北京郵電大學

操作系统《openEuler》实验报告

姓　　名 王何佳

学　　号 2023211603

贡 献 度 100%

班　　级 2023211804

邮　　箱 624772990@qq.com

2025 年 5 月 18 日

目录

一、 实验内容 3
二、 实验过程 3
1. 安装 openEuler 操作系统 3
2. 安装图形化界面 7
3. 安装 firefox 10
4. 安装 VMware Tools 11
5. 将内核更新至最新版 13
6. 内核模块编程 18
7. 内存管理 21
（1） 使用 kmalloc 分配 1KB，8KB 的内存，并打印指针地址 21
（2） 使用 vmalloc 分别分配 8KB、1MB、64MB 的内存，打印指针地址 23
8. 中断和异常处理 25
（1） 使用 tasklet 实现打印 helloworld 25
（2） 用工作队列实现周期打印 helloworld 26
（3） 编写一个信号捕获程序，捕获终端按键信号 27
9. 内核时间管理 28
（1） 调用内核时钟接口打印当前时间 28
（2） 编写 timer，在特定时刻打印 hello,world 30
（3） 调用内核时钟接口，监控累加计算代码的运行时间 31
三、 问题与解决 32
1. 权限问题 32
（1） 无法更改文件内容 32
（2） 对文件操作时出错 34
（3） 其他权限问题 34
2. 客户机操作系统已禁用 CPU 34
3. 代码报错，虚拟机界面编程不便 37

一、实验内容

1．完成 openEuler 操作系统的安装。
2．完成内核更新（源代码更新方式）。
3．添加其他功能：内核模块编程、内存管理、中断和异常处理、内核时间管理。

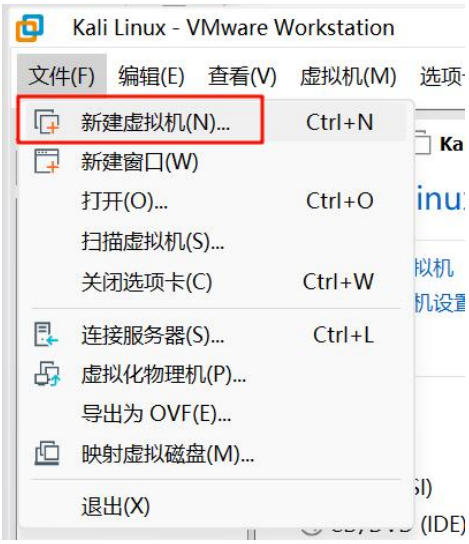
二、实验过程

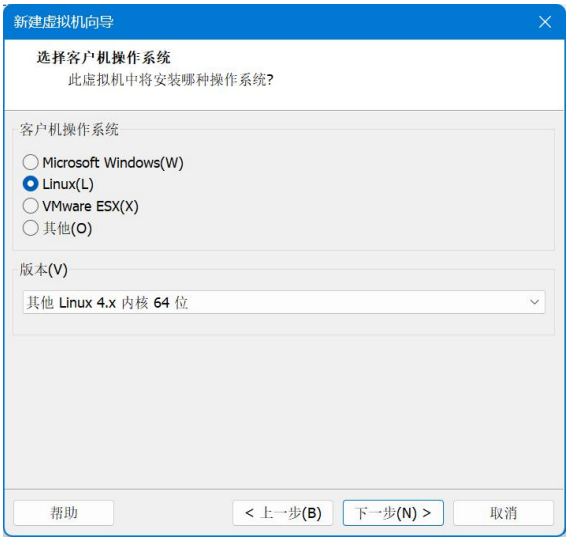
1．安装 openEuler 操作系统

采用虚拟机 Vmware 完成 openEuler 操作系统的安装，安装版本为 20.03-TLS，iso 镜像下载地址：

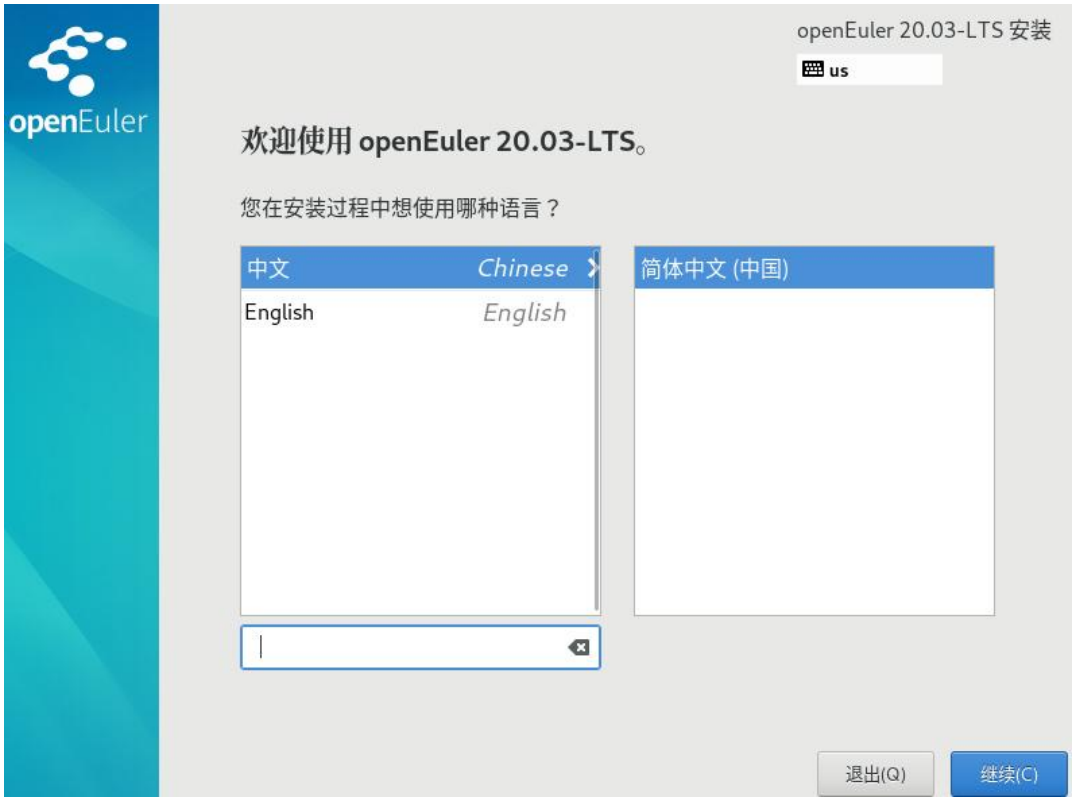
https://mirror.iscas.ac.cn/openeuler/openEuler-20.03-LTS/ISO/x86_64/openEuler-20.03-LTS-x86_64-dvd.iso

下载完镜像后，创建一个 Linux 操作系统（用 openEuler 镜像）

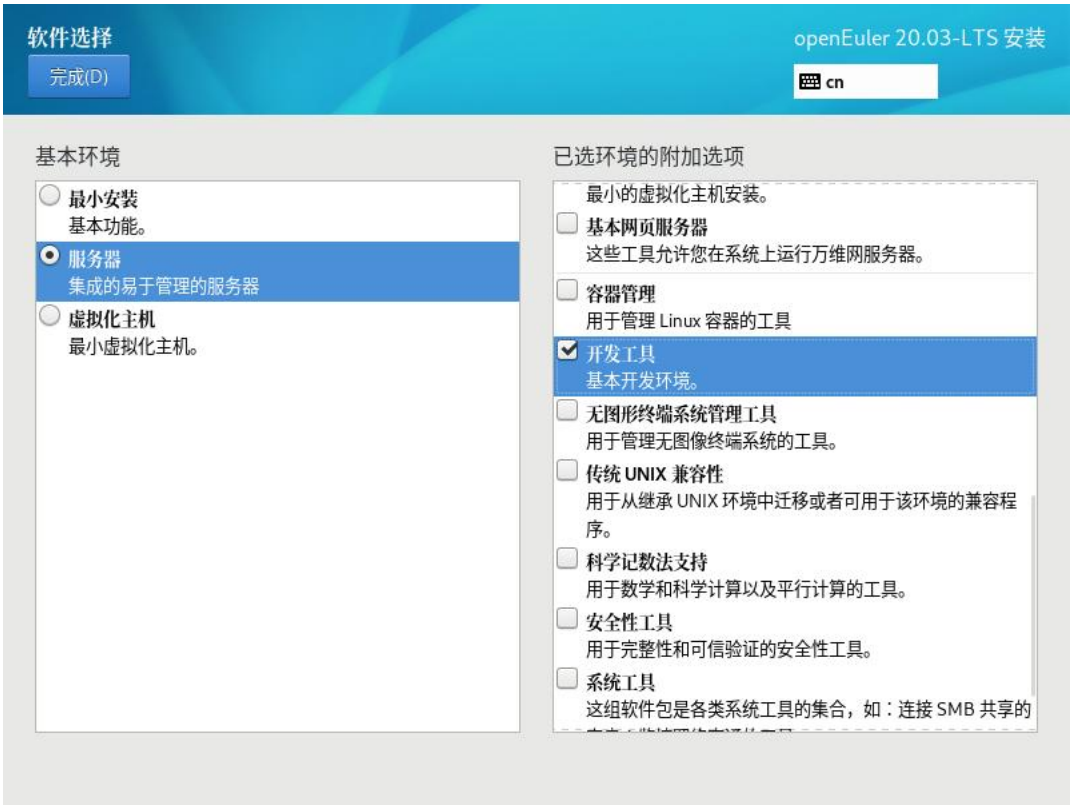




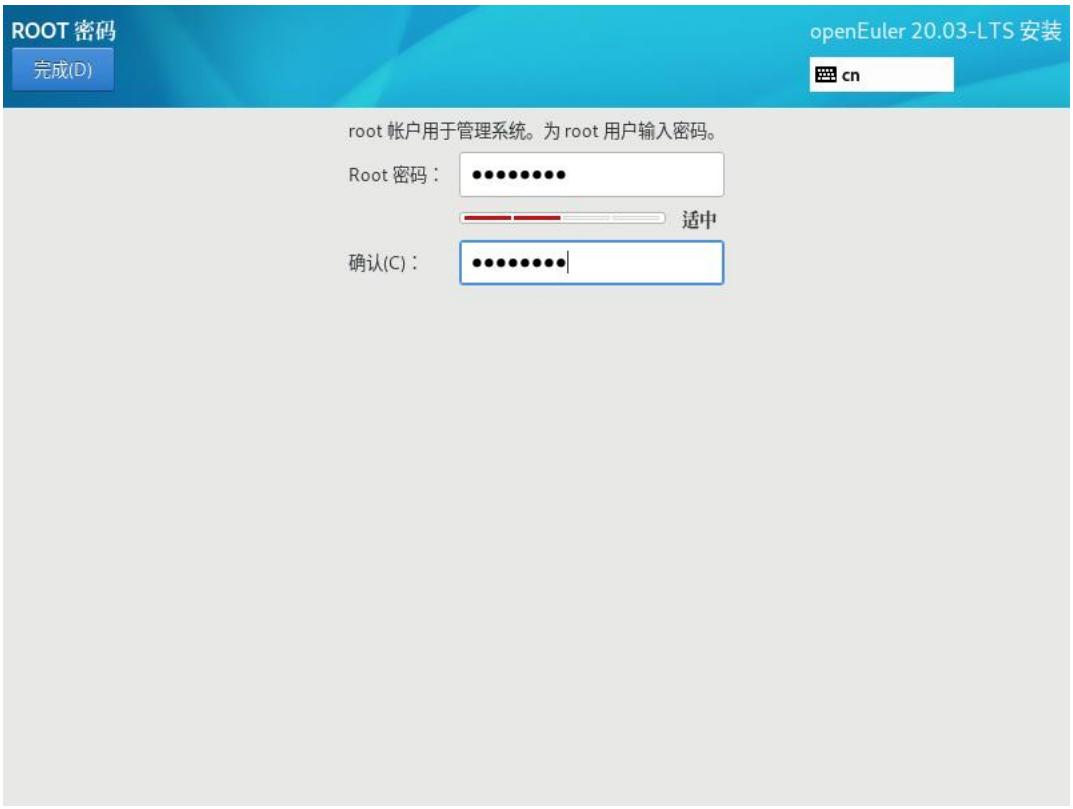
打开虚拟机，开始安装系统，配置 openEuler



“软件选择”→“服务器”→ 勾选“开发工具”



设置 ROOT 密码：whj@BUPT

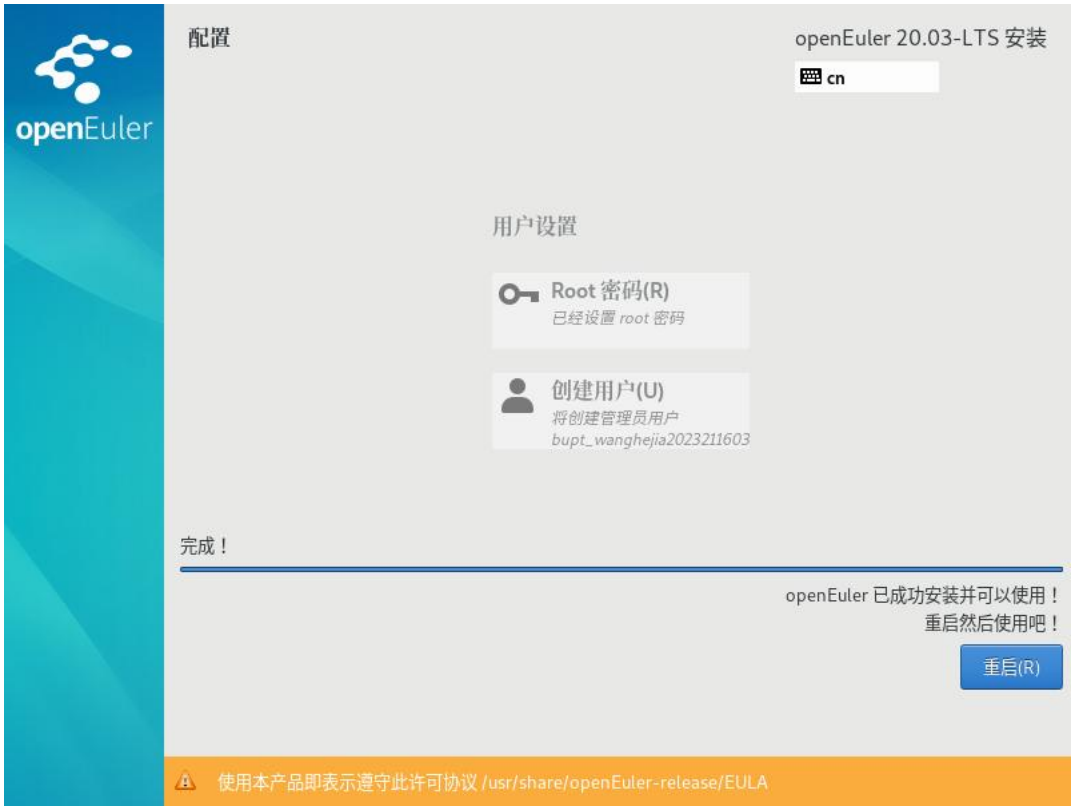


创建用户

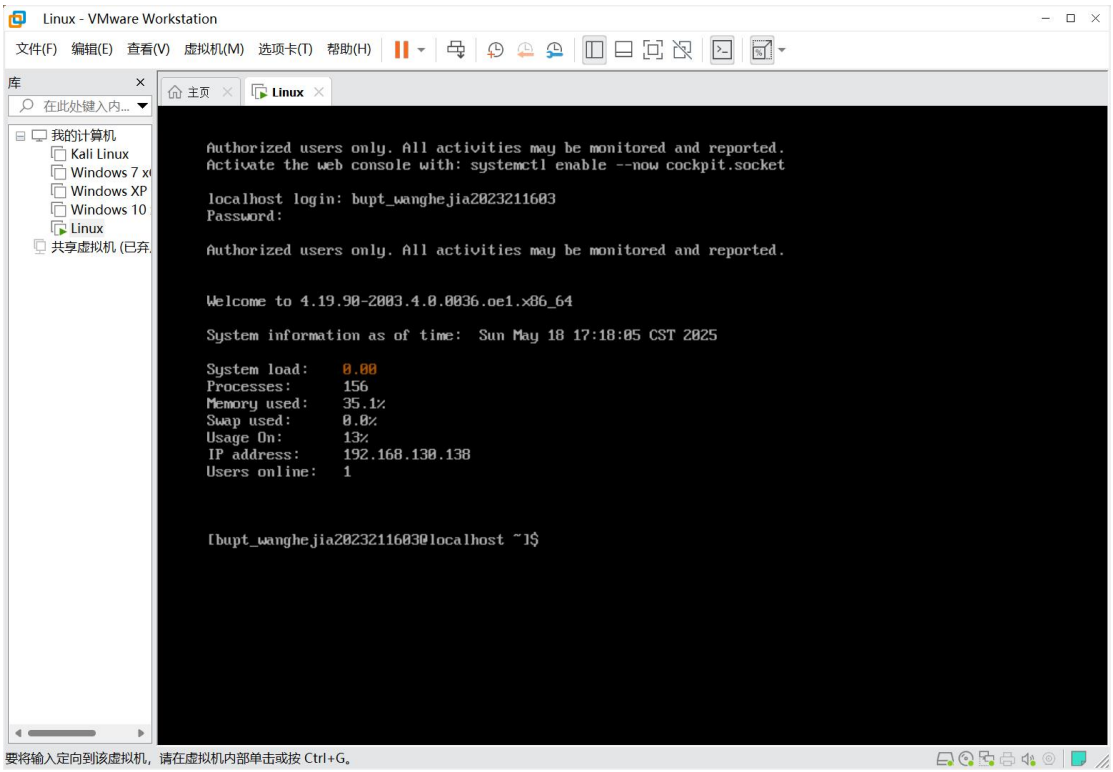
用户名：bupt_wanghejia2023211603

密码：whj@BUPT

配置完成后重启。



输入用户名和密码后，进入系统。



执行指令

```
uname -a
```

```
[bupt_wanghejia2023211603@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
```

查看 openEuler 分页大小

```
getconf PAGESIZE
```

```
[bupt_wanghejia2023211603@localhost ~]$ getconf PAGESIZE
4096
```

2. 安装图形化界面

首先配置清华源（root 模式下）

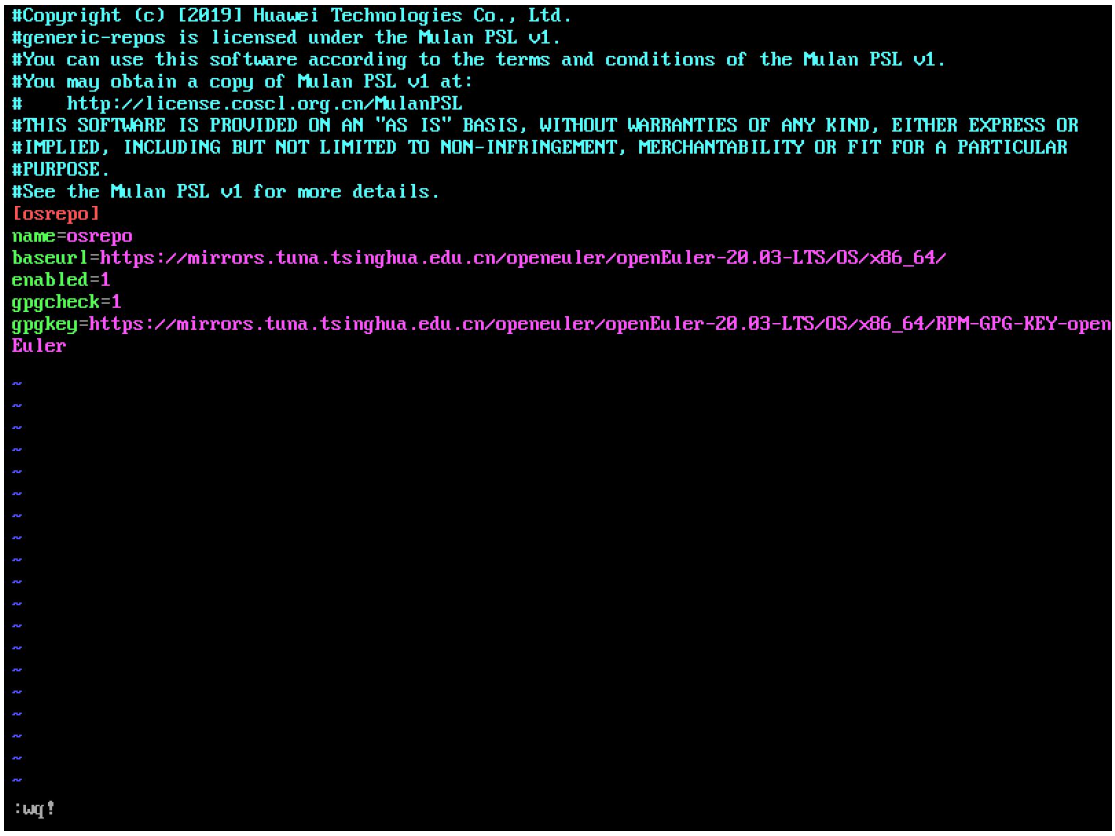
```
vim /etc/yum.repos.d/openEuler_x86_64.repo
```

```
[bupt_wanghejia2023211603@localhost ~]$ sudo vim /etc/yum.repos.d/openEuler_x86_64.repo

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for bupt_wanghejia2023211603:
```

按 i 进行插入，内容如下：

```
[osrepo]
name=osrepo
baseurl=https://mirrors.tuna.tsinghua.edu.cn/openeuler/openEuler-20.03-LTS/OS/x86_64/
enabled=1
gpgcheck=1
gpgkey=https://mirrors.tuna.tsinghua.edu.cn/openeuler/openEuler-20.03-LTS/OS/x86_64/RPM-GPG-KEY-openEuler
```

插入完按 ESC 退出 INSERT，再按:wq!保存并退出



安装 gnome 和相关组件

```
sudo dnf install gnome-shell gdm gnome-session
```

```
  libwacom-data-0.31-2.oe1.x86_64
  libxkbcommon-x11-0.8.4-3.oe1.x86_64
  libxkbfile-1.1.0-2.oe1.x86_64
  libxklavier-5.4-18.oe1.x86_64
  llvm-libs-7.0.0-9.oe1.x86_64
  lua-expat-1.3.0-16.oe1.x86_64
  lua-json-1.3.2-13.oe1.noarch
  lua-lpeg-1.0.2-2.oe1.x86_64
  lua-socket-3.0-0.19.oe1.x86_64
  mesa-demos-8.3.0-13.oe1.x86_64
  mesa-dri-drivers-18.2.2-6.oe1.x86_64
  mesa-filesystem-18.2.2-6.oe1.x86_64
  mesa-libGLU-9.0.1-1.oe1.x86_64
  mobile-broadband-provider-info-20190116-1.oe1.noarch
  mozilla-filesystem-1.9-21.oe1.x86_64
  mtdev-1.1.5-15.oe1.x86_64
  mutter-3.30.1-7.oe1.x86_64
  nm-connection-editor-1.8.22-2.oe1.x86_64
  pipewire-0.2.7-1.oe1.x86_64
  pulseaudio-12.2-3.oe1.x86_64
  rtkit-0.11-26.oe1.x86_64
  sbc-1.4-1.oe1.x86_64
  sound-theme-freedesktop-0.8-12.oe1.noarch
  speexdsp-1.2.0-1.oe1.x86_64
  switcheroo-control-1.1-7.oe1.x86_64
  totem-pl-parser-3.26.1-5.oe1.x86_64
  upower-0.99.8-5.oe1.x86_64
  webrtc-audio-processing-0.3.1-3.oe1.x86_64
  xorg-x11-drv-libinput-0.28.0-5.oe1.x86_64
  xorg-x11-server-1.20.6-4.oe1.x86_64
  xorg-x11-xauth-1:1.1-1.oe1.x86_64
  xorg-x11-xinit-1.4.0-5.oe1.x86_64
  xorg-x11-xkb-utils-7.7-28.oe1.x86_64
  zenity-3.30.0-2.oe1.x86_64

Complete!
[bupt_wanghejia20232116030localhost ~]$
```

安装 terminal

```
sudo dnf install gnome-terminal
```

```
  Verifying        : gvfs-1.40.2-6.oe1.x86_64                                    5/24
  Verifying        : gvfs-client-1.40.2-6.oe1.x86_64                             6/24
  Verifying        : libcdio-2.0.0-8.oe1.x86_64                                  7/24
  Verifying        : libcdio-paranoia-10.2+2.0.0-2.oe1.x86_64                    8/24
  Verifying        : libexif-0.6.21-20.oe1.x86_64                                9/24
  Verifying        : libgexiv2-0.10.8-5.oe1.x86_64                              10/24
  Verifying        : libgsf-1.14.43-4.oe1.x86_64                                11/24
  Verifying        : libgxps-0.3.1-1.oe1.x86_64                                 12/24
  Verifying        : libiptcdata-1.0.5-1.oe1.x86_64                             13/24
  Verifying        : libosinfo-1.2.0-9.oe1.x86_64                               14/24
  Verifying        : nautilus-3.33.90-3.oe1.x86_64                              15/24
  Verifying        : osinfo-db-20180920-2.oe1.x86_64                            16/24
  Verifying        : osinfo-db-tools-1.2.0-3.oe1.x86_64                         17/24
  Verifying        : poppler-0.67.0-5.oe1.x86_64                                18/24
  Verifying        : poppler-data-0.4.9-4.oe1.noarch                            19/24
  Verifying        : poppler-glib-0.67.0-5.oe1.x86_64                           20/24
  Verifying        : taglib-1.11.1-12.oe1.x86_64                                21/24
  Verifying        : tracker-2.1.5-3.oe1.x86_64                                 22/24
  Verifying        : tracker-miners-2.1.5-6.oe1.x86_64                          23/24
  Verifying        : vte291-0.54.1-4.oe1.x86_64                                 24/24

Installed:
  gnome-terminal-3.30.1-3.oe1.x86_64          exempi-2.4.5-4.oe1.x86_64
  exiv2-0.26-17.oe1.x86_64                    gnome-autoar-0.2.3-4.oe1.x86_64
  gvfs-1.40.2-6.oe1.x86_64                    gvfs-client-1.40.2-6.oe1.x86_64
  libcdio-2.0.0-8.oe1.x86_64                  libcdio-paranoia-10.2+2.0.0-2.oe1.x86_64
  libexif-0.6.21-20.oe1.x86_64                libgexiv2-0.10.8-5.oe1.x86_64
  libgsf-1.14.43-4.oe1.x86_64                 libgxps-0.3.1-1.oe1.x86_64
  libiptcdata-1.0.5-1.oe1.x86_64              libosinfo-1.2.0-9.oe1.x86_64
  nautilus-3.33.90-3.oe1.x86_64               osinfo-db-20180920-2.oe1.x86_64
  osinfo-db-tools-1.2.0-3.oe1.x86_64          poppler-0.67.0-5.oe1.x86_64
  poppler-data-0.4.9-4.oe1.noarch             poppler-glib-0.67.0-5.oe1.x86_64
  taglib-1.11.1-12.oe1.x86_64                 tracker-2.1.5-3.oe1.x86_64
  tracker-miners-2.1.5-6.oe1.x86_64           vte291-0.54.1-4.oe1.x86_64

Complete!
[bupt_wanghejia20232116030localhost ~]$ _
```

设置开机自启动

```
sudo systemctl enable gdm.service
sudo systemctl set-default graphical.target
```

```
[bupt_wanghejia20232116030localhost ~]$ sudo systemctl enable gdm.service
[sudo] password for bupt_wanghejia2023211603:
[bupt_wanghejia20232116030localhost ~]$ sudo systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
```

补全丢失文件

```
cd /tmp
wget https://gitee.com/name1e5s/xsession/raw/master/Xsession
mv Xsession /etc/gdm/
chmod 0777 /etc/gdm/Xsession
```

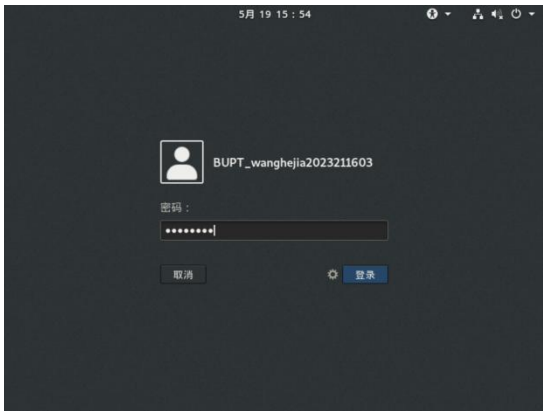
```
[root@localhost tmp]# wget https://gitee.com/name1e5s/xsession/raw/master/Xsession
--2025-05-19 15:42:41--  https://gitee.com/name1e5s/xsession/raw/master/Xsession
Resolving gitee.com (gitee.com)... 180.76.199.13, 180.76.198.77, 180.76.198.225
Connecting to gitee.com (gitee.com)|180.76.199.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'Xsession'

Xsession                    [ <=>                                ]   5.02K  --.-KB/s    in 0s

2025-05-19 15:42:42 (62.5 MB/s) - 'Xsession' saved [5145]

[root@localhost tmp]# mv Xsession /etc/gdm/Xsession
mv: overwrite '/etc/gdm/Xsession'? y
[root@localhost tmp]# chmod 0777 /etc/gdm/Xsession
```

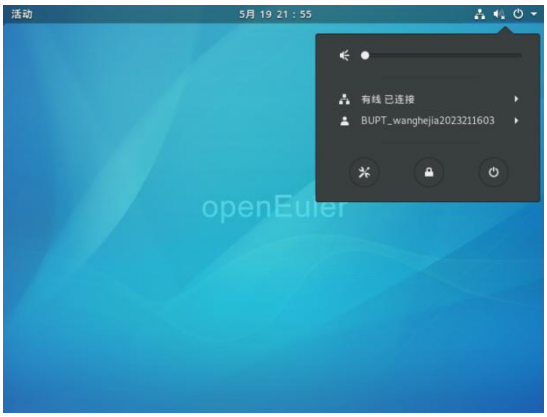
可以进入目录查看文件是否成功下载

```
ls /etc/gdm
```

```
[root@localhost tmp]# ls /etc/gdm
custom.conf  custom.confe  Init  PostLogin  PostSession  PreSession  Xsession
```

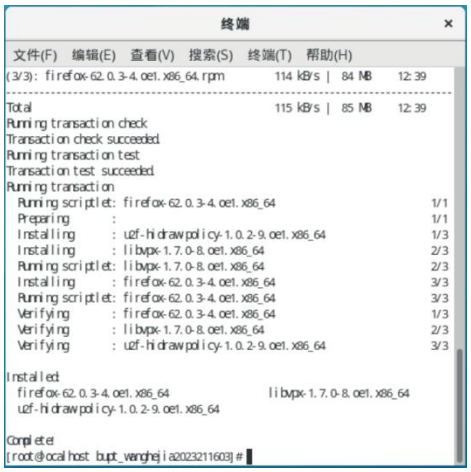
至此，可视化桌面已安装完成。重启虚拟机。

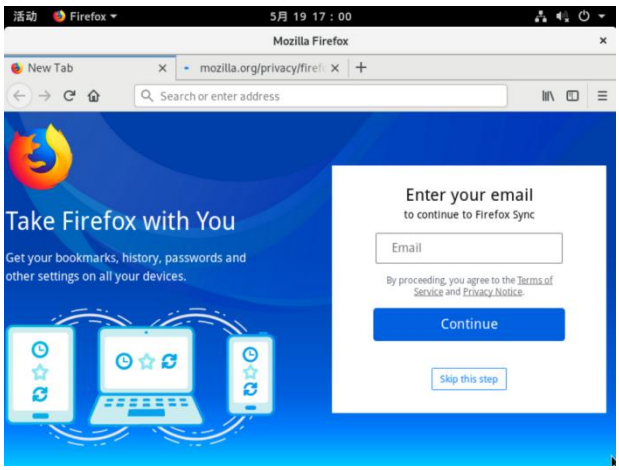




3．安装 firefox

```
yum -y install firefox
```

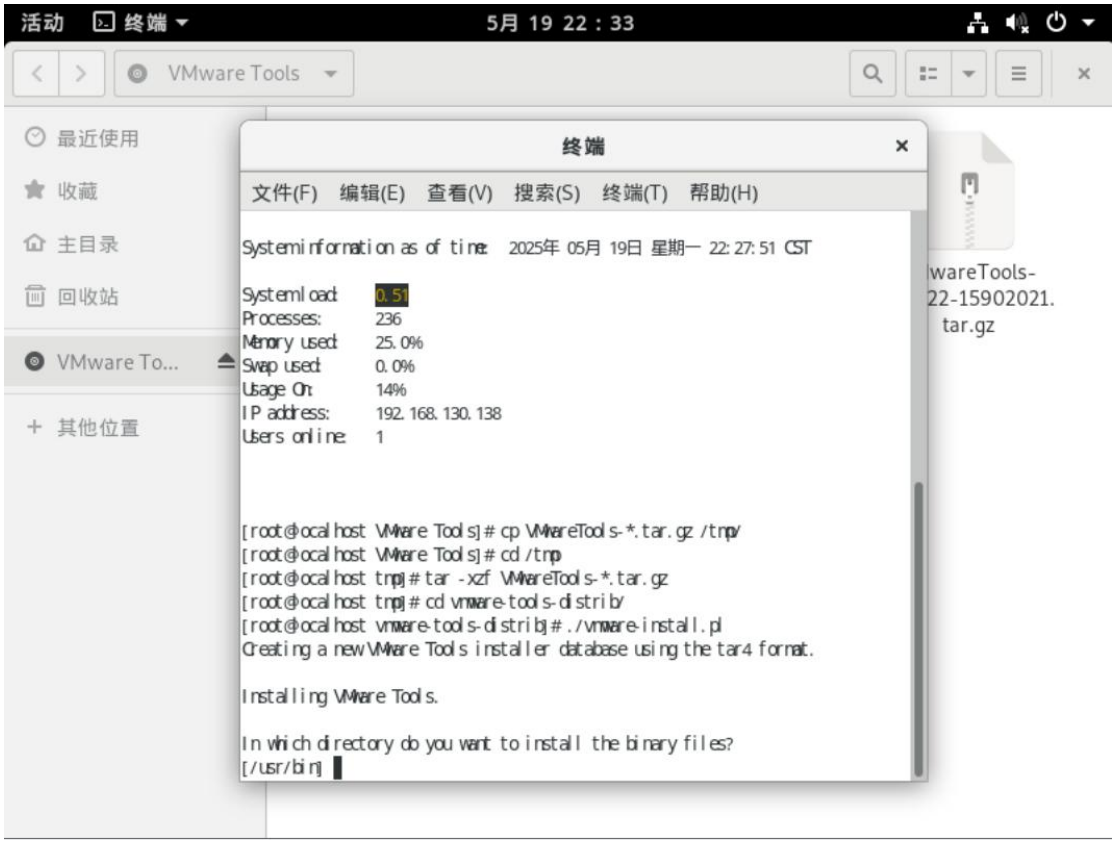




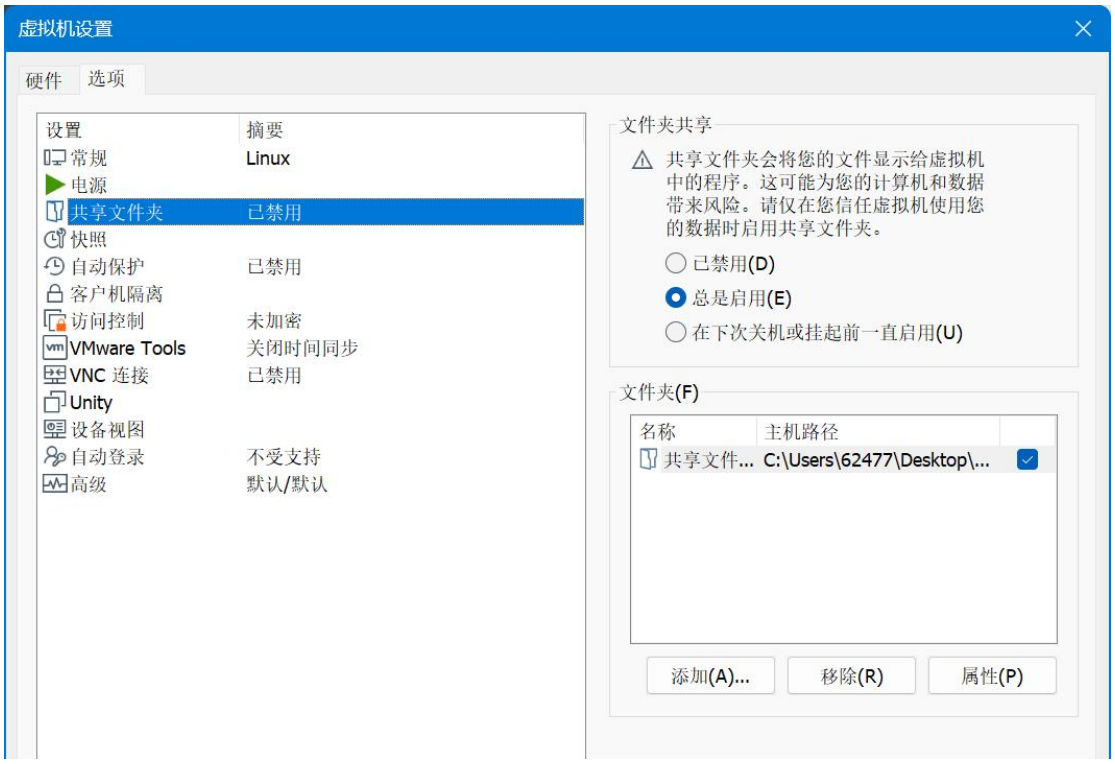
4. 安装 VMware Tools

在 VMwareTools 目录下打开终端

```
cp VMwareTools-*.tar.gz /tmp/
cd /tmp
tar -xzf VMwareTools-*.tar.gz
cd vmware-tools-distrib/
./vmware-install.pl
```



设置共享文件夹

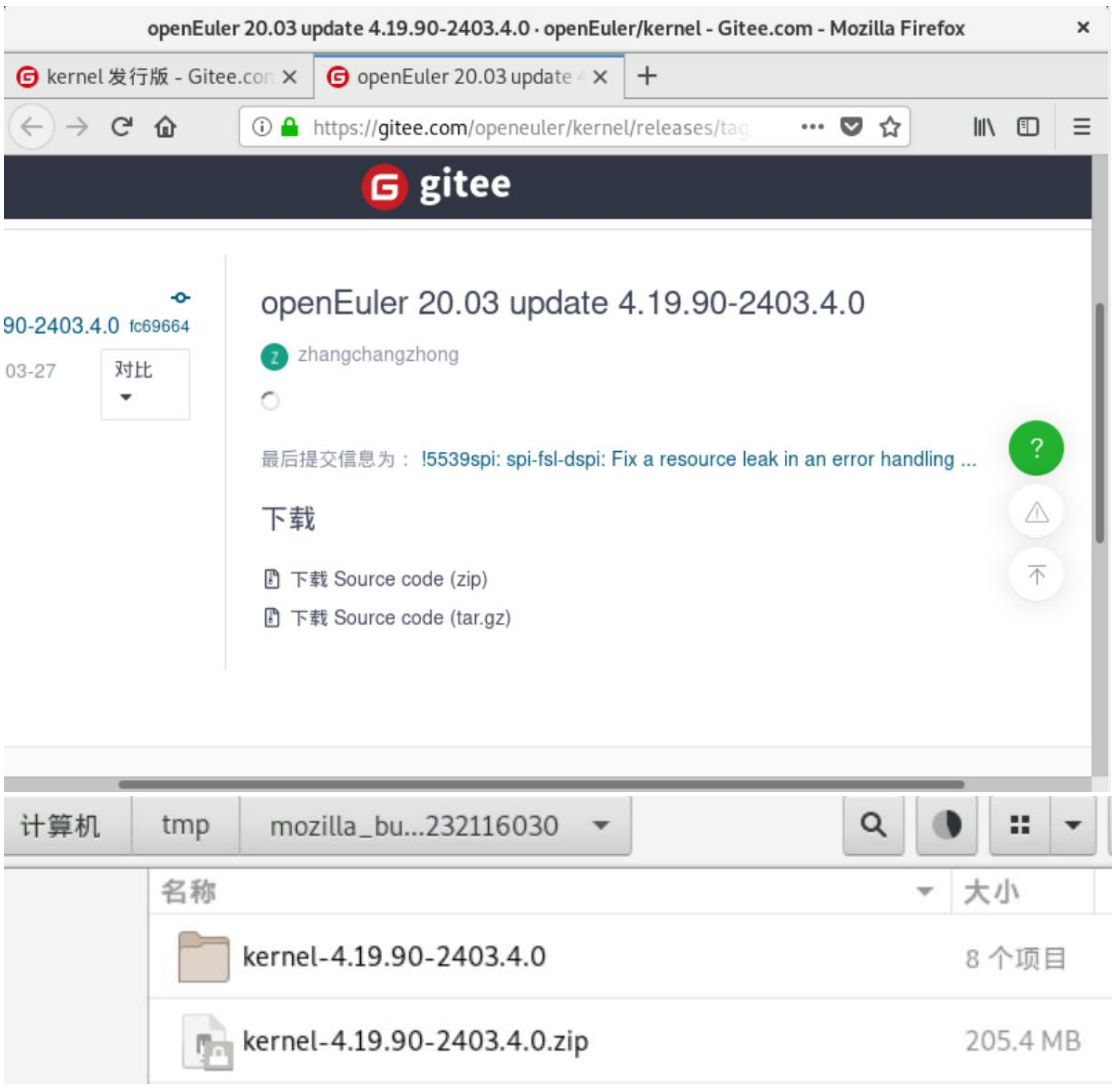


此时主机中的共享文件夹的内容会同步到 openEuler 中，方便后续操作。

5．将内核更新至最新版

采用重新编译源代码的方式将内核更新至最新版。下载最新版本的 openEuler 内核源码，见

```
https://gitee.com/openeuler/kernel/releases
```



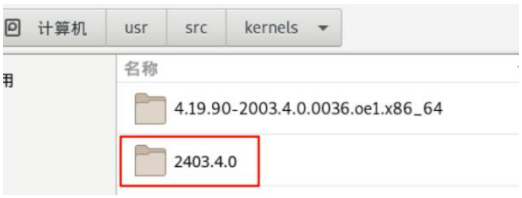
在安装目录下打开终端，移动到内核源码目录

重命名方便之后操作

```
mv kernel-4.19.90-2403.4.0 2403.4.0
```

移动文件位置

```
mv 2403.4.0/ /usr/src/kernels
```



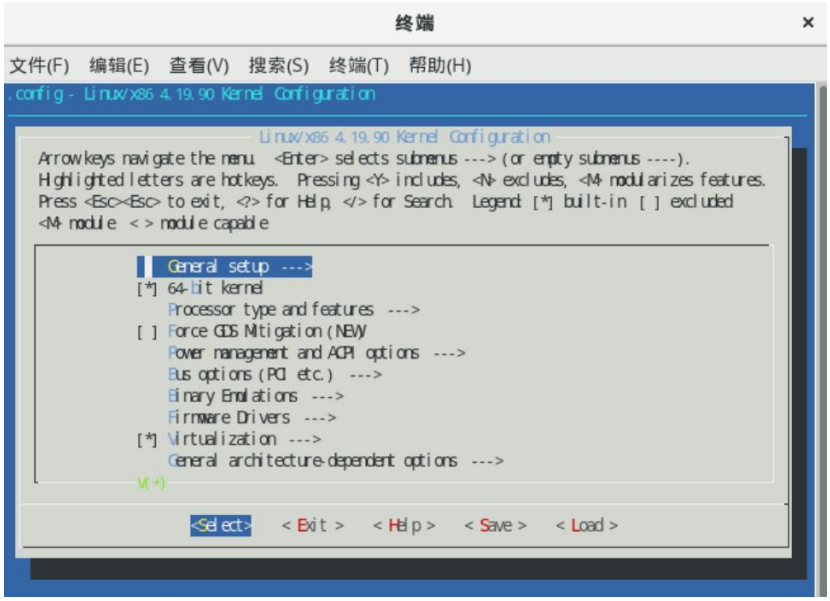
复制原配置文件

```
cp /boot/config-4.19.90-2003.4.0.0036.oe1.x86_64 /usr/src/kernels/2403.4.0
cd /usr/src/kernels/2403.4.0
ls
```

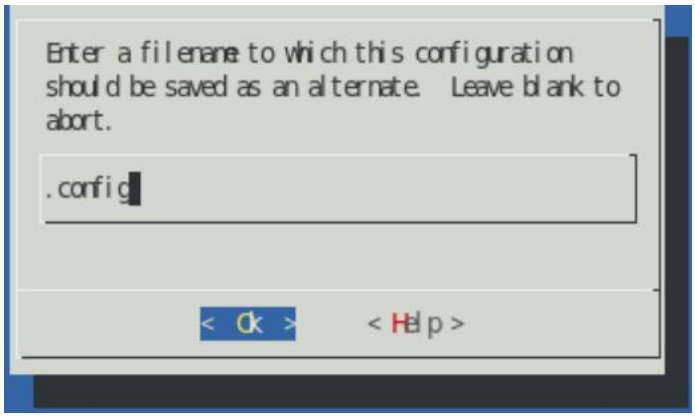
```
[root@localhost bupt_wanghejia20232116030]# cd
[root@localhost ~]# cp /boot/config-4.19.90-2003.4.0.0036.oe1.x86_64 /usr/src/kernels/2403.4.0
[root@localhost ~]# cd /usr/src/kernels/2403.4.0
[root@localhost 2403.4.0]# ls
arch                                CREDITS      fs       Kbuild   MAINTAINERS  samples  usr
block                               crypto       include  Kconfig  Makefile     scripts  virt
certs                               Documentation init    kernel   mm           security
config-4.19.90-2003.4.0.0036.oe1.x86_64 drivers  ipc      lib      net          sound
COPYING                             firmware     kabi     LICENSES README       tools
[root@localhost 2403.4.0]#
```

安装依赖，更新配置

```
yum install ncurses-devel
make menuconfig
```



选择“save”，生成配置文件.config



查看，生成成功。

```
[root@localhost 2403.4.0]# ls -l .config
-rw-------. 1 root root 184002  5月 20 00:28 .config
```

安装编译所需组件

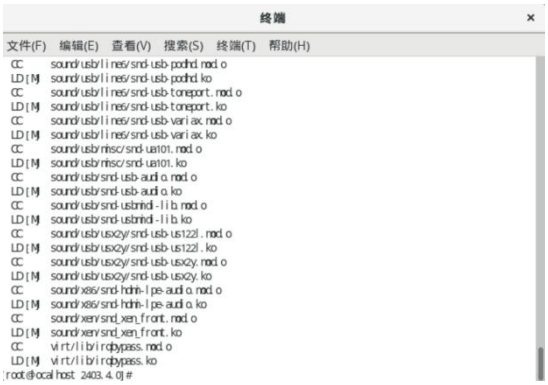
```
yum install elfutils-libelf-devel
yum install openssl-devel
yum install bc
```

```
Installed:
  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

Complete!
[root@localhost ~]# yum install bc
Last metadata expiration check: 2:28:46 ago on 2025年05月19日 星期一 22时04分02秒.
Package bc-1.07.1-10.oe1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

编译安装

```
make
```

```
[root@localhost ~]# cd /usr/src/kernels/2403.4.0
[root@localhost 2403.4.0]# make
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --syncconfig Kconfig
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
  HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  HOSTCC  arch/x86/tools/relocs_32.o
  HOSTCC  arch/x86/tools/relocs_64.o
```



安装完成

安装模块

```
make modules_install
```

安装内核

```
make install
```

```
[root@localhost 2403.4.0]# make install
sh ./arch/x86/boot/install.sh 4.19.90 arch/x86/boot/bzImage \
        System.map "/boot"
```

查看新内核是否安装成功

```
ll
```

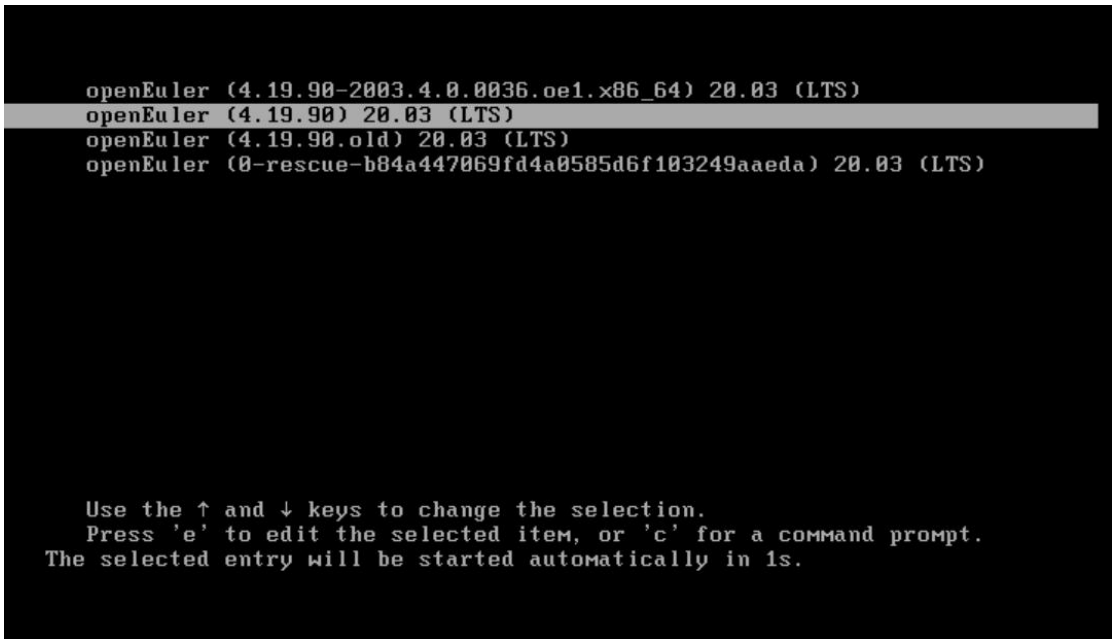
```
[root@localhost buptwanghejia2023211603]# cd /boot/
[root@localhost boot]# ll
总用量 224M
-rwxr--r--. 1 root root 179K 3月  24  2020 config-4.19.90-2003.4.0.0036.oe1.x86_64
drwxr-xr-x. 3 root root 4.0K 5月  20 05:03 efi
drwx------. 5 root root 4.0K 5月  20 13:29 grub2
-rw-------. 1 root root  68M 5月  19 21:12 initramfs-0-rescue-b84a447069fd4a0585d6f103249aaeda.img
-rw-------. 1 root root  23M 5月  19 21:13 initramfs-4.19.90-2003.4.0.0036.oe1.x86_64.img
-rw-------. 1 root root  20M 5月  19 21:54 initramfs-4.19.90-2003.4.0.0036.oe1.x86_64kdump.img
-rw-------. 1 root root  72M 5月  20 13:29 initramfs-4.19.90.img
drwxr-xr-x. 3 root root 4.0K 5月  19 21:09 loader
drwx------. 2 root root  16K 5月  20 05:02 lost+found
-rwxr--r--. 1 root root 326K 3月  24  2020 symvers-4.19.90-2003.4.0.0036.oe1.x86_64.gz
lrwxrwxrwx. 1 root root   24 5月  20 13:33 System.map -> /boot/System.map-4.19.90
-rw-------. 1 root root 3.6M 5月  20 13:33 System.map-4.19.90
-rwxr--r--. 1 root root 3.5M 3月  24  2020 System.map-4.19.90-2003.4.0.0036.oe1.x86_64
-rw-------. 1 root root 3.6M 5月  20 13:27 System.map-4.19.90.old
lrwxrwxrwx. 1 root root   21 5月  20 13:33 vmlinuz -> /boot/vmlinuz-4.19.90
-rwxr-xr-x. 1 root root 7.7M 5月  19 21:12 vmlinuz-0-rescue-b84a447069fd4a0585d6f103249aaeda
-rw-------. 1 root root 7.9M 5月  20 13:33 vmlinuz-4.19.90
-rwxr-xr-x. 1 root root 7.7M 3月  24  2020 vmlinuz-4.19.90-2003.4.0.0036.oe1.x86_64
```

更新引导

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

```
[root@localhost boot]# grub2-mkconfig -o /boot/grub2/grub.cfg
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-4.19.90.old
Found initrd image: /boot/initramfs-4.19.90.img
Found linux image: /boot/vmlinuz-0-rescue-b84a447069fd4a0585d6f103249aaeda
Found initrd image: /boot/initramfs-0-rescue-b84a447069fd4a0585d6f103249aaeda.img
done
```

重启，可以看到编译成功的新内核，选择新内核回车



成功进入系统后，查看默认启动内核

```
grub2-editenv list
```

```
[root@localhost bupt_wanghejia2023211603]# grub2-editenv list
saved_entry=openEuler (4.19.90) 20.03 (LTS)
boot_success=0
```

查看内核版本，成功安装了新内核

```
uname -a
```

```
[root@localhost bupt_wanghejia2023211603]# uname -a
Linux localhost.localdomain 4.19.90 #1 SMP Tue May 20 00:38:07 CST 2025 x86_64 x86_64 x86_64 GNU/Linux
```

6．内核模块编程

编写 helloworld.c 和 Makefile

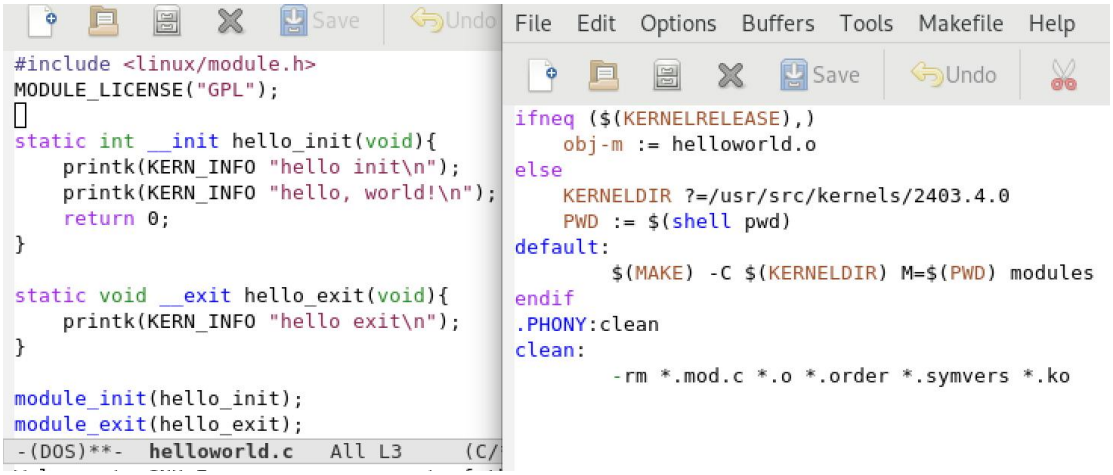
helloworld.c

```
#include <linux/module.h>
MODULE_LICENSE("GPL");

static int __init hello_init(void){
        printk(KERN_INFO "hello init\n");
        printk(KERN_INFO "hello, world!\n");
        return 0;
}
static void __exit hello_exit(void){
        printk(KERN_INFO "hello exit\n");
```

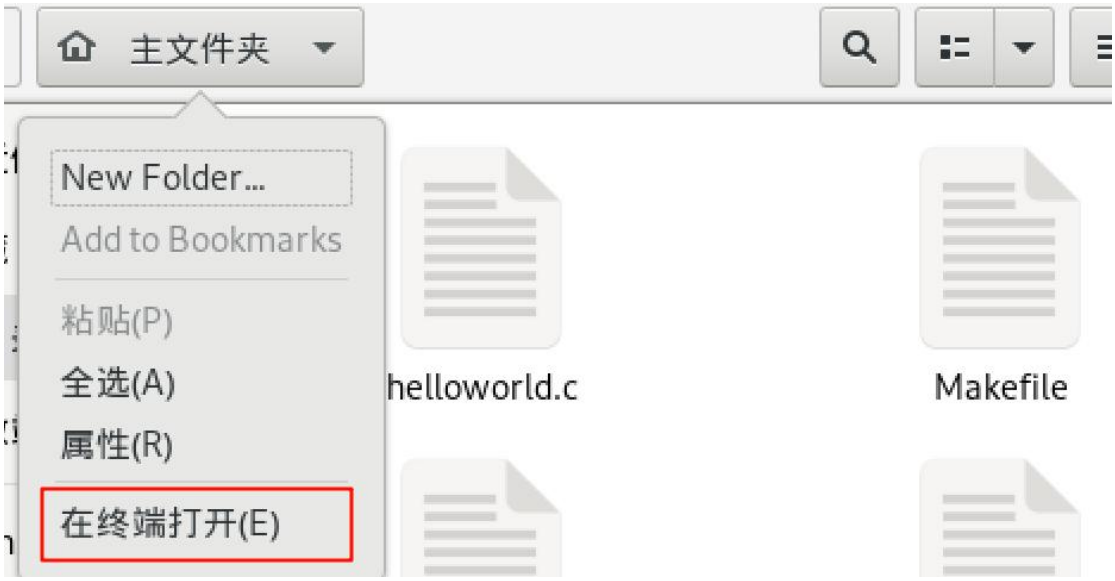
```
}
module_init(hello_init);
module_exit(hello_exit);
```

Makefile

```
ifneq ($(KERNELRELEASE),)
	obj-m := helloworld.o
else
	KERNELDIR ?=/usr/src/kernels/2403.4.0
	PWD := $(shell pwd)
default:
	$(MAKE) -C $(KERNELDIR) M=$(PWD) modules
endif
.PHONY:clean
clean:
	-rm *.mod.c *.o *.order *.symvers *.ko
```



在源代码目录下打开终端




```
make
```

```
[bupt_wanghejia2023211603@localhost ~]$ sudo make
[sudo] bupt_wanghejia2023211603 的密码：
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603 modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/helloworld.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/helloworld.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/helloworld.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
```

运行后查看文件列表

```
ll
```

```
[bupt_wanghejia2023211603@localhost ~]$ ll
总用量 452K
-rwxr-xr-x. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603   309 5月  20 14:25 #helloworld.c#
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603   331 5月  20 14:21 helloworld.c
-rw-------. 1 root                     root                     210K 5月  20 14:22 helloworld.ko
-rw-------. 1 root                     root                      843 5月  20 14:22 helloworld.mod.c
-rw-------. 1 root                     root                     106K 5月  20 14:22 helloworld.mod.o
-rw-------. 1 root                     root                     106K 5月  20 14:22 helloworld.o
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603   251 5月  20 14:09 Makefile
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603   254 5月  20 14:09 Makefile~
-rw-------. 1 root                     root                       52 5月  20 14:22 modules.order
-rw-------. 1 root                     root                        0 5月  20 14:22 Module.symvers
```

insmod helloworld.ko	加载内核模块
dmesg | tail -n 2	查看打印信息
lsmod | tail -n 2	查看内核模块
rmmod helloworld	卸载内核模块

```
[root@ocalhost bupt_wanghejia2023211603]# insmod helloworld.ko
[root@ocalhost bupt_wanghejia2023211603]# dmesg | tail -n 2
[ 3144.824612] hello init
[ 3144.824613] hello, world
[root@ocalhost bupt_wanghejia2023211603]# lsmod | tail -n 2
dm_log                 20480  2 dm_region_hash,dm_mirror
dm_mod                155648  8 dm_log,dm_mirror
[root@ocalhost bupt_wanghejia2023211603]# rmmod helloworld
```

7．内存管理

（1）使用 kmalloc 分配 1KB，8KB 的内存，并打印指针地址

kmalloc.c

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

MODULE_LICENSE("GPL");

unsigned char *kmallocmem1;
unsigned char *kmallocmem2;

static int __init mem_module_init(void)
{
	printk(KERN_INFO "Start kmalloc!\n");
	kmallocmem1 = (unsigned char*)kmalloc(1024, GFP_KERNEL);
	if (kmallocmem1 != NULL) {
		printk(KERN_ALERT "kmallocmem1 addr = %p\n", (void *)kmallocmem1);
	} else {
		printk(KERN_ERR "Failed to allocate kmallocmem1!\n");
	}
	kmallocmem2 = (unsigned char *)kmalloc(8192, GFP_KERNEL);
	if (kmallocmem2 != NULL) {
		printk(KERN_ALERT "kmallocmem2 addr = %p\n", (void *)kmallocmem2);
	} else {
		printk(KERN_ERR "Failed to allocate kmallocmem2!\n");
	}
	return 0;
}

static void __exit mem_module_exit(void)
{
	if (kmallocmem1) {
		kfree(kmallocmem1);
```

```
        }
        if (kmallocmem2) {
            kfree(kmallocmem2);
        }
        printk(KERN_INFO "Exit kmalloc!\n");
    }

    module_init(mem_module_init);
    module_exit(mem_module_exit);
```

Makefile 文件只需修改文件名（后续操作相同，不再赘述）

```
    obj-m := kmalloc.o
```

编译模块（操作相似，后续不再赘述）

```
    make
    insmod kmalloc.ko
    dmesg | tail -n 3
    rmmod kmalloc
    dmesg | tail -n 4
```

```
[root@localhost test]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test/kmalloc.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test/kmalloc.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test/kmalloc.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
[root@localhost test]# insmod kmalloc.ko
[root@localhost test]# dmesg | tail -n 3
[ 6140.864207] Start kmalloc!
[ 6140.864209] kmallocmem1 addr = 00000009d63dfc3
[ 6140.864213] kmallocmem2 addr = 00000000fb46d94
[root@localhost test]# rmmod kmalloc
[root@localhost test]# dmesg | tail -n 4
[ 6140.864207] Start kmalloc!
[ 6140.864209] kmallocmem1 addr = 00000009d63dfc3
[ 6140.864213] kmallocmem2 addr = 00000000fb46d94
[ 6161.699555] Exit kmalloc!
```

查看内存布局

```
    cd Documentation/x86/x86_64/
```

```
cat mm.txt
```

```
[root@ocalhost 2403.4.0]# cd Documentation/x86/x86_64/
[root@ocalhost x86_64]# ls
00-INDEX          boot-options.txt  fake-numa-for-cpusets  mm.txt
5level-paging.txt  cpu-hotplug-spec  machinecheck           uefi.txt
[root@ocalhost x86_64]# cat mm.txt

Virtual memory map with 4 level page tables:

0000000000000000 - 00007fffffffffff (=47 bits) user space, different per mm
hole caused by [47:63] sign extension
ffff800000000000 - ffff87ffffffffff (=43 bits) guard hole, reserved for hypervisor
ffff880000000000 - ffff887fffffffff (=39 bits) LDT remap for PTI
ffff888000000000 - ffffc87fffffffff (=64 TB) direct mapping of all phys. memory
ffffc88000000000 - ffffc8ffffffffff (=39 bits) hole
ffffc90000000000 - ffffe8ffffffffff (=45 bits) vmalloc/ioremap space
ffffe90000000000 - ffffe9ffffffffff (=40 bits) hole
ffffea0000000000 - ffffeaffffffffff (=40 bits) virtual memory map (1TB)
... unused hole ...
ffffec0000000000 - fffffbffffffffff (=44 bits) kasan shadow memory (16TB)
... unused hole ...
                                 vaddr_end for KASLR
fffffe0000000000 - fffffe7fffffffff (=39 bits) cpu_entry_area mapping
fffffe8000000000 - fffffeffffffffff (=39 bits) LDT remap for PTI
ffffff0000000000 - ffffff7fffffffff (=39 bits) %esp fixup stacks
```

结果分析得出，kmalloc 分配的内存地址位于内核空间

（2）使用 vmalloc 分别分配 8KB、1MB、64MB 的内存，打印指针地址

vmalloc.c

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

MODULE_LICENSE("GPL");

unsigned char *vmallocmem1;
unsigned char *vmallocmem2;
unsigned char *vmallocmem3;

static int __init mem_module_init(void)
{
    printk(KERN_INFO "Start vmalloc!\n");
    vmallocmem1 = (unsigned char*)vmalloc(8192);
    if (vmallocmem1 != NULL) {
        printk(KERN_INFO "vmallocmem1 addr = %lx\n", (unsigned long)vmallocmem1);
```

```
        } else {
            printk(KERN_ERR "Failed to allocate vmallocmem1!\n");
        }
        vmallocmem2 = (unsigned char*)vmalloc(1048576);
        if (vmallocmem2 != NULL) {
            printk(KERN_INFO "vmallocmem2 addr = %lx\n", (unsigned long)vmallocmem2);
        } else {
            printk(KERN_ERR "Failed to allocate vmallocmem2!\n");
        }
        vmallocmem3 = (unsigned char*)vmalloc(67108864);
        if (vmallocmem3 != NULL) {
            printk(KERN_INFO "vmallocmem3 addr = %lx\n", (unsigned long)vmallocmem3);
        } else {
            printk(KERN_ERR "Failed to allocate vmallocmem3!\n");
        }
        return 0;
    }

    static void __exit mem_module_exit(void)
    {
        vfree(vmallocmem1);
        vfree(vmallocmem2);
        vfree(vmallocmem3);
        printk(KERN_INFO "Exit vmalloc!\n");
    }

    module_init(mem_module_init);
    module_exit(mem_module_exit);
```

编译模块

```
[root@ocalhost test2]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test2 modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test2/vmalloc.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test2/vmalloc.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test2/vmalloc.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
```

```
[root@ocalhost test1]# insmod | tail -n 4
insmod: ERROR: missing filename.
[root@ocalhost test1]# insmod vmalloc.ko
[root@ocalhost test1]# dmesg | tail -n 4
[ 7216.464455] Start vmalloc!
[ 7216.464461] vmallocmem1 addr = ffffb84140679000
[ 7216.464487] vmallocmem2 addr = ffffb84142a59000
[ 7216.465653] vmallocmem3 addr = ffffb84150001000
[root@ocalhost test1]# rmmod vmalloc
[root@ocalhost test1]# dmesg | tail -n 5
[ 7216.464455] Start vmalloc!
[ 7216.464461] vmallocmem1 addr = ffffb84140679000
[ 7216.464487] vmallocmem2 addr = ffffb84142a59000
[ 7216.465653] vmallocmem3 addr = ffffb84150001000
[ 7231.903700] Exit vmalloc!
```

查看系统页表大小

```
getconf PAGE_SIZE
```

```
[root@ocalhost test1]# getconf PAGE_SIZE
4096
```

可知 vmalloc 分配的内存地址位于内核空间

8. 中断和异常处理

（1）使用 tasklet 实现打印 helloworld

tasklet_intertupt.c

```
#include <linux/module.h>
#include <linux/interrupt.h>
MODULE_LICENSE("GPL");
static struct tasklet_struct my_tasklet;
static void tasklet_handler(unsigned long data){
        printk(KERN_INFO "Hello World! tasklet is working...\n");
}
static int __init mytasklet_init(void){
        printk(KERN_INFO "Start tasklet module...\n");
        tasklet_init(&my_tasklet, tasklet_handler, 0);
        tasklet_schedule(&my_tasklet);
        return 0;
}
static void __exit mytasklet_exit(void){
        tasklet_kill(&my_tasklet);
        printk(KERN_INFO "Exit tasklet module...\n");
}
module_init(mytasklet_init);
```

```
module_exit(mytasklet_exit);
```

编译运行

```
[root@localhost test3]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test3 modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test3/tasklet_intertupt.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test3/tasklet_intertupt.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test3/tasklet_intertupt.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
[root@localhost test3]# insmod tasklet_intertupt.ko
[root@localhost test3]# dmesg | tail -n 2
[ 7858.368072] Start tasklet module...
[ 7858.368092] Hello World! tasklet is working..
[root@localhost test3]# rmmod tasklet_intertupt
[root@localhost test3]# dmesg | tail -n 3
[ 7858.368072] Start tasklet module...
[ 7858.368092] Hello World! tasklet is working..
[ 7878.687021] Exit tasklet module...
```

（2）用工作队列实现周期打印 helloworld

workqueue_text.c

```
#include <linux/module.h>
#include <linux/workqueue.h>
#include <linux/delay.h>
MODULE_LICENSE("GPL");

static struct workqueue_struct *queue = NULL;
static struct delayed_work mywork;
static int i = 0;
void work_handle(struct work_struct *work){
        printk(KERN_ALERT "Hello World!\n");
}
static int __init timewq_init(void){
        printk(KERN_ALERT "Start workqueue_test module.\n");
        queue = create_singlethread_workqueue("workqueue_test");
        if(queue == NULL){
                printk(KERN_ALERT "Failed to create workqueue_test!\n");
                return -1;
        }
        INIT_DELAYED_WORK(&mywork, work_handle);
        for(;i <= 3; i++){
                queue_delayed_work(queue, &mywork, 5 * HZ);
```

```
            ssleep(15);
        }
        return 0;
    }
    static void __exit timewq_exit(void){
        flush_workqueue(queue);
        destroy_workqueue(queue);
        printk(KERN_ALERT "Exit workqueue_test module.\n");
    }
    module_init(timewq_init);
    module_exit(timewq_exit);
```

编译运行

```
[root@ocalhost test]# dmesg | tail -n 5
[ 9342.994893] Start workqueue_test module.
[ 9348.049683] Hello World!
[ 9363.409676] Hello World!
[ 9378.763324] Hello World!
[ 9394.121182] Hello World!
```

打印 4 次“Hello World!\n”，模块加载之后 5 秒开始打印，每次打印之间休眠 15 秒

（3）编写一个信号捕获程序，捕获终端按键信号

catch_signal.c

```
#include <signal.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
void signal_handler(int sig){
    switch(sig) {
        case SIGINT:
            printf("\nGet a signal:SIGINT. You pressed ctrl+c.\n");
            break;
        case SIGQUIT:
            printf("\nGet a signal:SIGQUIT. You pressed ctrl+\\.\n");
            break;
        case SIGTSTP:
            printf("\nGet a signal:SIGTSTP. You pressed ctrl+z.\n");
            break;
    }
    exit(0);
}
int main(){
```

```
        printf("Current process ID is %d\n", getpid());
        signal(SIGINT, signal_handler);
        signal(SIGQUIT, signal_handler);
        signal(SIGTSTP, signal_handler);
        for(;;);
        return 0;
    }
```

编译运行

```
ll
gcc catch_signal.c -o catch_signal
ll
./catch_signal          重复捕捉几次
```

```
[root@ocalhost test]# ll
总用量 16K
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 791  5月 20 16:27 catch_signal.c
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 253  5月 20 16:28 Makefile
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 251  5月 20 14:09 Makefile~
-rw------. 1 root                      root                      59  5月 20 16:28 modules.order
[root@ocalhost test]# gcc catch_signal.c -o catch_signal
[root@ocalhost test]# ll
总用量 36K
-rwx------. 1 root                      root                     17K  5月 20 16:29 catch_signal
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 791  5月 20 16:27 catch_signal.c
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 253  5月 20 16:28 Makefile
-rwxrwxrwx. 1 bupt_wanghejia2023211603 bupt_wanghejia2023211603 251  5月 20 14:09 Makefile~
-rw------. 1 root                      root                      59  5月 20 16:28 modules.order
[root@ocalhost test]#./catch_signal
Current process ID is 28024
^C
Get a signal:SIGINT. You pressed ctrl+c.
[root@ocalhost test]#./catch_signal
Current process ID is 28037
^C
Get a signal:SIGINT. You pressed ctrl+c.
[root@ocalhost test]#./catch_signal
Current process ID is 28050
^C
Get a signal:SIGINT. You pressed ctrl+c.
[root@ocalhost test]#jobs
```

9．内核时间管理

（1）调用内核时钟接口打印当前时间

current_time.c

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

```
#include <linux/rtc.h>
MODULE_LICENSE("GPL");
struct timeval tv;
struct rtc_time tm;
static int __init currenttime_init(void){
        int year, mon, day, hour, min, sec;
        printk(KERN_INFO "Start current_time module...\n");
        do_gettimeofday(&tv);
        rtc_time_to_tm(tv.tv_sec, &tm);
        year = tm.tm_year + 1900;
        mon = tm.tm_mon + 1;
        day = tm.tm_mday;
        hour = tm.tm_hour + 8;
        min = tm.tm_min;
        sec = tm.tm_sec;
        printk(KERN_INFO "Current time: %d-%02d-%02d %02d:%02d:%02d\n", year, mon, day,
hour, min, sec);
        return 0;
}
static void __exit currenttime_exit(void){
        printk(KERN_INFO "Exit current_time module...\n");
}
module_init(currenttime_init);
module_exit(currenttime_exit);
```

编译运行

查看运行结果，成功再屏幕上打印出格式化时间、日期，并正确地加载和卸载

```
[root@ocalhost test]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test/current_time.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test/current_time.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test/current_time.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
[root@ocalhost test]# insmod current_time.ko
[root@ocalhost test]# dmesg | tail -n 2
[10417.678233] Start current_time module..
[10417.678236] Current time: 2025-05-20 16:39:45
[root@ocalhost test]# rmmod current_time
[root@ocalhost test]# dmesg | tail -n 3
[10417.678233] Start current_time module..
[10417.678236] Current time: 2025-05-20 16:39:45
[10433.776873] Exit current_time module..
```

（2）编写 timer，在特定时刻打印 hello,world

timer_example.c

```
#include <linux/module.h>
#include <linux/timer.h>
MODULE_LICENSE("GPL");
struct timer_list timer;
void print(struct timer_list *timer){
        printk(KERN_INFO "hello, world!\n");
}

static int __init timer_init(void){
        printk(KERN_INFO "Start timer_example module...\n");
        timer.expires = jiffies + 10 * HZ;
        timer.function = print;
        add_timer(&timer);
        return 0;
}
static void __exit timer_exit(void){
        printk(KERN_INFO "Exit timer_example module...\n");
}
module_init(timer_init);
module_exit(timer_exit);
```

编译运行

```
[root@localhost test]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test/timer_example.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test/timer_example.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test/timer_example.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
[root@localhost test]# insmod timer_example.ko
[root@localhost test]# dmesg -t | tail -n 2
perf: interrupt took too long (3229 > 3126), lowering kernel.perf_event_max_sample_rate to 61000
Start timer_example module...
[root@localhost test]# dmesg -T | tail -n 2
[二 5月 20 16:45:53 2025] Start timer_example module...
[二 5月 20 16:46:04 2025] hello, world!
[root@localhost test]# rmmod timer_example
[root@localhost test]# dmesg -T | tail -n 3
[二 5月 20 16:45:53 2025] Start timer_example module...
[二 5月 20 16:46:04 2025] hello, world!
[二 5月 20 16:46:41 2025] Exit timer_example module...
```

加载该内核模块 10 秒后打印 “hello，world！”，因为定时器执行了定时操作。

（3）调用内核时钟接口，监控累加计算代码的运行时间

sum_time.c

```
#include <linux/module.h>
#include <linux/time.h>
MODULE_LICENSE("GPL");
#define NUM 100000
struct timeval tv;
static long sum(int num){
	int i;
	long total = 0;
	for (i = 1; i <= num; i++)
		total = total + i;
	printk(KERN_INFO "The sum of 1 to %d is: %ld\n", num, total);
	return total;
}
static int __init sum_init(void){
	int start;
	int start_u;
	int end;
	int end_u;
	long time_cost;
	long s;
	printk(KERN_INFO "Start sum_time module...\n");
	do_gettimeofday(&tv);
	start = (int)tv.tv_sec;
	start_u = (int)tv.tv_usec;
	printk(KERN_INFO "The start time is: %d s %d us\n", start, start_u);
	s = sum(NUM);
	do_gettimeofday(&tv);
	end = (int)tv.tv_sec;
	end_u = (int)tv.tv_usec;
	printk(KERN_INFO "The end time is: %d s %d us\n", end, end_u);
	time_cost = (end - start) * 1000000 + (end_u - start_u);
	printk(KERN_INFO "The cost time of sum from 1 to %d is: %ld us\n", NUM, time_cost);
	return 0;
}

static void __exit sum_exit(void){
	printk(KERN_INFO "Exit sum_time module...\n");
}
module_init(sum_init);
module_exit(sum_exit);
```

编译运行

```
[root@ocalhost test]# make
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603/test modules
make[1]: 进入目录"/usr/src/kernels/2403.4.0"
  CC [M]  /home/bupt_wanghejia2023211603/test/sum_time.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/bupt_wanghejia2023211603/test/sum_time.mod.o
  LD [M]  /home/bupt_wanghejia2023211603/test/sum_time.ko
make[1]: 离开目录"/usr/src/kernels/2403.4.0"
[root@ocalhost test]# insmod sum_time.ko
[root@ocalhost test]# dmesg | tail -n 5
[11119.412791] Start sum_time module...
[11119.412793] The start time is: 1747731087 s 146999 us
[11119.412793] The sum of 1 to 100000 is: 5000050000
[11119.412794] The end time is: 1747731087 s 147000 us
[11119.412794] The cost time of sum from 1 to 100000 is: 1 us
[root@ocalhost test]# rmmod sum_time
[root@ocalhost test]# dmesg | tail -n 6
[11119.412791] Start sum_time module...
[11119.412793] The start time is: 1747731087 s 146999 us
[11119.412793] The sum of 1 to 100000 is: 5000050000
[11119.412794] The end time is: 1747731087 s 147000 us
[11119.412794] The cost time of sum from 1 to 100000 is: 1 us
[11131.308276] Exit sum_time module...
```

由程序运行结果可以看出，从 1 到 100000 的累加和所花时间是 1 us

三、问题与解决

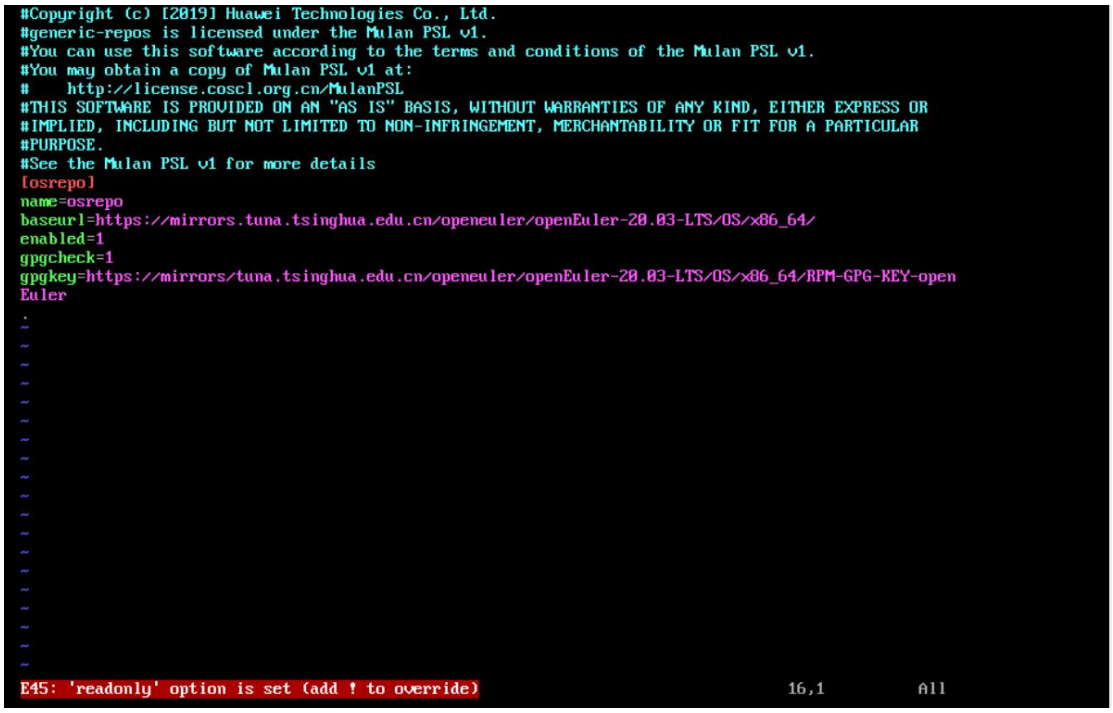
1．权限问题

（1）无法更改文件内容

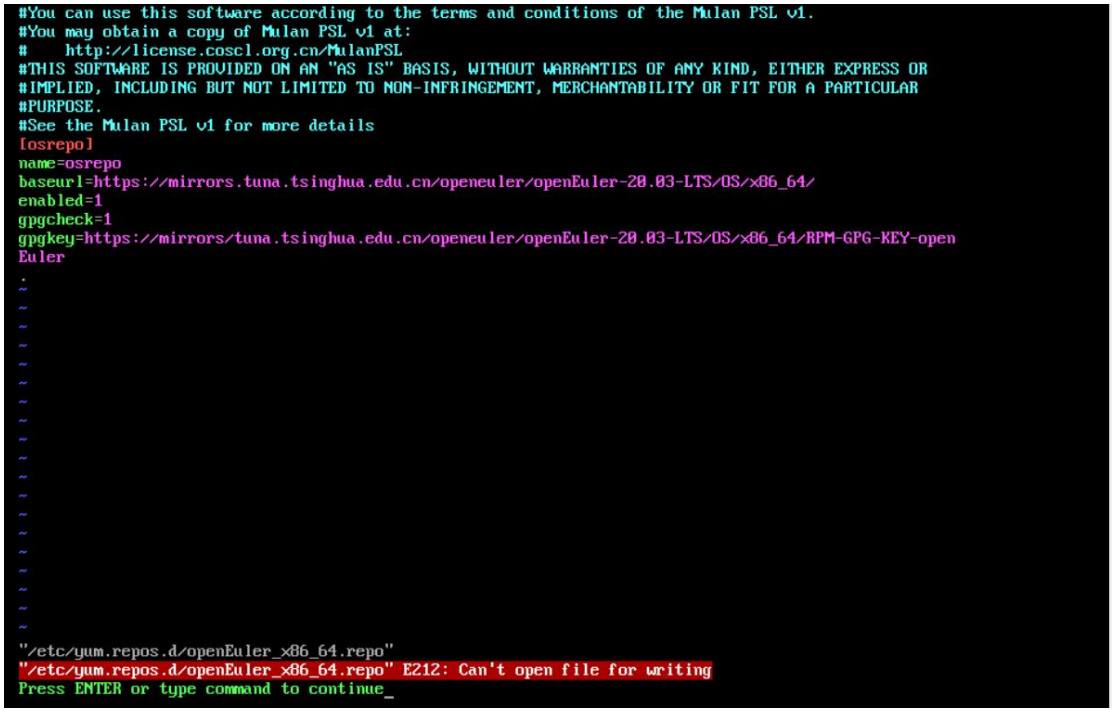
```
vim /etc/yum.repos.d/openEuler_x86_64.repo
```

```
[bupt_wanghejia2023211603@localhost ~]$ vim /etc/yum.repos.d/openEuler_x86_64.repo_
```

插入完，:wq 保存时报错



使用:wq!强制保存，又出现报错



权限不够，先:q!强制退出不保存修改，再启用 sudo 权限

```
sudo vim /etc/yum.repos.d/openEuler_x86_64.repo
```

```
[bupt_wanghejia2023211603@localhost ~]$ sudo vim /etc/yum.repos.d/openEuler_x86_64.repo

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for bupt_wanghejia2023211603:
```

（2）对文件操作时出错

移动文件夹时报错“No such file or directory”

下载时未开启 sudo 权限，重新在 sudo 权限下下载并执行 mv

提示“overwrite ‘/etc/gdm/Xsession’?”输入 y 确认覆盖即可

```
[root@localhost tmp]# wget https://gitee.com/name1e5s/xsession/raw/master/Xsession
--2025-05-19 15:42:41--  https://gitee.com/name1e5s/xsession/raw/master/Xsession
Resolving gitee.com (gitee.com)... 180.76.199.13, 180.76.198.77, 180.76.198.225
Connecting to gitee.com (gitee.com)|180.76.199.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: ‘Xsession’

Xsession                    [ <=>                                ]   5.02K  --.-KB/s    in 0s

2025-05-19 15:42:42 (62.5 MB/s) - ‘Xsession’ saved [5145]

[root@localhost tmp]# mv Xsession /etc/gdm/Xsession
mv: overwrite '/etc/gdm/Xsession'? y
```

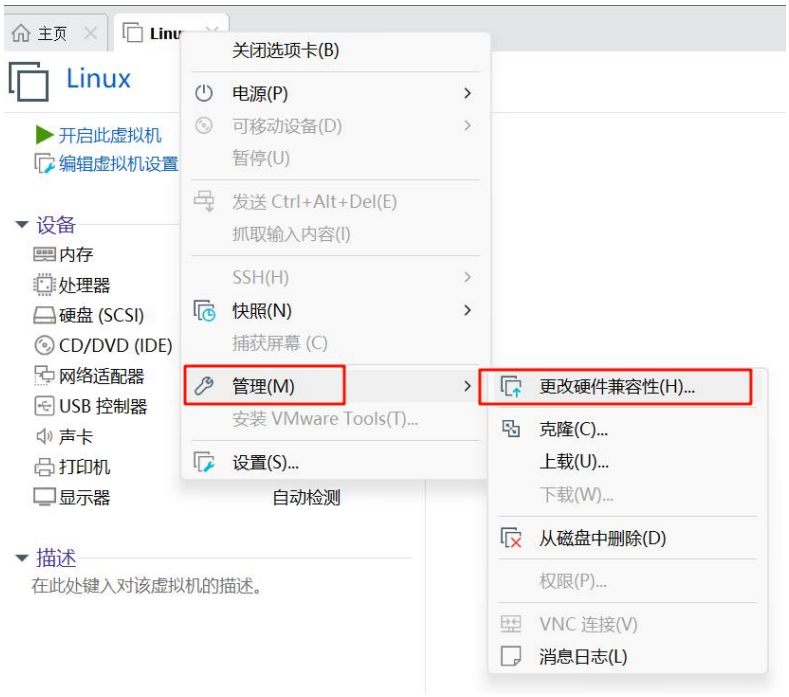
（3）其他权限问题

多数操作都需要 root 权限，因此直接 sudo su 进入 root 权限，再进行操作。

2．客户机操作系统已禁用 CPU

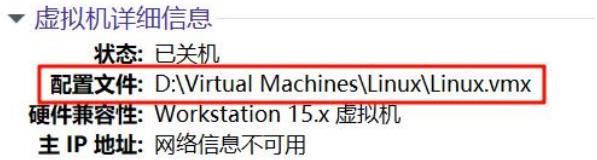
打开虚拟机设置，勾选全部

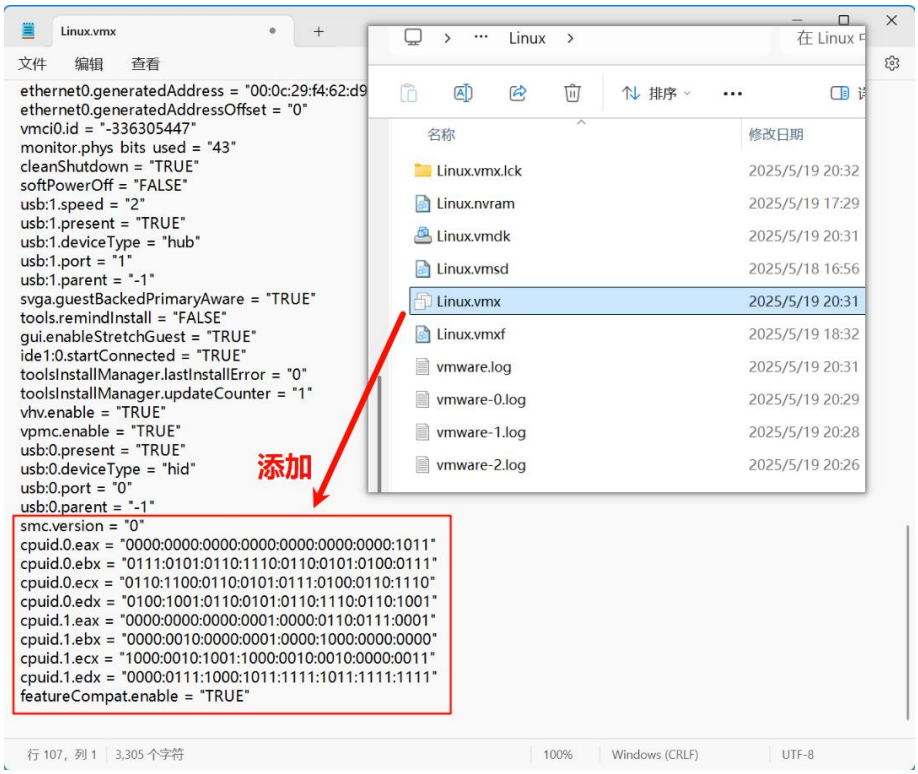
更改硬件兼容性



选择旧版

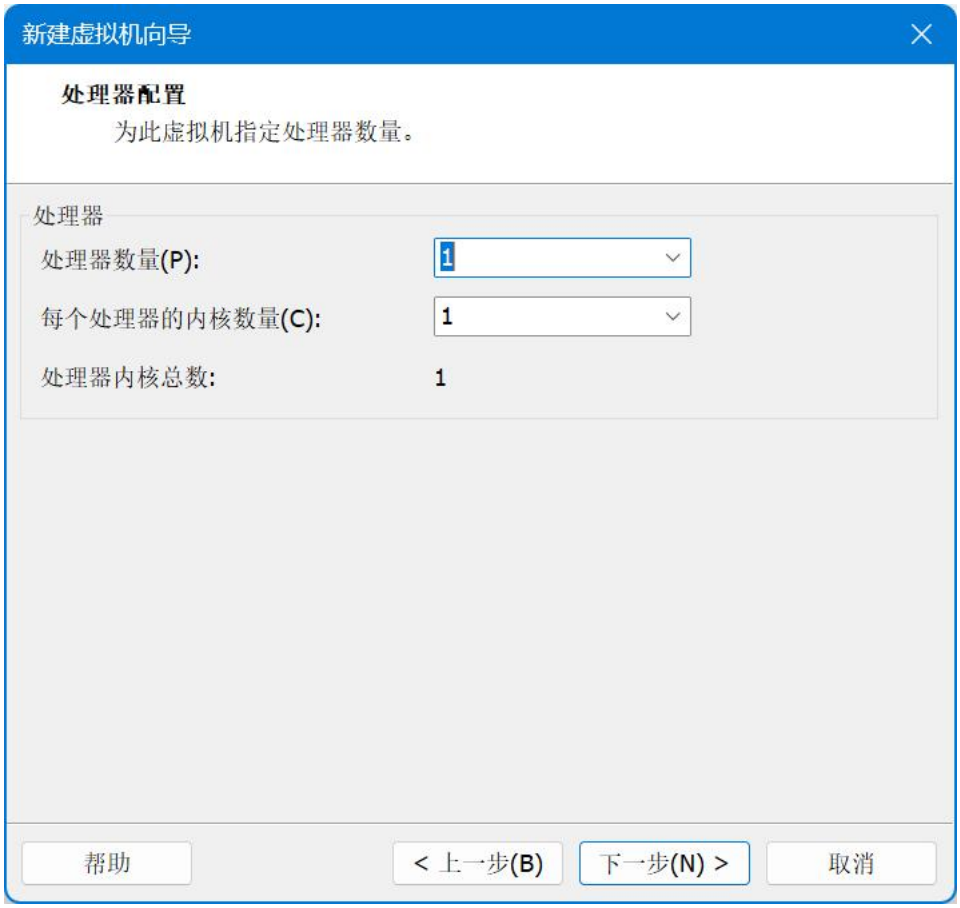
更改配置文件





然而，以上常规方法都没有解决问题

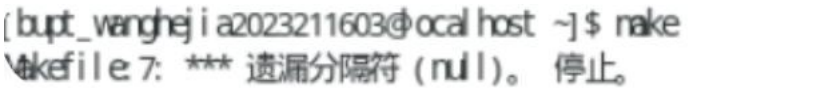
解决方法：重装系统，分配更少的 CPU 以保证稳定性



3．代码报错，虚拟机界面编程不便

安装 VMwareTools，使用共享文件夹在本机编程完同步到虚拟机。

同步之后仍有很多报错。根据提示进行修改即可。




```
[bupt_wanghejia2023211603@ocalhost ~]$ sudo make
[sudo] bupt_wanghejia2023211603 的密码：
make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603 modules
make[1]: 进入目录“/usr/src/kernels/2403.4.0”
make[2]: *** 没有规则可制作目标“/home/bupt_wanghejia2023211603/helloworld.c”，由
‘/home/bupt_wanghejia2023211603/helloworld.o’ 需求。 停止。
make[1]: *** [Makefile:1529：_module_/home/bupt_wanghejia2023211603] 错误 2
make[1]: 离开目录“/usr/src/kernels/2403.4.0”
make: *** [Makefile:7：default] 错误 2

make -C /usr/src/kernels/2403.4.0 M=/home/bupt_wanghejia2023211603 modules
make[1]: 进入目录“/usr/src/kernels/2403.4.0”
  CC [M]  /home/bupt_wanghejia2023211603/helloworld.o
/home/bupt_wanghejia2023211603/helloworld.c:2:1: 错误：程序中有游离的‘\342’
 ^
/home/bupt_wanghejia2023211603/helloworld.c:2:2: 错误：程序中有游离的‘\200’
  ^
/home/bupt_wanghejia2023211603/helloworld.c:2:3: 错误：程序中有游离的‘\213’
   ^
/home/bupt_wanghejia2023211603/helloworld.c:4:1: 错误：程序中有游离的‘\342’
 ^
/home/bupt_wanghejia2023211603/helloworld.c:4:2: 错误：程序中有游离的‘\200’
  ^
/home/bupt_wanghejia2023211603/helloworld.c:4:3: 错误：程序中有游离的‘\213’
   ^
/home/bupt_wanghejia2023211603/helloworld.c:11:1: 错误：程序中有游离的‘\342’
 ^
```