### 老师动员

今天主要为老师动员，讲解了实习注意事项，包括安全以及实习所需材料等等。暑期结束，学生返校后一周内进行考核并评定成绩，考核时应提交的实习材料包括：生产日志、生产实习报告和生产实习考核表。学生在离开实习单位时，一定请接受实习的单位在介绍信和实习大纲最后一页上盖章；在华中科技大学计算机学院各老师课题组从事科研实习的同学，可由实验室盖章，也可开学后由实习指导老师组织到计算机学院盖章。实习日志要求书写认真、必须手写，日志内容应与实习内容和报告内容相关，要求至少2天一次日志，必须包括老师动员、第一次接受具体实习任务情况、实习总结等部分的记录。

### 第一次接受具体实习任务7.10

今天主要由吴松老师与王坤博士在南一楼对要在服务计算教育重点实验室进行实习的同学讲解了具体实习内容。我们实习课题为：基于ARM64架构的容器环境下系统调用隔离性分析。实习内容为：首先，针对5.1.16版本linux内核分析系统调用执行流程和涉及到的资源；然后，分析以上资源在容器之间是否共享、是否会产生竞争，若有竞争会产生哪些影响；最后，根据以上分析结果对系统调用进行分类和总结。然后对该任务进行了分工，我的任务是对404-427号系统功能调用的分析与总结。

### 7.12

这两天主要完成了容器隔离性机制namespace与cgroups的了解，以帮助后面系统调用的分析与隔离性分类。其中，namespace提供了六种隔离机制，包括UTS（主机名与域名）、IPC（信号量等）、PID（进程号）、network（网络设备、端口等）、Mount（文件系统）、User（用户和用户组）。Cgroup 为​​​系​​​统​​​中​​​所​​​运​​​行​​​任​​​务​​​（进​​​程​​​）的​​​用​​​户​​​定​​​义​​​组​​​群​​​分​​​配​​​资​​​源​​​ — 比​​​如​​​ CPU 时​​​间​​​、​​​系​​​统​​​内​​​存​​​、​​​网​​​络​​​带​​​宽​​​或​​​者​​​这​​​些​​​资​​​源​​​的​​​组​​​合​​​。它主要提供了如下功能：Resource limitation: 限制资源使用，比如内存使用上限以及文件系统的缓存限制。Prioritization: 优先级控制，比如：CPU利用和磁盘IO吞吐。Accounting: 一些审计或一些统计，主要目的是为了计费。Control: 挂起进程，恢复执行进程。

### 7.18 //7.15（一）

今天主要完成了404号系统功能调用的流程以及所涉及的系统资源分析。

之前完成了namespace和cgroups实现的隔离性的学习与掌握，今天开始对系统调用进行隔离性分析。我的任务中第一个系统调用clock\_settime函数，系统调用号为404。在对系统调用的流程分析之前，需要掌握该函数相关的数据结构，包括结构体k\_clock、clock\_posix\_dynamic、timespec等等。此外，在该系统调用中也涉及到其他许多函数，包括clockid\_to\_kclock、get\_timespec64等等。最后大致了解该系统调用所涉及到的数据结构、函数后分析系统调用源码得到实现流程。分析完该函数调用后可知该系统功能调用主要实现设置系统时间秒数与纳秒数。参数分别为：时间类型、timespec类型的时间。时间类型根据定义有：CLOCK\_REALTIME、CLOCK\_MONOTONIC、CLOCK\_PROCESS\_CPUTIME\_ID、CLOCK\_THREAD\_CPUTIME\_ID四种。时间结构内包括秒以及纳秒信息。所以该系统调用主要影响系统的时间，后续需要判断是否已经实现隔离。

### 7.19 //7.16

今天主要完成了404号系统调用时间信息是否在容器中完成了隔离的尝试性测试。以及部分405号系统调用的流程分析。

测试时首先完成了docker的安装，以及使用方法的初步学习。然后在容器中安装必要的包后，编写测试程序。必要的包有：vim、gcc等。但是在运行测试程序时发现，系统调用404号执行失败，之后经过查询资料发现是设置的时间大小问题，尝试修改后，可以执行成功。该系统调用docker没有实现隔离，此外，在修改时间时还存在锁的竞争。

此外，今天也大致查找到了405号系统调用的源码以及涉及的数据结构及其函数的分析。涉及的数据结构与404号大致相同，函数包括copy\_from\_user：将相应内容拷贝到内核空间；copy\_to\_user：将对应内容拷贝到用户空间；do\_clock\_adjtime：主要函数，目前尚未分析。

### 7.22 //7.18

今天（这两天）主要完成了405号系统调用隔离性分析、406号系统调用功能流程和隔离性分析、407号系统调用功能流程分析。

经过分析405号系统调用与404号系统调用涉及的系统资源相似，容器下的隔离性也相似，时间上容器没有实现隔离，同时也存在对timekeeper的锁竞争。

406号系统调用涉及的数据结构与前面大致相同，该函数主要获取对应时钟类型的时间精度，只是对相关宏的读取，所以不存在隔离性问题。

407号系统调用比较复杂，参数也比较多。该系统调用主要提供设置进程纳秒级的休眠时间。由于该函数涉及到的函数较多，隔离性分析还未完成。

### 7.23//7.22（一）

今天（这两天）主要完成了407号系统调用的隔离性分析，408、409号系统调用的功能、流程、隔离性分析。

在昨天（之前）的基础上，~~今天~~继续结合源码分许出407号主要设计进程描述符task\_struct结构以及高精度定时器的使用。由于只跟进程相关，所以docker容器间也不存在隔离性问题。

408、409号都是定时器系列的系统调用，408号用于获取定时器剩余时间，409号用于设置定时器时间。这两个系统调用主要涉及定时器的读取与设置，这些内容都存储在一个哈希表中，而存储位置与本进程和定时器ID有关，所以不同容器之间不会发生冲突，这两个系统调用在容器环境下都不存在隔离性问题。

### 7.24

今天（这两天）主要完成了410、411、412号系统调用的功能、流程、隔离性分析。

410、411号两个系统调用属于同一系列，主要用于基于文件描述符的定时器。这两个系统调用涉及到了新的数据结构，包括fd、file、timerfd\_ctx等，fd是文件描述符包括文件指针file和标志，同时文件指针的成员private\_data指针指向了定时器相关结构timerfd\_ctx。二者实现时，即是通过文件描述符获得该定时器相关结构，从而获取或者设定定时器时间。根据实现流程，二者只读取基于文件描述符的定时器结构，而该结构只与进程相关~~，所以不同容器之间不存在隔离性问题。~~

412号系统调用用于纳秒级设置文件的时间戳。该系统调用主要通过文件描述符或者文件路径获取文件的inode，从而对其修改。因为涉及到文件系统并且修改了文件的inode，所以不同容器之间存在隔离性问题，但是由于namespace已经隔离了不同容器之间的文件系统，所以该问题已经解决。

### 7.25

今天主要完成了413号系统调用的相关数据结构、功能、流程、容器环境下隔离性分析。

413号系统调用pselect6涉及的数据结构与之前的数据结构都不太一样，并且该系统调用没有相关资料可以查阅，只能从linux源码中查找相关代码，把这个函数弄明白花费了不少时间。该系统调用涉及到了大量的宏定义，包括：sigset\_t、compat\_size\_t等等定义。此外，该系统调用可以设置进程阻塞时间以及信号，所以也涉及到了信号的数据结构，以及进程数据结构等等。该函数的参数包括6个，功能主要为设置进程纳秒级的阻塞时间并等待指定信号，如果超时或者信号到达继续执行。~~虽然，分析该系统调用花费了不少时间，但是，该系统调用只涉及到进程阻塞信号以及阻塞时间的修改，所以容器环境下隔离性不存在冲突问题。~~

### 7.26

今天主要完成了414号系统调用的分析，以及5.1.16版本的linux内核编译工作。

414号系统调用ppoll与413号系统调用类似，都是用来等待一组文件描述符上的信号，不同之处在413号系统调用使用三个参数表示三种信号，而该系统调用使用一个pollfd结构数组表示所有的信号，该结构中包括信号、对应文件描述符以及返回的结果。此外，在等待具体信号实现时，源码中使用了poll\_wqueues结构，但是只是初始化并使用了，并没有找到与全局数据结构相关的地方，内部实现原理没有搞太明白。其它地方与pselect6基本一致，所以，在hrtimer\_cpu\_base、runqueue上容器环境下存在锁隔离问题，poll\_wqueues上是否存在隔离性问题存疑。

此外，因为分析的是linux5.1.16版本的内核，所以今天我编译了该版本的内核并且成功安装了，以便后续使用。

### 7.29（一）

今天主要完成了416号系统调用的功能、流程、隔离性分析；以及linux最新版本的man-pages的更新。

416号系统调用io\_pgetevents又是一个单独种类的系统调用，是对之前aio种类的系统调用的补充。但是分析这一个系统调用，需要把该种类的所有系统调用都搞清楚。分析该系统调用的数据结构以及机制花费的很长时间，包括的数据结构包括aio\_context\_t、io\_event、kioctx等等。该系统调用的机制大致是通过io\_setup系统调用获取的aio\_context\_t类型句柄获取内核中对应的kioctx结构，该结构中的ring\_pages成员指针指向了用户空间存放异步事件处理的结果，然后从该内存中读取结果即可。

除此之外，由于分析过的系统调用涉及到了很多rcu锁，而之前没有认真分析该锁机制，所以今天也了解了下该锁的实现机制。

由于这一部分的系统调用在网上的资料很少，并且源码的注释较少，所以今天在之前编译的内核中安装了最新版本的man-pages便于查找相关函数，但是，安装之后发现一些系统调用仍然查不到。后续系统调用还是要主要依赖源码分析。

### 7.31

今天（这两天）主要完成了417号系统调用的分析，以及418号系统调用流程的部分分析。

417号系统调用recvmmsg用于从一个套接字接收多条信息，该系统调用在我的任务中又是一个单独的功能种类，是对socket系统调用的补充，需要重新分析该种类的数据结构。该系统调用的数据结构分析时间较长，相对而言，在数据结构分析完后流程的分析比较顺利。隔离性方面，~~该系统调用存在容器之间的冲突，但是namespace机制已经解决该问题。~~

418号系统调用mq\_timedsend用于向一个消息队列发送消息。同样的，在我的任务中该系统调用只与419号同类，需要重新分析。~~今天~~只完成了数据结构的分析，以及部分流程的分析（这个系统调用实现涉及的源码太长了！）。明天（之后）继续努力，正确把该系统调用以及419号的分析完成。

### 8.1//8.2

今天（这两天）主要完成了418号系统调用剩余部分的分析以及419号系统调用的分析。

在昨天（之前）分析418号的基础上，~~今天~~完成了该系统调用全部实现流程的分析，以及隔离性分析。该系统调用主要用于向一个消息队列发送消息，隔离性问题肯定存在，同时，namespace机制中存在IPC的隔离实现，该隔离性问题应该已经被处理了，但是后续仍然需要进一步验证。

419号系统调用与418号属于同一类，数据结构基本一致，在原来的基础上分析简便了许多。该系统调用主要为从消息队列接收消息，实现流程与418号也大致相同，隔离性问题同样也存在，但是是否已被IPC隔离处理，后续仍需验证。

### 8.5

今天（这两天）主要完成了420号系统调用发分析。

之前稍微看了一下该系统调用的功能，感觉挺好理解，但是今天发现该调用的源码是真的太长了，而且该系统调用在我的任务中又是单独一系列，是对semop函数的补充。该系统调用主要功能为实现对信号集的各种操作，并且提供阻塞时间的设置。源码中涉及了大量的新数据结构以及相关函数，分析非常费力，今天只是分析了该系统调用实现的大致思路，其中有些地方还没有完全搞明白，然后根据实现的流程分析了该系统调用的隔离性问题，因为每个信号集对应系统中的一个sem\_array结构，所以容器之间存在隔离性问题，但是IPC隔离机制已经实现了该隔离问题的处理。

### 8.7

这几天主要完成了421-427系统调用的分析。

这几个系统调用功能上基本都是零零散散的，其中包含一个linux5.x中新加入的系统调用类：io\_uring，总共有三个系统调用。该类系统调用内核实现中包括一个io\_ring\_ctx结构，提交队列、完成队列都在该结构中；此外，该类系统调用初始化时可以创建一个内核线程用于处理IO请求，这样不需调用专门的系统调用即可得到请求结果。io\_uring\_setup系统调用涉及到内核线程的创建，需要插入到内核数据结构kthread\_create\_list中，所以存在锁隔离问题；io\_uring\_enter系统调用可以唤醒创建的内核线程，所以存在对运行队列runqueue的修改，同样是锁隔离问题；而io\_uring\_register只需更改io\_ring\_ctx结构即可，不存在隔离性问题。

### 8.7//8.9

~~今天（这两天）主要完成了对413、414号系统调用的实现、隔离性重新分析。~~

~~之前完成了413号系统调用的分析，但是并没有详细了解内部实现原理以及相关数据结构；并且414号系统调用同样没有分析。所以今天就这两个进行补充。~~

这两天主要完成了对之前完成分析的所有系统调用进行修正。

由于完成分析任务后还有几天时间并且之前一些系统调用分析的不彻底，所以我又对所有的系统调用分析了一遍。重新分析时发现了如下问题：系统时钟类系统调用由于是第一个分析的，所以实现流程分析的不够彻底；413、414号系统调用内核中实现理解不够清晰导致隔离性分析错误等等问题。

### 总结答辩8.12（一）

今天主要在南一楼进行了实习总结答辩，到场的有王坤博士以及其他几位学长。答辩中我对自己完成的工作进行了一个汇总，我以系统功能调用的功能分类，分别讲述了404-427号系统调用的实现流程分析、内核中涉及到的数据结构以及隔离性分类。汇总完成后，学长们重点询问了linux5.x中新加入的io\_uring类系统调用，该类系统调用可以在内核中创建一个线程专门处理io请求。最后我的答辩顺利结束。