操作系统实验二—openEuler 实验

实验分工：

姓名：王鸣一 班级：2023211804 学号：2023211475 分工：33.34%

姓名：孙溯阳 班级：2023211804 学号：2023211502 分工：33.34%

姓名：章泽亮 班级：2023211803 学号：2023211490 分工：33.34%

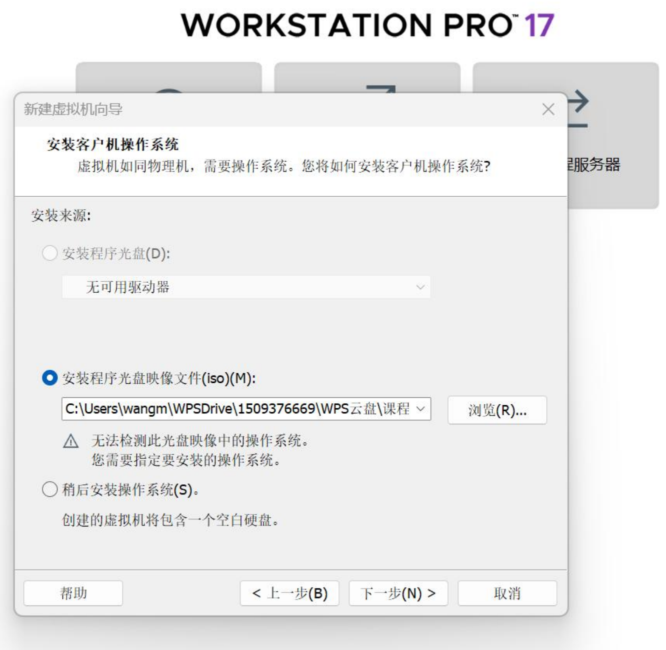
简介：《openEuler 实验》主要是面向国产操作系统 openEuler 的实验。该实验要求我们能够从零安装 openEuler 操作系统，采用重新编译源码的方式将内核更新至最新版，并且完成一些基础的实验。

一、 安装 openEuler 20.03-LTS

1、VMware 虚拟机配置

在 VMware 中自定义新建虚拟机

设置虚拟机的配置，如下图



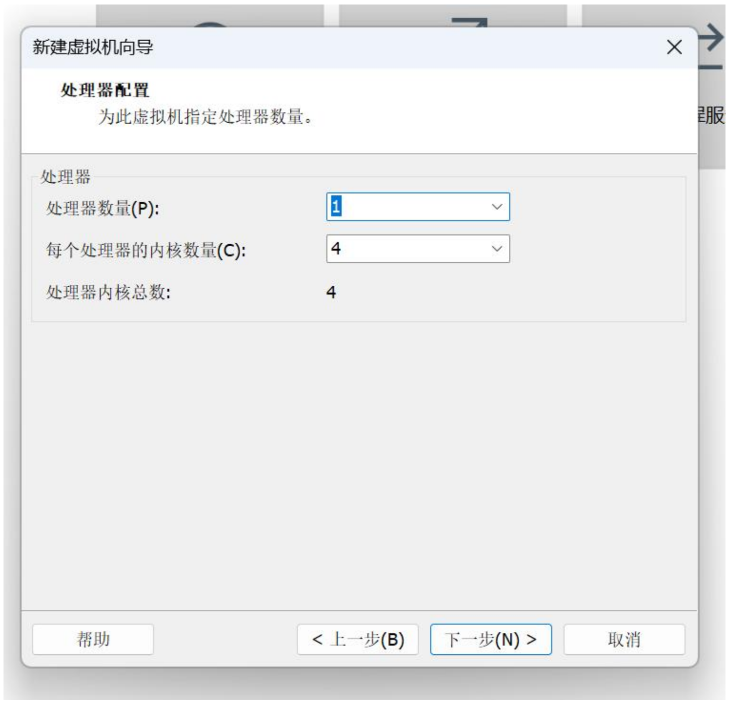
安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统？

安装来源：

○ 安装程序光盘(D):
无可用驱动器

● 安装程序光盘映像文件(iso)(M):
C:\Users\wangm\WPSDrive\1509376669\WPS云盘\课程
⚠ 无法检测此光盘映像中的操作系统。
您需要指定要安装的操作系统。

○ 稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。



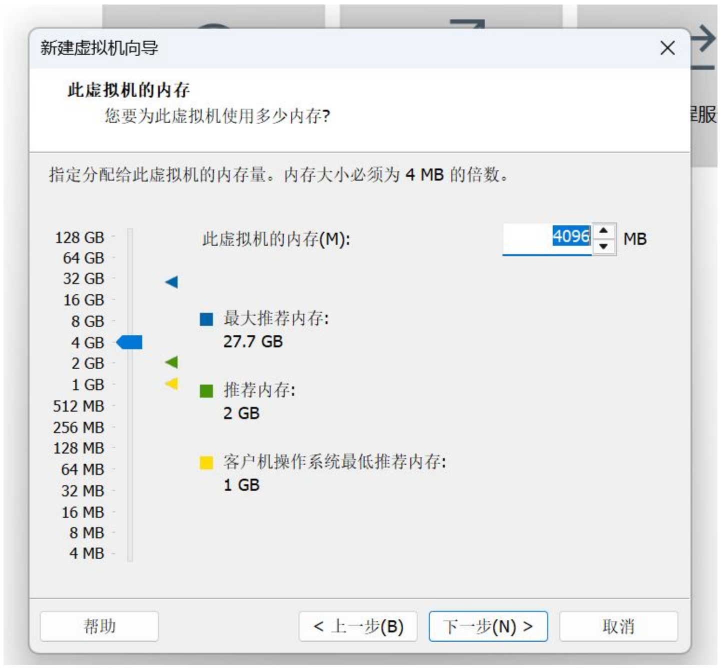
新建虚拟机向导

处理器配置

为此虚拟机指定处理器数量。

处理器

处理器数量(P):	1
每个处理器的内核数量(C):	4
处理器内核总数:	4



新建虚拟机向导

此虚拟机的内存

您要为此虚拟机使用多少内存?

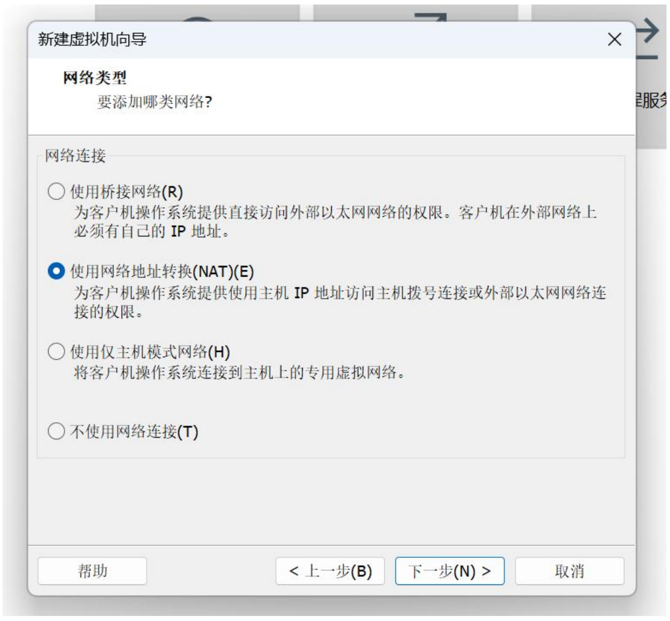
指定分配给此虚拟机的内存量。内存大小必须为 4 MB 的倍数。

此虚拟机的内存(M): 4096 MB

🟦 最大推荐内存:
27.7 GB

🟩 推荐内存:
2 GB

🟨 客户机操作系统最低推荐内存:
1 GB



新建虚拟机向导

网络类型

要添加哪类网络?

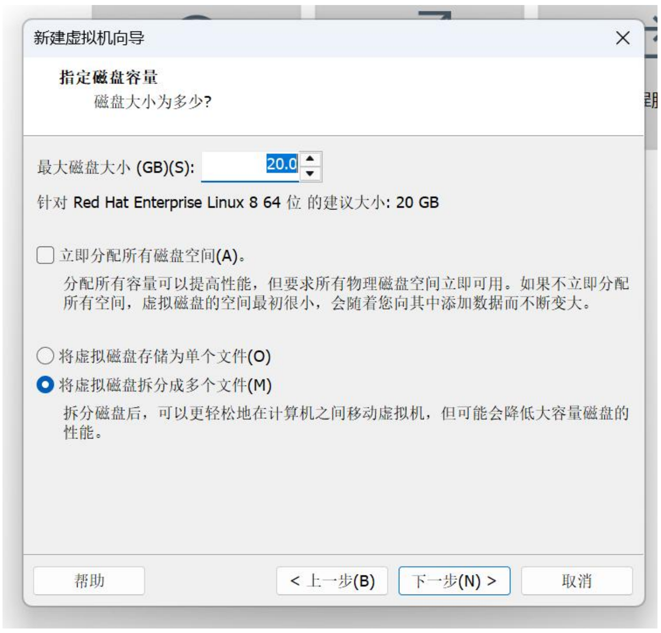
网络连接

( ) 使用桥接网络(R)
为客户机操作系统提供直接访问外部以太网网络的权限。客户机在外部网络上必须有自己的 IP 地址。

[✓] 使用网络地址转换(NAT)(E)
为客户机操作系统提供使用主机 IP 地址访问主机拨号连接或外部以太网网络连接的权限。

( ) 使用仅主机模式网络(H)
将客户机操作系统连接到主机上的专用虚拟网络。

( ) 不使用网络连接(T)



新建虚拟机向导

指定磁盘容量

磁盘大小为多少?

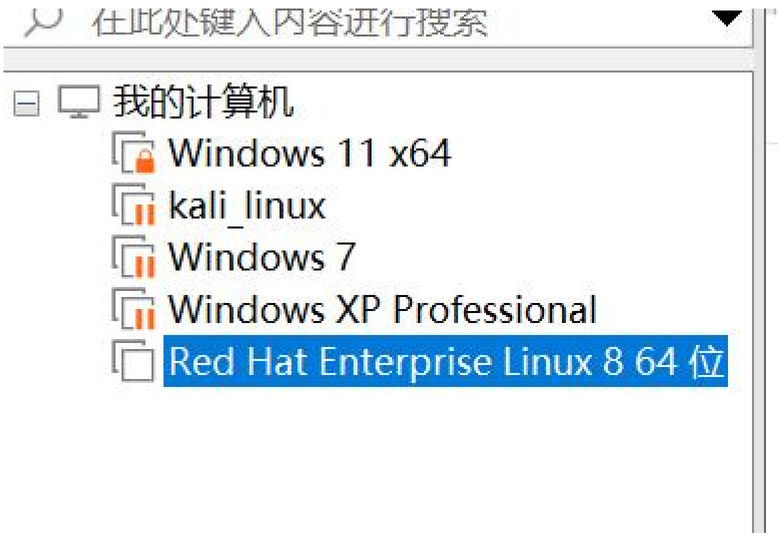
最大磁盘大小 (GB)(S): 20.0

针对 Red Hat Enterprise Linux 8 64 位 的建议大小: 20 GB

[ ] 立即分配所有磁盘空间(A)。
分配所有容量可以提高性能，但要求所有物理磁盘空间立即可用。如果不立即分配所有空间，虚拟磁盘的空间最初很小，会随着您向其中添加数据而不断变大。

( ) 将虚拟磁盘存储为单个文件(O)
[✓] 将虚拟磁盘拆分成多个文件(M)
拆分磁盘后，可以更轻松地在计算机之间移动虚拟机，但可能会降低大容量磁盘的性能。

基本虚拟机配置完成

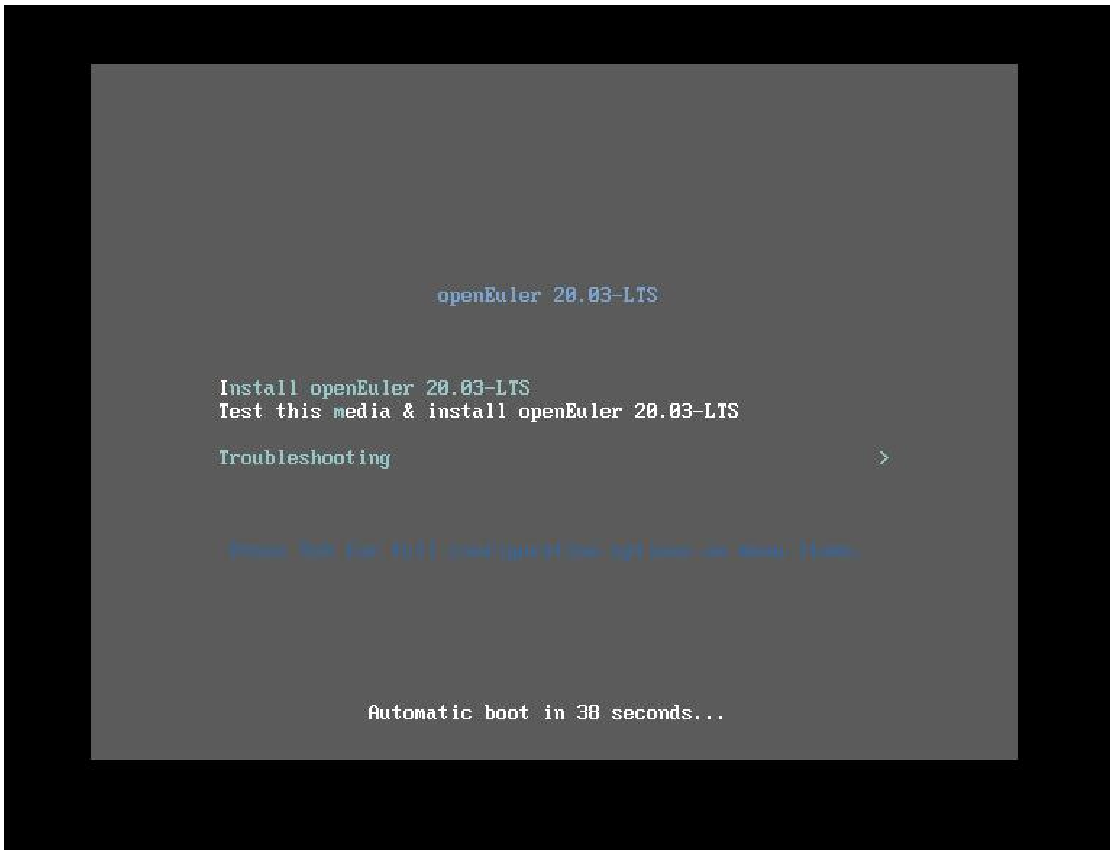


在此处键入内容进行搜索

我的计算机

- Windows 11 x64
- kali_linux
- Windows 7
- Windows XP Professional
- Red Hat Enterprise Linux 8 64 位

2、安装系统



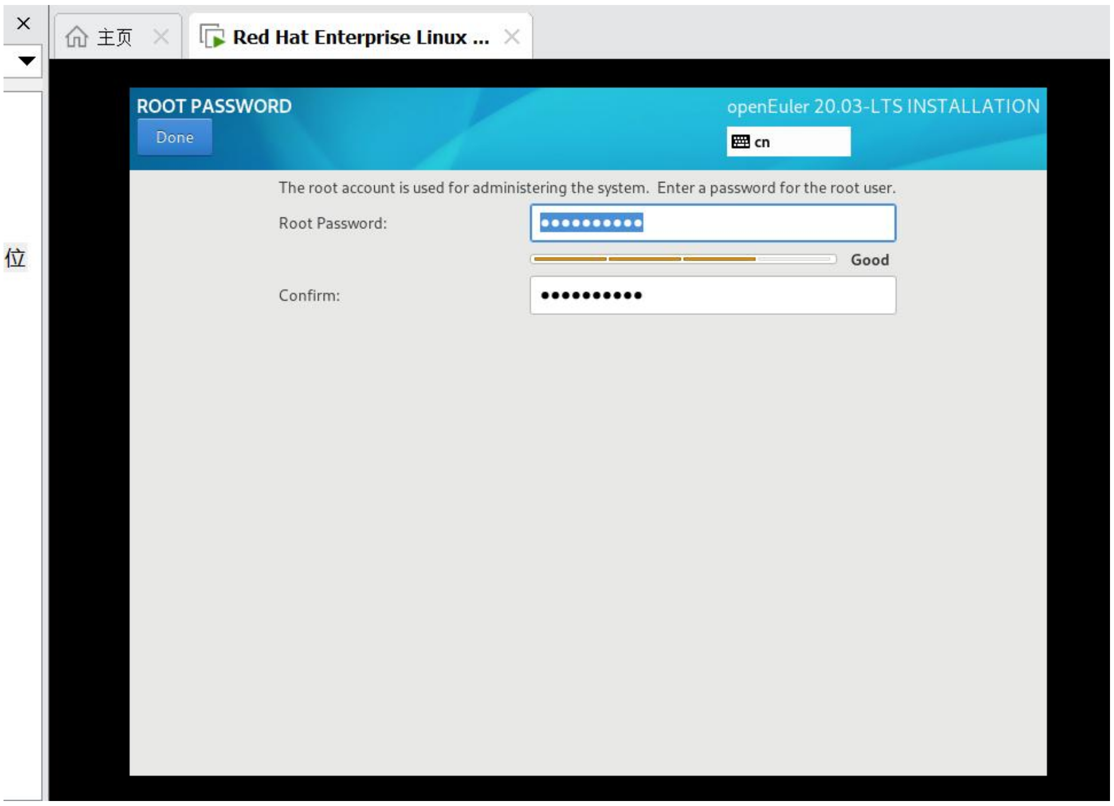
```
                    openEuler 20.03-LTS


      Install openEuler 20.03-LTS
      Test this media & install openEuler 20.03-LTS

      Troubleshooting                                          >


       Press Tab for full configuration options on menu items.



                 Automatic boot in 38 seconds...
```

启动虚拟机，选择 Install openEuler 20.03-LTS。

a)创建用户名

设密码

加载


主页
Red Hat Enterprise Linux ...
位
ROOT PASSWORD
Done
openEuler 20.03-LTS INSTALLATION
cn

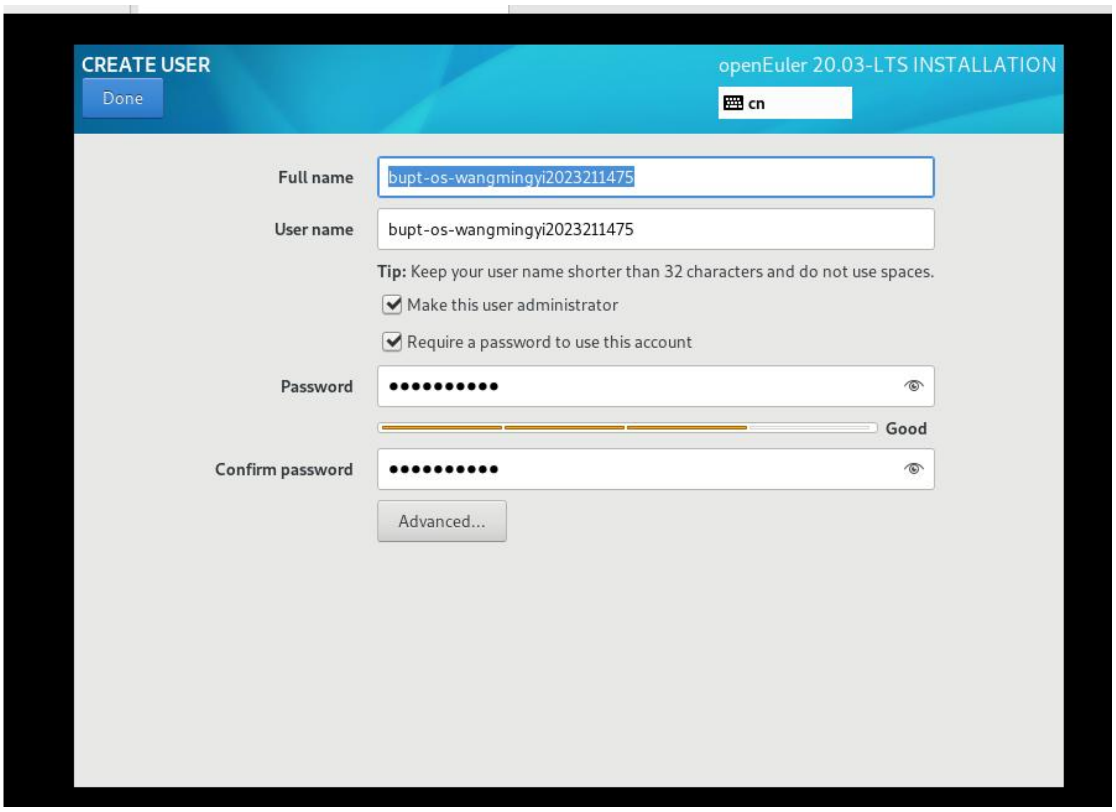
The root account is used for administering the system. Enter a password for the root user.

Root Password: ●●●●●●●●●●

Good

Confirm: ●●●●●●●●●●

设置用户名为 bupt-os-wangmingyi2023211475



Full name: bupt-os-wangmingyi2023211475

User name: bupt-os-wangmingyi2023211475

Tip: Keep your user name shorter than 32 characters and do not use spaces.

☑ Make this user administrator

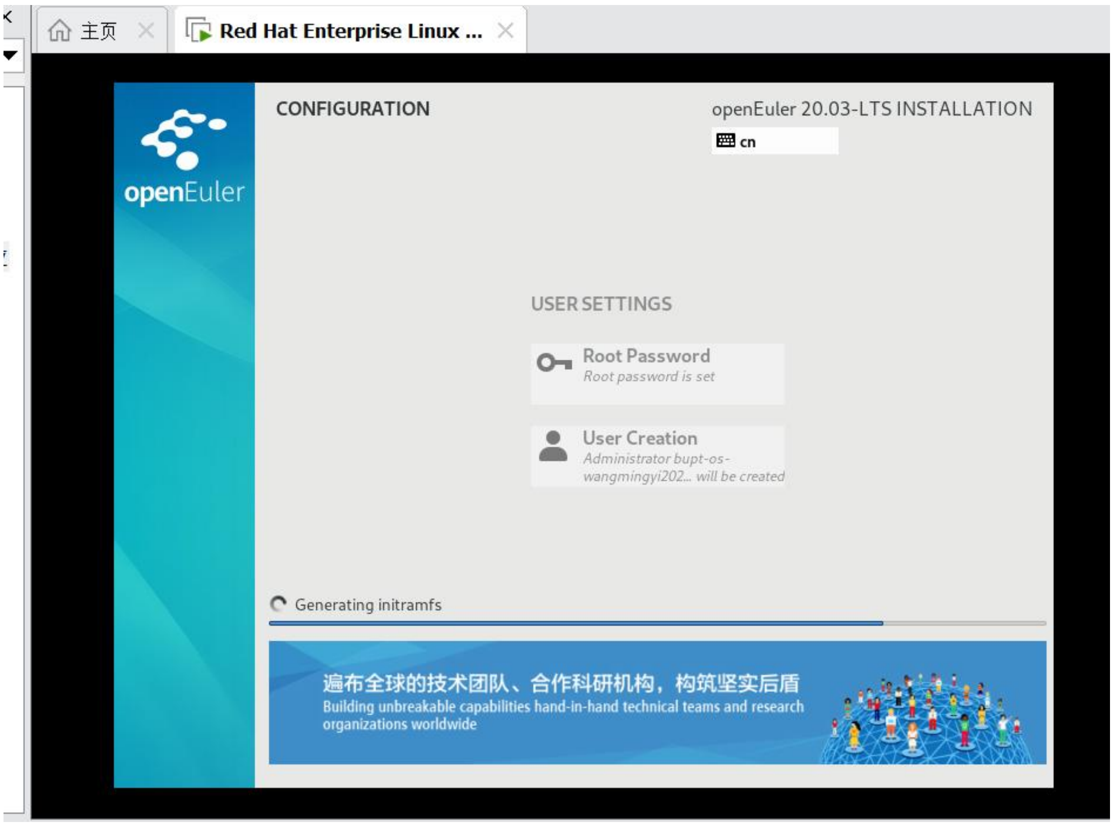
☑ Require a password to use this account

Password: ●●●●●●●●●●

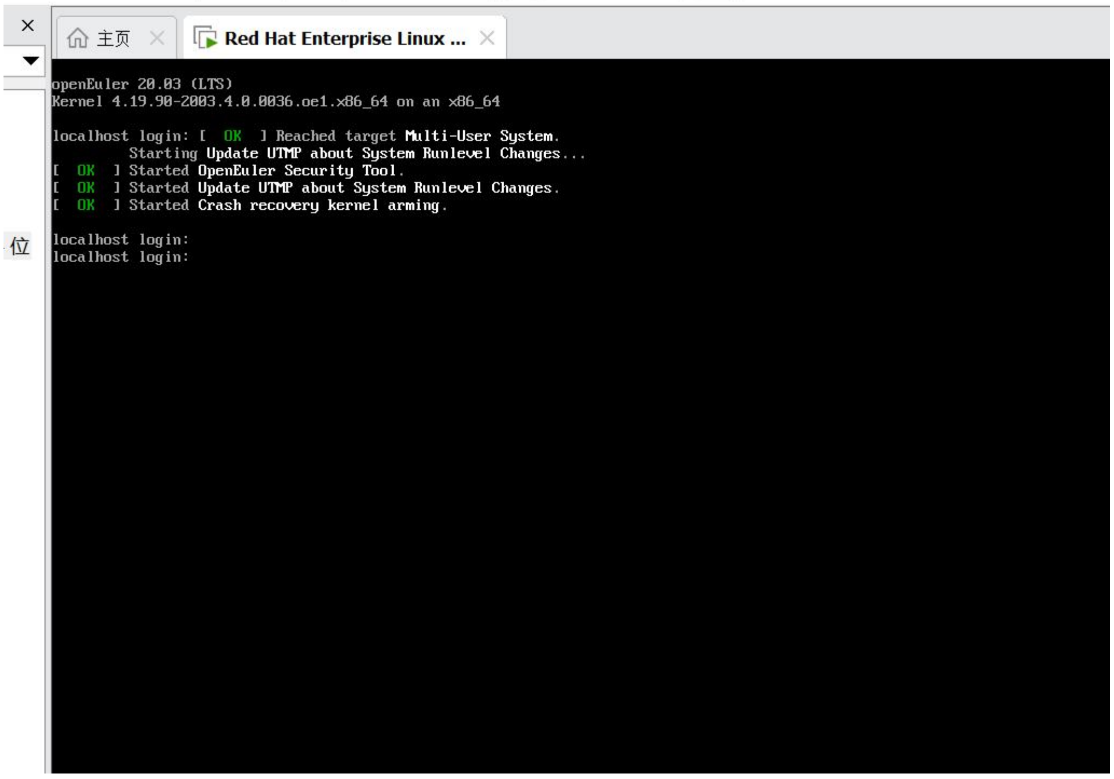
Good

Confirm password: ●●●●●●●●●●

加载

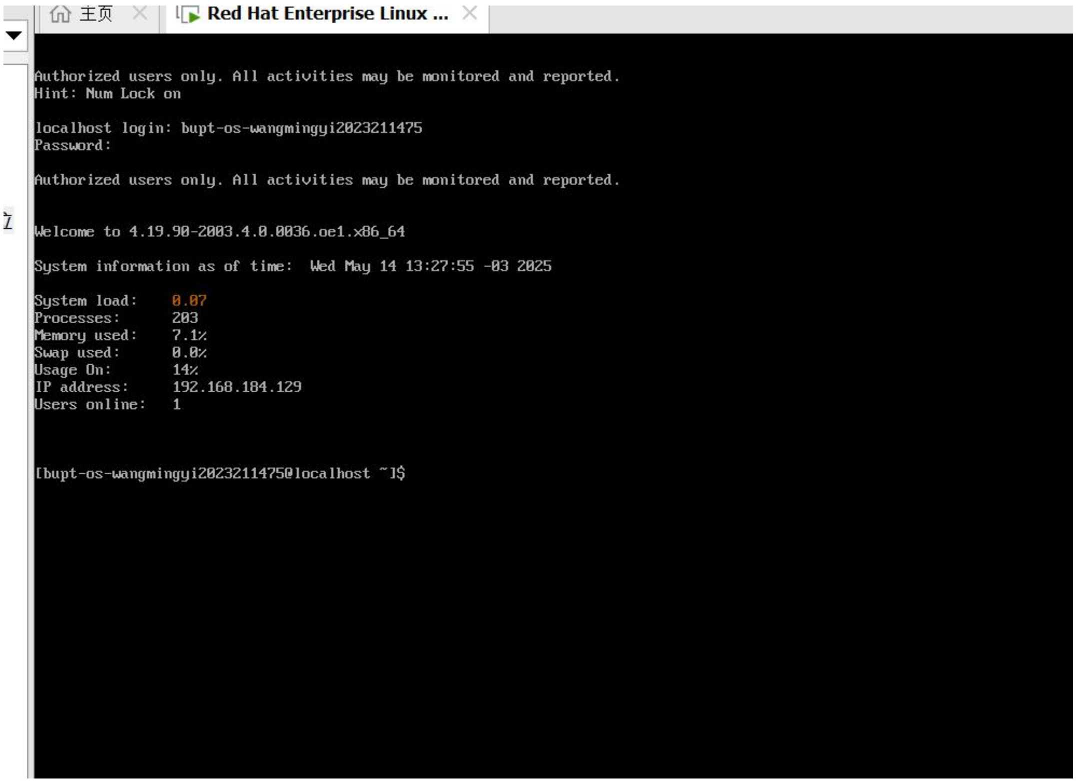

主页
Red Hat Enterprise Linux ...
openEuler
CONFIGURATION
openEuler 20.03-LTS INSTALLATION
cn
USER SETTINGS
Root Password
Root password is set
User Creation
Administrator bupt-os-wangmingyi202... will be created
Generating initramfs
遍布全球的技术团队、合作科研机构，构筑坚实后盾
Building unbreakable capabilities hand-in-hand technical teams and research organizations worldwide

安装好后登录



```
openEuler 20.03 (LTS)
Kernel 4.19.90-2003.4.0.0036.oe1.x86_64 on an x86_64

localhost login: [  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started OpenEuler Security Tool.
[  OK  ] Started Update UTMP about System Runlevel Changes.
[  OK  ] Started Crash recovery kernel arming.

localhost login:
localhost login:
```

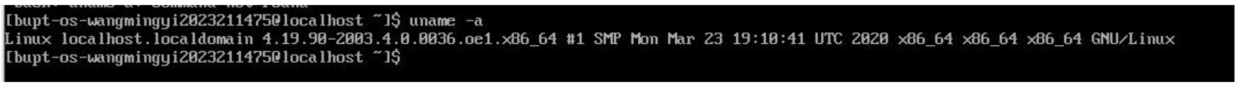
输入完用户名和密码，登陆成功



```
Authorized users only. All activities may be monitored and reported.
Hint: Num Lock on

localhost login: bupt-os-wangmingyi2023211475
Password:

Authorized users only. All activities may be monitored and reported.


Welcome to 4.19.90-2003.4.0.0036.oe1.x86_64

System information as of time:  Wed May 14 13:27:55 -03 2025

System load:    0.07
Processes:      203
Memory used:    7.1%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.184.129
Users online:   1


[bupt-os-wangmingyi2023211475@localhost ~]$
```

b) 执行 uname-a 指令，查看并记录内核版本、系统位数等信息.

Uname -a 指令截图如下：

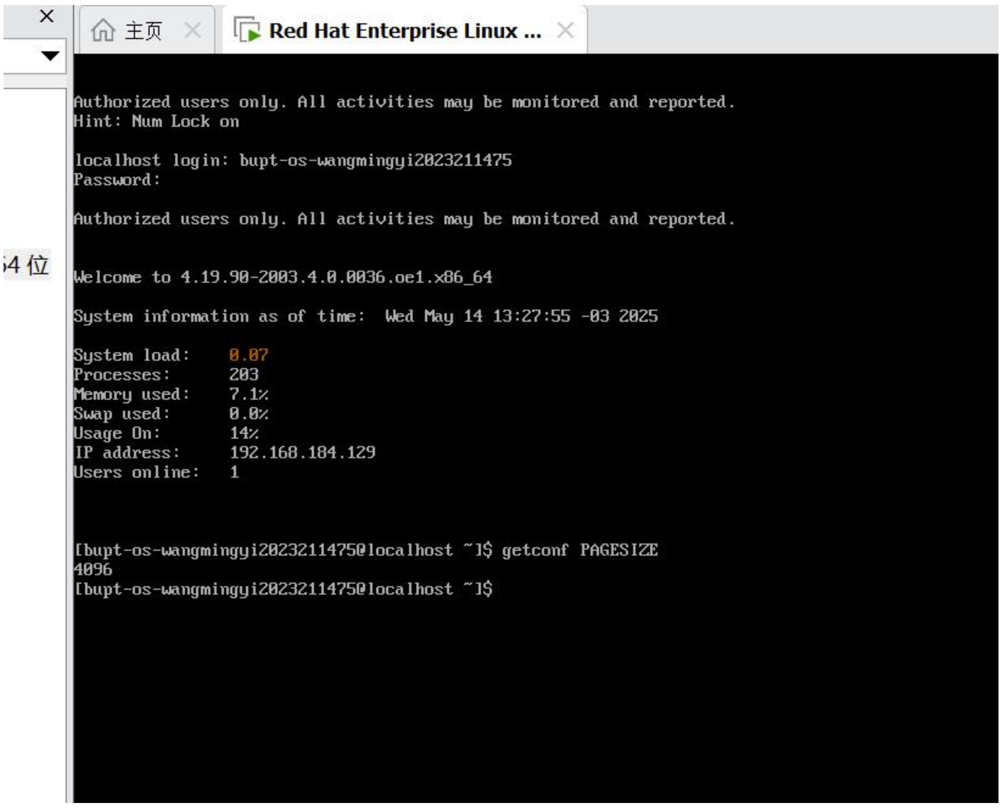


```
[bupt-os-wangmingyi2023211475@localhost ~]$ uname -a
Linux localhost.localdomain 4.19.90-2003.4.0.0036.oe1.x86_64 #1 SMP Mon Mar 23 19:10:41 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
[bupt-os-wangmingyi2023211475@localhost ~]$
```

c) 执行 getconf PAGESIZE 命令

getconf PAGESIZE 指令检查分页大小，为 4096。

命令执行结果如下：



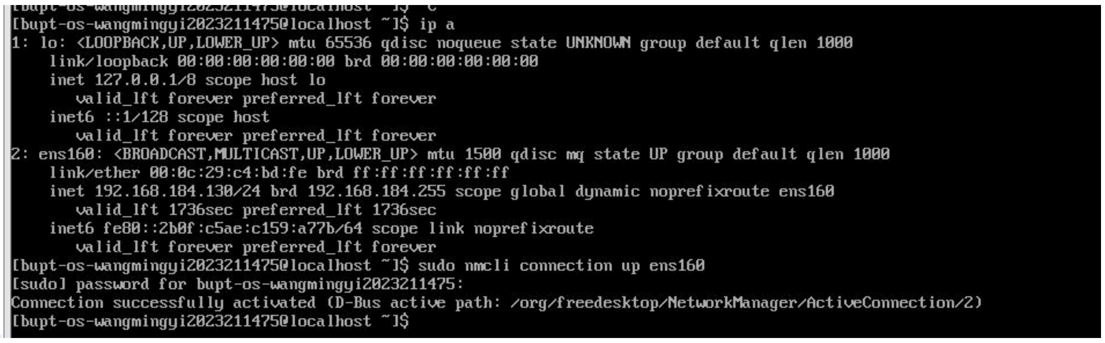
```
Authorized users only. All activities may be monitored and reported.
Hint: Num Lock on

localhost login: bupt-os-wangmingyi2023211475
Password:

Authorized users only. All activities may be monitored and reported.


Welcome to 4.19.90-2003.4.0.0036.oe1.x86_64

System information as of time:  Wed May 14 13:27:55 -03 2025

System load:    0.07
Processes:      203
Memory used:    7.1%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.184.129
Users online:   1


[bupt-os-wangmingyi2023211475@localhost ~]$ getconf PAGESIZE
4096
[bupt-os-wangmingyi2023211475@localhost ~]$
```

至此，已完成 openEuler 20.03-LTS 的安装

二、编译并更新内核

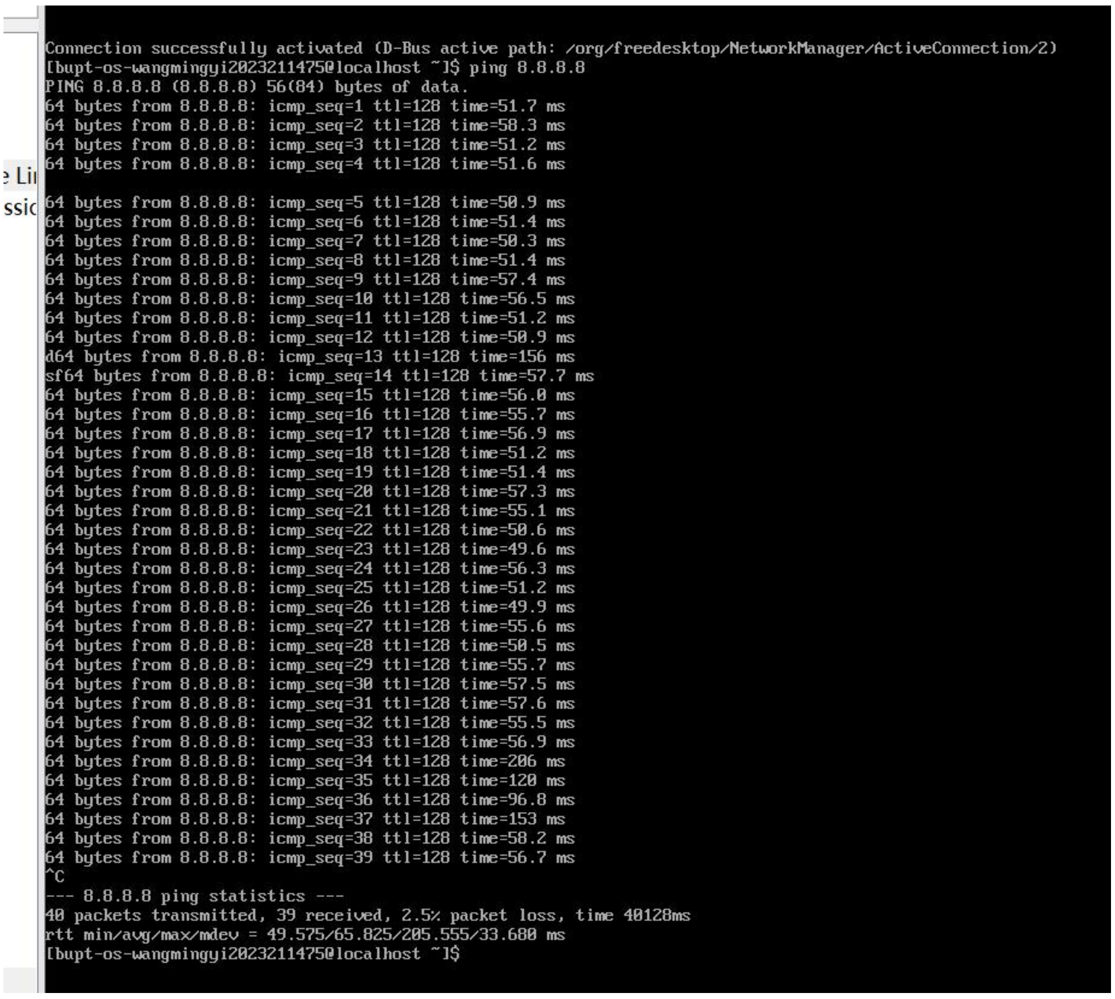
因为之后会下载 openEuler 内核源码，先查看虚拟机与网络的连接情况，输入 ip a，查看网卡

有 ens160 网卡，输入 sudo nmcli connection up ens160，连接



```
[bupt-os-wangmingyi2023211475@localhost ~]$ ^C
[bupt-os-wangmingyi2023211475@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:c4:bd:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.184.130/24 brd 192.168.184.255 scope global dynamic noprefixroute ens160
       valid_lft 1736sec preferred_lft 1736sec
    inet6 fe80::2b0f:c5ae:c159:a77b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[bupt-os-wangmingyi2023211475@localhost ~]$ sudo nmcli connection up ens160
[sudo] password for bupt-os-wangmingyi2023211475:
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/2)
[bupt-os-wangmingyi2023211475@localhost ~]$
```

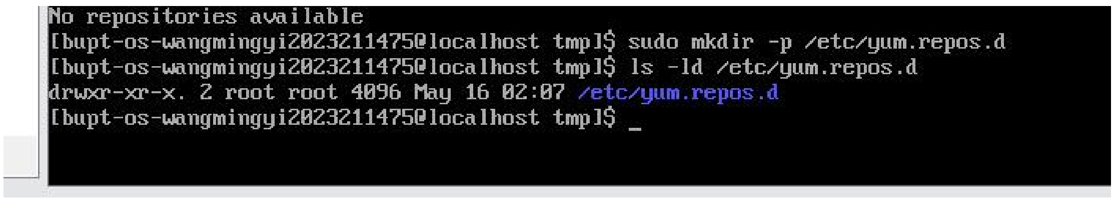
连接成功



```
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/2)
[bupt-os-wangmingyi2023211475@localhost ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=51.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=58.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=51.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=51.6 ms

64 bytes from 8.8.8.8: icmp_seq=5 ttl=128 time=50.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=128 time=51.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=128 time=50.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=128 time=51.4 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=128 time=57.4 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=128 time=56.5 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=128 time=51.2 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=128 time=50.9 ms
d64 bytes from 8.8.8.8: icmp_seq=13 ttl=128 time=156 ms
sf64 bytes from 8.8.8.8: icmp_seq=14 ttl=128 time=57.7 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=128 time=56.0 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=128 time=55.7 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=128 time=56.9 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=128 time=51.2 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=128 time=51.4 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=128 time=57.3 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=128 time=55.1 ms
64 bytes from 8.8.8.8: icmp_seq=22 ttl=128 time=50.6 ms
64 bytes from 8.8.8.8: icmp_seq=23 ttl=128 time=49.6 ms
64 bytes from 8.8.8.8: icmp_seq=24 ttl=128 time=56.3 ms
64 bytes from 8.8.8.8: icmp_seq=25 ttl=128 time=51.2 ms
64 bytes from 8.8.8.8: icmp_seq=26 ttl=128 time=49.9 ms
64 bytes from 8.8.8.8: icmp_seq=27 ttl=128 time=55.6 ms
64 bytes from 8.8.8.8: icmp_seq=28 ttl=128 time=50.5 ms
64 bytes from 8.8.8.8: icmp_seq=29 ttl=128 time=55.7 ms
64 bytes from 8.8.8.8: icmp_seq=30 ttl=128 time=57.5 ms
64 bytes from 8.8.8.8: icmp_seq=31 ttl=128 time=57.6 ms
64 bytes from 8.8.8.8: icmp_seq=32 ttl=128 time=55.5 ms
64 bytes from 8.8.8.8: icmp_seq=33 ttl=128 time=56.9 ms
64 bytes from 8.8.8.8: icmp_seq=34 ttl=128 time=206 ms
64 bytes from 8.8.8.8: icmp_seq=35 ttl=128 time=120 ms
64 bytes from 8.8.8.8: icmp_seq=36 ttl=128 time=96.8 ms
64 bytes from 8.8.8.8: icmp_seq=37 ttl=128 time=153 ms
64 bytes from 8.8.8.8: icmp_seq=38 ttl=128 time=58.2 ms
64 bytes from 8.8.8.8: icmp_seq=39 ttl=128 time=56.7 ms
^C
--- 8.8.8.8 ping statistics ---
40 packets transmitted, 39 received, 2.5% packet loss, time 40128ms
rtt min/avg/max/mdev = 49.575/65.825/205.555/33.680 ms
[bupt-os-wangmingyi2023211475@localhost ~]$
```

尝试 ping 8.8.8.8 测试

发现连接成功，按 Ctrl+C 结束 ping



```
No repositories available
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo mkdir -p /etc/yum.repos.d
[bupt-os-wangmingyi2023211475@localhost tmp]$ ls -ld /etc/yum.repos.d
drwxr-xr-x. 2 root root 4096 May 16 02:07 /etc/yum.repos.d
[bupt-os-wangmingyi2023211475@localhost tmp]$ _
```

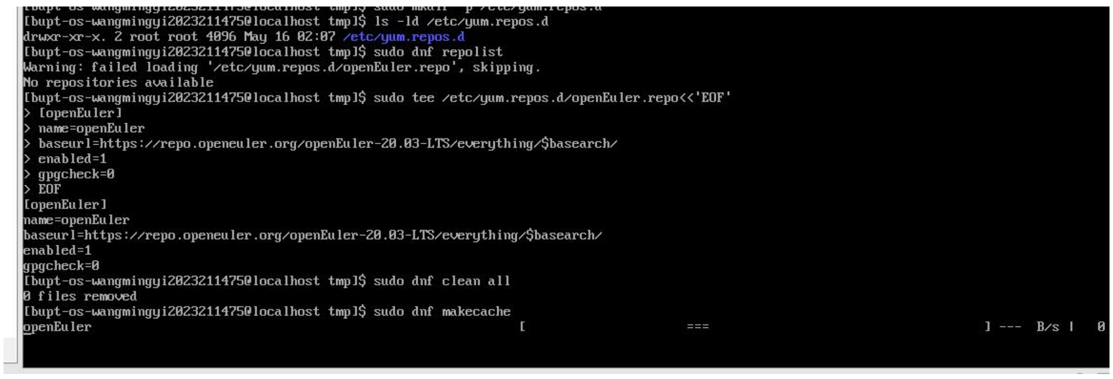
为确保 openEuler 系统能够正常获取内核编译所需的软件包，使用命令 sudo vi /etc/yum.repos.d/openEuler.repo 手动创建了 YUM 仓库配置文件 /etc/yum.repos.d/openEuler.repo

输入下面的指令编辑 openEuler.repo 文件

```
[openEuler]
name=openEuler
baseurl=https://mirrors.aliyun.com/openeuler/openEuler-20.03
-LTS/OS/$basearch/
enabled=1
gpgcheck=0
```

使用阿里云的 openEuler 镜像站。

编辑结果如下图



```
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo mkdir -p /etc/yum.repos.d
[bupt-os-wangmingyi2023211475@localhost tmp]$ ls -ld /etc/yum.repos.d
drwxr-xr-x. 2 root root 4096 May 16 02:07 /etc/yum.repos.d
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo dnf repolist
Warning: failed loading '/etc/yum.repos.d/openEuler.repo', skipping.
No repositories available
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo tee /etc/yum.repos.d/openEuler.repo<<'EOF'
> [openEuler]
> name=openEuler
> baseurl=https://repo.openeuler.org/openEuler-20.03-LTS/everything/$basearch/
> enabled=1
> gpgcheck=0
> EOF
[openEuler]
name=openEuler
baseurl=https://repo.openeuler.org/openEuler-20.03-LTS/everything/$basearch/
enabled=1
gpgcheck=0
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo dnf clean all
0 files removed
[bupt-os-wangmingyi2023211475@localhost tmp]$ sudo dnf makecache
openEuler                                        [                    ===                    ] --- B/s |   0
```

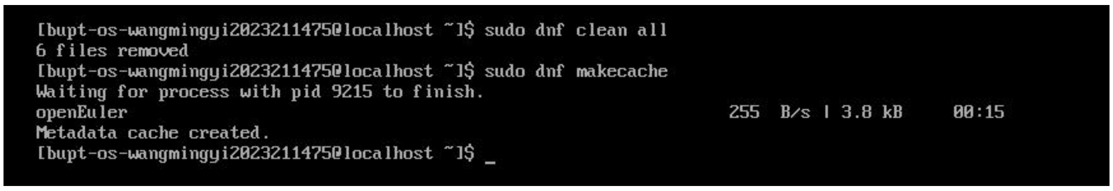
创建 openEuler 基础仓库文件

保险起见用 sudo dnf clean all，清除旧的缓存

重建仓库元数据：sudo dnf makecache

用 sudo dnf repolist 验证新仓库是否启用成功并可以使用

如下图，配置仓库完成



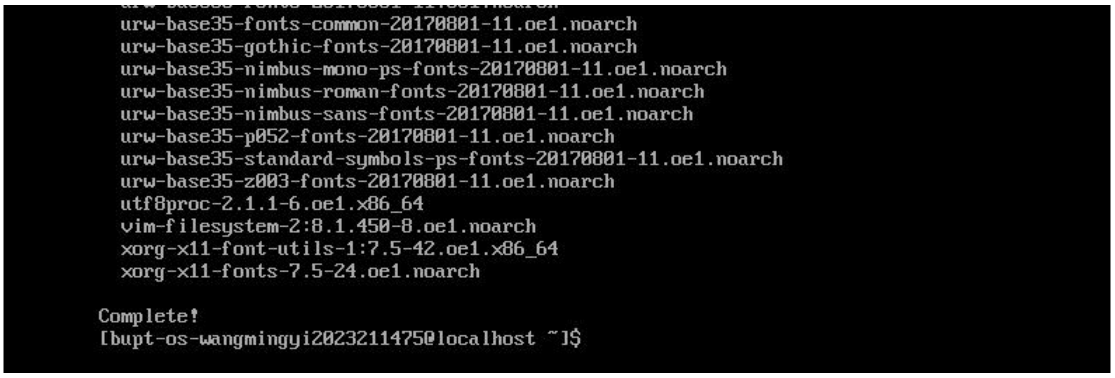
```
[bupt-os-wangmingyi2023211475@localhost ~]$ sudo dnf clean all
6 files removed
[bupt-os-wangmingyi2023211475@localhost ~]$ sudo dnf makecache
Waiting for process with pid 9215 to finish.
openEuler                                                255  B/s | 3.8 kB     00:15
Metadata cache created.
[bupt-os-wangmingyi2023211475@localhost ~]$ _
```

安装必要的依赖包

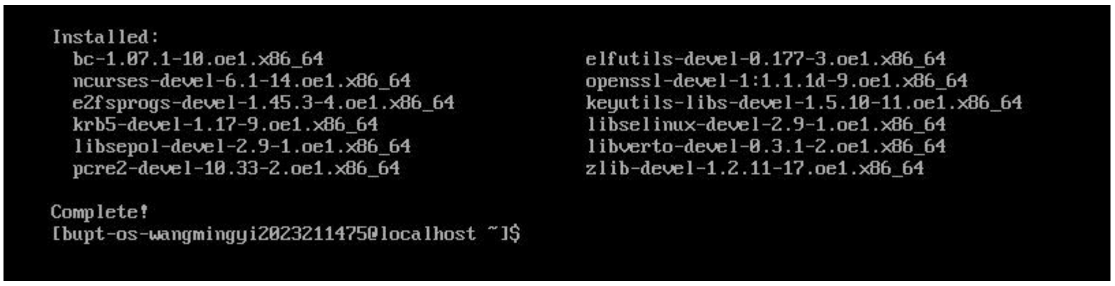
sudo dnf groupinstall "Development Tools" -y

sudo dnf install ncurses-devel bc openssl-devel elfutils-libelf-devel flex bison -y

完成情况如下图



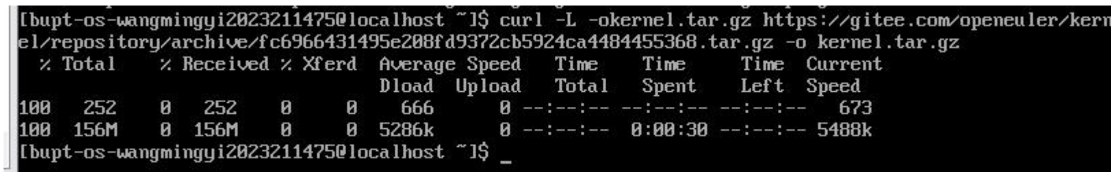
```
        urw-base35-fonts-common-20170801-11.oe1.noarch
        urw-base35-gothic-fonts-20170801-11.oe1.noarch
        urw-base35-nimbus-mono-ps-fonts-20170801-11.oe1.noarch
        urw-base35-nimbus-roman-fonts-20170801-11.oe1.noarch
        urw-base35-nimbus-sans-fonts-20170801-11.oe1.noarch
        urw-base35-p052-fonts-20170801-11.oe1.noarch
        urw-base35-standard-symbols-ps-fonts-20170801-11.oe1.noarch
        urw-base35-z003-fonts-20170801-11.oe1.noarch
        utf8proc-2.1.1-6.oe1.x86_64
        vim-filesystem-2:8.1.450-8.oe1.noarch
        xorg-x11-font-utils-1:7.5-42.oe1.x86_64
        xorg-x11-fonts-7.5-24.oe1.noarch

      Complete!
      [bupt-os-wangmingyi2023211475@localhost ~]$
```



```
Installed:
  bc-1.07.1-10.oe1.x86_64                      elfutils-devel-0.177-3.oe1.x86_64
  ncurses-devel-6.1-14.oe1.x86_64              openssl-devel-1:1.1.1d-9.oe1.x86_64
  e2fsprogs-devel-1.45.3-4.oe1.x86_64          keyutils-libs-devel-1.5.10-11.oe1.x86_64
  krb5-devel-1.17-9.oe1.x86_64                 libselinux-devel-2.9-1.oe1.x86_64
  libsepol-devel-2.9-1.oe1.x86_64              libverto-devel-0.3.1-2.oe1.x86_64
  pcre2-devel-10.33-2.oe1.x86_64               zlib-devel-1.2.11-17.oe1.x86_64

Complete!
[bupt-os-wangmingyi2023211475@localhost ~]$
```

完成上面两条语句。

用 curl 下载 fc6966431495e208fd9372cb5924ca4484455368.tar.gz 包用于更新内核，如下图

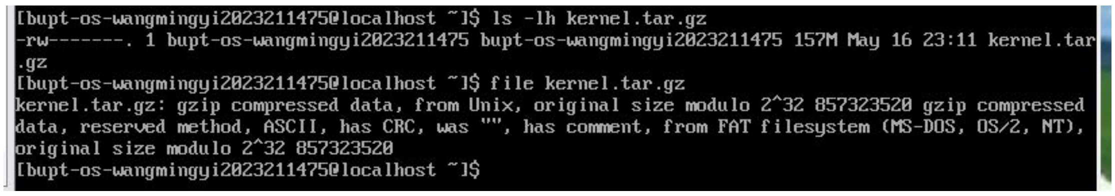


```
[bupt-os-wangmingyi2023211475@localhost ~]$ curl -L -okernel.tar.gz https://gitee.com/openeuler/kern
el/repository/archive/fc6966431495e208fd9372cb5924ca4484455368.tar.gz -o kernel.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   252    0   252    0     0    666      0 --:--:-- --:--:-- --:--:--   673
100  156M    0  156M    0     0  5286k      0 --:--:--  0:00:30 --:--:-- 5488k
[bupt-os-wangmingyi2023211475@localhost ~]$ _
```

验证下载完成：

用 ls -lh kernel.tar.gz

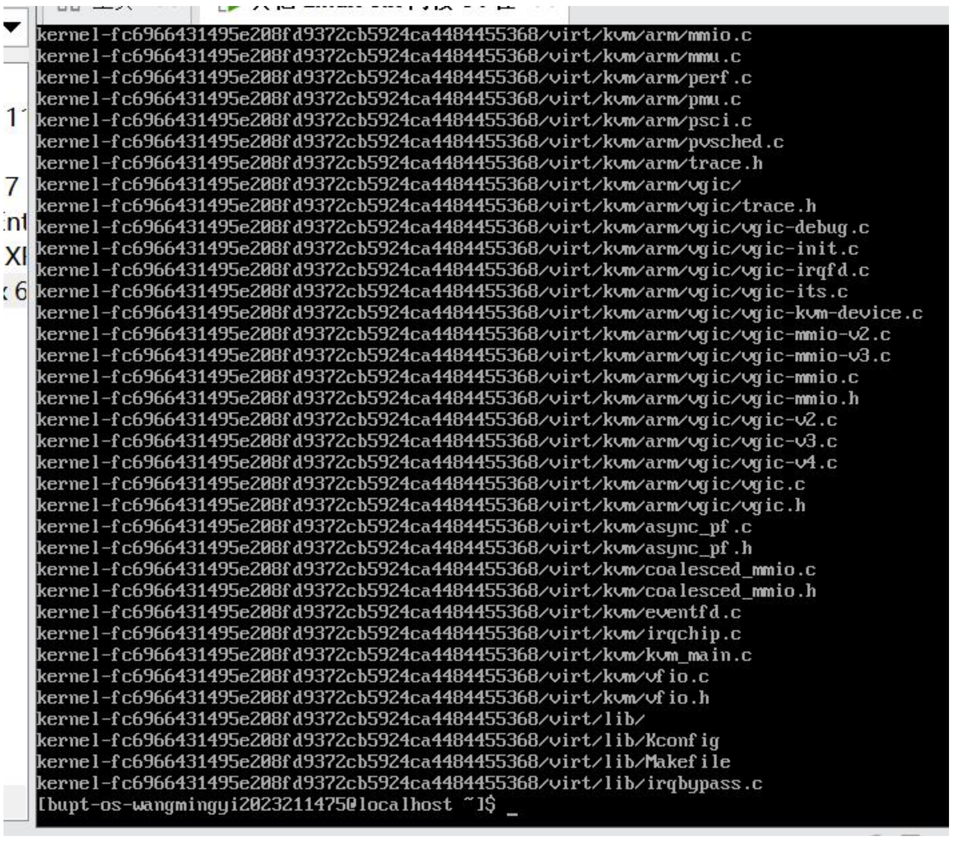
File kernel.tar.gz 验证，如下图



```
[bupt-os-wangmingyi2023211475@localhost ~]$ ls -lh kernel.tar.gz
-rw-------. 1 bupt-os-wangmingyi2023211475 bupt-os-wangmingyi2023211475 157M May 16 23:11 kernel.tar
.gz
[bupt-os-wangmingyi2023211475@localhost ~]$ file kernel.tar.gz
kernel.tar.gz: gzip compressed data, from Unix, original size modulo 2^32 857323520 gzip compressed
data, reserved method, ASCII, has CRC, was "", has comment, from FAT filesystem (MS-DOS, OS/2, NT),
original size modulo 2^32 857323520
[bupt-os-wangmingyi2023211475@localhost ~]$
```

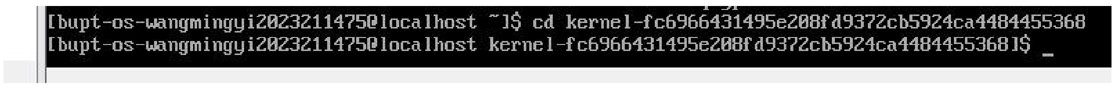
解压 fc6966431495e208fd9372cb5924ca4484455368.tar.gz 包

解压文件：tar -xzvf kernel.tar.gz，结果如下



```
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/mmio.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/mmu.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/perf.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/pmu.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/psci.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/pvsched.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/trace.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/trace.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-debug.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-init.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-irqfd.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-its.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-kvm-device.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-mmio-v2.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-mmio-v3.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-mmio.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-mmio.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-v2.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-v3.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic-v4.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/arm/vgic/vgic.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/async_pf.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/async_pf.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/coalesced_mmio.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/coalesced_mmio.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/eventfd.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/irqchip.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/kvm_main.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/vfio.c
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/kvm/vfio.h
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/lib/
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/lib/Kconfig
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/lib/Makefile
kernel-fc6966431495e208fd9372cb5924ca4484455368/virt/lib/irqbypass.c
[bupt-os-wangmingyi2023211475@localhost ~]$ _
```

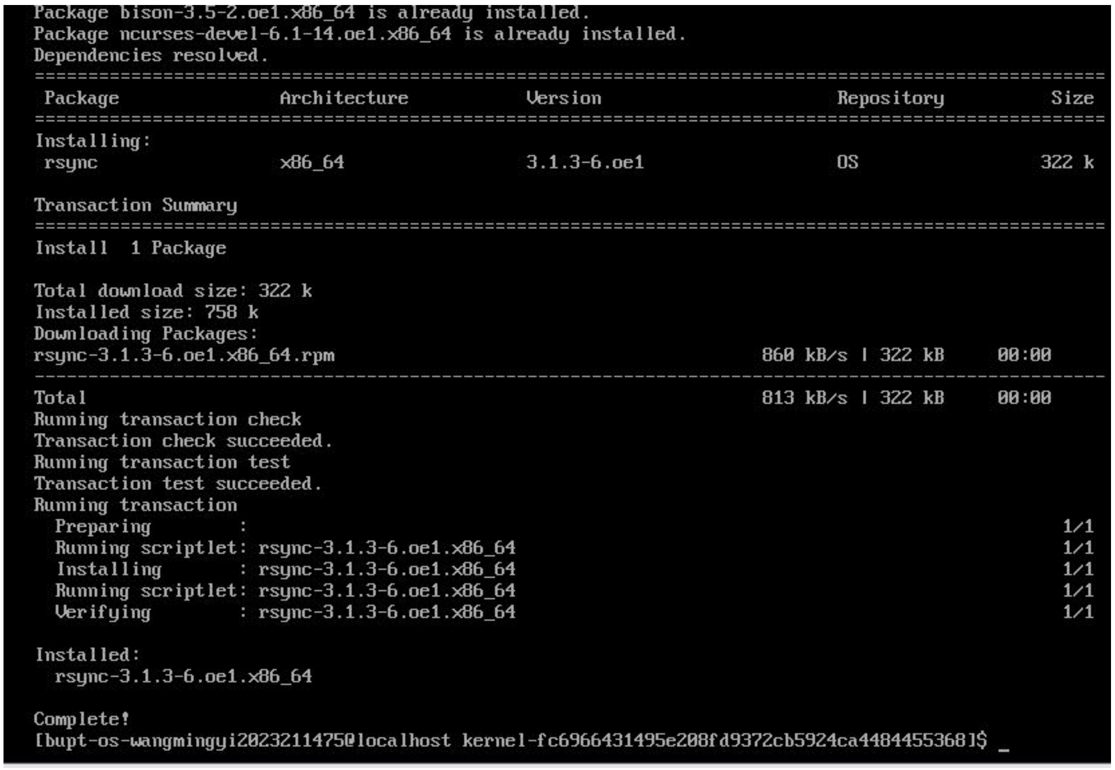
进入 kernel-fc6966431495e208fd9372cb5924ca4484455368 目录



```
[bupt-os-wangmingyi2023211475@localhost ~]$ cd kernel-fc6966431495e208fd9372cb5924ca4484455368
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ _
```

安装编译依赖工具：sudo dnf install -y bc openssl-devel elfutils-libelf-devel flex bison ncurses-devel rsync

结果如下

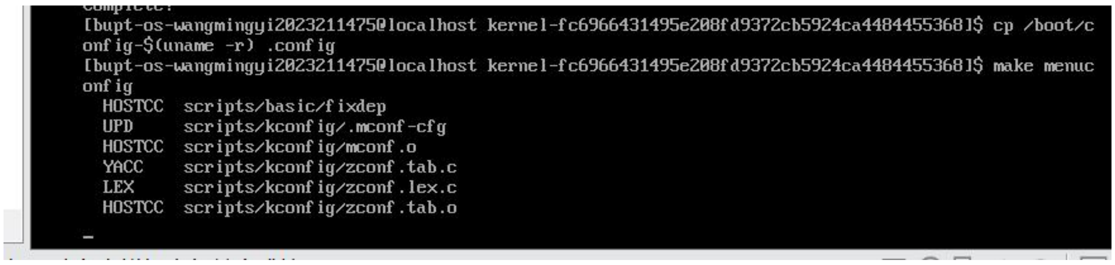


```
Package bison-3.5-2.oe1.x86_64 is already installed.
Package ncurses-devel-6.1-14.oe1.x86_64 is already installed.
Dependencies resolved.
================================================================================================
 Package              Architecture          Version                      Repository         Size
================================================================================================
Installing:
 rsync                x86_64                3.1.3-6.oe1                  OS                322 k

Transaction Summary
================================================================================================
Install  1 Package

Total download size: 322 k
Installed size: 758 k
Downloading Packages:
rsync-3.1.3-6.oe1.x86_64.rpm                                      860 kB/s | 322 kB     00:00
------------------------------------------------------------------------------------------------
Total                                                             813 kB/s | 322 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                         1/1
  Running scriptlet: rsync-3.1.3-6.oe1.x86_64                                                1/1
  Installing       : rsync-3.1.3-6.oe1.x86_64                                                1/1
  Running scriptlet: rsync-3.1.3-6.oe1.x86_64                                                1/1
  Verifying        : rsync-3.1.3-6.oe1.x86_64                                                1/1

Installed:
  rsync-3.1.3-6.oe1.x86_64

Complete!
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ _
```

配置内核：

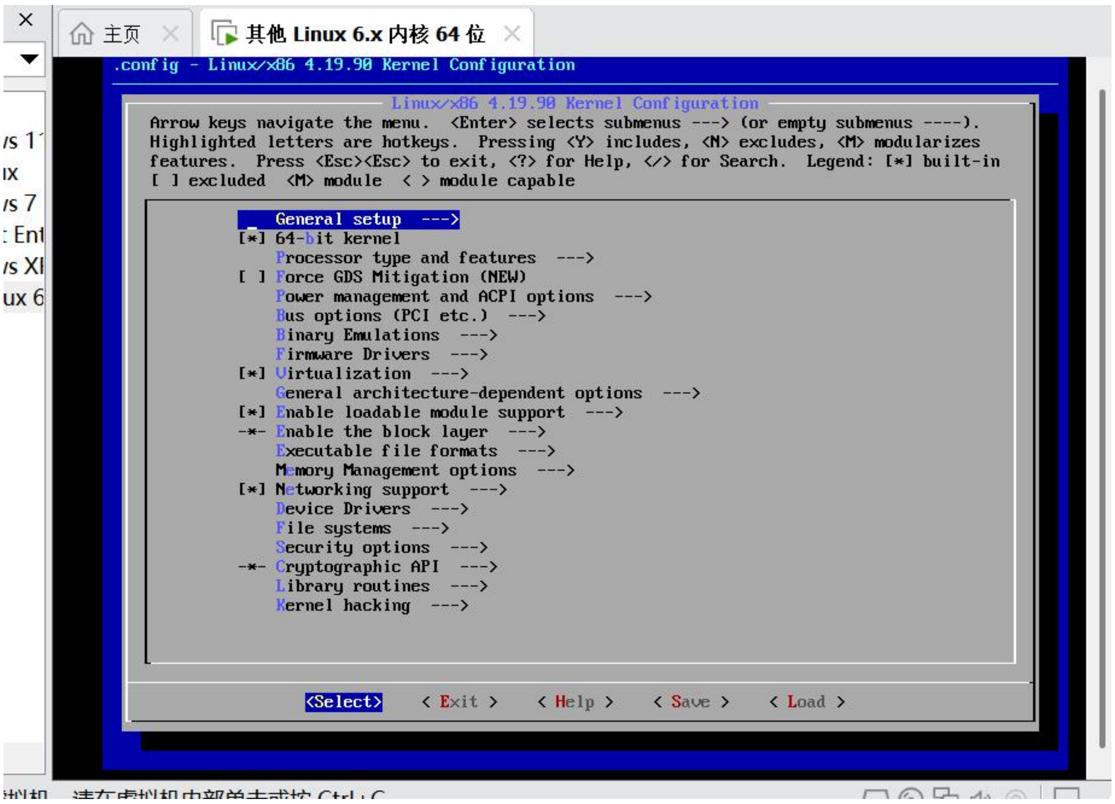
cp /boot/config-$(uname -r) .config

make menuconfig



```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ cp /boot/c
onfig-$(uname -r) .config
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ make menuc
onfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/.mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
_
```

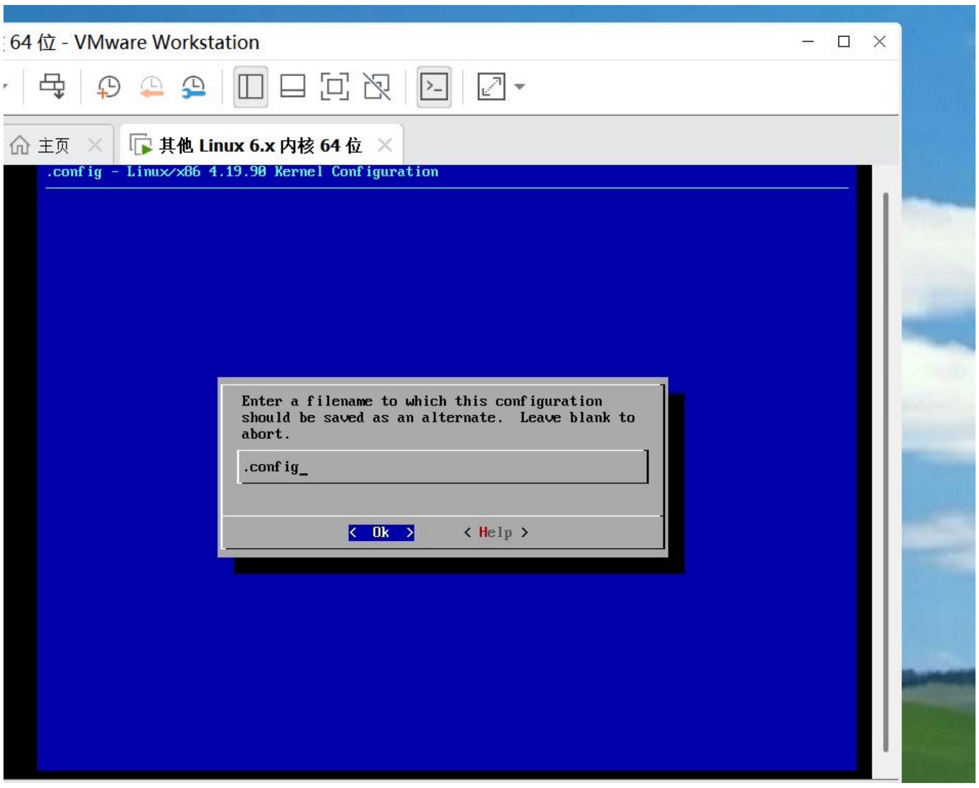
然后会弹出下图所示的界面，开始配置内核



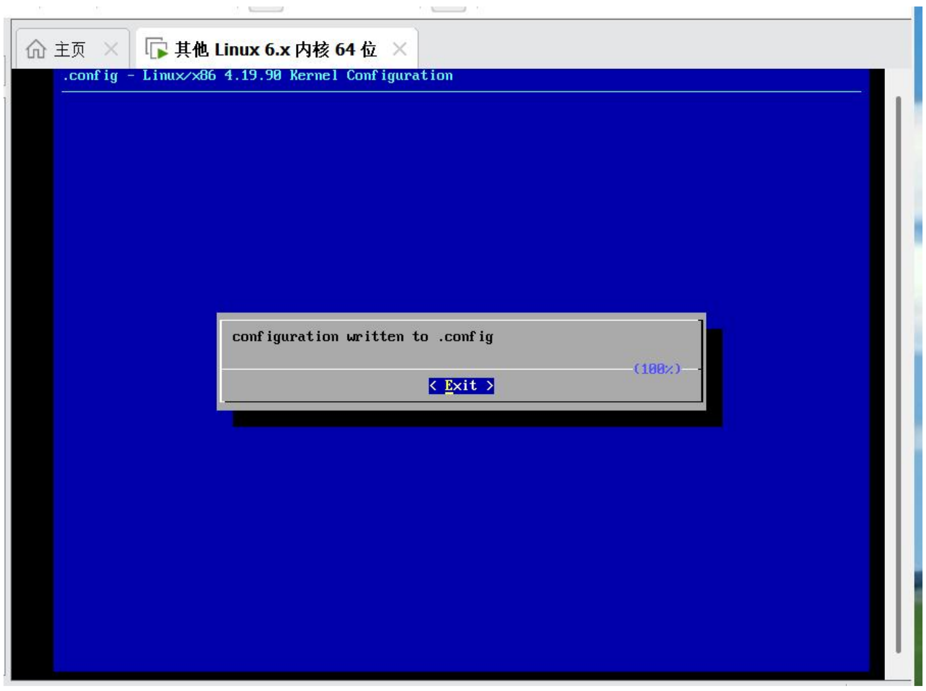
```
.config - Linux/x86 4.19.90 Kernel Configuration
                    Linux/x86 4.19.90 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable

           General setup  --->
       [*] 64-bit kernel
           Processor type and features  --->
       [ ] Force GDS Mitigation (NEW)
           Power management and ACPI options  --->
           Bus options (PCI etc.)  --->
           Binary Emulations  --->
           Firmware Drivers  --->
       [*] Virtualization  --->
           General architecture-dependent options  --->
       [*] Enable loadable module support  --->
       -*- Enable the block layer  --->
           Executable file formats  --->
           Memory Management options  --->
       [*] Networking support  --->
           Device Drivers  --->
           File systems  --->
           Security options  --->
       -*- Cryptographic API  --->
           Library routines  --->
           Kernel hacking  --->

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

先 Load 载入原始.config 配置

然后选择 Save，生成了一个.config 文件。



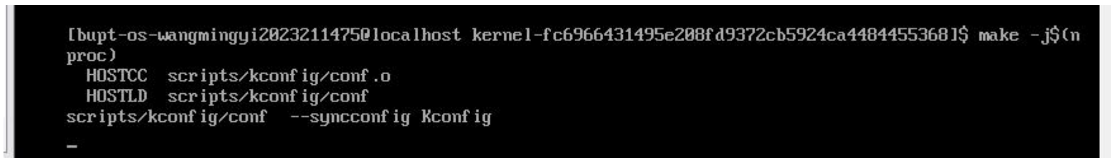
```
.config - Linux/x86 4.19.90 Kernel Configuration

        Enter a filename to which this configuration
        should be saved as an alternate.  Leave blank to
        abort.
        .config_

                  <  Ok  >        < Help >
```



```
.config - Linux/x86 4.19.90 Kernel Configuration

configuration written to .config
                                        (100%)
                  < Exit >
```

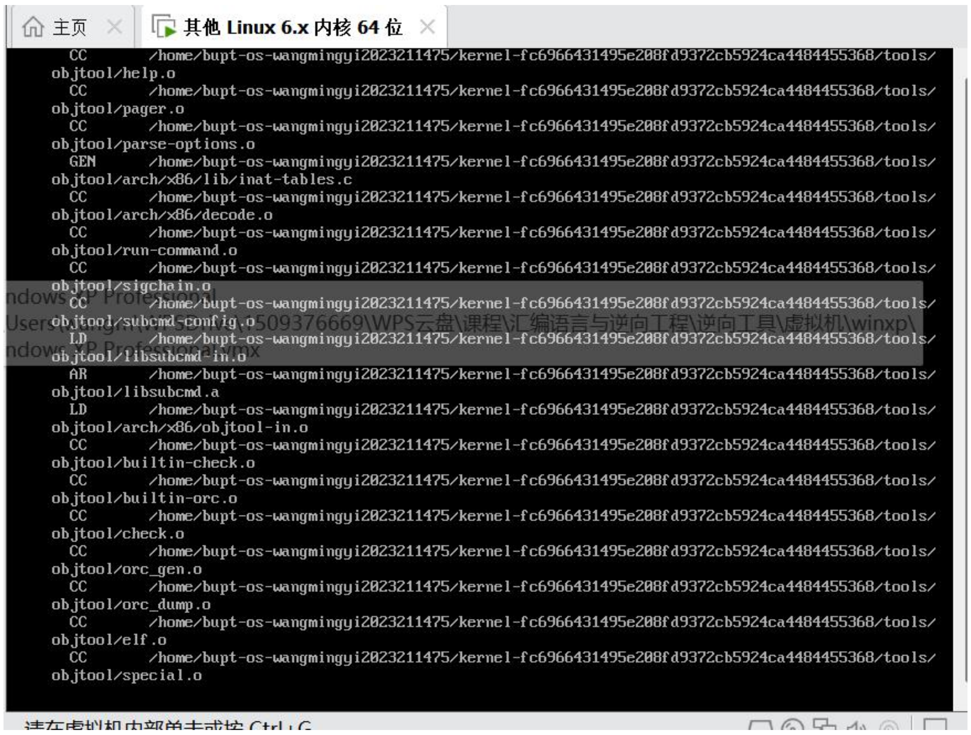
完成内核配置，保存配置用于后续的内核编译操作

编译内核，执行命令 make -j$(nproc)，如下图



```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ make -j$(n
proc)
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --syncconfig Kconfig
_
```

等待编译完成~

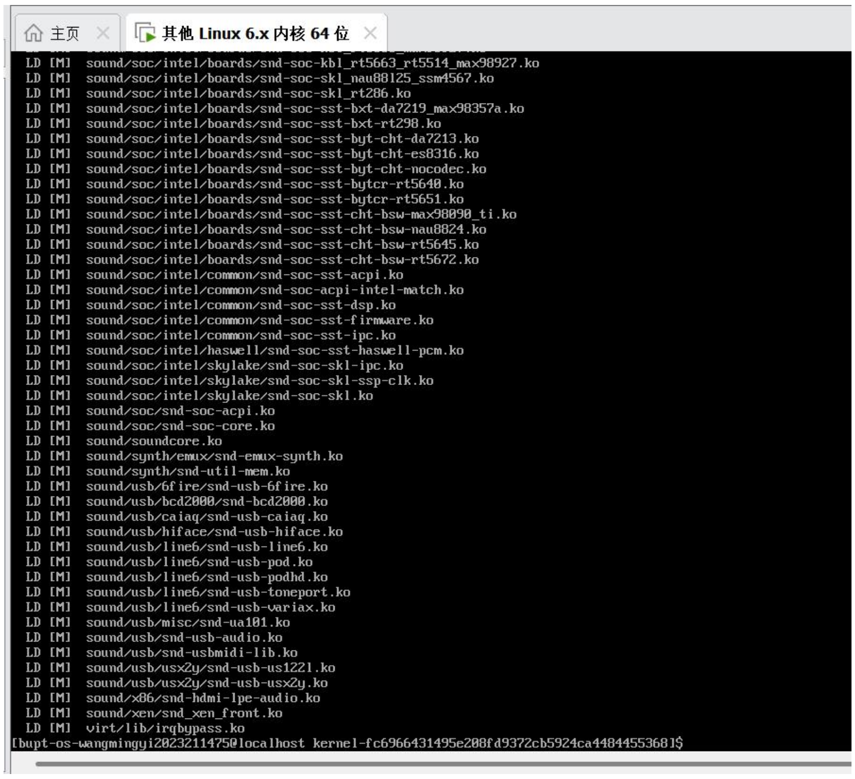


```
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/help.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/pager.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/parse-options.o
  GEN     /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/arch/x86/lib/inat-tables.c
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/arch/x86/decode.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/run-command.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/sigchain.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/subcmd-config.o
  LD      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/libsubcmd-in.o
  AR      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/libsubcmd.a
  LD      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/arch/x86/objtool-in.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/builtin-check.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/builtin-orc.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/check.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/orc_gen.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/orc_dump.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/elf.o
  CC      /home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368/tools/
objtool/special.o
```

ndows XP Professional
Users\...509376669\WPS云盘\课程\汇编语言与逆向工程\逆向工具\虚拟机\winxp\
ndows XP Professional.vmx

请在虚拟机内部单击或按 Ctrl+G

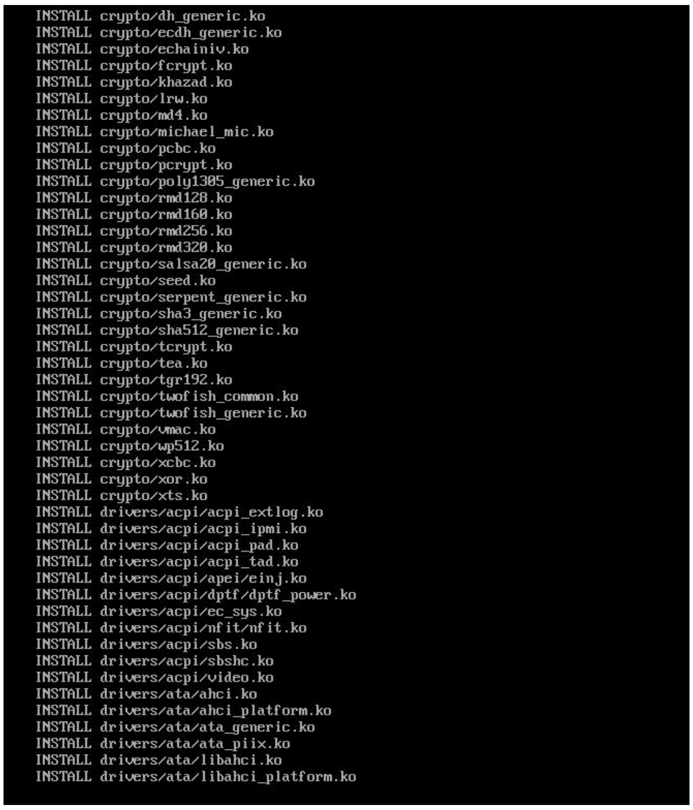
编译完成：



```
  LD [M]  sound/soc/intel/boards/snd-soc-kbl_rt5663_rt5514_max98927.ko
  LD [M]  sound/soc/intel/boards/snd-soc-skl_nau88125_ssm4567.ko
  LD [M]  sound/soc/intel/boards/snd-soc-skl_rt286.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bxt-da7219_max98357a.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bxt-rt298.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-byt-cht-da7213.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-byt-cht-es8316.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-byt-cht-nocodec.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5640.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5651.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-max98090_ti.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-nau8824.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5645.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5672.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-acpi.ko
  LD [M]  sound/soc/intel/common/snd-soc-acpi-intel-match.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-dsp.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-firmware.ko
  LD [M]  sound/soc/intel/common/snd-soc-sst-ipc.ko
  LD [M]  sound/soc/intel/haswell/snd-soc-sst-haswell-pcm.ko
  LD [M]  sound/soc/intel/skylake/snd-soc-skl-ipc.ko
  LD [M]  sound/soc/intel/skylake/snd-soc-skl-ssp-clk.ko
  LD [M]  sound/soc/intel/skylake/snd-soc-skl.ko
  LD [M]  sound/soc/snd-soc-acpi.ko
  LD [M]  sound/soc/snd-soc-core.ko
  LD [M]  sound/soundcore.ko
  LD [M]  sound/synth/emux/snd-emux-synth.ko
  LD [M]  sound/synth/snd-util-mem.ko
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  LD [M]  sound/usb/bcd2000/snd-bcd2000.ko
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  LD [M]  sound/usb/line6/snd-usb-line6.ko
  LD [M]  sound/usb/line6/snd-usb-pod.ko
  LD [M]  sound/usb/line6/snd-usb-podhd.ko
  LD [M]  sound/usb/line6/snd-usb-toneport.ko
  LD [M]  sound/usb/line6/snd-usb-variax.ko
  LD [M]  sound/usb/misc/snd-ua101.ko
  LD [M]  sound/usb/snd-usb-audio.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
  LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
  LD [M]  sound/xen/snd_xen_front.ko
  LD [M]  virt/lib/irqbypass.ko
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$
```

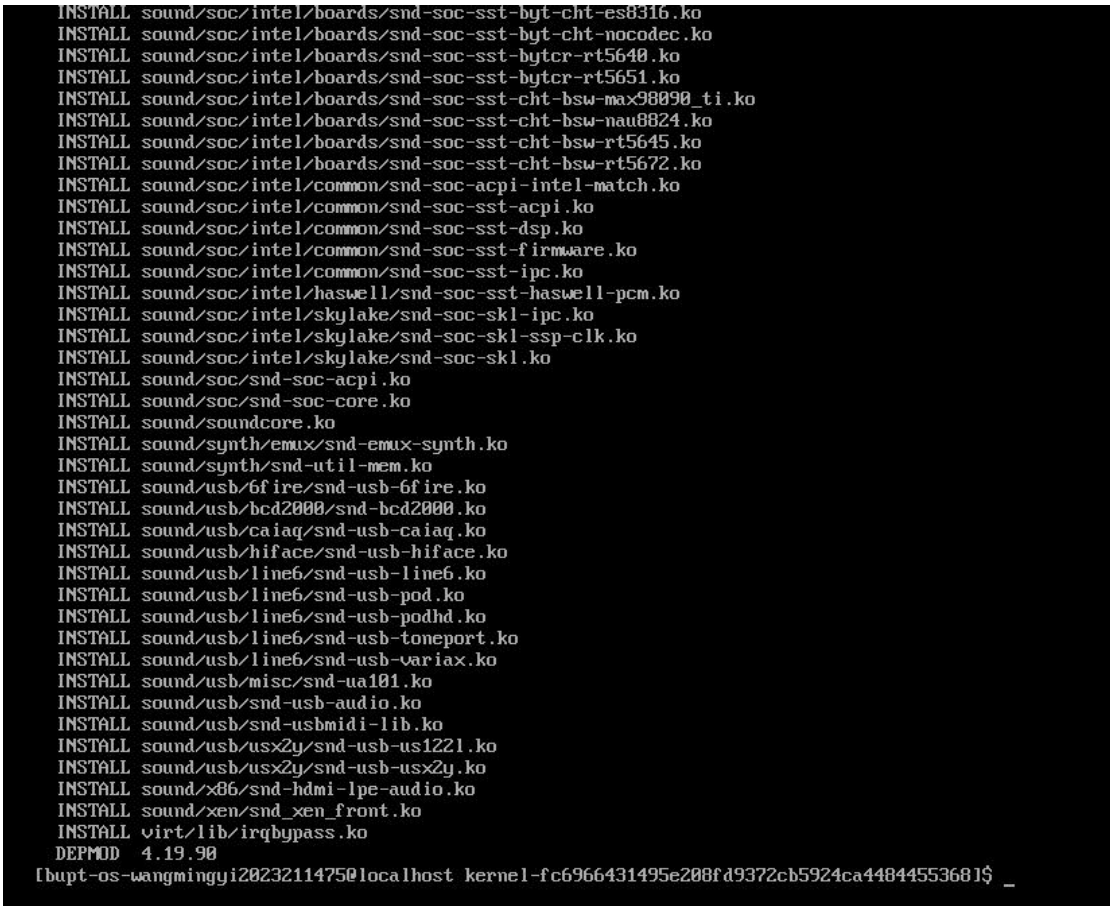
执行 make modules_install 安装模块

正在安装模块：



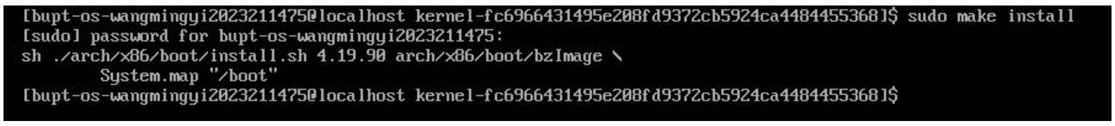
```
  INSTALL crypto/dh_generic.ko
  INSTALL crypto/ecdh_generic.ko
  INSTALL crypto/echainiv.ko
  INSTALL crypto/fcrypt.ko
  INSTALL crypto/khazad.ko
  INSTALL crypto/lrw.ko
  INSTALL crypto/md4.ko
  INSTALL crypto/michael_mic.ko
  INSTALL crypto/pcbc.ko
  INSTALL crypto/pcrypt.ko
  INSTALL crypto/poly1305_generic.ko
  INSTALL crypto/rmd128.ko
  INSTALL crypto/rmd160.ko
  INSTALL crypto/rmd256.ko
  INSTALL crypto/rmd320.ko
  INSTALL crypto/salsa20_generic.ko
  INSTALL crypto/seed.ko
  INSTALL crypto/serpent_generic.ko
  INSTALL crypto/sha3_generic.ko
  INSTALL crypto/sha512_generic.ko
  INSTALL crypto/tcrypt.ko
  INSTALL crypto/tea.ko
  INSTALL crypto/tgr192.ko
  INSTALL crypto/twofish_common.ko
  INSTALL crypto/twofish_generic.ko
  INSTALL crypto/vmac.ko
  INSTALL crypto/wp512.ko
  INSTALL crypto/xcbc.ko
  INSTALL crypto/xor.ko
  INSTALL crypto/xts.ko
  INSTALL drivers/acpi/acpi_extlog.ko
  INSTALL drivers/acpi/acpi_ipmi.ko
  INSTALL drivers/acpi/acpi_pad.ko
  INSTALL drivers/acpi/acpi_tad.ko
  INSTALL drivers/acpi/apei/einj.ko
  INSTALL drivers/acpi/dptf/dptf_power.ko
  INSTALL drivers/acpi/ec_sys.ko
  INSTALL drivers/acpi/nfit/nfit.ko
  INSTALL drivers/acpi/sbs.ko
  INSTALL drivers/acpi/sbshc.ko
  INSTALL drivers/acpi/video.ko
  INSTALL drivers/ata/ahci.ko
  INSTALL drivers/ata/ahci_platform.ko
  INSTALL drivers/ata/ata_generic.ko
  INSTALL drivers/ata/ata_piix.ko
  INSTALL drivers/ata/libahci.ko
  INSTALL drivers/ata/libahci_platform.ko
```

模块安装完成：



```
    INSTALL sound/soc/intel/boards/snd-soc-sst-byt-cht-es8316.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-byt-cht-nocodec.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-bytcr-rt5640.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-bytcr-rt5651.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-cht-bsw-max98090_ti.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-cht-bsw-nau8824.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5645.ko
    INSTALL sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5672.ko
    INSTALL sound/soc/intel/common/snd-soc-acpi-intel-match.ko
    INSTALL sound/soc/intel/common/snd-soc-sst-acpi.ko
    INSTALL sound/soc/intel/common/snd-soc-sst-dsp.ko
    INSTALL sound/soc/intel/common/snd-soc-sst-firmware.ko
    INSTALL sound/soc/intel/common/snd-soc-sst-ipc.ko
    INSTALL sound/soc/intel/haswell/snd-soc-sst-haswell-pcm.ko
    INSTALL sound/soc/intel/skylake/snd-soc-skl-ipc.ko
    INSTALL sound/soc/intel/skylake/snd-soc-skl-ssp-clk.ko
    INSTALL sound/soc/intel/skylake/snd-soc-skl.ko
    INSTALL sound/soc/snd-soc-acpi.ko
    INSTALL sound/soc/snd-soc-core.ko
    INSTALL sound/soundcore.ko
    INSTALL sound/synth/emux/snd-emux-synth.ko
    INSTALL sound/synth/snd-util-mem.ko
    INSTALL sound/usb/6fire/snd-usb-6fire.ko
    INSTALL sound/usb/bcd2000/snd-bcd2000.ko
    INSTALL sound/usb/caiaq/snd-usb-caiaq.ko
    INSTALL sound/usb/hiface/snd-usb-hiface.ko
    INSTALL sound/usb/line6/snd-usb-line6.ko
    INSTALL sound/usb/line6/snd-usb-pod.ko
    INSTALL sound/usb/line6/snd-usb-podhd.ko
    INSTALL sound/usb/line6/snd-usb-toneport.ko
    INSTALL sound/usb/line6/snd-usb-variax.ko
    INSTALL sound/usb/misc/snd-ua101.ko
    INSTALL sound/usb/snd-usb-audio.ko
    INSTALL sound/usb/snd-usbmidi-lib.ko
    INSTALL sound/usb/usx2y/snd-usb-us122l.ko
    INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
    INSTALL sound/x86/snd-hdmi-lpe-audio.ko
    INSTALL sound/xen/snd_xen_front.ko
    INSTALL virt/lib/irqbypass.ko
    DEPMOD  4.19.90
  [bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ _
```

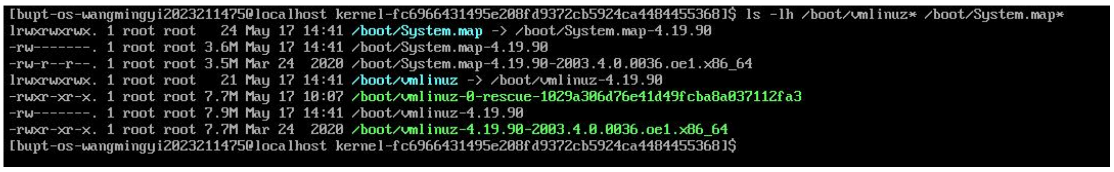
运行 make install 安装内核：



```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ sudo make install
[sudo] password for bupt-os-wangmingyi2023211475:
sh ./arch/x86/boot/install.sh 4.19.90 arch/x86/boot/bzImage \
        System.map "/boot"
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$
```

安装完成

输入“ls -lh /boot/vmlinuz* /boot/System.map*”检查/boot 目录确认文件生成

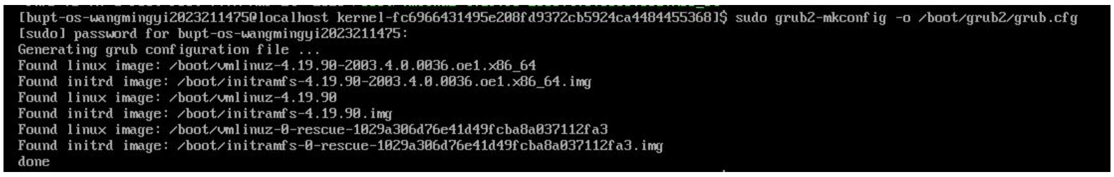


```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ ls -lh /boot/vmlinuz* /boot/System.map*
lrwxrwxrwx. 1 root root   24 May 17 14:41 /boot/System.map -> /boot/System.map-4.19.90
-rw-------. 1 root root 3.6M May 17 14:41 /boot/System.map-4.19.90
-rw-r--r--. 1 root root 3.5M Mar 24  2020 /boot/System.map-4.19.90-2003.4.0.0036.oe1.x86_64
lrwxrwxrwx. 1 root root   21 May 17 14:41 /boot/vmlinuz -> /boot/vmlinuz-4.19.90
-rwxr-xr-x. 1 root root 7.7M May 17 10:07 /boot/vmlinuz-0-rescue-1029a306d76e41d49fcba8a037112fa3
-rw-------. 1 root root 7.9M May 17 14:41 /boot/vmlinuz-4.19.90
-rwxr-xr-x. 1 root root 7.7M Mar 24  2020 /boot/vmlinuz-4.19.90-2003.4.0.0036.oe1.x86_64
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$
```

如图片所示：有“vmlinuz-4.19.98”和“System.map-4.19.98”文件，安装成功！

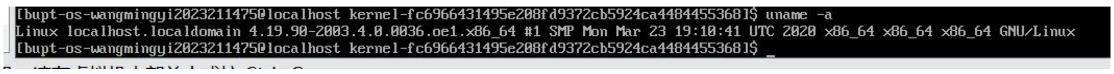
在安装完成新内核后，为使系统能够正确引导至新版本内核，需要重新生成 GRUB 启动引导配置文件。执行 sudo grub2-mkconfig -o /boot/grub2/grub.cfg 命令会自动将新安装的内核添加到启动菜单中。随后通过 sudo grub2-set-default 0 将新内核设为默认启动项。最后，执行 sudo reboot 重启系统，系统将在下一次引导中加载新内核

更新引导：



```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
[sudo] password for bupt-os-wangmingyi2023211475:
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-4.19.90-2003.4.0.0036.oe1.x86_64
Found initrd image: /boot/initramfs-4.19.90-2003.4.0.0036.oe1.x86_64.img
Found linux image: /boot/vmlinuz-4.19.90
Found initrd image: /boot/initramfs-4.19.90.img
Found linux image: /boot/vmlinuz-0-rescue-1029a306d76e41d49fcba8a037112fa3
Found initrd image: /boot/initramfs-0-rescue-1029a306d76e41d49fcba8a037112fa3.img
done
```

执行 uname -a 指令，好于之后的新内核做对比。

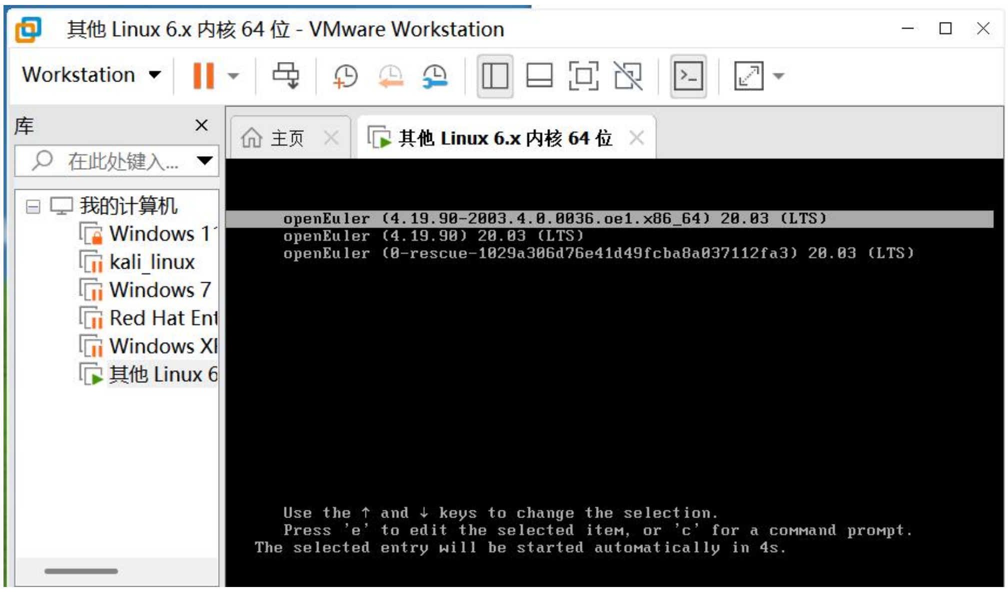


```
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ uname -a
Linux localhost.localdomain 4.19.90-2003.4.0.0036.oe1.x86_64 #1 SMP Mon Mar 23 19:10:41 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
[bupt-os-wangmingyi2023211475@localhost kernel-fc6966431495e208fd9372cb5924ca4484455368]$ _
```

重启系统：

然后重启系统就可以看到多个内核，其中一个就是我们新安装的内核：

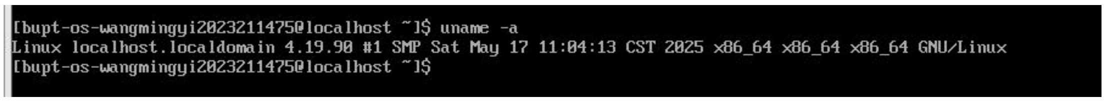
openEuler (4.19.90) 20.03 (LTS)



```
openEuler (4.19.90-2003.4.0.0036.oe1.x86_64) 20.03 (LTS)
openEuler (4.19.90) 20.03 (LTS)
openEuler (0-rescue-1029a306d76e41d49fcba8a037112fa3) 20.03 (LTS)

    Use the ↑ and ↓ keys to change the selection.
    Press 'e' to edit the selected item, or 'c' for a command prompt.
 The selected entry will be started automatically in 4s.
```

验证新内核

登录新内核后，执行 uname -a：



```
[bupt-os-wangmingyi2023211475@localhost ~]$ uname -a
Linux localhost.localdomain 4.19.90 #1 SMP Sat May 17 11:04:13 CST 2025 x86_64 x86_64 x86_64 GNU/Linux
[bupt-os-wangmingyi2023211475@localhost ~]$
```

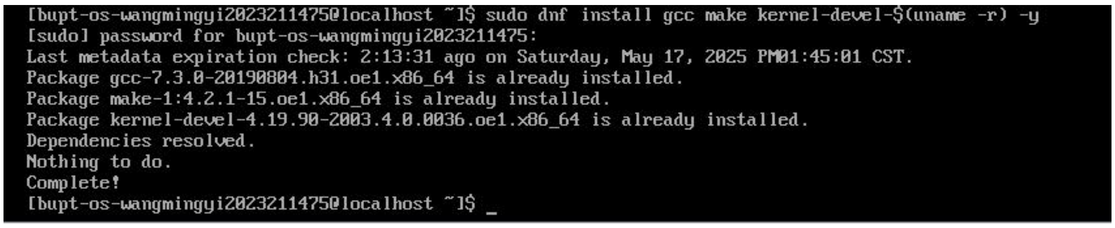
是新内核了。

三、基础操作系统实验

1、内核模块编程：

a）安装 C 语言编译器、make 工具、以及与当前内核版本对应的内核开发头文件

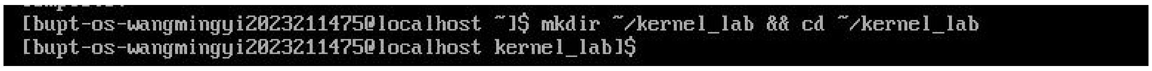
sudo dnf install gcc make kernel-devel-$(uname -r) -y



```
[bupt-os-wangmingyi2023211475@localhost ~]$ sudo dnf install gcc make kernel-devel-$(uname -r) -y
[sudo] password for bupt-os-wangmingyi2023211475:
Last metadata expiration check: 2:13:31 ago on Saturday, May 17, 2025 PM01:45:01 CST.
Package gcc-7.3.0-20190804.h31.oe1.x86_64 is already installed.
Package make-1:4.2.1-15.oe1.x86_64 is already installed.
Package kernel-devel-4.19.90-2003.4.0.0036.oe1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[bupt-os-wangmingyi2023211475@localhost ~]$ _
```

为管理内核模块实验相关的源代码与构建文件，在用户主目录下创建内核模块编程专用目录 kernel_lab，并立即进入该目录。

mkdir ~/kernel_lab && cd ~/kernel_lab



```
[bupt-os-wangmingyi2023211475@localhost ~]$ mkdir ~/kernel_lab && cd ~/kernel_lab
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$
```

b)用 vi helloworld.c 创建模块源文件

按 i 进入编辑模式，写入下面的代码

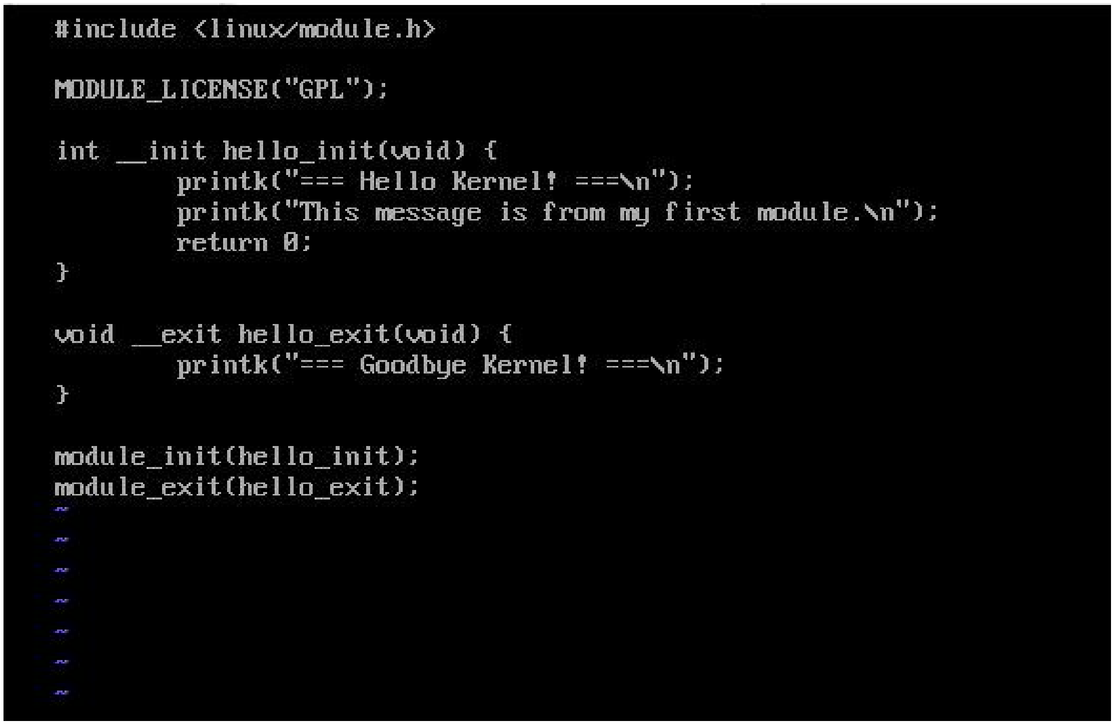
#include <linux/module.h>

MODULE_LICENSE("GPL");

int __init hello_init(void) {

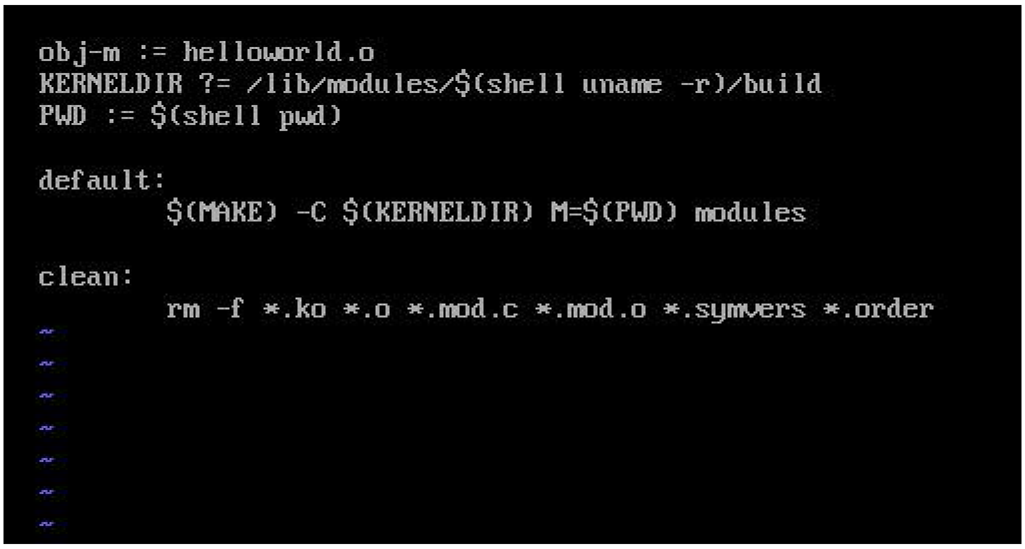
```
    printk("=== Hello Kernel! ===\n");
    printk("This message is from my first module.\n");
    return 0;
}
void __exit hello_exit(void) {
    printk("=== Goodbye Kernel! ===\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

该模块会在插入时打印“Hello Kernel”，在卸载时输出“Goodbye Kernel”，用于验证内核模块编写与加载机制是否正常运行



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

MODULE_LICENSE("GPL");

int __init hello_init(void) {
        printk("=== Hello Kernel! ===\n");
        printk("This message is from my first module.\n");
        return 0;
}

void __exit hello_exit(void) {
        printk("=== Goodbye Kernel! ===\n");
}

module_init(hello_init);
module_exit(hello_exit);
~
~
~
~
~
~
~
```

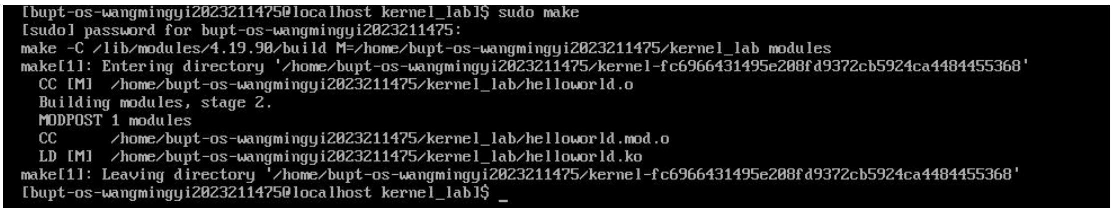
c)编写 Linux 内核模块的标准 Makefile，用于编译之前写的 helloworld.c 模块



```
obj-m := helloworld.o
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
	$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

clean:
	rm -f *.ko *.o *.mod.c *.mod.o *.symvers *.order
~
~
~
~
~
~
~
```

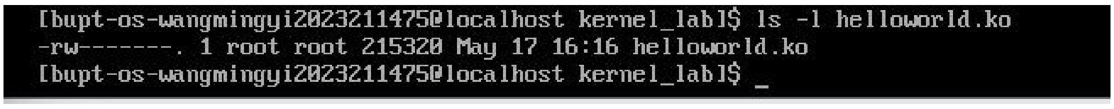
d)编译

调用 Makefile 中定义的规则完成对 helloworld.c 模块的编译，最终生成 helloworld.ko



```
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo make
[sudo] password for bupt-os-wangmingyi2023211475:
make -C /lib/modules/4.19.90/build M=/home/bupt-os-wangmingyi2023211475/kernel_lab modules
make[1]: Entering directory '/home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368'
  CC [M]  /home/bupt-os-wangmingyi2023211475/kernel_lab/helloworld.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt-os-wangmingyi2023211475/kernel_lab/helloworld.mod.o
  LD [M]  /home/bupt-os-wangmingyi2023211475/kernel_lab/helloworld.ko
make[1]: Leaving directory '/home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368'
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ _
```

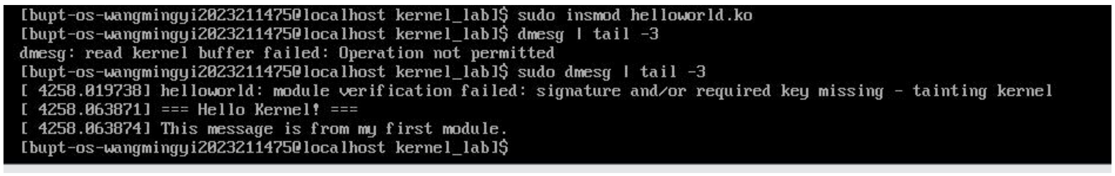
检查生成的文件



```
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ ls -l helloworld.ko
-rw-------. 1 root root 215320 May 17 16:16 helloworld.ko
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ _
```

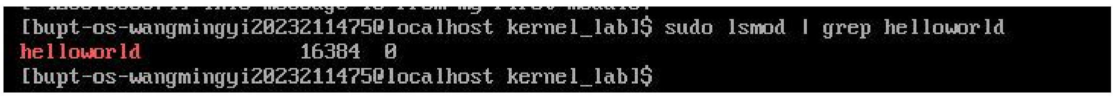
e)加载模块 sudo insmod helloworld.ko

查看日志 dmesg | tail -3



```
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo insmod helloworld.ko
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ dmesg | tail -3
dmesg: read kernel buffer failed: Operation not permitted
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo dmesg | tail -3
[ 4258.019738] helloworld: module verification failed: signature and/or required key missing - tainting kernel
[ 4258.063871] === Hello Kernel! ===
[ 4258.063874] This message is from my first module.
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$
```

f)使用 lsmod | grep helloworld 命令验证模块是否已成功加载至内核

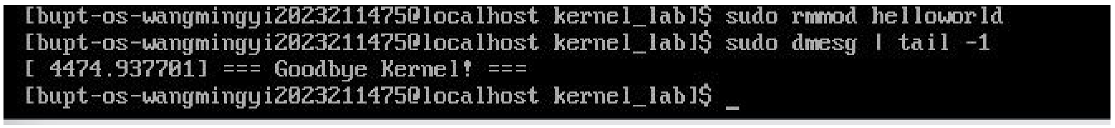


```
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo lsmod | grep helloworld
helloworld             16384  0
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$
```

输出 helloworld 说明成功。

g)卸载模块，sudo rmmod helloworld

用 dmesg | tail -1 查看内核日志的最新一条，查看是否输出了“Goodbye Kernel”



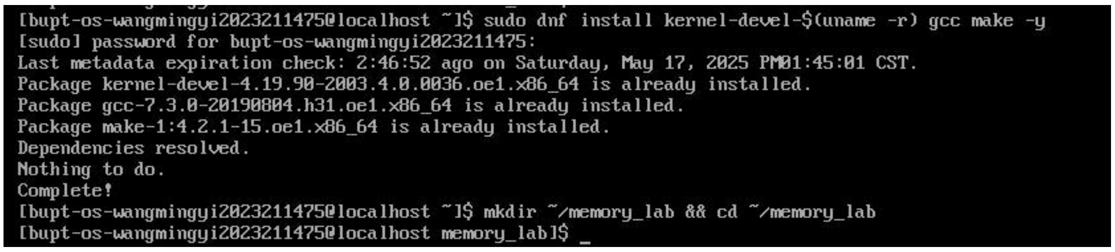
```
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo rmmod helloworld
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ sudo dmesg | tail -1
[ 4474.937701] === Goodbye Kernel! ===
[bupt-os-wangmingyi2023211475@localhost kernel_lab]$ _
```

至此已完成内核模块编程的实验。

2、内存管理：

(1)内核模块内存操作

a) 创建实验目录 mkdir ~/memory_lab && cd ~/memory_lab

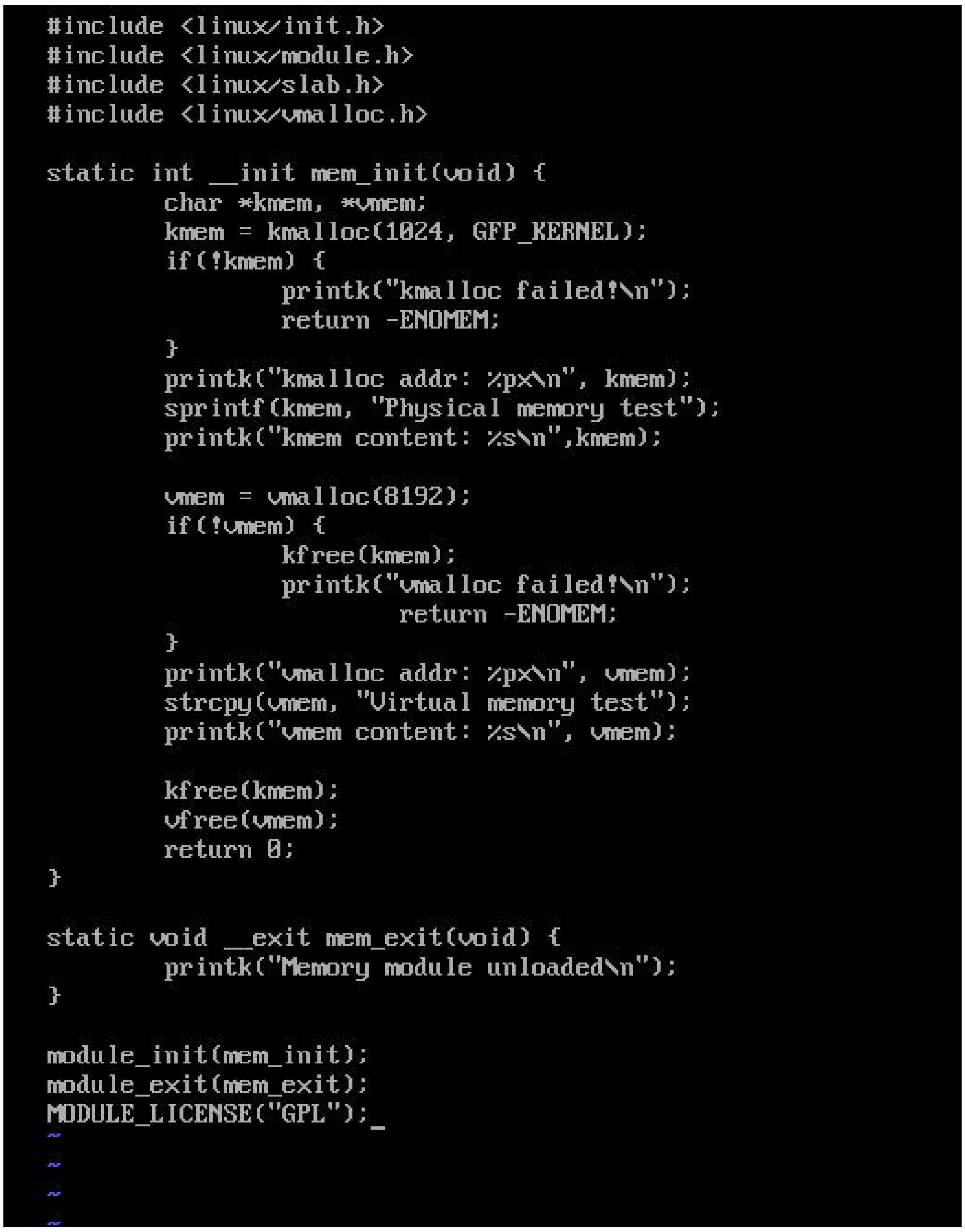


```
[bupt-os-wangmingyi2023211475@localhost ~]$ sudo dnf install kernel-devel-$(uname -r) gcc make -y
[sudo] password for bupt-os-wangmingyi2023211475:
Last metadata expiration check: 2:46:52 ago on Saturday, May 17, 2025 PM01:45:01 CST.
Package kernel-devel-4.19.90-2003.4.0.0036.oe1.x86_64 is already installed.
Package gcc-7.3.0-20190804.h31.oe1.x86_64 is already installed.
Package make-1:4.2.1-15.oe1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[bupt-os-wangmingyi2023211475@localhost ~]$ mkdir ~/memory_lab && cd ~/memory_lab
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ _
```

b)编写内核模块代码

创建文件：vi kmem.c

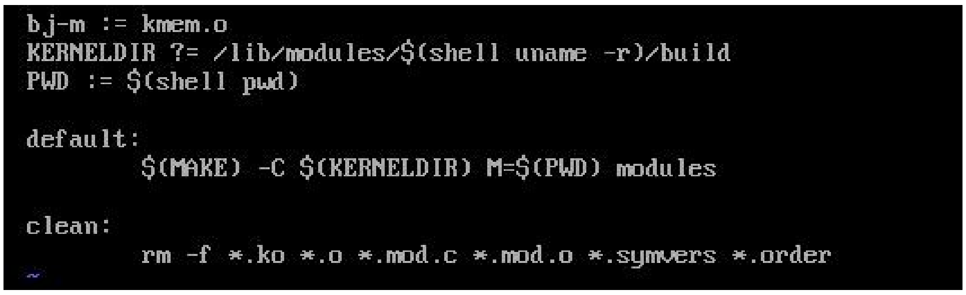
编写代码，如下图



```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/vmalloc.h>

static int __init mem_init(void) {
        char *kmem, *vmem;
        kmem = kmalloc(1024, GFP_KERNEL);
        if(!kmem) {
                printk("kmalloc failed!\n");
                return -ENOMEM;
        }
        printk("kmalloc addr: %px\n", kmem);
        sprintf(kmem, "Physical memory test");
        printk("kmem content: %s\n",kmem);

        vmem = vmalloc(8192);
        if(!vmem) {
                kfree(kmem);
                printk("vmalloc failed!\n");
                        return -ENOMEM;
        }
        printk("vmalloc addr: %px\n", vmem);
        strcpy(vmem, "Virtual memory test");
        printk("vmem content: %s\n", vmem);

        kfree(kmem);
        vfree(vmem);
        return 0;
}

static void __exit mem_exit(void) {
        printk("Memory module unloaded\n");
}

module_init(mem_init);
module_exit(mem_exit);
MODULE_LICENSE("GPL");_
~
~
~
~
```

该模块加载时，会分别分配 1024 字节的物理连续内存和 8192 字节的虚拟连续内存，并通过 printk 输出内存地址。

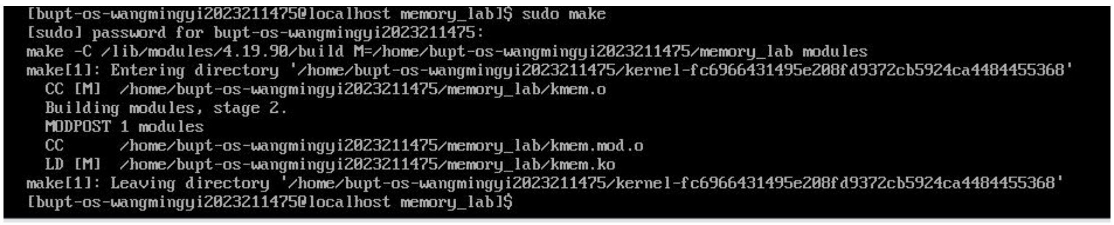
c)编写 Makefile，编译之前写 kmem.c,如下图



```
bj-m := kmem.o
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
        $(MAKE) -C $(KERNELDIR) M=$(PWD) modules

clean:
        rm -f *.ko *.o *.mod.c *.mod.o *.symvers *.order
~
```

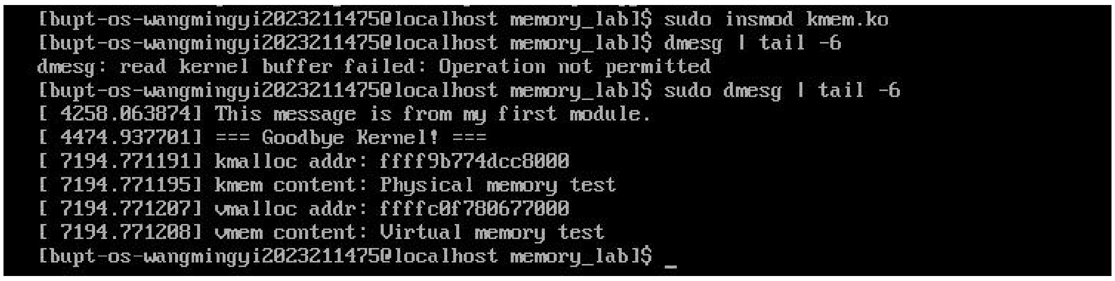
d)编译，查看结果

编译：



```
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ sudo make
[sudo] password for bupt-os-wangmingyi2023211475:
make -C /lib/modules/4.19.90/build M=/home/bupt-os-wangmingyi2023211475/memory_lab modules
make[1]: Entering directory '/home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368'
  CC [M]  /home/bupt-os-wangmingyi2023211475/memory_lab/kmem.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt-os-wangmingyi2023211475/memory_lab/kmem.mod.o
  LD [M]  /home/bupt-os-wangmingyi2023211475/memory_lab/kmem.ko
make[1]: Leaving directory '/home/bupt-os-wangmingyi2023211475/kernel-fc6966431495e208fd9372cb5924ca4484455368'
[bupt-os-wangmingyi2023211475@localhost memory_lab]$
```

查看输出：

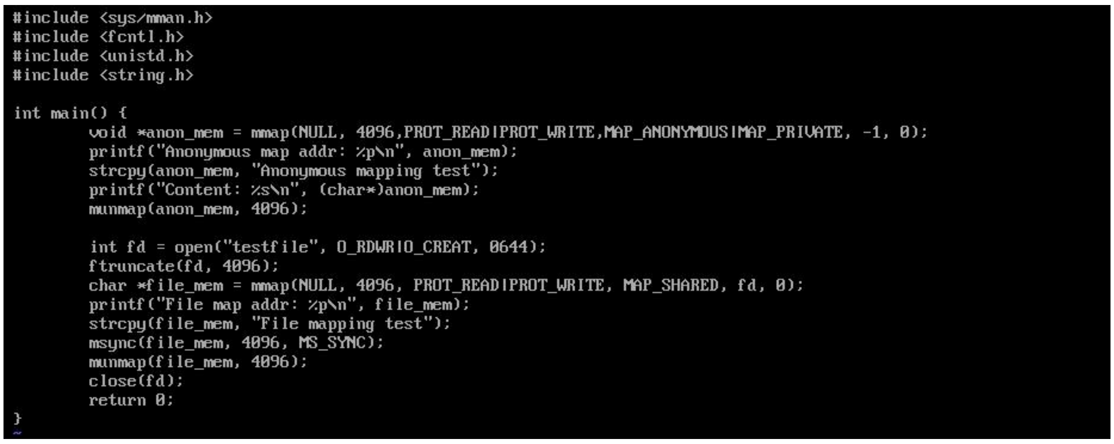


```
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ sudo insmod kmem.ko
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ dmesg | tail -6
dmesg: read kernel buffer failed: Operation not permitted
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ sudo dmesg | tail -6
[ 4258.063874] This message is from my first module.
[ 4474.937701] === Goodbye Kernel! ===
[ 7194.771191] kmalloc addr: ffff9b774dcc8000
[ 7194.771195] kmem content: Physical memory test
[ 7194.771207] vmalloc addr: ffffc0f780677000
[ 7194.771208] vmem content: Virtual memory test
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ _
```

（2）用户内存映射

a)创建 user_mmap.c 文件，vi user_mmap.c

b)编写文件内容，如下图



```
#include <sys/mman.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
        void *anon_mem = mmap(NULL, 4096,PROT_READ|PROT_WRITE,MAP_ANONYMOUS|MAP_PRIVATE, -1, 0);
        printf("Anonymous map addr: %p\n", anon_mem);
        strcpy(anon_mem, "Anonymous mapping test");
        printf("Content: %s\n", (char*)anon_mem);
        munmap(anon_mem, 4096);

        int fd = open("testfile", O_RDWR|O_CREAT, 0644);
        ftruncate(fd, 4096);
        char *file_mem = mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
        printf("File map addr: %p\n", file_mem);
        strcpy(file_mem, "File mapping test");
        msync(file_mem, 4096, MS_SYNC);
        munmap(file_mem, 4096);
        close(fd);
        return 0;
}
~
```

代码通过 mmap() 系统调用演示了匿名映射和文件映射两种内存管理机制。在匿名映射中，通过 MAP_ANONYMOUS|MAP_PRIVATE 获取一段私有内存，用于临时数据的读写。在文件映射中，先通过 open 和 ftruncate 创建并调整文件大小，再使用 MAP_SHARED

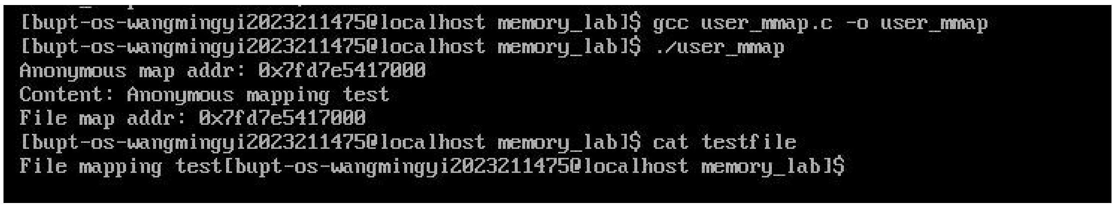
映射文件内容到内存，写入内容后通过 msync() 同步回磁盘。

c)编译运行

编译文件，gcc user_mmap.c -o user_mmap

运行程序，执行匿名映射和文件映射逻辑，./user_mmap

打印映射写入的文件内容，验证 "File mapping test" 是否写入成功，cat testfile



```
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ gcc user_mmap.c -o user_mmap
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ ./user_mmap
Anonymous map addr: 0x7fd7e5417000
Content: Anonymous mapping test
File map addr: 0x7fd7e5417000
[bupt-os-wangmingyi2023211475@localhost memory_lab]$ cat testfile
File mapping test[bupt-os-wangmingyi2023211475@localhost memory_lab]$
```

输出 "File mapping test"，说明 mmap 与 msync 调用已成功完成数据同步。

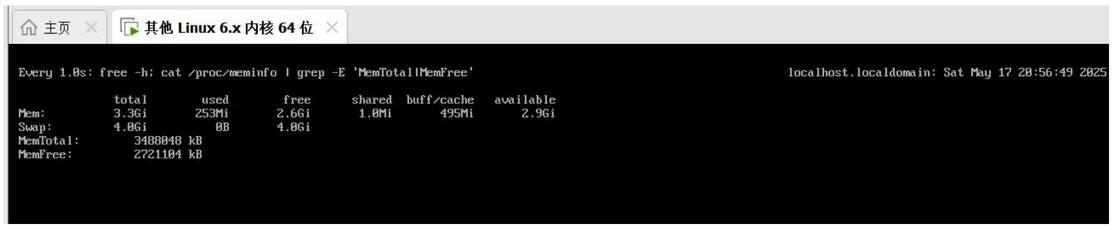
（3）系统内存监控

本实验采用 watch 命令实时监控 /proc/meminfo、/proc/buddyinfo 和 /proc/slabinfo。

a)

监控内存使用变化 watch -n 1 "free -h; cat /proc/meminfo | grep -E 'MemTotal|MemFree'"

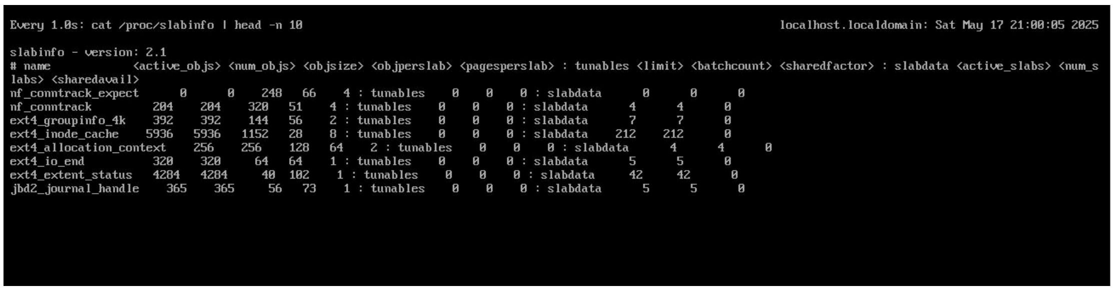
该命令每 1 秒刷新一次输出，通过 free -h 命令查看整体内存使用状态包括总量、使用、剩余，并通过 /proc/meminfo 提取精确的 MemTotal 与 MemFree 数值。



```
Every 1.0s: free -h; cat /proc/meminfo | grep -E 'MemTotal|MemFree'        localhost.localdomain: Sat May 17 20:56:49 2025

               total        used        free      shared  buff/cache   available
Mem:           3.3Gi       253Mi       2.6Gi       1.0Mi       495Mi       2.9Gi
Swap:          4.0Gi          0B       4.0Gi
MemTotal:        3488048 kB
MemFree:         2721104 kB
```

b) 实时监控系统的物理内存碎片情况，watch -n 1 "cat /proc/buddyinfo",

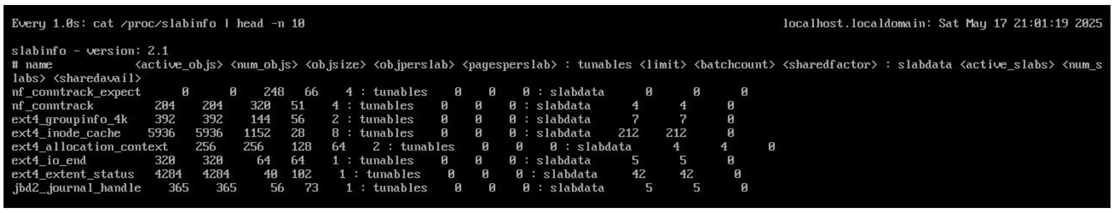
/proc/buddyinfo 文件记录了系统伙伴系统（buddy system）中各阶内存块的空闲数量，反映出不同大小连续内存块的可用性。



```
Every 1.0s: cat /proc/slabinfo | head -n 10                                          localhost.localdomain: Sat May 17 21:00:05 2025

slabinfo - version: 2.1
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> : tunables <limit> <batchcount> <sharedfactor> : slabdata <active_slabs> <num_s
labs> <sharedavail>
nf_conntrack_expect      0      0    248   66    4 : tunables    0    0    0 : slabdata      0      0      0
nf_conntrack           204    204    320   51    4 : tunables    0    0    0 : slabdata      4      4      0
ext4_groupinfo_4k      392    392    144   56    2 : tunables    0    0    0 : slabdata      7      7      0
ext4_inode_cache      5936   5936   1152   28    8 : tunables    0    0    0 : slabdata    212    212      0
ext4_allocation_context    256    256    128   64    2 : tunables    0    0    0 : slabdata      4      4      0
ext4_io_end            320    320     64   64    1 : tunables    0    0    0 : slabdata      5      5      0
ext4_extent_status    4284   4284     40  102    1 : tunables    0    0    0 : slabdata     42     42      0
jbd2_journal_handle    365    365     56   73    1 : tunables    0    0    0 : slabdata      5      5      0
```

c) 查看 SLAB 分配器

watch -n 1 "cat /proc/slabinfo | head -n 10



```
Every 1.0s: cat /proc/slabinfo | head -n 10                                          localhost.localdomain: Sat May 17 21:01:19 2025

slabinfo - version: 2.1
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> : tunables <limit> <batchcount> <sharedfactor> : slabdata <active_slabs> <num_s
labs> <sharedavail>
nf_conntrack_expect      0      0    248   66    4 : tunables    0    0    0 : slabdata      0      0      0
nf_conntrack           204    204    320   51    4 : tunables    0    0    0 : slabdata      4      4      0
ext4_groupinfo_4k      392    392    144   56    2 : tunables    0    0    0 : slabdata      7      7      0
ext4_inode_cache      5936   5936   1152   28    8 : tunables    0    0    0 : slabdata    212    212      0
ext4_allocation_context    256    256    128   64    2 : tunables    0    0    0 : slabdata      4      4      0
ext4_io_end            320    320     64   64    1 : tunables    0    0    0 : slabdata      5      5      0
ext4_extent_status    4284   4284     40  102    1 : tunables    0    0    0 : slabdata     42     42      0
jbd2_journal_handle    365    365     56   73    1 : tunables    0    0    0 : slabdata      5      5      0
```

四、出现问题及解决方法

（1）想要下载 VMware Tools 实现物理主机复制到虚拟机；放弃复制粘贴，全部字符手打

（2）无法连接网络；输入 ip a，查看网卡，发现有 ens160 网卡，然后执行命令：sudo nmcli connection up ens160，激活 ens 160 网络接口，使其能够发送与接收数据包。

（3）无法下载工具，压缩包；手动重新配置了 YUM 仓库，将软件源切换为阿里云提供的 openEuler 镜像站。

（4）内核编译空间不足（虚拟机创建只分配了 20G）；重现安装虚拟机，重做实验，分配了 40G 硬盘