记一次灵异般的 Bug 调试经历 - 文章

【伯乐在线导读】：说到程序员的噩梦，除了《程序员的 13 种噩梦，你遇到过哪些？》这篇提到的「无法重现的 Bug」，还有「遇到一个不懂技术又是掌控狂的项目经理」或「频繁变更需求」。自称有 35 年编程经历的 Mick Stute 对最大的噩梦有不同的体验。来看看他在 Quora 上拿到 16k 多顶的经历：

我曾经受雇于一位心理学家，去修复一个“输出有些奇怪的”软件，那个软件是他之前带过的一个研究生编写的。这款软件会读取一个数据文件，询问用户 50 个问题，进行一些计算并且根据该博士的研究给出一个分数。

这个程序运行在一台 3B2 电脑上。他向我示范了这个软件并且他确信在问题间切换时，屏幕上有奇怪的文字闪过，而且文字看起来并不友好。我同意去修复它，问题看上去比较直接。所以我可能得花几个小时去判断一下问题的严重性，这段时间的工作需要按小时进行支付，随后我们会就最终的报酬达成协议。

第1天

我坐在这台 3B2 电脑前，登录了那个研究生的账户，软件的代码都存在这里。我仔细检查这些 C 语言代码，这些代码故意写的让人看不懂。所有代码都挤在一行。软件共有 15 个文件，每个文件里面有三个函数——全在一行！所有的变量名都是三个看上去随机的字母。我和雇主谈了谈，决定花点时间处理一下（明智的决定）。我梳理了全部的代码并且做了排版优化，这让我读代码时容易些。

代码的梳理工作如期完成。软件使用了 curses 库，在屏幕上打印问题和答案并等待用户响应。但是它会首先移动到第一行的问题，打印一条鼓吹“白人至上”的消息，等待 1/2 秒，然后用一条问题覆盖了这串字符。这个问题应该比较简单。这里只有5个位置可以用来输出，每个位置都会闪过一条潜意识信息（译注：潜意识信息是指：参杂在正常信息中，用于对目标进行暗示的消息）。每条信息都是硬编码的。没问题，删除打印令人不适信息的`mvprintw()`函数一切就正常了。应该会正常吧。我重新编译代码，认为自己已经搞定了。但是当我再次运行软件的时候，问题又发生了——一条潜意识信息。这一次是同样的问题，只不过信息的内容不同了。

我再次查看代码，不管你信不信，它又回到了最初的状态。15个文件，混在一起，3个字母的变量——完完全全的变回了最初的状态。我真想一枪打死我自己，因为我忘了备份一份整理后的代码。我又重新梳理了代码，这一次我把它们放在了三个文件中，起了不同的名字。我把整个目录拷贝了一份，然后对其设置了只读。我再次编译了代码，看上去没什么问题。运行程序，这下好了，15个文件的原版代码，包括我自己修改过的代码以及输出的潜意识信息，又都回来了。

好吧，在硬盘上某处有一份源码，原来的作者设置了一个程序，每当你编译的时候，就把那份代码拷贝过来。我进行了一次全盘扫描，包括 (/usr/include)。因为这是一个研究用的程序，因此我们有除了内

核之外的全部源码。这里有很多的头文件，扫描它们需要很长的时间，所以，这是第一天。

第2天

硬盘扫描没有任何有用的消息。字符串显然要不是被加密了，要么就是被藏在哪个库里面了。因为我没有检查全部的可执行对象，我决定在所有库中搜索这段文本。这将花费更多的时间，所以，第二天结束了。

第3天

还是没有结果。字符串被加密了。这意味着我必须要跟踪全部头文件，从一个 #include 到另一个 #include 直到找到它。这么做需要花很多时间。

我们已经通知了学校的计算机部门，我们认为有人获取了进入 Phelp 博士研究用电脑的 root 权限，这部电脑是科研楼里的一台共享机。可以理解，他们并不太相信我说的话。

我开始分析 #include 文件，但我并没有找到那段代码。所以我知道，它被编译进了一个库中。没问题。为什么不重新编译一下呢，反正我们有全部的源码。

第4到6天

最困难的部分是让校园里的那帮书呆子知道他们遇到问题啦！但是最终我们说服了他们，Mark 是 Unix 管理员（他之所以被雇用是因为娶了院长的女儿），正在学习如何解决这个问题。最后，他决定由我来处理这件事，因为他怎么也搞不明白如何去编译这些东西。第六天结束了，所有的标准库都被重新编译了。哇！

我拿出了我修改并清理过的代码，开始进行重新编译。看上去没什么问题。运行程序，我去！又悲剧了！15 个混乱的源码文件和潜意识信息又回来了。这一切发生的太突然，就像魔法一样。我非常非常仔细的研究了这个问题，但是我还是陷入了困境。源码里面也没有那些代码，我觉得我可能被打败了。

Phelps 博士对我投入的这些时间并不满意，他认为我可能应该从头开始重写这个软件。“当然”，我说。我盯着终端，像一个受惊的小狗，深深的陷入思考。“你说的没错。这样可能更快。“很好”，他说，“我们可以明天开始重写。”

第7天

让始作俑者去死吧！我才没有被这家伙打败！我就是要编译他的代码，要不就干脆不搞了！“你不需要再给我付钱了，Phelps博士，我只需要一些研究时间”。这是一场黑客间的战争！

第8到14天

这回我变聪明了，我想，那个人不知如何的修改了这个 curses 库。我把这些代码编译为汇编，当时我还不懂 3B2 的汇编代码，于是我开始学习。我读了 6 天手册，试图从这些代码中找到问题，但是似乎一切正常。

第15天

突然我意识到，问题可能出在编译器。TM就是编译器！每次当你编译源码并且运行程序的时候，就会把这些潜意识信息拷贝到代码中。我好像听过这样的事。

啊哈！我知道了！！！我们正好也有这个编译器的源码。我读了读源码想要找点线索。你瞧，我找到了。的确，编译器和链接器里有一段代码会做下面这些事：

> 1) 检查所有`fopen()`调用，在打开的文件中搜索Phelp博士的问卷；如果找到了，那么 2) 在编译那个调用fopen()打开问卷文件的程序时，把这15个文件覆盖到当前编译目录。 3) 然后使用这15个文件编译Phelps博士的程序，在链接时用原始-o参数(即原始目标文件名)拿来作自己的目标文件名。
> 编译器被修改了，它会把代码添加到 Phelps 博士的程序中。修改编译器的人，就是编写该程序的人。

几天后，AT&T 的技术人员来了，带着硬盘，里面装有正常的编译器和链接器源码，我们对其进行了重新编译。问题解决啦。编译器源码中有害的代码段都没有了，我们有了一个全新的、正常的编译器。

不过它还是有问题。因为这个编译器还被其他的一些代码污染了，但是那些代码我们无法获取。这些代码现在只存在于可执行的编译器中，在编译器编译之前，它会把这些改变重新加入到源码中。但是这一次，它没有修改/usr/src中的副本，只是把它拷贝到了一个隐藏目录下，修改编译器源码、进行自我编译并删除了这个隐藏目录。这个研究生篡改了编译器，并让这编译器在被重新编译前篡改自身。我们必须在解决问题前使用另一台 3B2 机器上的二进制版本的编译器。

我们同时还发现，如果 /sbin/login 被编译，就会创建一个后门程序，任何使用特定密码的人都可以登录为 root 用户。该计算机可以通过调制解调器和Tymnet进行访问。最终，这一切受到了计算机中心的关注。

天才啊！但却导致了严重的后果。

- 安卓 - 专注Android技术分享
- 前端 - JavaScript, HTML5, CSS
- Java - 专注Java技术分享
- Python - 专注Python技术分享