高级操作系统课程报告

李家郡 2019104205

本学期的高级操作系统主要围绕《UNIX 环境高级编程》展开，课程的学习主要通过老师上课讲授、课后自行实现代码展开。关于 Unix 系统，再此之前，我只有一个粗略的认识，在本次课程中，除了学习到 Unix 系统的基础知识外，更主要的是让我意识到这样的一个底层系统，其内核的实现是非常完备的。苏老师经常在授课过程中谈及安全的重要性，虽然我们不做攻击系统的任务，但了解系统内部构成，是能进一步认识到如何防控风险的。例如，让我印象最深的一点是，但凡是攻击系统，必然首先要从普通用户，拿到 root 权限，至于使用注入，信号，溢出等等方式都是攻击的手段。Unix 知识的广博，实在不是我一学期的课程就能完全掌握的，因此我也已经建立了 github 上每章练习的 Repository，可能会在未来将书上的练习一一都写到里面，这样可能才算是真正入门 Unix。接下来，我将挑选本课程我印象比较深刻的几个知识点和我实际任务中产生的启发进行记录，以此反馈我所学习到的知识。

一、 UNIX 基础

AWK

在这节课之前，我并不知道什么是 AWK。让我认识到，原来在 Unix 自带的系统中，是可以通过脚本批量生成源代码的。这是妥妥的生产力呀，利用 awk 可以批量生成多组实验的脚本，再利用 shell 批量执行，就可以一次性完成很多实验。由此看来，Unix 系统还有许多值得研究的命令和功能。

标准文件与配置

本书给出了 UNIX 下，ISO C 标准定义的头文件，给出了一个完整的体系。在学校的 C 语言课程中，实际我们只关注于其中很小的一部分内容，这其实是不合适的。说起来，C 语言掌握的程度实际上是与这些库深切挂钩的。例如，在 python 中，我经常会用到的异常捕获机制，在一般的 C 语言编程中，我们却很少使用。实际上 C 语言的错误机制应该更底层，更精准。其实老师课程上说的错误代码给了我很深的体会，从错误码判断程序的错误是一件很体现程序员编程能力的事。一方面是知道错误代码背后所实际代表的含义，另一方面是解决这个错误，需要使用什么样的手段。本科与研究生的鸿沟可能就在于此，现有的培养方案其实是比较缺乏的。现有的培养方案依然在照顾那些本科没有学好的知识，而太少着力于更深入知识的讲解。虽然这部分知识，并不能完全依赖于老师的讲解，更多的需要自己去查找和体会，本课程则是给出了一个学习的方向。其实本课程的展开思路也就依赖于这些库。标准 I/O 库支撑起了 I/O 部分的内容，信号库也是我们平常完全没接触过的库，这也就是我们为什么信号知识匮乏的原因。

另一方面，通过对本书的环境配置，我认识到每个 Unix 系统头文件所在的位置是不太一样的。用./来代表当前目录也是之前忽略的一件事。以前只认识到了自己写的可执行文件需要用./xxx 来运行，而常用的程序是使用其名字就可以执行的原因。这让我意识到，现实编程问题中存在着很多我忽略的异常的部分，但由于经验、习惯，我将其看成很自然的事，但实际上这些事并不是那么自然。在最近师弟问我在一种我不熟悉的语言下，编译 c 语言不成功的事件中，我深刻体会到了这点。不知道 C 语言的原理，想要直接使用，很可能导致连出错都不知道错误在哪，更不必说如何修正这个错误。他的错误在于，没有理解.h 文件、.so 文件、.a 文件这些文件的具体用法，和对应关系。通过本课程，我也确实是第一

次理解了动态库、静态库、头文件之间的关系。之前我是都听过这些概念的，但都没去仔细的研究，但在课程讲解的过程中，从运行实例的过程中，我自发地开始意识到这些东西都不像表面的那么显然，而是存在着一些非常合理的规则。

I/O、线程、信号、进程、锁等等这些概念，虽然通过短短的课程，我可能只是听了个大概的印象。但是在我比较好的理解了环境配置，出错问题需要从哪方面去研究，遇到问题需要寻找哪个库去解决（如果 Unix 自带的库不存在的功能自然知道了需要去找第三方库），那么我觉得之后我再继续自己慢慢体会这些内容也是很容易的事。只是需要一定的时间的耐心去积累。

二、 I/O

文件 I/O 对算法效率的启示

我主要研究的内容为算法效率，所以对 IO 效率关注得比较多。常规算法通常都是“内存”算法，也就是研究内存情况下的运行效率，但这样的算法往往在实践中没有外存算法应用要广泛。就以搜索树为例，虽然平衡二叉搜索树有非常好的理论效果，但几乎不会有人在写二分查找时，建立一棵红黑树。在数据库中使用最多的，通常也是 B 树以及 B 树的变种，这其实是和课程中的文件 IO 息息相关的。再比如说，虽然基数排序有很好的理论性质，效率为 O(n)，堆排序、快速排序的效率都为 O（log n），但在实际数据库实践中，我们却都是使用基于归并法的外排序。

这里提及这些算法的原因是在于以前的认知中，我会认为缓存越大越好，也就是说，例如在 B 树中，我朴素地认为将页换入到内存中就能提高效率了。但是文件的 IO 告诉我，并不是这样的。最让我印象深刻的是第 5 章 5.8 的实验，原来缓存并不是越大越好，这也就意味着，以上的算法，也并不是将越多页换入到

内存中，就会有更高的效率，而这个实验则能很好的给予我一条探索的道路。

在做数据库频率估计的时候，我就遇到了这样的实例。通常的随机采样都是需要生成随机数，不断读数据到内存中，再进行估计，这样的效率其实是比较低下的。比如，若是均匀采样，采到的样本分布在所有的块上，那就意味需要从所有块分别提取数据，那就必须要将所有的数据分批都读如内存。但我们如果将算法改进，通过块采样去替代均匀随机采样，那效率将成倍提高。实际上，这也是一种利用缓冲的技术，那么需要进一步研究的就是到底一次将多少块数据放到内存是比较合适的。

三、　线程与进程

算法很多都会追求需要多线程和可并行化。在我实习时，对这点深有体会。但是我对线程与数据库的结合体会并没有那么深。整个课程是复习，也有新知识的引入。关于锁的概念，初次学习还是在大二学操作系统的时候，但由于掌握得不够深刻，很多细节都并没有记得非常清楚。最难得的是，苏老师能将这一系列的知识串联起来，看到整个知识的系统脉络和联系。例如线程的死锁是和数据库的死锁息息相关的。线程的关系和信号传递也是息息相关的。线程存在着父子关系，线程的具体结构说起来复杂，却很有章法和结构。其次是让我看到了安全与线程紧密的关系。通过线程栈的溢出，和改变信号结构，可以有许多攻击的手段，这点是只学线程表面的东西做并行计算所无法感受到的事。

守护进程

关于守护进程也是课程讲解得比较多的一个部分。在此之前，我完全没有听过守护进程的概念。虽然确实在 top 中见到过很多写着 daemon 的进程。相信在之后看到这些守护进程不会再陌生。

进程通信

进程通信也是编程实战中很重要的一个部分。一个多进程的程序效率高不高，很多时候是取决于进程通信做的好不好。现在的代码通常都会封装得很好，导致我们很多时候都忽略了进程通信在底层的实现。这不得不提到网络的通信。网络的通信也是一样，我们大多都是使用已经封装好的通信方式。加密和封装好的通信有其安全上的优点，但我们也不能只知道表面上的协议方式，而忽略底层的实现，只有对底层实现足够理解，才能明白协议设计的必要性，和合理性。我们始终是在应用层看问题，但要改进应用层的效果，有时还是需要深入到实现层去看。

四、 信号

信号过去就不是很懂，现在感觉依然一知半解，但是已经可以指导我解决一些问题了。这学期从信号原理理解了一件以前一直没明白的事。在我们写代码时，通常都会因为需要长时间执行，所以会将程序挂起到后台执行。这时候挂起执行的方式是使用&，一个运行的程序放在后台则是需要利用 control + Z 将其挂起，在用 bg 命令放到后台执行。甚至在本学期之前，我不是很能理解 control+Z 和 control+C 的区别，更不必说还有 control+D 之类的。过去遇到一个代码卡住时，通常就会按住 control 再按其他的好多按键，总有一个能将其停下来的。但是，从这学期的信号学习中，我才认识到，原来信号的种类有非常多，信号之间也是有级别的，还可以屏蔽键盘的操作。我认为这才是计算机专业学生应该学会的操作系统知识，这门课变成选修课确实十分可惜。比起学科基础的操作系统，我认为这些知识才是我们欠缺的。那门基础的操作系统，一方面是在学习各种 linux 操作命令，另一方面是在学习分布式系统，但这些都脱离了操作系统的基础。实际上，我们在工作或者科研的过程中，首先需要面对的其实是自己手中的机器，

而不是分布式的机器。我们并没有想象中对单机操作系统掌握得很好，另一方面是我们也并不是老师想象中对操作系统掌握得那么差。举信号的例子来说，虽然我以前确实不知道 control+Z 和 control+C 是信号且有区别，但是并不阻止我使用它。研究生的课程确实应该从教命令，知识点改变为教授背后的原理，虽然这确实很难。

借用进程和信号原理，我终于明白了，为什么在将任务放到后台运行前，需要加一个 nohup 的命令，那是需要屏蔽来自父进程的中断信号。我也终于明白，挂起后再放到后台的进程和没使用 nohup 就放在后台执行的进程，在终端关闭后，为什么会随之结束。这是我在本科知道，却不明白原理的一件事。理解到整个系统是由一个又一个进程所组成，进程又有着其自己的父进程，而终端开启的进程，则需要收到终端的控制。传说中的挖矿程序也就不那么不可知了。而由这个理论，我也很容易搜到了解决由于终端关闭，所传递中断信号，使后台程序关闭的解决方案。其实也就是将信号屏蔽即可，一方面是使用 nohup 命令，另一方面是使用类似于 screen 之类的命令，创建一个与当前终端不想干的“界面”，自然就不受终端的影响，最后是直接忽略信号，即利用 disown。

五、 总结与课程建议

本次课程还是学到很多东西的，总体来说，就是对 Unix 系统有了更深刻的认识。但实际上，我对 Unix 的理解还处于一个比较浅层的地步，还需要花大量的时间好好啃这本书。稍微对本课程有点建议，我个人感觉我理解比较深刻的地方还是 IO 的部分，主要原因是老师带着做了对应的练习。由于课程时间的限制，这课时间实在是太短了，感觉练习的时间不够。其实 EJB 那种，先投屏抄代码，再花时间去解析的方式挺好的，如果能将讲解串接到练习中，可能我的理解会更

加深刻。虽然我们自己也能在下面运行测试代码，但是有时候虽然跑了，但没有意识到问题所在，没有创造性的发现。