课堂讲稿

王晓林

2020年2月22日

目录

1 [2020-02-20 Thu] **老生常谈** 1

2 [2020-02-21 Fri] **关于作业** 3

3 [2020-02-21 Fri] **幻灯片第7页，什么是操作系统?** 4

4 [2020-02-22 Sat] **幻灯片第8页，OS的功能模块** 6

5 [2020-02-22 Sat] **幻灯片第9页，选个OS** 9

1 [2020-02-20 Thu] 老生常谈

我提供的讲义、幻灯片、参考书等所有资料都是英文的，这有几方面原因。首先，英文真的很重要。怎么算是重要？我们来举个例子，假设我现在给你留一个作业，比如写一个简单的system call吧。面对这个作业，你会发现有诸多困难要克服：

1. 不知道什么是system call；（缺乏操作系统的相关知识）
2. C编程啊？不会；
3. 你说的这个编辑器（Emacs, Vim）我也没用过；
4. 没标准答案？要我上网搜什么？
5. 你提供的参考资料都是英文的？

6. 这作业太难了……（我没这份耐心）

也就是说，想完成作业的话，你要克服上面这6个困难才行。而且，上面这6条，是我按它们的重要程度排好了顺序的。其中最重要的是“耐心”，最不重要的是“操作系统知识”。为什么？因为，你们敷衍完这门课之后，恐怕没有谁会去搞操作系统开发。可是，无论你将来做什么，只要你想稍微做出点样子，你都必须要有“耐心”。

1. 耐心，无论干什么，都需要；
2. 英文，只要想找个好工作，就需要；
3. 上网搜，也就是科研能力，如果你将来搞技术工作，那么肯定需要；
4. 编辑器，也就是软件工具，只要你用电脑就需要；
5. 编程，只有程序员才需要；
6. 操作系统，只有搞系统开发才需要。

也就是说，最通用的才最有用。尽管你以后不搞系统开发，甚至都不搞技术，但既然选了这门课，总该有点收获，那就借着这门课的机会，补习一下你的英文吧。如果你打算将来吃技术饭，那么英文就更加重要了。现代IT技术，都是西方人搞出来的，低头看看键盘，上面就没有一个中国字，不老老实实把英文学好，你的专业技术也不会有什么前途。

面对这份作业，我相信大家完成作业的速度和质量也会大不一样。通常，好同学会完成得又快又好，为什么？因为学习有“捷径”呗。好同学的捷径是什么？是这样：

1. 他做事的耐心早就锻炼出来了；
2. 他的英文已经学得不错了；
3. 他经常用Google查英文资料，而不是百度中文资料（没错，google比百度强太多了；英文技术资料的数量和质量都比中文的强）；
4. 他Linux用得很熟；
5. 他C编程也不错；
6. 他唯一需要现学的就是一点操作系统知识。

也就是说，貌似你们是同时开始做作业，同时开始克服这6个困难，但实际上，他已经提前把5个困难克服掉了，而你是从现在才开始启动，所以他当然比你做得快、做得好。所谓捷径，无非是笨鸟先飞罢了。

「也不全是吧，老师，那孩子真的比我聪明」。聪明不重要，至少在本科学习阶段，真的不重要。这不是说聪明不好，就好比电脑的CPU，当然是越快越好。但只是硬件好，软件很烂，电脑也做不出什么正经事。如果你真感觉聪明不足，那么可以学点不需要聪明的东西，比如英文。英国的傻子都比我们英文好，不是吗？可你的英文为什么这么烂？很简单，因为你懒。

「英国的傻子有良好的“语境”」。那么，你的高中同学英语都和你一样烂吗？和你同龄的年轻人，他们考进了清华、北大、云大……英语显然比你强。所以，别扯什么“语境”的淡了，懒就是懒。发现问题，正视问题，才可能解决问题。真想学好，就好好学。学习是需要耐心的，像长跑一样，只有气喘吁吁、大汗淋漓，才算是锻炼，才能锻炼出强健的体魄。

2 [2020-02-21 Fri] 关于作业

下周一，我应该可以收到你们的作业了吧？「还什么都没讲，怎么做作业啊？」因为我布置的作业很简单，就是每周问一个问题而已。因为我还什么都没讲，你还什么都没学，所以你满脑子都应该是问题，问一个出来有什么难的？话虽如此，但提问有提问的规矩。在第4页幻灯片上，我把规矩列出来了。提问要有条有理：

1. 我要做一件什么事情？
2. 我是怎样做的？（也就是步骤）
3. 做到哪一步的时候，我期待看到怎样的结果，可实际上看到的却是……
4. 于是我尝试了如此种种若干办法去解决它；
5. 我在这个问题上花费了多少小时？

很显然，这个作业的意义就在于“做事情”。如果你能保证每周做一件稍微像样的事情，花费若干小时真正去克服一些困难，那么你在毕业的时候就可以傲视身边这群懒蛋了。“傲视懒蛋”，这真的算不上有出息，但这至少是第一步。想有出息，就要时刻提醒自己「我未来的竞争对手绝不是身边这

些比我更懒的游戏帅哥、追剧美眉」。想有出息，就要去和校外比，和省外比，和国外比。

「可是，我真的想不到有什么正经事要做啊」。如果你打算有出息，只是对操作系统缺乏兴趣，那么没关系，我说过了，操作系统是学习中最不重要的事情。你尽管去做你感兴趣的事情，然后在作业里告诉我，你是怎样做的，花了多少小时，克服了哪些困难？更直白地说，只要你给我一个好印象，期末考试是不成问题的。

「我还是不知道该做啥，你分配个任务让我做吧」。千万不要找我要任务，因为我对「这是老师让我写的，那是家长逼我学的」深恶痛绝。为什么？因为这些都是“借口”，是为失败找的借口，是为敷衍了事、推卸责任找的借口。扯淡的是，从幼儿园到大学，我们的教育一直在为你提供这些借口。换言之，我们的教育一直在培养“敷衍了事、推卸责任、毫无担当”的人。只有“尊重个人权利、尊重个人自由”的教育才能培养出“负责任、有胆当”的人。显然，全世界没有哪个国家100%实现了这种理想化教育。但我们也看到，欧美国家的孩子动手能力更强，更有团队精神。为什么？因为这些孩子从小就被鼓励去做他们自己喜欢做的事情。既然是自己的事情，自然会认真去做。「我要做什么，我该怎样做」这样的问题对他们来讲，是家常便饭。久而久之，就养成了认真、负责的做事习惯。

我一个人没本事改变教育现状，但至少在我的课堂上，我们可以尝试一下。现在我给你机会，自由自在地去做自己想做的事情。我希望你珍惜它。

3 [2020-02-21 Fri] 幻灯片第7页，什么是操作系统?

我们还是按着幻灯片的顺序来上课吧。其实，大家心里都明白，上课和学习是两码事，就好像“做一天和尚”和“撞一天钟”是两码事一样。来课堂不等于就学习，学习也未必非要上课。毕竟，在课堂上我能告诉你的充其量就是「你该学些什么」，而真正的学习肯定是课下进行的。“修行”不是“撞钟”能取代的。

好了，我们现在步入正题。前面说了，正题（也就是操作系统知识）并不重要，如果你真要学习的话，千万不要绕过前面五个困难，直奔第六个，也就是最不重要的操作系统。你应该静下心来，克服幻灯片里的每一个生词，认真理解这张幻灯片要表达的意思，这需要你去查阅参考书里的相关章节。然后，在你认为「我终于搞明白了」的时候，把书合上，把幻灯片关掉，然

后用自己的话把幻灯片里的内容复述出来。没错，这才叫“学习”。

好了，假装你还没被吓跑，我开始讲第7张幻灯片，What's an OS？第一句说，OS就是当你网购一个OS的时候，人家寄给你的东西，那肯定就是OS。没错，这话很正确啊，虽然只是个玩笑，但并非毫无意义。起码这句话让我意识到，要和流氓正经讲道理该有多困难。人家只要回你一句这么“正确”的话，就能噎得你想撞墙。人嘴两张皮，人嘴是多么邪恶的两张皮啊！所以说，道理永远是讲给“讲道理的人”听的。对于不讲道理的人呢？用他听得懂的语言去教训他！说白了，给他不及格呗。

好了，再看第二句，OS是从一开机就开始跑，直到关机（或者死机，如果你用Windows的话）才会结束的程序。这句话算是很讲道理了。这的确是OS的重要特征之一，可以算是操作系统定义的一部分了。

第三句，它是资源的管理者。那么，什么是资源？电脑诞生之前，资源这个词就存在了。水、土地、煤炭、石油、空气、人、动物……貌似没有什么东西不能被当作资源。没错，就连“垃圾”也可以被认为是“放错了地方的资源”。但通常，当我们谈到资源的时候，“空气”和“垃圾”都不太容易被想到，为什么？因为它们不“紧缺”。当我们谈资源的时候，通常是在谈那些“大家都想抢的东西”。

回到电脑里面来，也一样，everything is a resource，但大家（众多进程）无时无刻不想抢的东西就两样，一是time，时间；二是space，空间。和时间相关的资源就是CPU，和空间相关的就是内存。好了，现在打开终端，跟我学一个小命令：

```
ps aux | wc -l
```

用这条命令可以数出你的电脑里正运行着多少个进程。我的是191个。而我的CPU是8核，也就是说，191个进程要抢着用8个CPU，没人管肯定是不行的吧。任何一个程序要运行的话，先要把它加载到内存中去，而我的内存只有8G，如果不够用怎么办呢？操作系统最重要的工作就是负责CPU和内存的分配与管理，它是电脑里的resource manager。

第四句，它是个控制程序。假如那191个进程里，谁和谁发生了不愉快，比如一个流氓进程非要往我的地址空间里写东西，那么，操作系统肯定要出手干预，或者在流氓得手之后，帮它洗地。每当此时，你心爱的Windows就会呈现出著名的Blue screen of death。Unix没这么夸张，它通常会在你启动

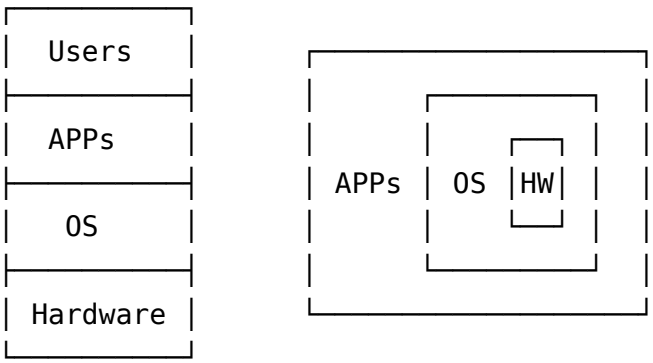
流氓程序的瞬间，就告诉你“Segmentation fault”，也就是所谓“段错误”，这通常是访问非法内存地址造成的。

最后一句，关于操作系统，并不存在一个放之四海而皆准的定义。为什么呢？因为有个问题，操作系统除了要一直转着不停，除了要管理资源，除了要洗地之外，还应该有那些功能呢？这就见仁见智了。举例而言吧，苹果和微软都把图形界面放到了操作系统内核里，因为这样“打开窗户的速度”更快；而其它的Unix，还有Linux，内核里都不包含图形界面，因为并不是所有人都需要图形界面，比如服务器就不需要图形界面。庞大的图形界面会给内核带来更多的bug，降低系统的稳定性、安全性、和效率。因此，专业的服务器都不太会选用Windows或者苹果系统，毕竟Windows和苹果的设计初衷，就是面向个人电脑用户。服务器的话，基本上都是Linux和Unix的天下（除了苹果，虽然它也算是Unix）。Linux、Windows、苹果，这些都是通用型操作系统，这世界上还有很多专用型系统呢，比如用于流水线控制的单板机。不同的系统，面向不同的工作场景，有不同的设计需求，所以，操作系统该如何定义呢？还是见仁见智吧。

好了，现在关掉幻灯片，把你的理解复述出来吧，最好用英文。

4 [2020-02-22 Sat] 幻灯片第8页，OS的功能模块

第8页上这张图，简化一下，就是下面这副样子：



左、右两张图是一回事，表达的是同一个意思，我们，生物意义上的人，从来不直接使用操作系统，我们只使用应用程序，应用程序才会去和操作系统打交道。任何应用程序如果想使用硬件（比如键盘、鼠标、显示器）的话，

都要向操作系统发出请求，然后操作系统帮你把键盘输入的字符显示到屏幕上。

「干吗这么费事？没有操作系统不行吗?」，其实前面我们已经回答过这个问题了，电脑里的资源（比如键盘、鼠标、显示器）很紧缺，若干进程都要抢着用，所以必然需要有人来维持一下秩序。换言之，如果你的系统里只跑一个程序的话，也就是所谓“单任务系统”，那么操作系统的确就显得多余了。

把上图中的OS放到显微镜下，看到的就是第8页的幻灯片。片中上下两条虚线之间就是我们最关心的部分，操作系统。它是个软件，可以很庞大而复杂，也可以小巧而简单，因设计需求的不同而变化。通常，讲课的时候，都选庞大而复杂的来说，而具体编程实现的时候，都是怎么简单怎么来。为什么？因为说起来容易，做起来难呗。

无论如何，一个通常意义的操作系统，它里面会有进程控制、内存管理、文件系统、输入输出等功能模块。教科书上，一般也是着重讲这几个模块。我们16周的课，通常只能讲完前三个，输入输出就靠自学了。本来嘛，上课也就能告诉你“该学什么”，不是吗?

现在来看看，APPs是如何向OS请求服务的? APPs和OS都是软件，软件之间怎么相互通信，或者说传递数据啊? 函数调用呗。所以说，APPs只要调用OS提供的函数，就可以把信息发送给OS了。每个操作系统都为用户进程提供了一整套函数，或者说一个“函数库”，这个函数库就是图中的system call interface。Linux的库比较小，里面有大约400个syscall。Meanwhile，Windows的库里有4000多个。是大点好，还是小点好呢？其实很好回答，就问问你自己，「做为一个程序员，我是愿意看一本400页的手册呢，还是看4000页的?」。分手吧，她真的太胖了！而且，不止是手册的厚薄问题，软件的代码量越大，bug数量必然就越多。这可是操作系统啊，它蓝屏，我一点都不觉得意外。

再注意一下，图中的syscall interface有两条路通往用户程序。一条路是直接的；另一条是间接的，要通过libraries（函数库）。其实，这个函数库，差不多就是专指C函数库。它不在OS里面，它在用户层（user level），是一个普通用户就能随意安装、卸载、替换的软件包。既然有一条直接的路径，为什么还需要它？有两个主要原因：

1. 跨平台。前面说了，不同的OS提供了不同的syscall函数库。那么，如果你在windows平台写了个程序，里面自然要用到Windows提供的syscall，比如说`CreateProcess()`，用来产生一个子进程。写好了之后，你编

译、运行，一切良好。于是，你把它拿到Linux平台，直接运行肯定是不行的。换了平台，要先编译。「完蛋了！在Windows上一切都好好的，为什么到Linux上就编译通不过？Linux太难用了！」没错，Linux对你不友好，也只是对“你”不友好。为什么？因为（此处略去500脏字）。

想想看，Windows提供了4000多个syscall，其中包括`CreateProcess()`；而Linux只提供了400个，你保证它也有`CreateProcess()`吗？Too simple, sometimes naive！在Linux平台，想要产生子进程的话，你要调用`fork()`。于是，累了，你不得不把程序中用到的成百上千个Windows syscall都替换成相应的Linux syscall！（我相信你的脏字也会很多的）结论，由于你直接调用了OS提供的syscall，导致你的软件可移植性极差，根本不跨平台。

如果你不直接调用syscall，而是“走弯路”，调用Library里的函数，生活就美好多了。比如说，我们最常用的Library是POSIX提供的LIBC，它既有Windows版，也有Linux版。于是，你只要在Windows和Linux上都装好POSIX LIBC，编程的时候调用里面的函数，让它去帮你调用底层的syscall，就没问题了。

2. 方便。函数库存在的意义，不止是把syscall包装一下，以便于你的软件可以跨平台。它还提供了很多广受用户欢迎，而syscall没有提供的功能。比如说`printf()`吧，

```
printf("Hello, world!\n");
```

说简单了，它的功能就是“屏幕输出”。但直接调用syscall（不考虑跨平台的话）`write()`也可以实现屏幕输出。实际上，`printf()`最终就是通过调用`write()`来完成屏幕输出的。

```
write(1, "Hello, world!\n", 14);
```

这么绕弯的好处是什么呢？`printf()`提供了带格式的输出，而`write()`不行。为什么`write()`不提供格式支持呢？原因既浅显又重要，Linux的设计者认为，OS只应该提供“不得不提供的功能”，所有非必须的功能都应该由用户层软件提供。这也是Linux内核不提供GUI（图形界面）支持的原因。这也是它更适合用来做服务器的原因。

再来看看图中的这两条路，边上都有一个单词，trap，当名词用的时候被翻译成“陷阱”。但是，在计算机专业，它经常被用做动词，是“触发”的意思。一个陷阱挖好了，只要别踩它，什么事都没有。一旦你踩上去，就会触发你的噩梦……同样，用户程序以调用函数的方式触发操作系统的某个功能，所以这里用trap一词。

后面的课程里，我们还会经常接触system call。课上用到的例子，自己去尝试一下吧。

- https://cs6.swfu.edu.cn/~wx672/lecture_notes/os/src.tgz
- https://cs6.swfu.edu.cn/~wx672/lecture_notes/linux-app/src.tgz

5 [2020-02-22 Sat] 幻灯片第9页，选个OS

这张画在技术圈里还是挺著名的，而且有好几个版本。不论哪个版本，都是个笑话。能看懂的，就笑呗；看不懂的呢，“加油！不哭！”，画一个自己的版本，咱们再笑呗。

「我是学技术的，到底该用哪个系统啊?」。肯定不该用Windows。为什么?

1. 学外科的，总得解剖过尸体吧；学操作系统的，总得解剖过操作系统吧。Windows根本就不开源，你看不到它里面的东西，所以无法拿它来解剖、学习。如果你真的有本事把它解剖了，那么你就要收到法院的传票了，因为这犯法。

2. Windows是要花钱买的，不便宜，而且在它的版权声明里还罗列着种种限制，包括不能修改、不能送人、不能私自买卖……而Linux是自由软件，你可以自由获取、自由使用、自由学习、自由送人、自由买卖、自由修改……你一个穷学生，不该为爹妈省点钱吗？而且，这绝不只是钱的事情，「自由」才是最重要的。武汉的李文亮大夫用生命告诉我们「言论自由关系到每个人的生命安全」。西谚有云「Live free or die」，先贤译之为「不自由，毋宁死」。看看李大夫，我的翻译是「不自由，真要命」。

3. 「可是我身边的同学和老师，他们都用Windows啊」。没错，而且我劝过他们了，就像我劝你一样。他们会说「自由了，早晚不也得死」，「不觉

不自由，也就自由了」，「现在不挺舒服的，自由又不能当饭吃」，「干吗非得跟人不一样啊」……的确，各有各的活法，至少我也要尊重他们「选择不自由」的自由。而且，被这些人环绕着，你会感觉自己才是个怪物。又想起一句西谚「Birds born in a cage think flying is an illness」，笼子里长大的鸟会认为飞翔是一种病。你的翅膀还在吗？愿意做一个飞翔的“怪物”吗？还是收起翅膀，和他们一起做“正常人”呢？西洋人又说了，「Why fit in, when you were born to stand out?」，一个生来就该与众不同、卓尔不群的人，干吗又要去随大流呢？

4. 用Linux的同学，毕业后的前景都很好。这并不意外吧。首先，肯用Linux的同学都普遍比较好学；其次，用Windows的人太多，竞争自然就大。竞争大，老板就会把工资压得很低。而用Linux的人少，所以需求缺口巨大，工作自然要容易找，工资也相对较高。

好了，晓之以理、动之以情、诱之以利，我都做到了。「好吧，我试试，那我到底该装哪款Linux呢?」。很简单，你周围的人用哪个，你就用哪个。为什么？容易得到帮助呗。所以，最好是我用哪个，你就用哪个。跟着我的安装指导一步步走，应该不会很费事。

- https://cs6.swfu.edu.cn/~wx672/debian-install/install.html