操作系统实验报告

实验（三）欧拉操作系统内核操作

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2023年5月19日
学　　院	信息学院
课程名称	操作系统

实验（三）欧拉操作系统内核操作

目录

1　实验内容

本实验通过安装openEuler操作系统、编译安装openEuler操作系统新内核以及简单的内核模块编程任务操作带领大家了解操作系统及内核编程。

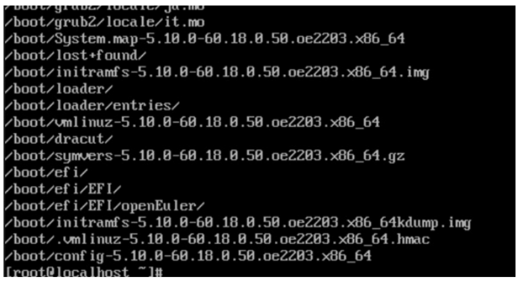
2　目的

1. 学习掌握如何安装操作系统
2. 学习掌握如何编译操作系统内核
3. 了解内核模块编程

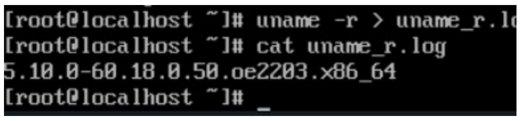
3　编译和安装

首先，备份boot目录和版本信息，如图 1

图 1: 备份



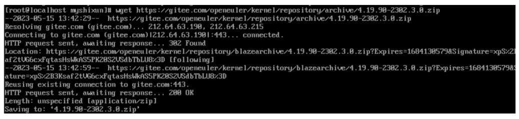
```
/boot/grub2/locale/it.mo
/boot/System.map-5.10.0-60.18.0.50.oe2203.x86_64
/boot/lost+found/
/boot/initramfs-5.10.0-60.18.0.50.oe2203.x86_64.img
/boot/loader/
/boot/loader/entries/
/boot/vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64
/boot/dracut/
/boot/symvers-5.10.0-60.18.0.50.oe2203.x86_64.gz
/boot/efi/
/boot/efi/EFI/
/boot/efi/EFI/openEuler/
/boot/initramfs-5.10.0-60.18.0.50.oe2203.x86_64kdump.img
/boot/.vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64.hmac
/boot/config-5.10.0-60.18.0.50.oe2203.x86_64
[root@localhost ~]#
```



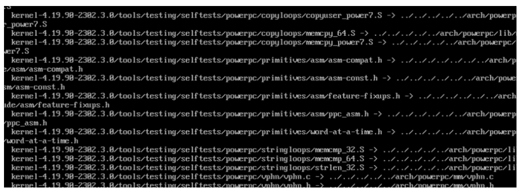
```
[root@localhost ~]# uname -r > uname_r.l
[root@localhost ~]# cat uname_r.log
5.10.0-60.18.0.50.oe2203.x86_64
[root@localhost ~]# _
```

然后获取内核源码并解压，如图 2

图 2: 获取内核源码并解压



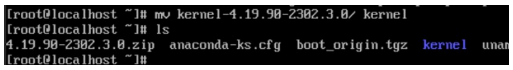
```
[root@localhost myshixun]# wget https://gitee.com/openeuler/kernel/repository/archive/4.19.90-2302.3.0.zip
--2023-05-15 13:42:29--  https://gitee.com/openeuler/kernel/repository/archive/4.19.90-2302.3.0.zip
Resolving gitee.com (gitee.com)... 212.64.63.190, 212.64.63.215
Connecting to gitee.com (gitee.com)|212.64.63.190|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://gitee.com/openeuler/kernel/repository/blazearchive/4.19.90-2302.3.0.zip?Expires=1684130579&Signature=xpS%2
af2tVG6cxFqtasHsWkAS5PK20S2VSdbTbLU8%3D [following]
--2023-05-15 13:42:59--  https://gitee.com/openeuler/kernel/repository/blazearchive/4.19.90-2302.3.0.zip?Expires=1684130579&S
ature=xpS%2B3Ksaf2tVG6cxFqtasHsWkAS5PK20S2VSdbTbLU8%3D
Reusing existing connection to gitee.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: '4.19.90-2302.3.0.zip'
```



```
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/copyloops/copyuser_power7.S -> ../../../../../arch/powerp
r_power7.S
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/copyloops/memcpy_64.S -> ../../../../../arch/powerpc/lib/
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/copyloops/memcpy_power7.S -> ../../../../../arch/powerpc/
wer7.S
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/primitives/asm/asm-compat.h -> ../../../../../../arch/p
e/asm/asm-compat.h
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/primitives/asm/asm-const.h -> ../../../../../../arch/powe
sm/asm-const.h
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/primitives/asm/feature-fixups.h -> ../../../../../../arch
ude/asm/feature-fixups.h
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/primitives/asm/ppc_asm.h -> ../../../../../../arch/powerp
/ppc_asm.h
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/primitives/word-at-a-time.h -> ../../../../../arch/powerp
/word-at-a-time.h
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/stringloops/memcmp_32.S -> ../../../../../arch/powerpc/li
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/stringloops/memcmp_64.S -> ../../../../../arch/powerpc/li
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/stringloops/strlen_32.S -> ../../../../../arch/powerpc/li
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/vphn/vphn.c -> ../../../../../arch/powerpc/mm/vphn.c
 kernel-4.19.90-2302.3.0/tools/testing/selftests/powerpc/vphn/vphn.h -> ../../../../../arch/powerpc/mm/vphn.h
```

然后，把源码目录命名为kernel。如图 3

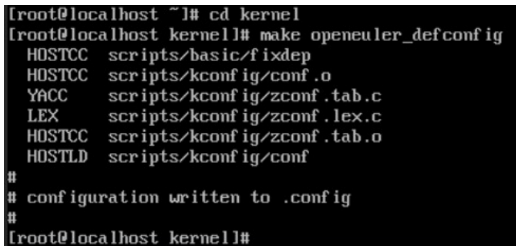
图 3: 把源码目录命名为kernel



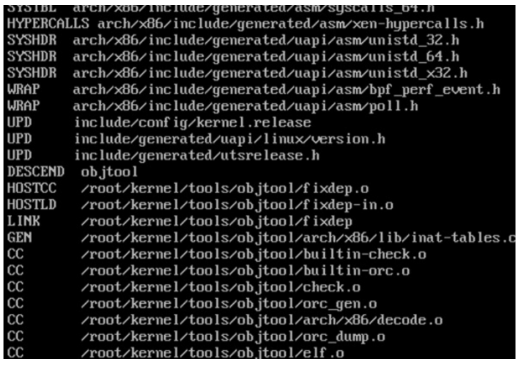
```
[root@localhost ~]# mv kernel-4.19.90-2302.3.0/ kernel
[root@localhost ~]# ls
4.19.90-2302.3.0.zip  anaconda-ks.cfg  boot_origin.tgz  kernel  unam
[root@localhost ~]#
```

接着，进入源码目录，配置内核并编译，如图 4

图 4: 配置内核并编译



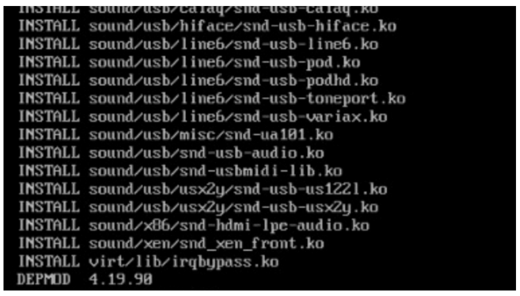
```
[root@localhost ~]# cd kernel
[root@localhost kernel]# make openeuler_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
[root@localhost kernel]#
```



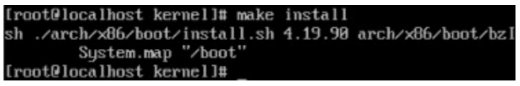
```
SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
WRAP    arch/x86/include/generated/uapi/asm/bpf_perf_event.h
WRAP    arch/x86/include/generated/uapi/asm/poll.h
UPD     include/config/kernel.release
UPD     include/generated/uapi/linux/version.h
UPD     include/generated/utsrelease.h
DESCEND  objtool
HOSTCC   /root/kernel/tools/objtool/fixdep.o
HOSTLD   /root/kernel/tools/objtool/fixdep-in.o
LINK     /root/kernel/tools/objtool/fixdep
GEN      /root/kernel/tools/objtool/arch/x86/lib/inat-tables.c
CC       /root/kernel/tools/objtool/builtin-check.o
CC       /root/kernel/tools/objtool/builtin-orc.o
CC       /root/kernel/tools/objtool/check.o
CC       /root/kernel/tools/objtool/orc_gen.o
CC       /root/kernel/tools/objtool/arch/x86/decode.o
CC       /root/kernel/tools/objtool/orc_dump.o
CC       /root/kernel/tools/objtool/elf.o
```

最后，进行安装，如图 5。

图 5: 配置内核并编译



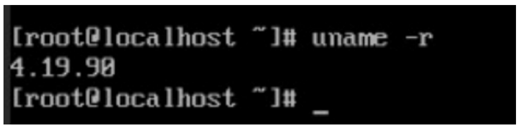
```
 INSTALL sound/usb/caiaq/snd-usb-caiaq.ko
 INSTALL sound/usb/hiface/snd-usb-hiface.ko
 INSTALL sound/usb/line6/snd-usb-line6.ko
 INSTALL sound/usb/line6/snd-usb-pod.ko
 INSTALL sound/usb/line6/snd-usb-podhd.ko
 INSTALL sound/usb/line6/snd-usb-toneport.ko
 INSTALL sound/usb/line6/snd-usb-variax.ko
 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
```



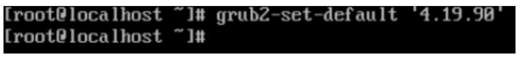
```
[root@localhost kernel]# make install
sh ./arch/x86/boot/install.sh 4.19.90 arch/x86/boot/bzI
        System.map "/boot"
[root@localhost kernel]# _
```

重启之后检查内核版本，并在grub中设置默认内核，如图 6

图 6: 使用新内核



```
[root@localhost ~]# uname -r
4.19.90
[root@localhost ~]# _
```



```
[root@localhost ~]# grub2-set-default '4.19.90'
[root@localhost ~]#
```

4 任务一

查找相关资料，解释`hello_world.c`文件中以下代码的含义和作用：

1. MODULE_LICENSE：在Linux内核文档有如下表述：

   > The sole purpose of this tag is to provide sufficient information whether the module is free software or proprietary for the kernel module loader and for user space tools.

   即这个tag提供了以便内核模块加载器和用户空间工具判断模块是自由软件还是专有软件的信息。

2. module_param：是用于定义模块参数的宏，三个参数分别为变量名、类型、权限。

3. MODULE_PARM_DESC：用于定义模块参数的描述的宏。描述会在使用`modinfo`命令时显示出来

4. module_init：用于通知内核初始化模块的时候，要使用哪个函数进行初始化。

5. module_exit：用于告诉内核，当卸载模块时，需要调用哪个函数。

6. __init：用于将函数放在可执行文件的init段中，将在初始化时被执行。

7. __exit：将函数放在exit段中，在模块卸载阶段执行。

内容出处：

1. https://tldp.org/LDP/lkmpg/2.6/html/x323.html
2. https://www.kernel.org/doc/html/latest/process/license-rules.html#id1

5 任务二

请参考hello_world.c和Makefile文件，在右侧代码文件区域编写hello_magic_student.c和 Makefile文件，完成以下任务：

1. 在hello_magic_student.c文件中定义函数 hello_student(…)，该函数包含3个参数：id, name, age，分别代表学号、姓名和年龄，并通过printk输出：

   ```
   My name is ${name}, student id is ${id}. I am ${age} years old.
   ```

2. 在hello_magic_student.c文件中定义函数 my_magic_number(…)，该函数包含2个参数：id和age，分别代表学号和年龄。请你在该函数里将学号的每一位数字相加后再与年龄求和，将求和结果的个位数作为magic_number，并使用printk输出：

   ```
   My magic number is ${magic_number}.
   ```

完成hello_magic_student.c文件的编写后，系统会自动保存。参考hello_world模块的Makefile并适当调整，在加载内核时提供学号、姓名和年龄，通过dmesg命令查看printk的输出。

首先，在文件开始处的三个参数定义进行修改，写入自己的姓名学号年龄：

```
        /*id*/
static char* id = "22920202204622";
module_param(id, charp, 0644);
MODULE_PARM_DESC(id, "char* param\n");

/*name*/
static char* name = "KeZheng Xiong";
module_param(name, charp, 0644);
MODULE_PARM_DESC(name, "char* param\n");

/*age*/
static int age = 21;
module_param(age, int, 0644);
MODULE_PARM_DESC(age, "int param\n");
```

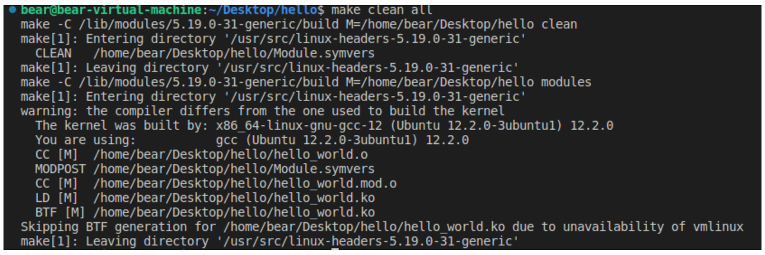
然后依题意实现两个函数即可。前者直接调用printk输出，该函数的使用方式与printf一致。后者遍历学号、转化为整数并按规则计算：

```
void hello_student(char* id,char *name,int age)
{
    printk("My name is %s, student id is %s. I am %d years old.",name,id,age);
}
void my_magic_number(char* id,int age)
```

```
{
    int ret=0;
    char *p=id;
    while(*p){
        ret+=(*p++ -'0');
    }
    ret+=age;
    printk("My magic number is %d.",ret%10);
}
```

首先，编译该模块，输出信息如图 7，无错误后安装模块。

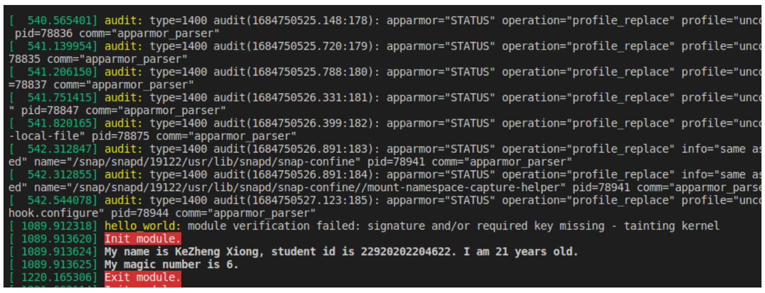
图 7: 编译模块



```
● bear@bear-virtual-machine:~/Desktop/hello$ make clean all
  make -C /lib/modules/5.19.0-31-generic/build M=/home/bear/Desktop/hello clean
  make[1]: Entering directory '/usr/src/linux-headers-5.19.0-31-generic'
    CLEAN   /home/bear/Desktop/hello/Module.symvers
  make[1]: Leaving directory '/usr/src/linux-headers-5.19.0-31-generic'
  make -C /lib/modules/5.19.0-31-generic/build M=/home/bear/Desktop/hello modules
  make[1]: Entering directory '/usr/src/linux-headers-5.19.0-31-generic'
  warning: the compiler differs from the one used to build the kernel
    The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.2.0-3ubuntu1) 12.2.0
    You are using:           gcc (Ubuntu 12.2.0-3ubuntu1) 12.2.0
    CC [M]  /home/bear/Desktop/hello/hello_world.o
    MODPOST /home/bear/Desktop/hello/Module.symvers
    CC [M]  /home/bear/Desktop/hello/hello_world.mod.o
    LD [M]  /home/bear/Desktop/hello/hello_world.ko
    BTF [M] /home/bear/Desktop/hello/hello_world.ko
  Skipping BTF generation for /home/bear/Desktop/hello/hello_world.ko due to unavailability of vmlinux
  make[1]: Leaving directory '/usr/src/linux-headers-5.19.0-31-generic'
```

然后，使用insmod安装内核模块，然后等待一段时间，使用dmesg查看输出，如图 8。

图 8: 输出信息



```
[  540.565401] audit: type=1400 audit(1684750525.148:178): apparmor="STATUS" operation="profile_replace" profile="unco
 pid=78836 comm="apparmor_parser"
[  541.139954] audit: type=1400 audit(1684750525.720:179): apparmor="STATUS" operation="profile_replace" profile="unco
78835 comm="apparmor_parser"
[  541.206150] audit: type=1400 audit(1684750525.788:180): apparmor="STATUS" operation="profile_replace" profile="unco
=78837 comm="apparmor_parser"
[  541.751415] audit: type=1400 audit(1684750526.331:181): apparmor="STATUS" operation="profile_replace" profile="unco
" pid=78847 comm="apparmor_parser"
[  541.820165] audit: type=1400 audit(1684750526.399:182): apparmor="STATUS" operation="profile_replace" profile="unco
-local-file" pid=78875 comm="apparmor_parser"
[  542.312847] audit: type=1400 audit(1684750526.891:183): apparmor="STATUS" operation="profile_replace" info="same as
ed" name="/snap/snapd/19122/usr/lib/snapd/snap-confine" pid=78941 comm="apparmor_parser"
[  542.312855] audit: type=1400 audit(1684750526.891:184): apparmor="STATUS" operation="profile_replace" info="same as
ed" name="/snap/snapd/19122/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=78941 comm="apparmor_parse
[  542.544078] audit: type=1400 audit(1684750527.123:185): apparmor="STATUS" operation="profile_replace" profile="unco
hook.configure" pid=78944 comm="apparmor_parser"
[ 1089.912318] hello_world: module verification failed: signature and/or required key missing - tainting kernel
[ 1089.913620] Init module.
[ 1089.913624] My name is KeZheng Xiong, student id is 22920202204622. I am 21 years old.
[ 1089.913625] My magic number is 6.
[ 1220.165306] Exit module.
```

可见该实现正确地输出了相关信息。计算结果也是正确的（按照规则求和得56，个位为6）。

6 实验总结

在本次实验中，我编译并运行了Linux内核。在自行编译运行的基础上，我实现了一个简单的内核模块。了解了Linux内核模块的编程并进行了实际体验。在实验中，我对Linux内核编程的细节有了更多把握，并且注意了Makefile中需要使用Tab而非空格，否则会出现错误。