**《操作系统原理》实验报告**

**一、实验目的**

1. 理解页面淘汰算法原理，编写程序演示页面淘汰算法。
2. 验证Linux虚拟地址转化为物理地址的机制
3. 理解和验证缺页处理的流程。
4. 理解设备是文件的概念。
5. 掌握Linux模块、驱动的概念和编程流程
6. **实验内容**
7. Windows/Linux模拟实现FIFO或LRU页面淘汰算法。
8. 编写一个Linux内核模块，并完成模块的安装/卸载等操作。
9. 编写Linux驱动程序（字符类型或杂项类型）并编程应用程序测试。 功能：write几个整数进去，read出其和或差或最大值。

**三、实验过程**

**3.1 模拟FIFO或LRU页面淘汰**

FIFO实现过程中的核心在于记录每个页表项上一次访问的时间；此处利用数组waittimes [bsize]记录物理块中的进程等待次数，同时用maxProc记录存放最久未被使用的进程，maxWait用来记录未被使用进程的时间最大值。实现能够记录每一个页表项目的访问时间后，每次需要淘汰时只需要依次遍历页表项并比较页表项之间等待次数的大小，更新此时的waitTimes。对每一次需要访问的目标，在当前队伍列表中查询是否有空闲物理块，是否命中，并找出此时等待次数最大的进程序列。



更新此轮页表情况后，分为三种情况进行替换。若存在空闲的物理块且该页表未命中，则将当前进程号填入该空闲物理块，并将缺页标志增加；若不存在空闲物理块且未命中，则按照记录的waitTimes进行页表项的替换，淘汰最早进来的进程并记录缺页；若物理块中存在相同的进程，则可以不进行淘汰。需要注意的是，不管是哪一种情况，在此轮结束后waitTimes数组均需要自加一。





与FIFO相似，模拟实现LRU函数中也分为三种情况，存在相同进程时直接记录结果即可，在此不做过多赘述。当页表项中不存在相同进程时，函数利用页表对象中的时间记录成员，取得在内存中停留最久的页面,默认状态下为最早调入的页面，访问后该页面时间记录成员清零并记录缺页。





除此之外，对于辅助函数，main函数实现菜单功能；buildNum()函数用来生成随机数并该序列存入进程数组；Init()用来在每次执行前初始化内存单元、缓冲区；output()函数将result数组中存储的结果以更形象的方式输出。

**3.2 完成Linux简单模块的安装/卸载**

编写模块函数如下所示。



Makefile文件代码如下图所示。其中KERNEL\_PATH为内核的绝对路径，PWD需要安装进去的内核文件所在的路径。



执行编译并安装代码如下。



卸载该模块代码如下。



**3.3 编写Linux驱动程序**

对于驱动程序的编写核心在于代码中定义的一个file\_operations类型的结构体pstruct，该结构体内的成员通过键值对进行赋值，表明了编写时对该驱动自定义的函数与Linux给出的标准读写等API之间的映射关系。具体指明了owner，以及标准的读写函数、装卸时调用函数与编写时自定义函数之间的对应关系。



与模块相同，驱动程序也有相应装卸时的调用函数，使用module\_init()和module\_exit()进行注册。其中，在安装驱动调用的函数dv\_init()中，使用了register\_chrdev()函数来注册一个字符型的设备，需要传入参数主设备号，设备名以及定义的驱动函数映射的结构体pstruct;而在删除驱动调用的函数dv\_exit()中，使用了unregister\_chrdev()对注册的设备进行删除。



分别实现驱动程序中的自定义读写函数，其中要注意函数copy\_from\_user()和copy\_to\_user()的合理调用，同时需要注意的是在my\_write()函数中，需要检查用户的输入字符数量，限制在一定长度内，防止过长。



编译安装该模块。编写Makefile文件，与上一节中的Makefile文件相同，详见3.2节中Makefile文件的编写代码，生成一系列的编译文件后，对模块进行安装。



查看节点的驱动号，如图3-1可知此设备my\_dv驱动号为240，进一步创建设备节点和设备挂钩。其中/dev/my\_dv为驱动模块的名称，c表明为字符型设备，240为主设备号，0为次设备号,成功创建设备并挂钩后如图3-2所示。



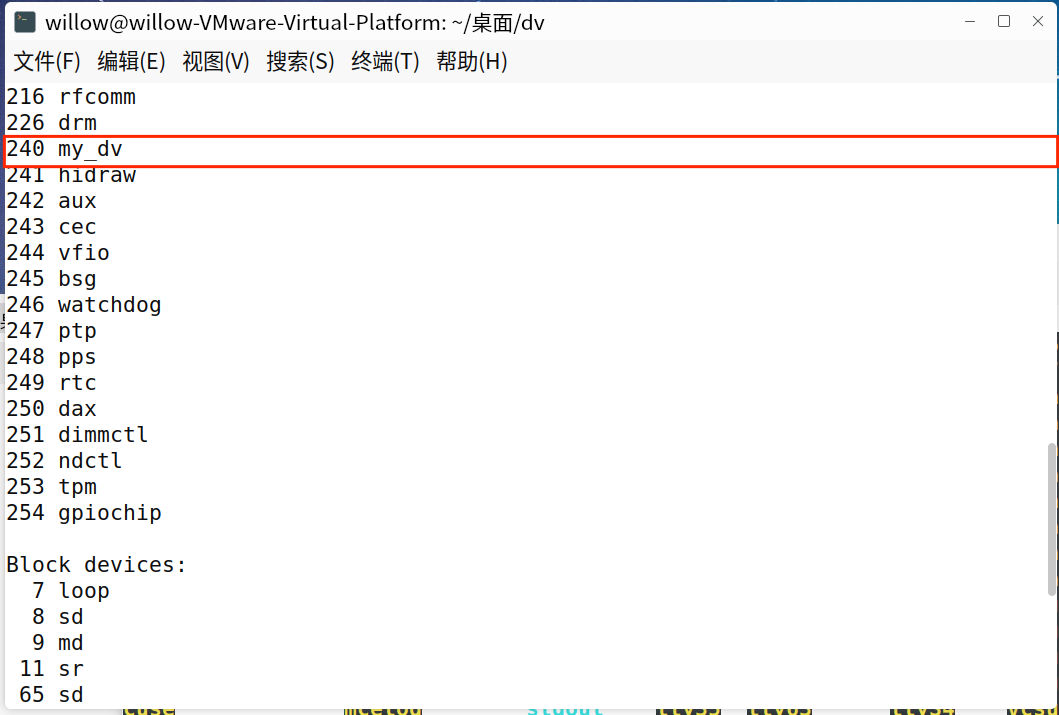


图3-1 查看驱动号

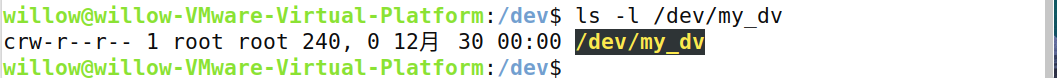


图3-2 查看设备

编写测试文件test2.c后进行编译运行，对模块进行测试，执行如下命令。





**四、实验结果**

**4.1 模拟FIFO或LRU页面淘汰**

完成代码编写后运行，选择“0”则退出，如图4-1所示。



图4-1 退出界面

选择“1”后进入FIFO模拟界面，系统随机产生一系列的随机数，来模拟序列号的生成，随后运用函数output()来打印出易于理解的过程，最后显示出缺页次数以及缺页率。如图4-2所示。

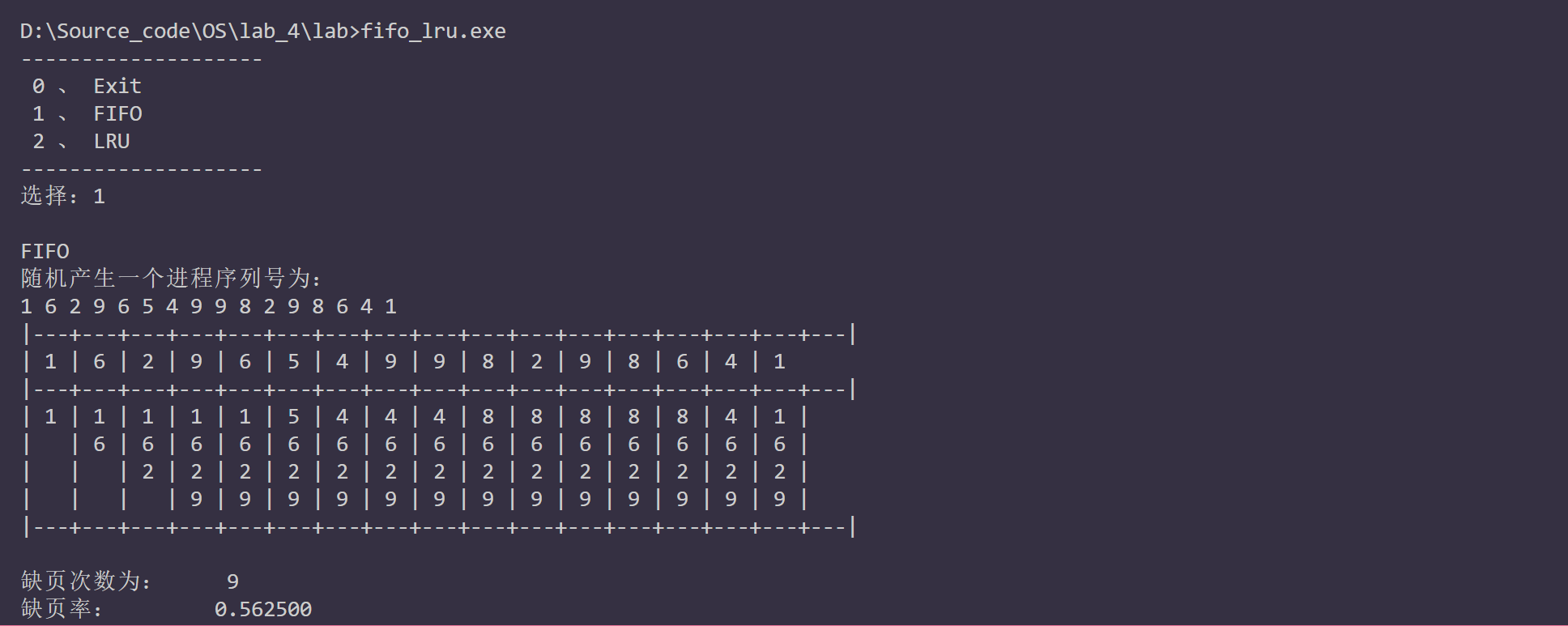


图4-2 FIFO运行结果

选择“2”后进入LRU模拟界面，与FIFO运行结果显示结构类似，如图4-3所示。

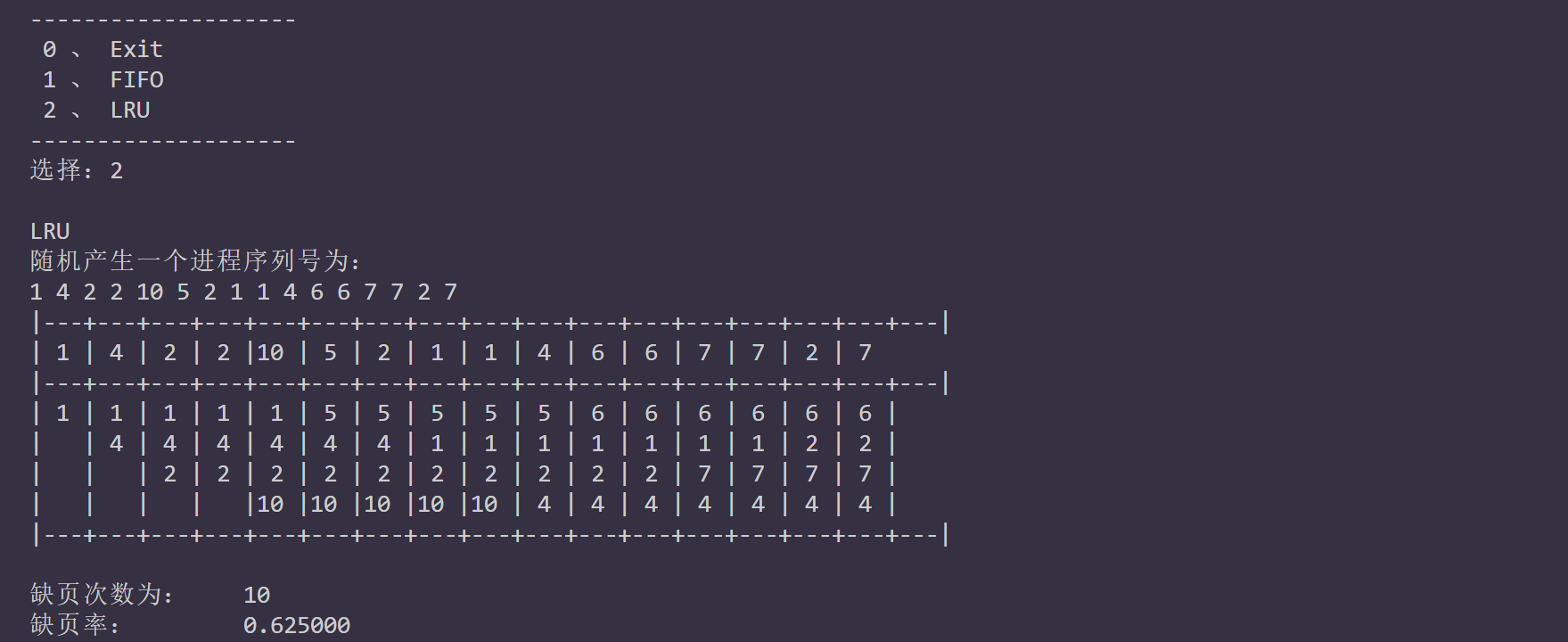


图4-3 LRU运行结果

**4.2 完成Linux简单模块的安装/卸载**

完成自行编写模块的编写、编译和装卸过程之后，使用dmesg命令查看装卸时调用的printk函数输出的内容，可以看到与我们编写的提示信息一致，实验达到预期。



图4-4 LRU安装/卸载模块

**4.3 编写Linux驱动程序**

测试该模块的安装情况，如图可以显示出模块my\_dv.ko的详细信息，证明该模块安装成功。

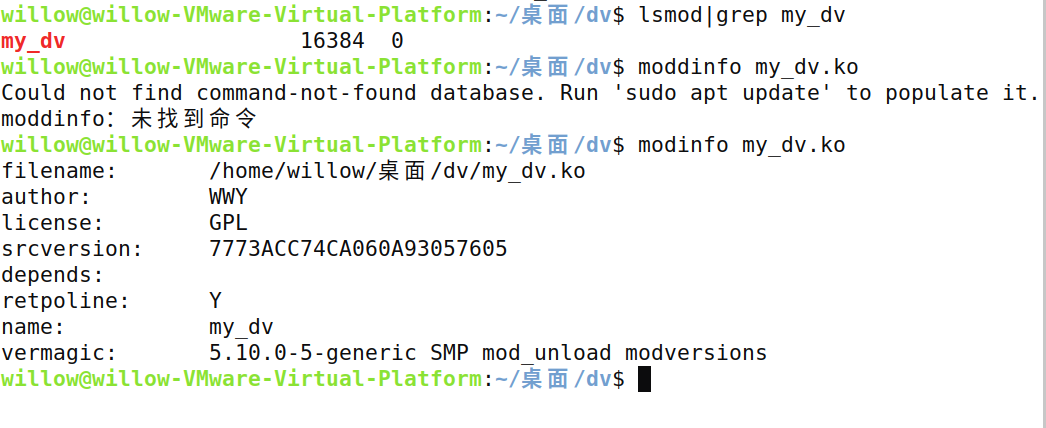


图4-5 模块详细信息

对测试程序test2.c进行编译并运行后，根据提示输入两个需要参与运算的数3和4，可以得到正确的运行结果7，表明驱动程序成功装载并正常工作，达到实验预期。

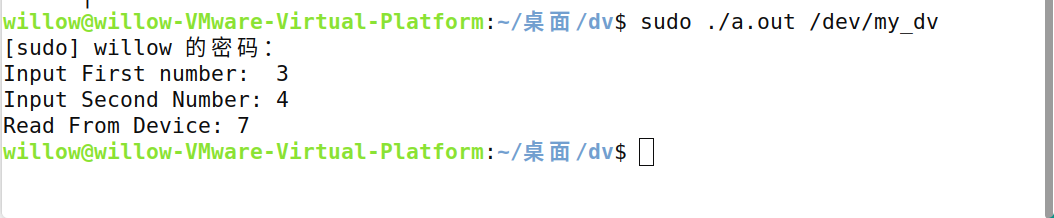


图4-6 运行结果

完成自行编写模块的编写、编译过程之后，使用dmesg命令查看装卸时调用的printk函数输出的内容，可以看到与我们编写的提示信息一致，实验达到预期。



图4-6 dmesg查看信息

**五、实验错误排查和解决方法**

**5.1 模拟FIFO或LRU页面淘汰**

Q：理论变成写代码实现较困难。

A：缺页中断情况可以分成两种：内存中存在空闲位置，直接将当前调度的页面调入内存中（一般出现在调度的开始）内存中不存在空闲位置，就需要选择内存中的一个页面调出，让当前调度页面进入内存，此过程就是页面置换，淘汰的依据就是上述visit值大者。我查阅了网上的一些资料和一些代码之后，写出了相应的代码算法。

**5.2 完成Linux简单模块的安装/卸载**

A：在编写Linux内核模块的过程中，由于不清楚一些具体命令应该要怎么写和对相关指令的不熟悉，我一开始并不知道要怎么下手。后来我在网上查找了大量的相关资料后，弄明白了内核模块程序所需要的一些内容，与一些具体指令module\_init和module\_exit等的用法后，我成功完成了内核模块程序的编写。

Q：makefile文件的编写

A：在网上查询到了一些可以使用的Makefile文件并仿照进行修改后，成功完成了make编译和内核模块的安装与卸载。

**5.3 编写Linux驱动程序**

Q：相关API不清楚使用方法。

A:在任务1的基础上，需要另外完成驱动程序设备的申请创建，write相关功能和read相关的功能操作。但是对于一些具体功能操作我其实并不清楚，还有设备申请创建和连接的相关命令也完全不知晓。后来花了很多时间在网上查阅了大量的资料后我弄清楚了相应的过程，并搜索到了一些具体参数如file\_operation字符设备操作集合的一系列操作指令。关于设备注册可以通过静态定义并注册设备号或者动态申请并注册设备号的方法，我在综合考虑之后选择了动态申请的方法。此方法无需自己去确定哪个设备号可用，内核会查询哪个设备号没有被使用，然后分配给当前驱动进行注册。

Q：只能输出第一个输入的数据，后面数据的正确性会出问题

A：增加缓冲区buf，将输入的两个数据读入后，进行相加并存入缓冲区中，之后read即可读出相加后的数据。

Q：无法编译成功一直显示找不到设备

A：没有创建设备并挂钩，只是安装了模块。需要创建设备后再运行测试程序。

**六、实验参考资料和网址**

**（1）教学课件**

**（2）https://blog.csdn.net/weixin\_44518102/article/details/124951165**

**（3）https://blog.csdn.net/hanp\_linux/article/details/90445164**

**（4）https://blog.csdn.net/hanp\_linux/article/details/90441869**