毕设工作记录

从第一学期20周开始，每3天左右记录一次

2018年1月15日（第20周，周一）

初步确定毕设课题作文件系统方向的研究，开始学习Linux文件系统的基础知识。根据老师的指导和学长给的文献资料，首先学习了《Operating System：Three Easy Pieces》中Persistence部分，阅读了I/O deveices、disk和raid这三节的内容，其中I/O devices这一节中学习了I/O在系统架构中的地位，规范设备和规范协议，中断和DMA，设备的交互方法以及设备驱动；在disk中学习了磁盘的物理结构以及磁盘I/O时间的计算等，而在raid这一节重温了磁盘冗余阵列的知识。。

2018年1月18日（第20周，周四）

阅读了files&directories，file system implementation， ffs这三个部分的内容。学习了文件系统中文件和目录的区别及各自的组织方式，关键数据结构和实现方式，还学习了vsfs这一简单的文件系统，了解了文件系统的组织方式和布局，以及文件系统对于空闲块的管理，磁盘写入、缓存等问题，最后学习了一个相较于vsfs更为高效的文件系统fast file system，在ffs中着重考虑了性能问题以及磁盘碎片化问题，提出了磁盘感知的解决方案，并定义了一种新的磁盘组织结构——cylinder group，将磁盘划分为不同的组来进行管理，并利用了局部性的概念。

2018年1月22日（第21周，周一）

阅读了journaling、lfs、integrity和distributed system这几节的内容。了解了文件系统的日志功能和两种不同的日志方式——撤销和重做，了解了记录式的文件系统的概念和组织方式，学习了数据完整性和数据保护的各种方案，以及分布式系统的概念。

2018年1月24日（第21周、周四）

阅读了《深入理解Linux内核》中ext2和ext3文件系统部分及其他一些相关的部分。

2018年1月30日（已放假，周二）

阅读S R Dullor等人的论文《System Software for Persistent Memory》，并进行翻译。

2018年2月4日（已放假，周日）

完成文献翻译工作

2018年2月28日（第1周，周三）

开展开题报告的撰写工作，广泛查阅国内外关于NVMFS的论文。

2018年3月2日（第1周，周五）

完成开题报告中课题背景意义、国内外研究现况及发展趋势这两个部分的撰写。

2018年3月5日（第2周，周一）

完成开题报告全部内容的撰写，进行格式修订和检查，并开始准备开题答辩的PPT。

2018年3月8日（第2周，周四）

完成开题答辩，并根据答辩组老师的指导建议整理修改开题报告，对于课题进行思考和挖掘。

2018年3月12日 周一

回顾了S.R Dullor等人的论文<System Software for Persistent Memory>以及文件系统基础相关的知识，开展了PMFS源码分析工作，初步完成了源码super.c文件中，pmfs\_fill\_super()函数和pmfs\_init()函数的流程分析工作。

2018年3月13日 周二

分析了balloc.c文件中pmfs\_new\_block()和pmfs\_free\_block()函数，理解了pmfs分配器的工作方式和分配策略。

2018年3月16日 周五

尝试在电脑上安装centos7，但是在安装过程中由于不明原因导致windows操作系统无法正常启动，centos也只安装了内核（只有命令行，没有图形或者文本界面），在网上查找各种解决方法尝试恢复windows启动项。

2018年3月19日 周一

尝试了在centos内使用root权限修改/boot/grub2/grub.cfg文件和用u盘PE系统恢复win10的MBR主引导记录，但是问题都没能得到解决，最后查阅了大量资料，发现自己的电脑引导和分区分别是UEFI和GPT，因而无法通过重建MBR来修复引导，导致问题的原因可能是centos不正确的安装覆盖了win10的启动文件，最终只能在PE系统下，格式化了centos和ubuntu的分区（电脑上原已安装win10和ubuntu双系统），并且重装了win10，才成功启动了windows，所幸除了C盘安装的软件等有所损失，其他资料都还完整。

2018年3月27日 周二

分析了super.c文件中的pmfs\_ioremap()函数，在pmfs\_fill\_super()中调用了该函数进行虚实地址的映射，即内存映射I/O，在pmfs中对内存映射I/O进行了优化，可以支持大页面的透明映射。此外，在网上查阅资料，重新温习了文件系统四大对象（超级块、目录项、文件、索引节点）的特征和相关关系。

2018年3月30日 周五

与学长交流了毕设的进度和现状，并就代码上的疑惑向学长请教，学长给我重点讲了分配器的数据结构，并指出该分配器存在性能问题，需要重点研究并加以改进，另外还给我演示了如何安装pmfs，如何对pmfs进行性能测试等，督促我注意毕设的进度，不要陷入到所有代码中，有针对性的研究并优化。于是重新重点分析了balloc.c文件（分配器的实现文件），发现学长之前给的方案可能存在一定问题，在与学长沟通后得到了肯定并自己设计了一种新的方案，准备进行尝试。

2018年4月2日 周一

根据自己的想法和所设计的方案，画出方案的设计图，设计以及相关数据结构，并仔细分析方案的可行性和可能碰到的问题，想法成型之后自己觉得没有太大问题，就开始动手实现，设计了新的数据结构pmfs\_blockp以及新的分配器总体结构和分配策略，重新实现了pmfs\_init\_blockmap()，pmfs\_new\_block()，pmfs\_free\_block()等函数。

2018年4月8日 周日

在自己的笔记本电脑上安装了centos6.8 64位虚拟机，尝试进行pmfs的安装和使用，但是从github上下载下来的源码包含了Linux 3.11.0的内核源码，在进行编译内核时出现了诸多问题，还没找出具体原因。这几天毕设工作没有取得实质性的进展，内核方面的问题，虚拟机的问题，设备的问题层出不穷。大致总结了以下问题：1）sh文件的unix-dos格式问题。这个问题是由于git clone命令基于不同平台下载下来的源码中sh文件格式不同造成的，最开始我使用的时在win10平台上下载下来的源码，其中sh脚本文件的格式为dos，但是在linux中只能执行dos格式的sh脚本，解决方法是在centos6.8虚拟机中重新下载源码。2）ncurses库问题。在执行make menuconfig时需要ncurses库支持才能弹出图形界面，但是ubuntu和centos默认都没有安装这个库，需要执行yum install ncurses ncurses-devel命令安装对应的库。但后来这个问题又有了新的曲折，实验室的服务器不能联网，因此yum命令没法用，解决的方法是在我笔记本上下载ncurses的源码包，然后用U盘拷贝到服务器中，通过./configure-make-make install三步命令完成ncurses的安装。3）压缩和解压缩问题。这个问题至今没有得到解决。由于换了新的服务器，需要重新搭建环境，学长让我把之前搭好环境的那台服务器中的linux 3.11.0内核拷贝出来（需要先make clean），最开始尝试直接拷贝文件夹（/usr/src/kernels/linux-3.11），但在拷贝一些文件时会报错（如/net/…/ni65.h，aux.c），还有一些文件出现覆盖问题，最后直接拷贝出来的文件夹也无法完成make命令。后来又尝试将内核文件夹压缩之后再拷贝，结果解压的时候会报错（无论在原机器上解压还是在新的服务器上解压都会），原因不明，报错信息各种各样。4）windows下无法拷贝，移动和创建文件名为aux.c的文件，而内核源码中包含该文件，因此无法通过windows操作该源码，这一问题是在使用xftp5在虚拟机和win10之间传送源码的时候发现的。5）xftp传送文件修改时间不一致问题。前面讲过pmfs源码是在centos虚拟机中通过git clone命令下载下来的，但是虚拟机的时间是有问题的，所有文件的修改时间显示为4月9日 4点多，但是当天的时间是4月8号，当我用xftp传送完源码到win10，在用u盘拷贝至centos7服务器上执行make menuconfig时，提示make的时间不一致，并且无法执行make操作，后来通过命令find ./ -type d –exec touch {} \;和find ./ -type f –exec touch {} \;两个命令将全部文件和文件夹的修改时间改为当前时间。

2018年4月9日 周一

今天上午用从github上下载下来的pmfs的源码，完成了3.11.0内核的编译，但是内核无法启动，原因不明，尝试了许多网上查到的方法，但都没有效果，学长也没跟我说明具体原因，只是让我从之前已经安装好内核的虚拟机上讲内核文件夹拷贝下来，我提出了拷贝时遇到的问题，学长中午过来帮我解决问题，未果。余亚学长提出内核文件夹的压缩格式不能用tar.gz，应该用zip格式（后来证明是对的）。中午，学长发现我在服务器上装的系统是centos7，说系统版本太高，容易出问题，于是给我找了一个centos6.4的镜像，下午在服务器上重装centos6.4，但是用u盘安装最后都会报错（cannot find image #1什么的），原因不明，根据网上的说法，可能是6.4版本太低，也可能是U盘的文件系统vfat格式不支持过大的ISO镜像复制（关于碰到的问题，这个网站说的比较清楚<http://www.zhetao.com/content218>，但并没有尝试），余亚学长提出的解决方法是用光盘安装centos6.5，实验室的机器之前安装过centos6.5没问题，并且有现成的光盘，于是用光盘安装了centos6.5，安装成功！在新的系统中，解压了zip格式的内核，成功！make menuconfig； make all; make modules\_install; make install四个步骤全部成功，在make install时提示名为power\_meter的模块无法找到，但并没有大的影响，重启，3.11.0内核启动成功！修改grub.conf，添加memmap=1G$4G，挂载pmfs到/mnt/pmfs，成功。随后用postmark测试pmfs的性能，发现postmark的所有操作都报error，问了学长，学长说是selinux的问题，要关掉selinux。命令setenforce 0可以临时关闭selinux，可修改配置文件在重启后也关闭selinux（<https://jingyan.baidu.com/article/6d704a131ba67828da51ca73.html> ）。至此，所需的环境和测试工具终于配置成功了。

2018年4月10日 周二

昨晚配置好了环境后，今天过来主要进行代码调试。用U盘将之前已经写好的代码拷到服务器上，进行第一次make尝试。根据学长的说法，由于我的pmfs采用内核模块方式，因此每次只需要在我自己的mypmfs中进行编译和安装模块就可以了。上午第一次make自己的代码，根据调试信息改了不少的问题，主要有：1）pmfs\_blockp\_alloc()函数定义的参数为空，但是要写void在里面；2）balloc.c中的BUG\_ON;3)一些指针初始化为NULL；4）在pmfs.h中添加自己定义的函数的声明；5）添加kmem\_cache\_alloc相关的函数，主要有create，destroy等，修改了init\_pmfs\_fs()函数。在改完调试信息里面能看到的一些bug后，我尝试insmod和mount命令，但是在mount的时候出了问题，进程直接被kill了。通过dmesg查看日志信息，发现报错是在我的自定义函数pmfs\_blockp\_alloc()中，后来经过仔细分析源码，发现是我自己定义的pmfs\_blockp\_cachep没有进行初始化，于是仿照pmfs\_blocknode\_cache相关的几个函数，进行初始化，解决了这个问题，于是也成功的将pmfs挂载上去了。但是后面在用postmark进行测试的时候，又出现了新的问题，报错信息比较模糊，通过dmesg查看日志发现错误定位在\_\_pmfs\_free\_block()函数中，并且有精确定位balloc.c第144行，查看是BUG\_ON(list\_empty(head))，而这个head之前初始化为sbi->block\_inuse\_head，是空的，猜想可能是这里出发了内核的预警，中断了postmark的操作。并且导致无法umount。

2018年4月11日 周三

今天主要是解决postmark无法完成测试的情况，在使用配置文件的情况下执行./postmark pm.cfg时，postmark程序会直接被kill掉，用dmesg查看日志信息发现错误是由于引用空指针触发了内核的BUG，导致程序崩溃，并且postmark崩溃，导致pmfs文件系统无法从卸载点umount，pmfs模块也无法用rmmod指令移除，在网上查看了大量的资料后，发现pmfs文件系统可以用umount –fl /mnt/pmfs指令强制卸载，但是pmfs模块的卸载却遇到了很大的问题，在移除pmfs文件系统后，用lsmod命令查看pmfs模块，发现引用计数为1，这也就是pmfs模块无法卸载的原因（提示module is in use），根据网上的资料，需要另外写一个force\_rmmod模块，来将pmfs模块的引用计数清零，但是我在尝试这种办法的时候，发现force\_rmmod被插上之后，该模块也会直接崩溃，并且无法使用reboot命令重启机器（这是最严重的！），并且用不同的方法实现force\_rmmod模块有不同的效果，在使用complete函数时，甚至会导致机器直接重启（算是一种间接重启的方法）。后来放弃了模块的卸载，每次都是重启后再编译测试（没办法的办法）。自己写了个small.cfg配置文件，测试10个事务，并且在自己写的几个函数中添加了大量的debug信息，最终才定位到是自己在写pmfs\_new\_block()时，有一种情况中，没有将二级索引节点Pmfs\_blockp与一级索引pmfs\_blocknode相互连接起来，导致发生了空指针引用，解决完这个问题之后，postmark程序才真正能够完全运行测试，但是新的问题是，测试结果竟然与优化前的方案一模一样！性能既没有降低，也没有提升，一番考虑后认为可能是由于postmark的序列化操作，以及配置文件中设置的文件size过小导致这种结果，于是重新设置了新的配置文件my.cfg，并重新分配了4G内存给pmfs，测试后仍然是一样的，这让我不得不怀疑postmark程序并不能测试出分配器的优化，需要寻求其他测试工具的辅助。