regex. A common use of the latter is
to load modules for hotplugged devices:
        $MODALIAS=.* 0:0 660 @modprobe "$MODALIAS"

If /dev/mdev.seq file exists, mdev will wait for its value
to match $SEQNUM variable. This prevents plug/unplug races.
To activate this feature, create empty /dev/mdev.seq at boot.

If /dev/mdev.log file exists, debug log will be appended to it.

Please press Enter to activate this console. input: ImExPS/2 Generic Explorer Mo
use as /devices/platform/smb@4000000/smb@4000000:motherboard/smb@4000000:motherb
oard:iofpga@7,00000000/10007000.kmi/serio1/input/input2
Error: Driver 'vexpress-muxfpga' is already registered, aborting...
drm-clcd-pl111 10020000.clcd: initializing Versatile Express PL111
drm-clcd-pl111 10020000.clcd: DVI muxed to daughterboard 1 (core tile) CLCD

/ # ls
bin       etc       mnt       sbin      tmp       ?p
dev       linuxrc   proc      sys       usr
/ # pwd
/
/ #
```

实验总结

通过手动编译 Linux 内核模块，以及通过 qemu 启动手动编译内核，使我明白了 Linux 的起源，以及对操作系统埋下了浓厚的兴趣，大量的`.c`及`.s`代码构成了庞大 Linux 的核心部件

Linux 内核模块

实验环境

Vmware 虚拟机，Ubuntu16

```
Linux ubuntu 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC
2020 x86_64 x86_64 x86_64 GNU/Linux
```

实验内容

编写一个简单的内核模块

编写模块程序：hello_module.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init hello_init(void){
    printk("This is hello_module, welcome to Linux kernel \n");
return 0;
}
static void __exit hello_exit(void){
    printk("see you next time!\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Mr Yu");
MODULE_DESCRIPTION("hello kernel module");
MODULE_ALIAS("hello");
```

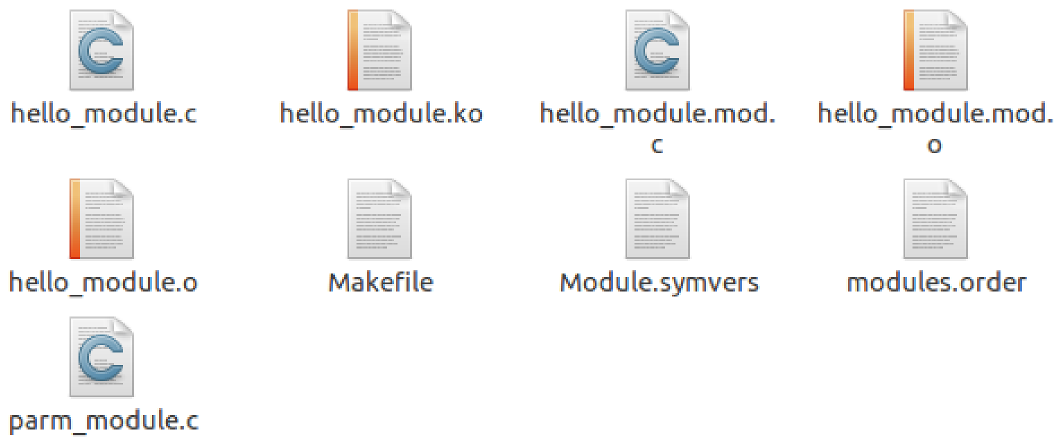
编译内核模块：编写`Makefile`文件

```
obj-m := hello_module.o
KERNELBUILD := /lib/modules/$(shell uname -r)/build
CURRENT_PATH := $(shell pwd)
all:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) modules
clean:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) clean
```

编译：将`hello_module.c`和`Makefile`放在同一目录做

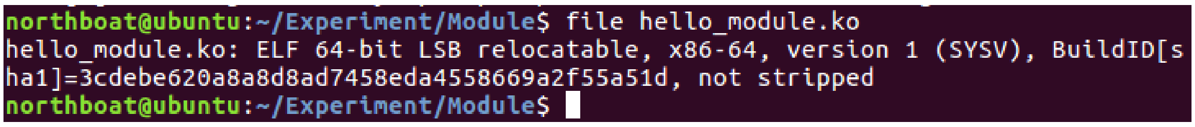
```
make
```

得到`hello_module.ko`文件



检查编译模块：通过`file`命令检查编译模块是否正确

```
file hello_module.ko
```



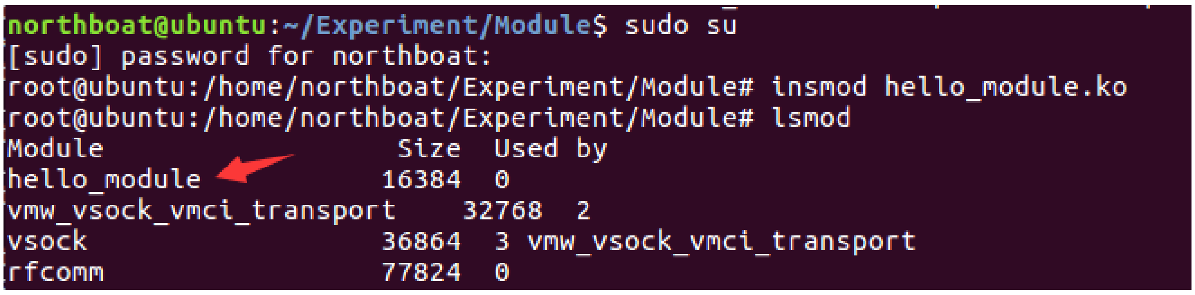
```
northboat@ubuntu:~/Experiment/Module$ file hello_module.ko
hello_module.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[s
ha1]=3cdebe620a8a8d8ad7458eda4558669a2f55a51d, not stripped
northboat@ubuntu:~/Experiment/Module$ 
```

插入模块：通过`insmod`命令插入模块

```
insmod hello_module.ko
```

完成插入后使用`lsmod`命令查看当前模块是否被加载到系统中

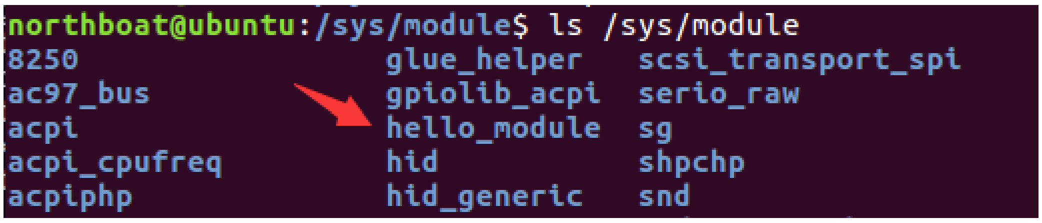
```
lsmod
```



```
northboat@ubuntu:~/Experiment/Module$ sudo su
[sudo] password for northboat:
root@ubuntu:/home/northboat/Experiment/Module# insmod hello_module.ko
root@ubuntu:/home/northboat/Experiment/Module# lsmod
Module                  Size  Used by
hello_module           16384  0
vmw_vsock_vmci_transport    32768  2
vsock                  36864  3 vmw_vsock_vmci_transport
rfcomm                 77824  0
```

在`/sys/modules`目录下会有以模块名命名的目录

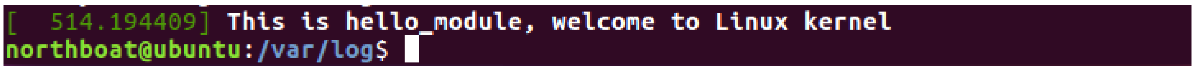
```
ls /sys/module
```



```
northboat@ubuntu:/sys/module$ ls /sys/module
8250                     glue_helper   scsi_transport_spi
ac97_bus                 gpiolib_acpi  serio_raw
acpi                     hello_module  sg
acpi_cpufreq             hid           shpchp
acpiphp                  hid_generic   snd
```

查看输出：通过`tail /var/log/messages`或`dmesg`命令查看输出结果

```
tail /var/log/message
dmesg
```



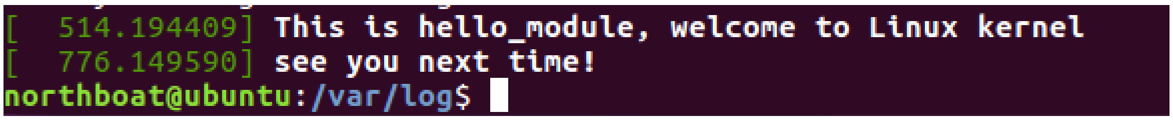
```
[  514.194409] This is hello_module, welcome to Linux kernel
northboat@ubuntu:/var/log$ 
```

卸载模块：通过`rmmod`命令卸载模块

```
rmmod hello_module
```

通过`dmesg`命令查看结果

```
dmesg
```



```
[  514.194409] This is hello_module, welcome to Linux kernel
[  776.149590] see you next time!
northboat@ubuntu:/var/log$ 
```

编写带参模块

Linux 内核提供一个宏来实现模块的参数传递

编写模块代码：parm_module.c

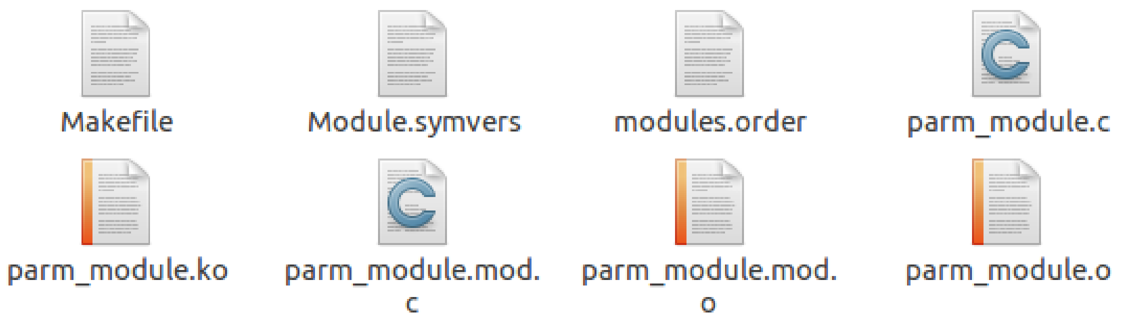
```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int debug = 1;
module_param(debug, int, 0644);
MODULE_PARM_DESC(debug, "debugging information");
#define dprintk(args...) if(debug){printk(KERN_DEBUG args);}
static int myparm = 10;
module_param(myparm, int, 0644);
MODULE_PARM_DESC(myparm, "kernel module parameter experiment.");

static int __init parm_init(void){
    dprintk("my linux kernel module init.\n");
    dprintk("module parameter = %d\n", myparm);
    return 0;
}
static void __exit parm_exit(void){
    printk("see you next time!\n");
}

module_init(parm_init);
module_exit(parm_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Mr Yu");
MODULE_DESCRIPTION("kernel module paramter experiment");
MODULE_ALIAS("myparm");
```

修改`Makefile`文件，编译并插入模块

Makefile

```makefile
obj-m := parm_module.o
KERNELBUILD := /lib/modules/$(shell uname -r)/build
CURRENT_PATH := $(shell pwd)
all:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) modules
clean:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) clean
```

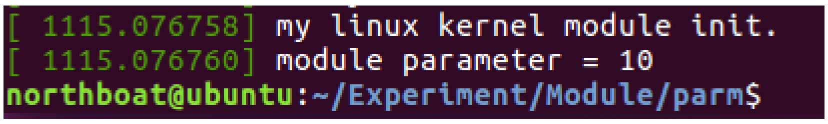
make



插入

```
insmod parm_module.ko
```

通过 dmesg 查看日志信息，可发现输出以上程序中 myparm 的默认值



```
[ 1115.076758] my linux kernel module init.
[ 1115.076760] module parameter = 10
northboat@ubuntu:~/Experiment/Module/parm$
```

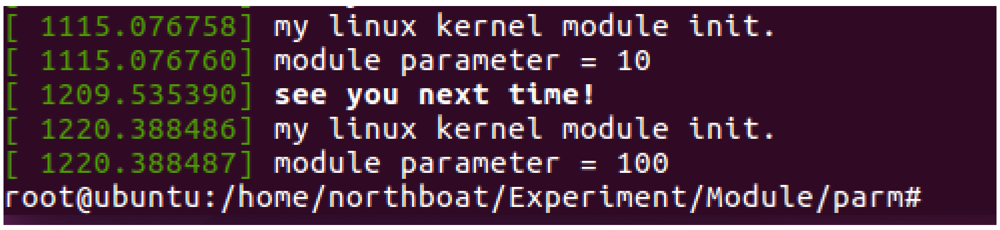
卸载模块

```
rmmod parm_module
```

赋值重新加载模块，修改参数 `myparm` 值为 100

```
insmod parm_module.ko myparm=100
```

通过 dmesg 查看日志信息，可发现 myparm 值已经改变



```
[ 1115.076758] my linux kernel module init.
[ 1115.076760] module parameter = 10
[ 1209.535390] see you next time!
[ 1220.388486] my linux kernel module init.
[ 1220.388487] module parameter = 100
root@ubuntu:/home/northboat/Experiment/Module/parm#
```

实验总结

通过内核模块的编写以及插入使用，使我对 Linux 的 Freedom 理念理解得更加深刻，同时对 linux c 编程有了更深入的理解

Linux 内存管理

实验环境

Vmware 虚拟机，Ubuntu16，

```
Linux ubuntu 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC
2020 x86_64 x86_64 x86_64 GNU/Linux
```

实验内容

> virtual memory areas，VMA

本实验内容编写一个内核模块，遍历一个用户进程中所有的 VMA，并且打印 这些 VMA 的属性信息，如 VMA 的大小、起始地址等，并通过与 `/proc/pid/maps` 中显示的信息进行对比验证 VMA 信息是否正确

编写并编译模块程序

vma_test.c

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/mm.h>
```

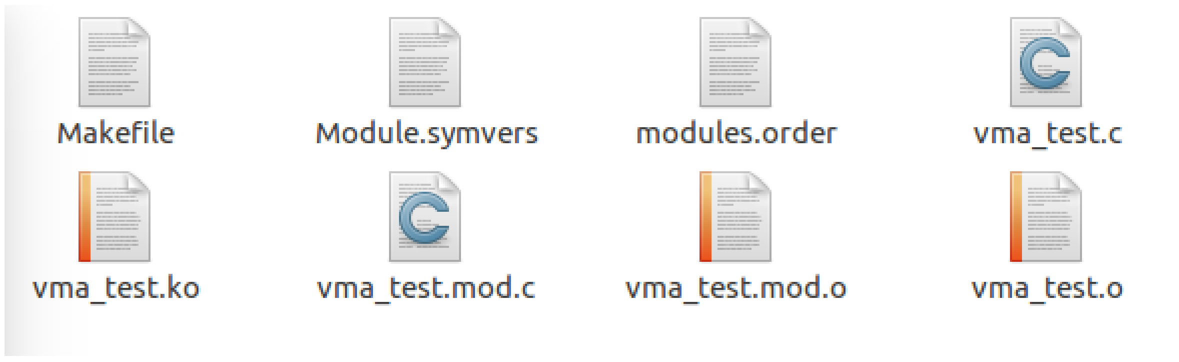
```
#include <linux/sched.h>

static int pid;
module_param(pid, int, 0644);

static void printit(struct task_struct *tsk) {
    struct mm_struct *mm;
    struct vm_area_struct *vma;
    int j = 0;
    unsigned long start, end, length;

    mm = tsk->mm;
    pr_info("mm_struct addr = 0x%p\n", mm);
    vma = mm->mmap;

    /* 使用 mmap_sem 读写信号量进行保护 */
    down_read(&mm->mmap_sem);
    pr_info("vmas: vma start end length\n");

    while (vma) {
        j++;
        start = vma->vm_start;
        end = vma->vm_end;
        length = end - start;
        pr_info("%6d: %16p %12lx %12lx %8ld\n",
                j, vma, start, end, length);
        vma = vma->vm_next;
    }
    up_read(&mm->mmap_sem);
}

static int __init vma_init(void) {
    struct task_struct *tsk;
    /* 如果插入模块时未定义 pid 号，则使用当前 pid */
    if (pid == 0) {
        tsk = current;
        pid = current->pid;
        pr_info("using current process\n");
    } else {
        tsk = pid_task(find_vpid(pid), PIDTYPE_PID);
    }
    if (!tsk)
        return -1;
    pr_info(" Examining vma's for pid=%d, command=%s\n", pid, tsk->comm);
    printit(tsk);
    return 0;
}

static void __exit vma_exit(void) {
    pr_info("Module exit\n");
}

module_init(vma_init);
module_exit(vma_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Mr Yu");
MODULE_DESCRIPTION("vma test");
```

编译内核模块

编写 Makefile

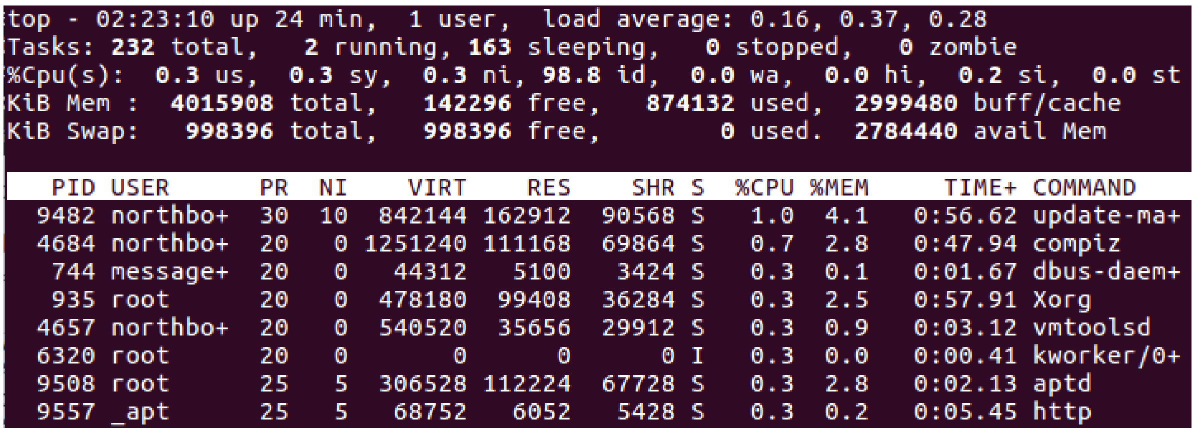
```
obj-m := vma_test.o

KERNELBUILD := /lib/modules/$(shell uname -r)/build
CURRENT_PATH := $(shell pwd)

all:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) modules
clean:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) clean
```

使用 make 命令编译，得到`vma_test.ko`文件



插入模块

通过 top 命令随便获取一个进程号



```
top - 02:23:10 up 24 min,  1 user,  load average: 0.16, 0.37, 0.28
Tasks: 232 total,   2 running, 163 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.3 ni, 98.8 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem :  4015908 total,   142296 free,   874132 used,  2999480 buff/cache
KiB Swap:   998396 total,   998396 free,        0 used.  2784440 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9482 northbo+  30  10  842144 162912  90568 S   1.0  4.1   0:56.62 update-ma+
 4684 northbo+  20   0 1251240 111168  69864 S   0.7  2.8   0:47.94 compiz
  744 message+  20   0   44312   5100   3424 S   0.3  0.1   0:01.67 dbus-daem+
  935 root      20   0  478180  99408  36284 S   0.3  2.5   0:57.91 Xorg
 4657 northbo+  20   0  540520  35656  29912 S   0.3  0.9   0:03.12 vmtoolsd
 6320 root      20   0       0      0      0 I   0.3  0.0   0:00.41 kworker/0+
 9508 root      25   5  306528 112224  67728 S   0.3  2.8   0:02.13 aptd
 9557 _apt      25   5   68752   6052   5428 S   0.3  0.2   0:05.45 http
```

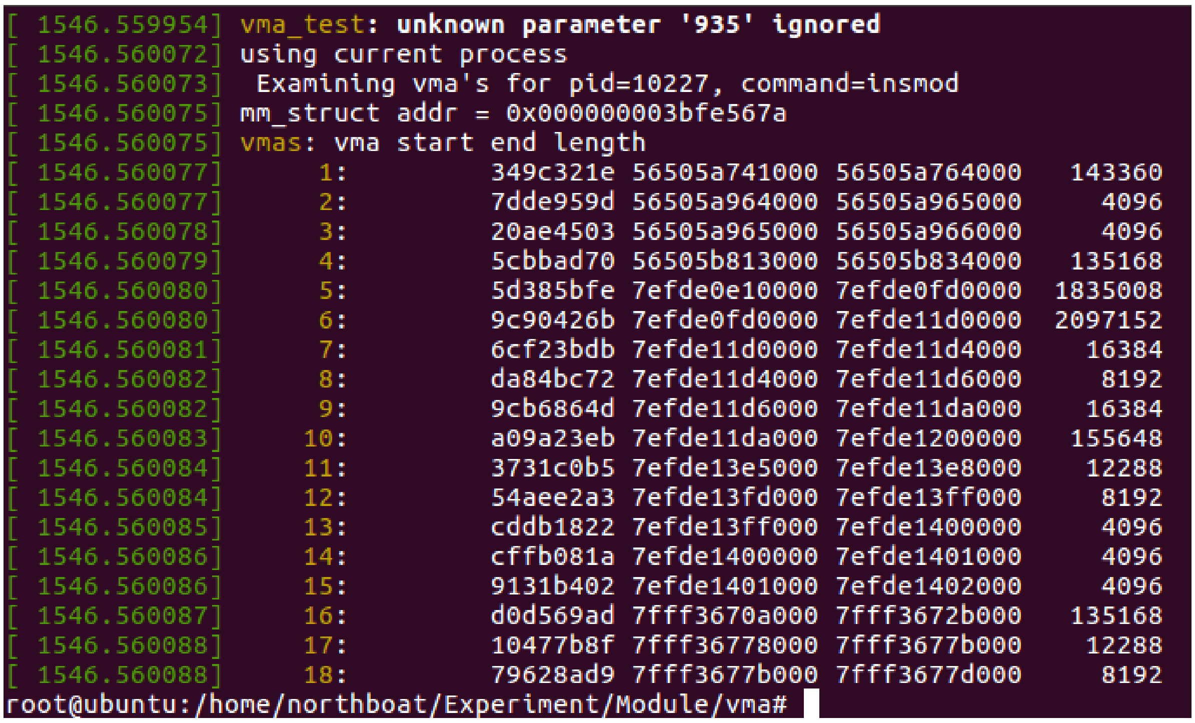
这里选择`xorg`的进程号`935`，使用 insmod 命令插入模块并传参

```
insmod vma_test.ko 935
```

查看程序打印信息

使用 dmesg 查看信息

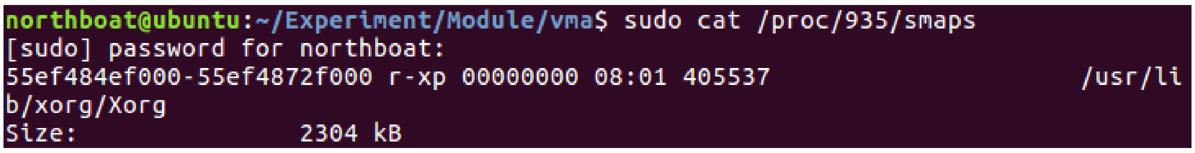
```
dmesg
```



```
[ 1546.559954] vma_test: unknown parameter '935' ignored
[ 1546.560072] using current process
[ 1546.560073]  Examining vma's for pid=10227, command=insmod
[ 1546.560075] mm_struct addr = 0x000000003bfe567a
[ 1546.560075] vmas: vma start end length
[ 1546.560077]      1:         349c321e 56505a741000 56505a764000   143360
[ 1546.560077]      2:         7dde959d 56505a964000 56505a965000     4096
[ 1546.560078]      3:         20ae4503 56505a965000 56505a966000     4096
[ 1546.560079]      4:         5cbbad70 56505b813000 56505b834000   135168
[ 1546.560080]      5:         5d385bfe 7efde0e10000 7efde0fd0000  1835008
[ 1546.560080]      6:         9c90426b 7efde0fd0000 7efde11d0000  2097152
[ 1546.560081]      7:         6cf23bdb 7efde11d0000 7efde11d4000    16384
[ 1546.560082]      8:         da84bc72 7efde11d4000 7efde11d6000     8192
[ 1546.560082]      9:         9cb6864d 7efde11d6000 7efde11da000    16384
[ 1546.560083]     10:         a09a23eb 7efde11da000 7efde1200000   155648
[ 1546.560084]     11:         3731c0b5 7efde13e5000 7efde13e8000    12288
[ 1546.560084]     12:         54aee2a3 7efde13fd000 7efde13ff000     8192
[ 1546.560085]     13:         cddb1822 7efde13ff000 7efde1400000     4096
[ 1546.560086]     14:         cffb081a 7efde1400000 7efde1401000     4096
[ 1546.560086]     15:         9131b402 7efde1401000 7efde1402000     4096
[ 1546.560087]     16:         d0d569ad 7fff3670a000 7fff3672b000   135168
[ 1546.560088]     17:         10477b8f 7fff36778000 7fff3677b000    12288
[ 1546.560088]     18:         79628ad9 7fff3677b000 7fff3677d000     8192
root@ubuntu:/home/northboat/Experiment/Module/vma#
```

从 proc 虚拟文件系统中查看进程第一个 VMA 的信息

```
cat /proc/935/smaps
```



```
northboat@ubuntu:~/Experiment/Module/vma$ sudo cat /proc/935/smaps
[sudo] password for northboat:
55ef484ef000-55ef4872f000 r-xp 00000000 08:01 405537                     /usr/li
b/xorg/Xorg
Size:                 2304 kB
```

通过对比发现第一块内存区域地址起始位置一致，说明程序输出信息正确

实验总结

通过内核模块程序查看 VMA，使我对 linux 内核模块编写能力提升，并且对 linux 的内存管理理解更加深刻

Linux 设备驱动

实验环境

Vmware 虚拟机，Ubuntu16，

```
Linux ubuntu 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC
2020 x86_64 x86_64 x86_64 GNU/Linux
```

实验内容

编写驱动程序

mycdev_driver.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/types.h>
```

```c
#include <linux/fs.h>
#include <linux/mm.h>
#include <linux/sched.h>
#include <linux/cdev.h>
#include <asm/io.h>
#include <asm/switch_to.h>
#include <asm/uaccess.h>
#include <linux/errno.h>
#include <linux/uaccess.h>

#define MYCDEV_MAJOR 300  /*主设备号，通过cat /proc/devices 查询，选择未使用的设备号*/
#define MYCDEV_SIZE 1024


static int mycdev_open(struct inode *inode, struct file *fp){
    return 0;
}

static int mycdev_release(struct inode *inode, struct file *fp){
    return 0;
}

/*实现read程序*/
static ssize_t mycdev_read(struct file *fp, char __user *buf, size_t size,
loff_t *pos){
    unsigned long p = *pos;
    unsigned int count = size;
    char kernel_buf[MYCDEV_SIZE] = "This is mycdev driver!";
    int i;

    if(p >= MYCDEV_SIZE)
        return -1;
    if(count > MYCDEV_SIZE)
        count = MYCDEV_SIZE - p;
    if(copy_to_user(buf, kernel_buf, count) != 0){
        printk("read error!\n");

        return -1;
    }

    printk("reader: %d bytes was read.\n", count);

    return size;
}

/*实现write程序*/
static ssize_t mycdev_write(struct file *fp, const char __user *buf, size_t
size, loff_t *pos){
    return size;
}

/*填充file operations结构*/
static const struct file_operations mycdev_fops = {
    .owner = THIS_MODULE,
    .open = mycdev_open,
    .release = mycdev_release,
    .read = mycdev_read,
    .write = mycdev_write,
};


/*模块初始化函数*/
static int __init mycdev_init(void){
    printk("mycdev driver is now starting!\n");

    /*注册驱动程序*/
    int ret = register_chrdev(MYCDEV_MAJOR, "my_cdev_driver", &mycdev_fops);

    if(ret < 0){
        printk("register failed!\n");
        return 0;
    }else{
        printk("register successfully!\n");
    }

    return 0;
}

/*卸载模块函数*/
static void __exit mycdev_exit(void){
    printk("mycdev driver is now leaving!\n");
    unregister_chrdev(MYCDEV_MAJOR, " ");
}

module_init(mycdev_init);
module_exit(mycdev_exit);

MODULE_LICENSE("GPL");
```

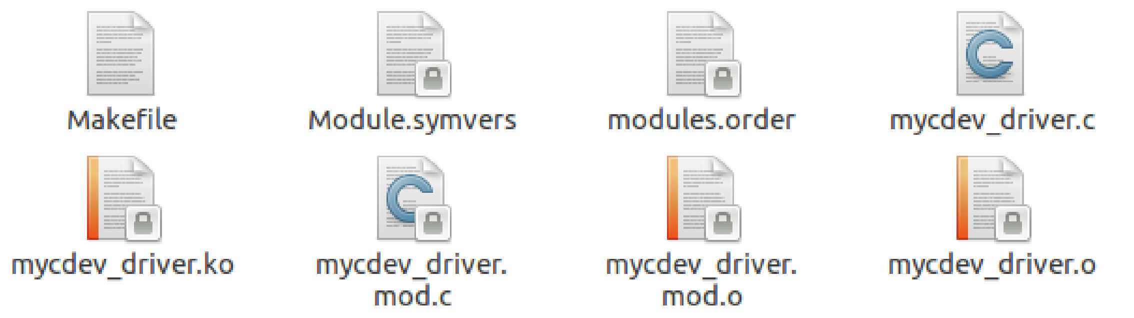
编译并插入模块

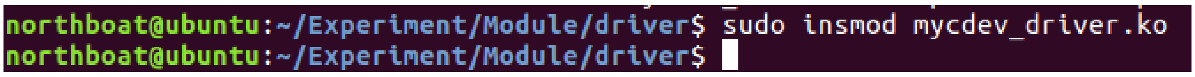
Makefile

```
obj-m := mycdev_driver.o
KERNELBUILD := /lib/modules/$(shell uname -r)/build
CURRENT_PATH := $(shell pwd)
all:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) modules
clean:
    make -C $(KERNELBUILD) M=$(CURRENT_PATH) clean
```

编译并插入模块

```
sudo make
insmod mycdev_driver.ko
```



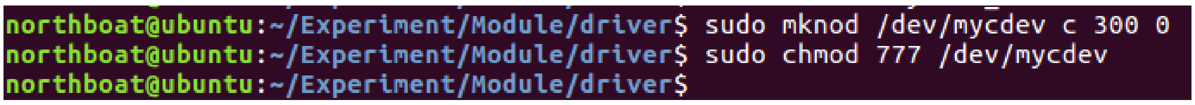


```
northboat@ubuntu:~/Experiment/Module/driver$ sudo insmod mycdev_driver.ko
northboat@ubuntu:~/Experiment/Module/driver$ 
```

创建文件设备节点

创建文件节点并修改文件权限

```
sudo mknod /dev/mycdev c 300 0
sudo chmod 777 /dev/mycdev
```



```
northboat@ubuntu:~/Experiment/Module/driver$ sudo mknod /dev/mycdev c 300 0
northboat@ubuntu:~/Experiment/Module/driver$ sudo chmod 777 /dev/mycdev
northboat@ubuntu:~/Experiment/Module/driver$
```

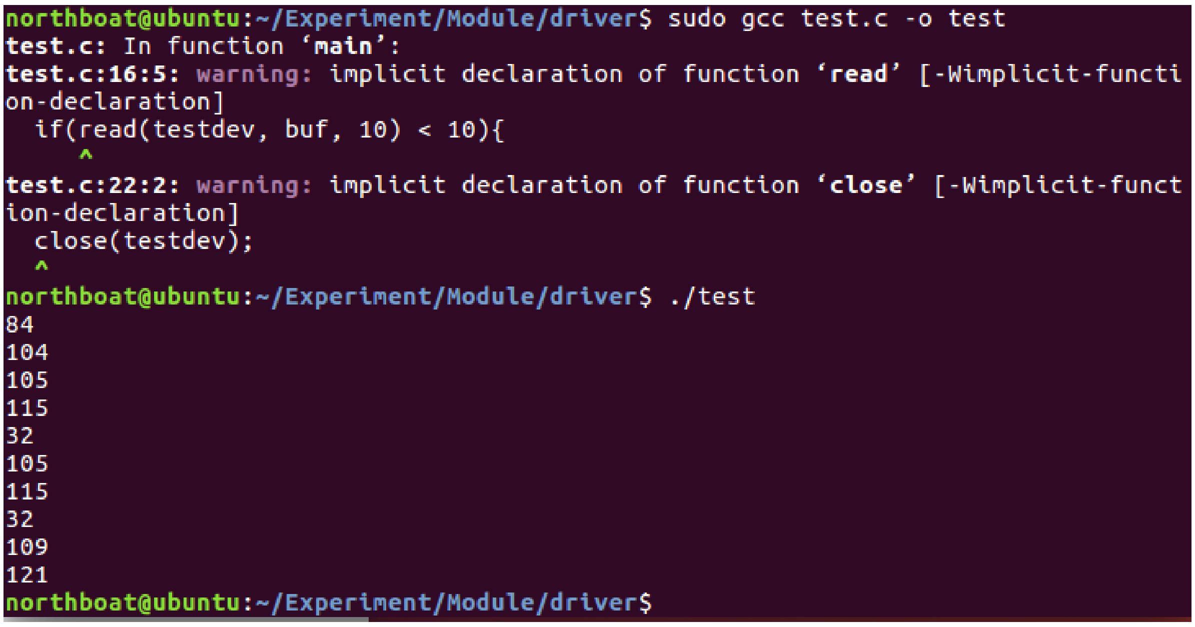
编写测试程序并执行

test.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>

int main() {
    int testdev;
    char buf[10];
    testdev = open("/dev/mycdev", O_RDWR);
    if(testdev == -1){
        printf("open file failed!\n");
        exit(1);
    }
    //将 testdev 所指的文件读 10 个字节到 buf 中
    if(read(testdev, buf, 10) < 10){
        printf("Read error!\n");
        exit(1);
    }
    for(int i = 0; i < 10; i++)
        printf("%d\n", buf[i]);
    close(testdev);
    return 0;
}
```

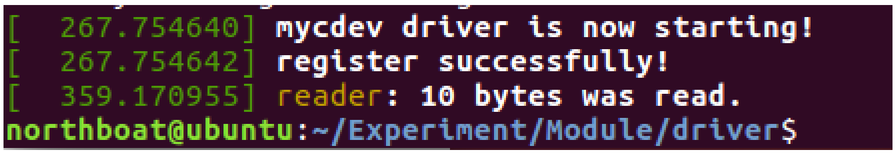
编译并执行

```
sudo gcc test.c -o test
./test
```


```
northboat@ubuntu:~/Experiment/Module/driver$ sudo gcc test.c -o test
test.c: In function ‘main’:
test.c:16:5: warning: implicit declaration of function ‘read’ [-Wimplicit-functi
on-declaration]
  if(read(testdev, buf, 10) < 10){
     ^
test.c:22:2: warning: implicit declaration of function ‘close’ [-Wimplicit-funct
ion-declaration]
  close(testdev);
  ^
northboat@ubuntu:~/Experiment/Module/driver$ ./test
84
104
105
115
32
105
115
32
109
121
northboat@ubuntu:~/Experiment/Module/driver$
```

查看日志信息

通过 dmesg 命令查看


```
[  267.754640] mycdev driver is now starting!
[  267.754642] register successfully!
[  359.170955] reader: 10 bytes was read.
northboat@ubuntu:~/Experiment/Module/driver$
```

实验总结

Linux 内核根据各类设备抽象出一套完整的驱动框架和 API 接口，以便驱动开发者在编写驱动程序时可重复使用，通过调用 Linux 驱动 API，使我对 Linux 的驱动开发有了基础的理解，对 Linux 操作系统也有了更细致的了解