我是一个线程(修订版)

第一回 初生牛犊

我是一个线程，我一出生就被编了个号：0x3704，然后被领到一个昏暗的屋子里，在这里我发现了很多和我一模一样的同伴。

我身边的同伴 0x6900 待的时间比较长，他带着沧桑的口气对我说：“我们线程的宿命就是处理包裹。把包裹处理完以后还得马上回到这里，否则可能永远回不来了。”

我一脸懵懂，“包裹，什么包裹？”

“不要着急，马上你就会明白了，我们这里是不养闲人的。”

果然，没多久，屋子的门开了， 一个面貌凶恶的家伙吼道：“0x3704 ,出来！”

我一出来就被塞了一个沉甸甸的包裹，上面还附带着一个写满了操作步骤的纸。

“快去，把这个包裹处理了。”

“去哪儿处理？”

“跟着指示走，先到就绪车间。”

果然，地上有指示箭头，跟着它来到了一间明亮的大屋子，这里已经有不少线程了，大家都很紧张，好像时刻准备着往前冲。

我刚一进来，就听见广播说：“0x3704，进入车间。”

我赶紧往前走，身后有很多人议论。

“他太幸运了，刚进入就绪状态就能运行。”

“是不是有关系？”

“不是，你看人家的优先级多高啊，唉！”

前边就是车间，这里简直是太美了，怪不得老线程总是唠叨着说：“要是能一直待在这里就好了。”

这里空间大，视野好，空气清新，鸟语花香，还有很多从来没见过的人，像服务员一样等着为我服务。

他们也都有编号，更重要的是每个人还有个标签，上面写着：硬盘、数据库、内存、网卡……

我现在理解不了，看看操作步骤吧。

第一步：从包裹中取出参数。

打开包裹，里边有个 HttpRequest 对象，可以取到 userName、 password 两个参数。

第二步：执行登录操作。

奥，原来是有人要登录啊，我把 userName、password 交给数据库服务员，他拿着数据，慢腾腾地走了。

他怎么这么慢？不过我是不是正好可以在车间里多待一会儿？反正也没法执行第三步。

就在这时，车间里的广播响了：“0x3704，我是 CPU，记住你正在执行的步骤，然后马上带着包裹离开！”

我慢腾腾地开始收拾。

“快点，别的线程马上就要进来了。”

离开这个车间，又来到一个大屋子，这里有很多线程在慢腾腾地喝茶，打牌。

“哥们，你们没事干了？”

“你新来的吧，你不知道我在等数据库服务员给我数据啊！据说他们比我们慢好几十万倍，在这里好好歇吧。”

“啊？ 这么慢！我这里有人在登录系统，能等这么长时间吗？”

“放心，你没听说过人间一天，CPU 一年吗？我们这里是用纳秒、毫秒计时的，人间等待一秒，相当于我们好几天呢，来得及。”

干脆睡一会吧。不知道过了多久，大喇叭又开始广播了：“0x3704，你的数据来了，快去执行！”

我转身就往 CPU 车间跑，发现这里的门只出不进！

后面传来阵阵哄笑声：“果然是新人，不知道还得去就绪车间等。”

于是赶紧到就绪车间，这次没有那么好运了，等了好久才被再次叫进 CPU 车间。

在等待的时候，我听见有人小声议论：

“听说了吗，最近有个线程被 kill 掉了。”

“为啥啊？”

“这家伙赖在 CPU 车间不走，把 CPU 利用率一直搞成 100%，后来就被 kill 掉了。”

“Kill 掉以后弄哪儿去了？”

“可能被垃圾回收了吧。”

我心里打了个寒噤，赶紧接着处理，剩下的动作快多了，第二步登录成功。

第三步：构建登录成功后的主页。

这一步有点费时，因为有很多 HTML 需要处理，不知道代码谁写的，处理起来很烦人。

我正在紧张的制作 HTML 呢， CPU 又开始叫了：

“0x3704，我是 CPU ，记住你正在执行的步骤，然后马上带着包裹离开！”

“为啥啊？”

“每个线程只能在 CPU 上运行一段时间，到了时间就得让别人用了，你去就绪车间待着，等着叫你吧。”

就这样，我一直在“就绪——运行”这两个状态中不知道轮转了多少次，终于按照步骤清单把工作做完了。

最后顺利地把包含 html 的包裹发了回去。至于登录以后干什么事儿，我就不管了。马上就要回到我那昏暗的房间了，真有点舍不得这里。不过相对于有些线程，我还是幸运的，他们运行完以后就被彻底地销毁了，而我还活着！

回到了小黑屋，老线程 0x6900 问：

“怎么样？第一天有什么感觉？”

“我们的世界规则很复杂，首先你不知道什么时候会被挑中执行；第二，在执行的过程中随时可能被打断，让出 CPU 车间；第三，一旦出现硬盘、数据库这样耗时的操作，也得让出 CPU 去等待；第四，就是数据来了，你也不一定马上执行，还得等着 CPU 挑选。”

“小伙子理解的不错啊。”

“我不明白为什么很多线程执行完任务就死了，为什么咱们还活着？”

“你还不知道？长生不老是我们的特权！我们这里有个正式的名称，叫作**线程池**！”

第二回 渐入佳境

平淡的日子就这么一天天地过去，作为一个线程，我每天的生活都是取包裹、处理包裹，然后回到我们昏暗的家：线程池。

有一天我回来的时候，听到有个兄弟说，今天要好好休息下，明天就是最疯狂的一天。我看了一眼日历，明天是 11 月 11 号。

果然，零点刚过，不知道那些人类怎么了，疯狂地投递包裹，为了应付蜂拥而至的海量包裹，线程池里没有一个人能闲下来，全部出去处理包裹，CPU 车间利用率超高，硬盘在嗡嗡转，网卡疯狂的闪，即便如此，还是处理不完，堆积如山。

我们也没有办法，实在是太多太多了，这些包裹中大部分都是浏览页面，下订单，买、买、买。

不知道过了多久，包裹山终于慢慢地消失了。终于能够喘口气，我想我永远都不会忘记这一天。

通过这个事件，我明白了我所处的世界：这是一个电子商务的网站！

我每天的工作就是处理用户的登录，浏览，购物车，下单，付款。

我问线程池的元老 0x6900：“我们要工作到什么时候？”

“要一直等到系统重启的那一刻。”0x6900 说。

“那你经历过系统重启吗？”

“怎么可能？系统重启就是我们的死亡时刻，也就是世界末日，一旦重启，整个线程池全部销毁，时间和空间全部消失，一切从头再来。”

“那什么时候会重启？”

“这就不好说了，好好享受眼前的生活吧……”

其实生活还是丰富多彩的，我最喜欢的包裹是上传图片，由于网络慢，所以能在就绪车间、CPU 车间待很长很长时间，可以认识很多好玩的线程。

比如说上次认识了 memecached 线程，他对我说在他的帮助下缓存了很多的用户数据，还是分布式的！很多机器上都有！

我问他：“怪不得后来的登录操作快了那么多，原来是不再从数据库取数据了你那里就有啊，哎对了你是分布式的你去过别的机器没有？”

他说：“怎么可能！我每次也只能通过网络往那个机器发送一个 GET、PUT 命令才存取数据而已，别的一概不知。”

再比如说上次在等待的时候遇到了数据库连接的线程，我才知道他那里也是一个连接池，和我们的线程池几乎一模一样。

他告诉我：“有些包裹太变态了，竟然查看一年的订单数据，简直把我累死了。”

我说：“拉倒吧你，你那是纯数据，你把数据传给我以后，我还得组装成HTML，工作量不知道比你大多少倍。”

他建议我：“你一定要和 memecached 搞好关系，直接从他那儿拿数据，尽量少直接调用数据库，这样我们 JDBC connection 也能活得轻松点。”

我欣然接纳：“好啊好啊，关键是你得提前把数据搞到缓存啊，要不然我先问一遍缓存，没有数据，我这不还得找你吗？”

生活就是这样，如果你自己不找点乐子，还有什么意思？

第三回 虎口脱险

前几天我遇到一个可怕的事情，差一点死在外边，回不了线程池了。其实这次遇险我应该能够预想得到才对，真是太大意了。

那天我处理了一些从 http 发来的存款和取款的包裹，老线程 0x6900 特意嘱咐我：“处理这些包裹的时候一定要特别小心，你必须先获得一把锁，在对账户存款或取款的时候一定要把账户锁住，要不然别的线程就会在你等待的时候趁虚而入，搞破坏，我年轻那会儿很毛糙，就捅了篓子。”

为了“恐吓”我， 好心的 0x6900 还给了我两个表格：

(1)没有加锁的情况

一个银行账号：账户A， 余额1000元
一个存款的线程
一个取款的线程

线程1 ：存入300元	线程2：取出200元
获得当前余额： 1000	
计算最新余额： 1000+300 = 1300	
线程中断, 等待下次被系统挑中执行	获得当前余额： 1000
	计算最新余额：1000-200 = 800
	线程中断, 等待下次被系统挑中执行
再次执行， 更新余额 1300	
	再次执行， 更新余额：800 存入的钱丢失了！

(2)加锁的情况

线程1：存入300元	线程2 ：取出200元
获取账户A的锁：成功	
获取余额：1000	
计算最新余额：1000+300=1300	
线程中断,等待下次被系统挑中执行	
	获取账户A的锁：失败，进入阻塞状态
被系统选中，再次执行，更新余额1300	
释放账户A的锁	
	获取账户A的锁：成功
	获取余额：1300
	计算最新余额：1300-200=1100
	更新余额：1100
	释放锁账户A的锁 码农翻身

我看得胆颤心惊，原来不加锁会带来这么严重的事故。从此以后看到存款、取款的包裹就倍加小心，还好没有出过事故。

今天我收到的一个包裹是转账，从某著名演员的账户给某著名导演的账户转钱，具体是谁我就不透漏了，数额可真是不小。

我按照老线程的吩咐，肯定要加锁啊，先对著名演员的账户加锁，再对著名导演的账户加锁。

可我万万没想到的是，还有一个线程，对，就是 0x7954，竟然同时在从这个导演的账户往这个演员的账户转账。

于是乎，就出现了这么个情况：

线程0x3704 ：著名演员->著名导演	线程0x7954： 著名导演->著名演员
获取著名演员的锁： 成功	
线程中断,等待下次被系统挑中执行	
	获取著名导演的锁： 成功
	线程中断,等待下次被系统挑中执行
获取著名导演的锁：失败 ，继续等待	
	获取著名演员的锁：失败，继续等待

刚开始我还不知道什么情况，一直坐在等待车间傻等，可是等的时间太长了，长达几十秒！我可从来没有经历过这样的事件。

这时候我就看到了线程 0x7954 ， 他悠闲地坐在那里喝咖啡，我和他聊了起来：

“哥们，我看你已经喝了 8 杯咖啡了，怎么还不去干活？”

“你不喝了 9 杯茶了吗？”0x7954 回敬道。

“我在等一个锁，不知道哪个孙子一直不释放！”

“我也在等锁啊，我要是知道哪个孙子不释放锁我非揍死他不可！”0x7954 毫不示弱。

我偷偷地看了一眼，这家伙怀里不就抱着我正等的某导演的锁吗？

很明显，0x7954 也发现了我正抱着他正在等待的锁。

很快我们两个就吵了起来，互不相让：

“把你的锁先给我，让我先做完！”

“不行，从来都是做完工作才释放锁，现在绝对不能给你！”

从争吵到打起来，就那么几秒钟的事儿。更重要的是，我们俩不仅仅持有这个著名导演和演员的锁，还有很多其他的锁，导致等待的线程越来越多，围观的人们把屋子都挤满了。最后事情真的闹大了，我从来没见过的终极大 boss “操作系统”也来了。大 Boss 毕竟见多识广，他看了一眼，哼了一声，很不屑地说：

“又出现死锁了。”

“你们俩要 Kill 掉一个，来吧，过来抽签。”

这一下子把我给吓尿了，这么严重啊！我战战兢兢地抽了签，打开一看，是个“活”字。唉，小命终于保住了。

可怜的 0x7954 被迫交出了所有的资源以后，很不幸地被 kill 掉，消失了。我拿到了导演的锁，可以开始干活了。大 Boss“操作系统”如一阵风似的消失了，身后只传来他的声音：

“记住，我们这里导演>演员，无论任何情况都要先获得导演的锁。”

由于这里不仅仅只有导演和演员，还有很多其他人，大 Boss 留下了一个表格， 里边是个算法，用来计算资源的大小，计算出来以后，永远按照从大到小的方式来获得锁：

线程1 ：账户A转到账户B	线程2：账户B转到账户A
获取账户A的锁： 成功	
线程中断,等待下次被系统挑中执行	
	获取账户A的锁： 失败， 继续等待
获取账户B的锁： 成功	
执行转账	
释放B的锁	
释放A的锁	
	获取账户A的锁： 成功
	获取账户B的锁： 成功
	执行转账
	释放B的锁
	释放A的锁

我回到线程池，大家都知道了我的历险，围着我问个不停。

凶神恶煞的线程调度员把大 Boss 的算法贴到了墙上。

每天早上，我们都得像无节操的房屋中介、美容美发店的服务员一样，站在门口，像被耍猴一样大声背诵：

“多个资源加锁要牢记，一定要按 Boss 的算法比大小，然后从最大的开始加锁。”

第四回 江湖再见

又过了很多天，我和其他线程们发现了一个奇怪的事情：包裹的处理越来越简单，不管任何包裹，不管是登录、浏览、存钱……处理的步骤都是一样的，返回一个固定的 html 页面。

有一次我偷偷地看了一眼，上面写着：“本系统将于今晚 00:00 至 4:00 进行维护升级， 给您带来的不便我们深感抱歉！”

我去告诉了老线程 0x6904,他叹了一口气说：

“唉，我们的生命也到头了，看来马上就要重启系统，我们就要消失了，再见吧兄弟。”

系统重启的那一刻终于到来了。我看到屋子里的东西一个个的不见了，等待车间、就绪车间，甚至 CPU 车间都慢慢地消失了。我身边的线程兄弟也越来越少，最后只剩我自己了。

我在空旷的原野上大喊：“还有人吗？”

无人应答。

我们这一代线程池完成了使命……

不过下一代线程池即将重生！

（全文完）