第 3 组进度报告

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 已完成任务 | 下周计划 | 备注 |
| 20090121 | 陈立标 | 1. 了解对非连续内存的分配和释放。清楚vmalloc()和vfree()的作用、非连续内存地址空间由vmlist链表进行管理。 2. 了解系统可以配置多个交换设备，可以用swapon()和swapoff()增加删除交换设备或文件，对于所对应的交换空间，使用swap\_info\_struct进行描述。 3. 了解linux的页面替换策略是双表针的时钟算法和页面记龄算法。 4. 了解linux为尽量保证系统中有一定量的空闲页框，使用kswapd每隔10s向free\_area链表中加入可用的空闲页框。 5. 了解页换入函数do\_swap\_page()和页换出函数try\_to\_swap\_out(). | 1. 组内交流沟通，一起构建内存管理模块的函数调用关系图，合理分配任务。 2. 通过对多本参考书籍的阅读了解自己负责的部分，并对源代码进行整体上的简单注释。 |  |
| 20020203 | 王思哲 | 0.了解i386硬件组成、用途，以及i386的分段分页机制  1.了解linux在i386中使用的分段和分页：  分段机制的核心是两张表GDT（系统段）和LDT（每个用户任务的段），要访问每一个段必须通过段选择寄存器；  分页机制由4个控制寄存器控制。当cr0的pg为1且pe为1时，启用分页机制，采用三级分页模式。 但实际上i386分页组织实际上是两级，为了代码的可移植性，使用pmd\_alloc函数完成转换。除此之外了解操作页目录和页表的函数和宏。  2.了解buddy算法原理  3.了解内存页框page的数据结构：  使用prev，next把物理页框插入双向循环链表，使用count记录是否被分配以及分配的数量，使用flag描述页框的状态。  4.了解内存页框page保存在mem\_map数组中，该数组由free\_area\_init()创建，然后会调用mem\_init()函数继续对数组各页框进行初始化，最后调用free\_page()函数计算出mem\_map的free页框总数，并把它保存在变量nr\_free\_pages中。  5. 了解使用buddy算法进行内存页框管理用到的数据结构：mem\_map数组，以free\_area\_struct（\*next,\*prev,\*map）为结构的free\_area数组（10个元素）  6.了解buddy算法分配内存块，使用函数\_\_get\_free\_pahes()实现，释放内存块由free\_pages()函数执行。  7.了解基于slab算法的内存区管理:  了解slab算法的必要性;  了解slab分配器的组成，了解其高速缓存的数据结构(struct kmem\_cache\_s)、高速缓存中每个slab的数据结构(struct kmem\_slab\_s);  了解高速缓存的两种类型，其中通用高速缓存由kmem\_cache\_init()和kmem\_cache\_sizes\_init()创建，专用高速缓存由kmem\_cache\_create()创建;  了解slab分配器通过调用kmem\_getpages()函数(以及里面的子函数)实现与buddy算法的交互，获得一组连续的页框;  了解通过kmalloc()和kfree()完成连续内存区的分配和释放; | 1.组内沟通交流，按照老师提供的建议，绘制函数调用图，梳理清楚函数之间的关系  2.继续阅读书籍，比照示例代码进行进一步的理解，组内探讨理清分析思路  3.开始尝试看源代码，并做简要注释，不懂的地方组内积极交流讨论 |  |
| 20105225 | 张瑶清 | 基本读完第二章，了解了Linux的硬件基础结构，启动内存装入流程。阅读了11.13-11.14内容，进程空间部分，了解了mm\_struct, vm\_area struct的结构和组成。了解了进程地址空间的存取权限，地址分配回收的基本函数，了解了缺页后系统处理异常和调页的流程。 | 详细了解mm\_struct, vm\_area struct的结构和组成，  详细阅读该部分代码即底层操作函数。  下载代码分析工具，进一步分析代码结构 |  |