

内核开发日志

我们的三名成员为来自杭州电子科技大学的

- 1. 队长 徐薪雨
- 2. 组员 高坚
- 3. 组员 陈亭材

我们的赛道为模块实验方向，致力于为原生的xv6系统上添加各种奇妙且实用的功能。

下图为仓库成员，其中OS为自动添加的机器人，其余为三名组员的自己的gitlab帐号以及比赛为我们分配的官方帐号。

























Account	Source	Max role	Expiration	Activity
<div><div>B</div><div><div>BOURBAKI04</div><div>it's you</div><div>@T202410336994369</div></div></div>	Direct member by BOURBAKI04	Owner	Expiration date 📅	User created: Nov 15, 2024 Access granted: Nov 15, 2024 Last activity: Dec 25, 2024
<div><div>O</div><div><div>OS Bot</div><div>@project_23833_bot_948edec26d8adcfa0c1abb3a62d4778a</div></div></div>	Direct member	Owner ▾	2025-01-03 📅	User created: Dec 04, 2024 Access granted: Dec 04, 2024
<div><div>W</div><div><div>Whale</div><div>@ChenTingcai</div></div></div>	Direct member by BOURBAKI04	Maintainer ▾	Expiration date 📅	User created: Nov 18, 2024 Access granted: Nov 18, 2024 Last activity: Dec 05, 2024
<div><div>B</div><div><div>bourbaki0404</div><div>@XuXinyu</div></div></div>	Direct member by BOURBAKI04	Maintainer ▾	Expiration date 📅	User created: Nov 16, 2024 Access granted: Nov 18, 2024 Last activity: Dec 25, 2024
<div><div>H</div><div><div>hardstone</div><div>@hardstone</div></div></div>	Direct member by BOURBAKI04	Owner ▾	Expiration date 📅	User created: Nov 15, 2024 Access granted: Nov 15, 2024 Last activity: Dec 22, 2024

11.16 ~ 11.17

在11月16日的时候，我们在仓库的main分支上传了xv6的源代码。YuanShen OS是基于裸的xv6系统，在其上增添系统调用与各种模块而成。在最开始的两天里我们为YuanShen OS加入了四个系统调用，分别为

- 1. sys_getcwd 打印当前目录路径
- 2. sys_nanosleep 精细化的睡眠函数
- 3. sys_trace 打印进程调用过的系统调用
- 4. sys_sysinfo 打印操作系统的进程数量以及剩余内存大小。

gitlab的提交记录由下图所示:

17 Nov, 2024 6 commits		
 syscall sysinfo added and tested. bourbaki0404 authored 1 month ago	a48576a3	 
 a cleaned version bourbaki0404 authored 1 month ago	8217521c	 
 syscall and user program trace implemented bourbaki0404 authored 1 month ago	a7118499	 
 Clear compilation intermediate files and write README hardstone authored 1 month ago	bab818fd	 
 add syscall sys_nanosleep and user program nanosleep. Add timer.h to restructure code. bourbaki0404 authored 1 month ago	1199fae7	 
16 Nov, 2024 3 commits		
 add syscall getcwd and user program pwd. bourbaki0404 authored 1 month ago	4b532c28	 
 add syscall getcwd and user program pwd. bourbaki0404 authored 1 month ago	7f8a32d2	 
 initial commit - a bare xv6 kernel without any modification yet. bourbaki0404 authored 1 month ago	6d00d54f	 

11.17 ~ 11.20



















在这段时间里我们跟着mit的实验完成了sigalarm和sigreturn系统调用的实现。它们的作用是设置一个极为原始的信号处理程序，这个信号处理程序甚至不能接收一个参数。

做完这个实验后，组员徐薪雨意识到这个简单的信号机制也许可以被用于实现抢占式调度的用户级，只需要用这个信号处理程序不断的打断用户进程的执行，并在这个过程中切换用户线程的上下文即可。

后来他意识到的一个难点是这个信号处理程序必须要获得进程在trapframe上保存的上下文才能做到这一点。然而信号处理程序是在用户态执行的，trapframe却只能在内核态才能访问到。通过查阅unix系统的信号机制，他意识到只需要让内核把信号处理程序所需的上下文复制到用户栈上，即合理的设置信号栈帧，并为信号处理程序设置一个类型为u_context_t的入参，就可以实现这一点。随即他利用改进后的信号机制完成了用户级多线程的设计。

组员高坚发现之前组员徐薪雨完成的系统调用sys_getcwd存在致命的错误，先前的做法只是在cd的时候将cd的入参做简单转化得到新的路径名。高坚将检索当前目录路径名的方法改为逐级查询父目录的方式，得到了正确的sys_getcwd。

这段时间，我们仓库的提交记录由下图所示:






















20 Nov, 2024 1 commit		
 add basic user-level thread support to xv6 bourbaki0404 authored 1 month ago	0228ad5a	 
19 Nov, 2024 2 commits		
 feat: add some basics of sys_getcwd hardstone authored 1 month ago	da87e395	 
 feat: add some basics of sys_getcwd hardstone authored 1 month ago	2c625409	 
18 Nov, 2024 3 commits		
 tidy version bourbaki0404 authored 1 month ago	8d4017c9	 
 vmprint added bourbaki0404 authored 1 month ago	b8e746fc	 
 syscall sigalarm & sigreturn implemented and tested bourbaki0404 authored 1 month ago	67734a01	 

11.22 ~ 12.12

这段时间隔了这么长是因为我们在完成学校布置的大量课程作业。在11.22日的时候组员徐薪雨根据浙江大学操作系统实验的文档意识到，xv6只有一个全局的进程内核页表，这导致在用户空间和内核空间转移数据要通过进程的页表手动翻译用户地址的方法进行，这非常低效率。而在浙江大学操作系统实验里，用户进程的地址空间可以出现在内核空间的下部分的未映射区域，而每个用户进程都要随着配备一个内核页表，这个内核页表上部是内核空间，下部是用户空间的一部分。随后他完成了这个设计，使得内核态的进程可以直接通过用户地址访问用户内存空间，大大提升了效率。

在12.11日的时候，组员徐薪雨参照MIT的实验文档network driver为YuanShen OS配置了e1000以太网卡并跟随实验文档实现了简单的UDP-IP协议。随之在第二天，他了解到伙伴系统在linux内存管理系统中的作用，于是他花费一天的时间设计并实现了一个新型的伙伴系统，其核心思想在于在一个完全二叉树的同一高度的节点建立的free list本质就是伙伴系统的free list，而通过维护不同树的节点的状态信息可以大大提升伙伴系统回收区间的效率。

这段时间，我们仓库的提交记录由下图所示:

12 Dec, 2024 2 commits		
 improved version of buddy system, use complete binary tree to speed up coalescence bourbaki0404 authored 1 week ago	4964a626	 
 buddy system (first version) added bourbaki0404 authored 1 week ago	3e4b5586	 
11 Dec, 2024 2 commits		
 tidy version bourbaki0404 authored 1 week ago	89ae5d1a	 
 add simple udp support to xv6 bourbaki0404 authored 1 week ago	9e22ab58	 
22 Nov, 2024 3 commits		
 cleaned version bourbaki0404 authored 1 month ago	469e1fd1	 
 exec fixed for copyout bourbaki0404 authored 1 month ago	443bd8b4	 
 add per-process kernel pagetable to support direct dereference of user address. bourbaki0404 authored 1 month ago	d4a93c1e	 

12.13 ~ 12.17

接下来的时间我们实现的功能主要围绕内存管理，包括虚拟内存和物理内存。

组员徐薪雨在12.14日参考linux操作系统关于虚拟内存的优化方案，确定了接下来虚拟内存方向的所有工作目标:

1. Copy-on-Write fork
2. 关于堆内存的惰性分配
3. 关于mmap系统调用的惰性分配
4. 最终目标，借鉴linux中vma(virtual memory address)的设计组织进程的用户态空间地址范围，并实现exec函数的惰性加载，延迟加载进程的各种程序段。





































随后他于当天完成了COW fork的代码编写与测试。

之后他承接之前完成的伙伴系统的部分，考虑到伙伴系统最小只能分配一个页面，而内核中的很多动态数据结构的大小是以字节为单位的，因此他决定实现一个具有更细粒度空间配置能力的内存分配器，例如linux里的slub分配器。

了解到slub分配器的契机是浙江大学操作系统实验2024年实验五的初版，里面简略的提到了slub分配器与kmallocc函数，不过这个文档在更新后删除了slub分配器的内容。于是他通过两天时间在网上查阅资料了解slub

分配器的基本原理，并删除了其中关于kmem_cache_node和kmem_cache_cpu的二级缓存设计，决定采用一级缓存的设计，所有的cpu共用一个kmem_cache进行内存分配，并通过一个锁来保护共享变量。在12.17日他完成了这个简化的slub分配器，并编写了测试代码。

这段时间的仓库提交记录如下图所示:

 slub allocator implemented and fully tested bourbaki0404 authored 1 week ago	d9a4331f		
 slub allocator implemented and fully tested bourbaki0404 authored 1 week ago	73b8217e		
 slub allocator half completed bourbaki0404 authored 1 week ago	7c342890		
 slub allocator half completed bourbaki0404 authored 1 week ago	1ae70319		
16 Dec, 2024 4 commits			
 repaired bourbaki0404 authored 1 week ago	24a3fb6d		
 repaired bourbaki0404 authored 1 week ago	8e245450		
 new backup bourbaki0404 authored 1 week ago	6cf8d87f		
 new backup bourbaki0404 authored 1 week ago	d593ea21		
14 Dec, 2024 4 commits			
 error repaired by adding page fault handler to kerneltrap bourbaki0404 authored 1 week ago	7ed0b017		
 COW fork supported bourbaki0404 authored 1 week ago	6591df7f		
 COW fork supported bourbaki0404 authored 1 week ago	ac9fd18b		
 directory structure improved readability bourbaki0404 authored 1 week ago	2d47161b		

12.20 ~ 12.23

这段时间的主要工作为:

组员徐薪雨注意到xv6原生的调度器与别的模块联系较少，修改起来可以不用担心要修改很多的别的模块的代码。原先他准备在YuanShen OS上实现linux的O(1)调度器，使用多个不同优先级的链表组织管理所有进程。然而他意识到这样实现不够有意思，因此他决定实现linux目前使用的CFS调度器(顺便写写红黑树练练手)。在参考了youtube和zhihu上对CFS调度算法的解析后他花了两天时间实现了CFS调度器。






















在此期间他解决了两个bug，第一个是因为多个进程可能有相同的vruntime(cfs调度算法要用到的一个指标)，但是直接的红黑树是不能容纳重复索引的，因此他为红黑树的节点增加了一个next字段，让红黑树的一个节点上能建立起一个链表用于安置具有相同vruntime的其他进程。

第二个问题在于他之前的代码在删除红黑树的根节点的时候少写了一个对空指针的判断。当时由于虚拟内存方面的设计存在一点小问题，导致对空指针解引用并不会报异常，而只是让整个内核卡住，没有其他任何的提示信息。他花了一天的时间在各种地方打printf才发现这个问题。

在12月22日，他为YuanShen OS添加了简单的内核信号量支持(只需要一个睡眠锁，再修改下wakeup函数的逻辑就可以轻松完成)。之后他又在12月23号完成了mmap和munmap系统调用，并把最简单的，类型为VMA_FILE的vma引入了YuanShen OS。

同一时间，组员高坚则根据mit的文件系统实验，为YuanShen OS增加了大文件和符号链接的功能。通过大文件的拓展，本来的用户文件夹只能放置13个测试程序，现在可以放置更多的测试程序。

这段时间的提交记录如下图所示:



24 Dec, 2024 1 commit		
<div><div></div><div><div>add mmap & munmap</div><div>bourbaki0404 authored 2 days ago</div></div></div>	05711969	<div></div>
22 Dec, 2024 1 commit		
<div><div></div><div><div>add mmap & munmap</div><div>bourbaki0404 authored 2 days ago</div></div></div>	1af49eed	<div></div>
20 Dec, 2024 2 commits		
<div><div></div><div><div>On branch feature/add-bigfiles-and-symbolic-links</div><div>hardstone authored 3 days ago</div></div></div>	5515357b	<div></div>
17 Dec, 2024 4 commits		
<div><div></div><div><div>CFS fixed and tested</div><div>bourbaki0404 authored 5 days ago</div></div></div>	9c001de5	<div></div>
<div><div></div><div><div>CFS fixed and tested</div><div>bourbaki0404 authored 5 days ago</div></div></div>	1f6a843f	<div></div>
20 Dec, 2024 5 commits		
<div><div></div><div><div>test</div><div>bourbaki0404 authored 4 days ago</div></div></div>	f20f9ce8	<div></div>
<div><div></div><div><div>semaphore added</div><div>bourbaki0404 authored 4 days ago</div></div></div>	6683444c	<div></div>

12.23 ~ 12.24

在这段时间内，组员徐薪雨拓展了原有的vma功能，并把之前完成的堆内存惰性分配，cow fork重新用vma加以组织。参考浙江大学2024年的操作系统实验5,他意识到可以用vma来实现用户进程的程序段的惰性加载，直到程序访问到代码或者自己的数据时才把它们加载到进程的地址空间里。

中间他遇到的一个主要问题是没有正确理解elf格式中程序段的filesz和memsz的具体含义，导致他在为进程添加VMA_TEXT和VMA_DATA的时候，从可执行文件里多读了不该读取的内容，导致用户进程的那些本该初始化为0的数据段内容几乎完全错乱。在修改了添加vma的核心逻辑后他完成了这块功能，并将原先usertrap里杂乱无章的缺页异常处理方法通过vma进行了有效的组织。

提交记录如下:

24 Dec, 2024 1 commit		
<div><div></div><div><div>lazy loading implemented</div><div>bourbaki0404 authored 21 hours ago</div></div></div>	1f07492a	<div></div>
23 Dec, 2024 7 commits		