



中山大學

SUN YAT-SEN UNIVERSITY

实 验 报 告

课程名称：_____操作系统_____

姓 名：_____孙广岩_____

学 号：_____20354242_____

专业班级：_____智能科学与技术专业 5 班_____

任课教师：_____吴贺俊_____

_____2022_____年_____11_____月_____11_____日

实验报告成绩评定表

| 评定项目 | 内 容 | 满 分 | 评 分 | 总 分 |
|--|---|--------|--------|--------|
| 实验态度 | 态度端正、遵守纪律、出勤情况 | 10 | | |
| 实验过程 | 按要求完成算法设计、代码书写、注释清晰、运行结果正确 | 30 | | |
| 实验记录 | 展示讲解清楚、任务解决良好、实验结果准确 | 20 | | |
| 报告撰写 | 报告书写规范、内容条理清楚、表达准确规范、上交及时、无抄袭，抄袭记 0 分，提供报告供抄袭者扣分。 | 40 | | |
| 评语： | | | | |
| 指导老师签字：_____年 月 日 | | | | |

实验 7 虚拟内存管理（内核源码）实验

一、 实验目的

1. 进一步加深对虚拟内存管理方式及其实现过程的理解

二、 实验内容

1. 任务描述

- Linux 是当今流行的操作系统之一。由于其源码的开放性，现代操作系统设计的思想和技术能够不断运用于它的新版本中。因此，读懂并修改 Linux 内核源代码无疑是学习操作系统设计技术的有效方法。
- Linux 内核：内核指的是一个提供设备驱动、文件系统、进程管理、网络通信等功能的系统软件，内核并不是一套完整的操作系统，它只是操作系统的核心。
- Linux 发行版本：一些组织或厂商将 Linux 内核与各种软件和文档包装起来，并提供系统安装界面和系统配置、设定与管理工具，就构成了 Linux 的发行版本。

2. 实验方案

首先进行内核编译，之后统计系统缺页次数。

三、实验记录

1. 内核编译

① 查看内核版本

```
haiden@haiden-virtual-machine: ~  
haiden@haiden-virtual-machine:~$ uname -r  
4.15.0-142-generic  
haiden@haiden-virtual-machine:~$
```

可以看到原来的版本是 4.15.0

② 下载所需内核版本

下载地址：<https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.16.10.tar.xz>



我使用了火狐浏览器下载，当然也可以直接 wget 下载/

③ 解压

使用如下命令来解压：

```
cp linux-4.16.10.tar.xz /usr/src  
cd /usr/src/  
sudo tar -xvf linux-4.16.10.tar.xz
```

```
root@george-virtual-machine: /usr/src
linux-4.16.10/virt/kvm/arm/vgic/vgic-mmio.h
linux-4.16.10/virt/kvm/arm/vgic/vgic-v2.c
linux-4.16.10/virt/kvm/arm/vgic/vgic-v3.c
linux-4.16.10/virt/kvm/arm/vgic/vgic-v4.c
linux-4.16.10/virt/kvm/arm/vgic/vgic.c
linux-4.16.10/virt/kvm/arm/vgic/vgic.h
linux-4.16.10/virt/kvm/async_pf.c
linux-4.16.10/virt/kvm/async_pf.h
linux-4.16.10/virt/kvm/coalcesced_mmio.c
linux-4.16.10/virt/kvm/coalcesced_mmio.h
linux-4.16.10/virt/kvm/eventfd.c
linux-4.16.10/virt/kvm/irqchip.c
linux-4.16.10/virt/kvm/kvm_main.c
linux-4.16.10/virt/kvm/vfio.c
linux-4.16.10/virt/kvm/vfio.h
linux-4.16.10/virt/lib/
linux-4.16.10/virt/lib/Kconfig
linux-4.16.10/virt/lib/Makefile
linux-4.16.10/virt/lib/irqbypass.c
root@george-virtual-machine: /usr/src# ls
linux-4.16.10          linux-headers-5.15.0-56-generic
linux-4.16.10.tar.xz   linux-hwe-5.15-headers-5.15.0-46
linux-headers-5.15.0-46-generic  linux-hwe-5.15-headers-5.15.0-56
root@george-virtual-machine: /usr/src#
```

④ 配置内核

安装一些需要使用的包：

```
sudo apt install make
```

```
sudo apt install gcc
```

```
sudo apt install libgtk2.0-dev libglib2.0-dev libglade2-dev
```

```
sudo apt install flex
```

```
sudo apt install bison
```

```
apt install libncurses5-dev
```

然后使用 `make menuconfig` 配置内核：（只需要直接 `save` 即可）

```
root@george-virtual-machine: /usr/src/linux-4.16.10
config - Linux/x86 4.16.10 Kernel Configuration

Linux/x86 4.16.10 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenu --->). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

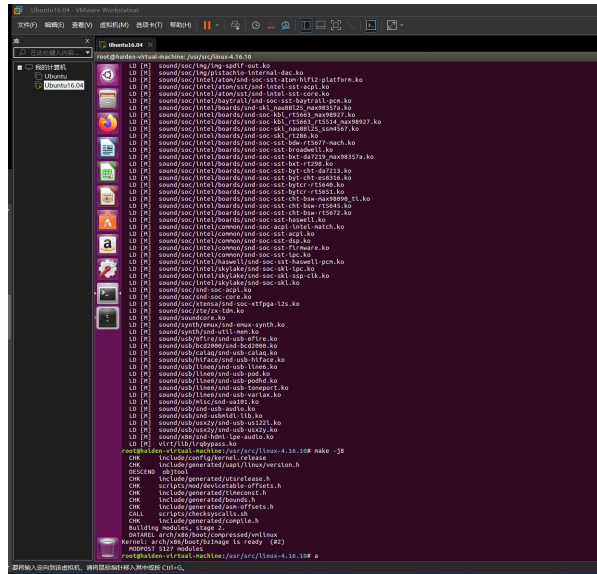
[*] 64-bit kernel
  General setup --->
    [*] Enable loadable module support --->
    [*] Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
    [*] Networking support --->
    Device Drivers --->
  ↵(+)
```

⑤ 编译内核

```
sudo apt install libelf-dev
```

```
sudo apt install libelf-dev
```

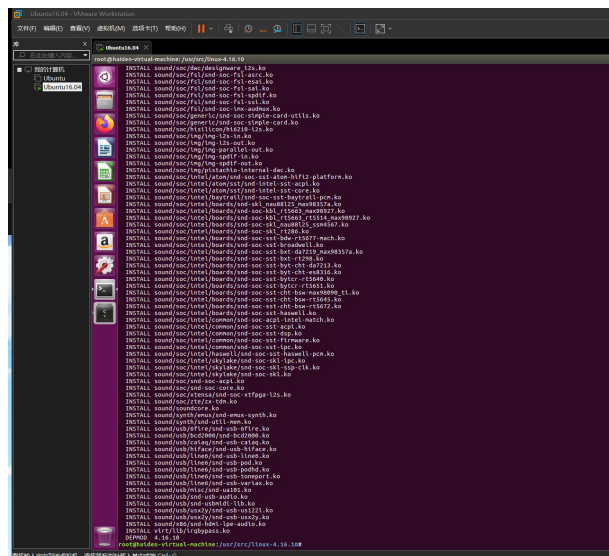
```
sudo make -j8
```



⑥ 编译和安装

```
root@haiden-virtual-machine: /usr/src/linux-4.16.10# make modules
CHK       include/config/kernel.release
CHK       include/generated/uapi/linux/version.h
CHK       include/generated/utsrelease.h
CHK       include/generated/bounds.h
CHK       include/generated/timeconst.h
CHK       include/generated/asm-offsets.h
CALL      scripts/checksyscalls.sh
DESCEND   objtool
CHK       scripts/mod/devicetable-offsets.h
Building modules, stage 2.
MODPOST 5127 modules
```

make modules_install



⑦ 安装内核

```

root@haiden-virtual-machine: /usr/src/linux-4.16.10# make install
sh ./arch/x86/boot/install.sh 4.16.10 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.16.10 /boot/vmlinuz-4.16.10
run-parts: executing /etc/kernel/postinst.d/dkms 4.16.10 /boot/vmlinuz-4.16.10
* dkms: running auto installation service for kernel 4.16.10
Kernel preparation unnecessary for this kernel. Skipping...

Building module:
cleaning build area... (bad exit status: 2)
make KERNELRELEASE=4.16.10 VMLINUZ=4.16.10 MODULEBUILDDIR=/var/lib/dkms/open-vn-tools/10.0.7/build -C vxnet... (bad exit status: 2)
ERROR (dkms support): kernel package linux-headers-4.16.10 is not supported
Error! Bad return status for module build on kernel: 4.16.10 (x86_64)
Consult /var/lib/dkms/open-vn-tools/10.0.7/build/make.log for more information.

run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.16.10 /boot/vmlinuz-4.16.10
update-initramfs: Generating /boot/initrd.img-4.16.10
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.16.10 /boot/vmlinuz-4.16.10
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.16.10 /boot/vmlinuz-4.16.10
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.16.10 /boot/vmlinuz-4.16.10
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.16.10 /boot/vmlinuz-4.16.10
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is set is no longer supported.
Found linux image: /boot/vmlinuz-4.16.10
Found initrd image: /boot/initrd.img-4.16.10
Found linux image: /boot/vmlinuz-4.15.0-142-generic
Found initrd image: /boot/initrd.img-4.15.0-142-generic
Found linux image: /boot/vmlinuz-4.15.0-140-generic
Found initrd image: /boot/initrd.img-4.15.0-140-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done

```

⑧ 重新启动，检查新内核

```

haiden@haiden-virtual-machine: ~
haiden@haiden-virtual-machine:~$ uname -r
4.16.10
haiden@haiden-virtual-machine:~$

```

可以看到成功更换了内核。

2. 统计系统缺页次数

① 在内核源码中找到 include/linux/mm.h 文件，声明变量 pfcount，用于统计缺页次数。

```

root@haiden-virtual-machine: /usr/src/linux-4.16.10/include/linux
struct file_ra_state;
struct user_struct;
struct writeback_control;
struct bdi_writeback;

void init_mm_internals(void);

#ifdef CONFIG_NEED_MULTIPLE_NODES /* Don't use mapnr's, do it properly */
extern unsigned long max_mapnr;

static inline void set_max_mapnr(unsigned long limit)
{
    max_mapnr = limit;
}
#else
static inline void set_max_mapnr(unsigned long limit) {}
#endif

extern unsigned long totalram_pages;
extern void * high_memory;
extern int page_cluster;
extern unsigned long volatile pfcount;
#endif CONFIG_SYSCTL

```

② 在/arch/x86/mm/fault.c 文件中定义变量 pfcount，并在 do_page_fault()函数中找到 good_area，让变量 pfcount 递增 1，实现了缺页次数的统计。

```

root@haiden-virtual-machine: /usr/src/linux-4.16.10/arch/x86/mm
*/
if (major) {
    tsk->maj_flt++;
    perf_sw_event(PERF_COUNT_SW_PAGE_FAULTS_MAJ, 1, regs, address);
} else {
    tsk->min_flt++;
    perf_sw_event(PERF_COUNT_SW_PAGE_FAULTS_MIN, 1, regs, address);
}

check_v8086_mode(regs, address, tsk);
}
NOKPROBE_SYMBOL(__do_page_fault);
unsigned long volatile pfcount;
static nokprobe_inline void
trace_page_fault_entries(unsigned long address, struct pt_regs *regs,
                        unsigned long error_code)
{
    if (user_mode(regs))
        trace_page_fault_user(address, regs, error_code);
    else
        trace_page_fault_kernel(address, regs, error_code);
}

```



```

root@haiden-virtual-machine: /usr/src/linux-4.16.10/arch/x86/mm
    }
    if (unlikely(expand_stack(vma, address))) {
        bad_area(regs, error_code, address);
        return;
    }
    /*
     * Ok, we have a good vm_area for this memory access, so
     * we can handle it..
     */
good_area:
    if (unlikely(access_error(error_code, vma))) {
        bad_area_access_error(regs, error_code, address, vma);
        return;
    }
    pfcount++;
    /*
     * If for any reason at all we couldn't handle the fault,
     * make sure we exit gracefully rather than endlessly redo
     * the fault. Since we never set FAULT_FLAG_RETRY_NOWAIT, if
     * we get VM_FAULT_RETRY back, the mmap_sem has been unlocked.
     */

```

- ③ 修改 kernel/kallsyms.c 文件，在文件最后插入 EXPORT_SYMBOL(pfcount);

```

root@haiden-virtual-machine: /usr/src/linux-4.16.10/kernel
        if (kdb_walk_kallsyms_iter.name[0])
            return kdb_walk_kallsyms_iter.name;
    }
}
#endif /* CONFIG_KGDB_KDB */

static const struct file_operations kallsyms_operations = {
    .open = kallsyms_open,
    .read = seq_read,
    .llseek = seq_lseek,
    .release = seq_release_private,
};

static int __init kallsyms_init(void)
{
    proc_create("kallsyms", 0444, NULL, &kallsyms_operations);
    return 0;
}
device_initcall(kallsyms_init);
EXPORT_SYMBOL(pfcount);
~
~
~

```

- ④ 重新编译内核，与前面操作一致。
- ⑤ 编写测试程序和 Makefile，请参见提供的示例代码。
- ⑥ 编译测试程序，并加载内核

```

root@haiden-virtual-machine:~/source# make
make -C /lib/modules/4.16.10/build M=/home/haiden/source modules
make[1]: Entering directory '/usr/src/linux-4.16.10'
CC [M] /home/haiden/source/pf.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/haiden/source/pf.mod.o
LD [M] /home/haiden/source/pf.ko
make[1]: Leaving directory '/usr/src/linux-4.16.10'

```

- ⑦ 加载内核成功后，输入如下命令，测试实验结果。

```

root@haiden-virtual-machine:~/source# insmod pf.ko
insmod: ERROR: could not insert module pf.ko: File exists
root@haiden-virtual-machine:~/source# cat /proc/readpfcount
The pfcount is 771537 and jiffies is 4295161847!

```

insmod 命令这里有 error 是因为之前运行了一次所以已经激活了，当时没有截到图，可以看到最终成功打印出实验结果。

测试程序中我将第 9 行的 extern unsigned long pfcount; 改为了 extern unsigned long volatile pfcount; 不然会报错。