有着 1 万个全局变量的一大坨代码 - 文章 - 伯乐在线

2013 年 10 月，丰田公司匆匆了结了“意外突然加速”（以下简称”突然加速“）诉讼案。经过数小时的讨论，俄克拉荷马法庭陪审团得出结论：丰田汽车制造商”贸然不顾“（用户的安全），并裁定丰田公司赔偿原告 300 万美元。这还不包括对丰田公司可能做出的大额惩罚性赔偿。

陪审团是根据所掌握的证据给丰田公司下达“严重忽视安全责任”判决的。原告方的两位软件专家针对丰田公司的软件设计以及设计过程给出了证词，其内容令人惊讶。在审阅了 2005 版的丰田凯美瑞汽车的软件开发过程和源代码之后，两位软件专家得出一致结论：丰田公司的系统不但有缺陷，而且达到了危险的程度。因为故障保护机制里充斥着错误和不一致，这是导致事故的根源。

Bookout 和 Schwarz 起诉丰田案件起因于 2007 年 9 月的“丰田车突然加速导致严重车祸”事件。当时 Jean Bookout 和她的朋友 Barbara Schwarz（当时坐在副驾驶）正准备从俄克拉荷马 69 号洲际公路驶出，结果她驾驶的 2005 版本丰田凯美瑞的油门失去了控制。当时她踩刹车却没用，于是不得不拉起了手刹，结果车的右后轮留下了 150 英尺的刹车印，左后轮留下了 25 英尺的刹车印。然而丰田凯美瑞并没有停下来，而是加速冲下了匝道，穿过了底部的道路，并撞上了路堤。Schwarz 重伤不治身亡；Bookout在医院里躺了5个多月，才从严重的头部和背部伤势中恢复。

Beasley Allen 律师事务所的 Graham Esdale（原告的律师）是最早做出类似最终裁决判断的人。他的判断就是基于匝道的路面上那两条刹车印做出的。

“丰田公司无法解释清楚这是怎么回事，”Esdale说，“有刹车印就证明了她当时确实有刹车动作。”

尽管技术讨论已经（客观）决定了证词的内容，陪审团的态度还是非常谨慎细致。在陪审团意识到此案已经结案后，陪审员又询问Patricia Parrish 法官是否可以对庭审内容继续进行讨论。十二位陪审员、Parrish 法官和被告律师又对此案进行了讨论。被告表示，从他们讨论的内容看来，陪审团明显是要对丰田公司的所作所为和企图掩饰的行动开展进一步的惩罚。

尽管已经有了刹车痕迹作为证据，原告方依然聘请了两位软件专家：Phillip Koopman 和 Michael Barr 作为专家证人，让他们以程序员的独特视角，来洞察丰田公司汽车软件研发过程和源代码中的各种问题：例如 位翻转（bit flips），任务终止导致的故障保护机制失效，对爆栈和内存溢出的防护机制不足，单一的故障隔离区域，滥用全局变量（全局变量达上万个之多）等等。（调查结果显示，）丰田在软件开发过程以及软件产品上的缺陷，多得难以逐一列举。

- Michael Barr 是一位广受尊敬的嵌入式软件工程师专家。他花了超过 20 个月的时间审查丰田的源代码，审查地点设在一个酒店套间大小的房间内，房间一共有五个隔间，Barr 就在这五个隔间中的一个里进行审查工作。审查过程由保安全程监视，参与者工作时不能穿皮带或者戴手表，而且连一张纸都无法带进带出。Barr 对丰田源代码给出了法庭证词，这些证词都基于他所做的长达 800 多页的测试报告。
- Pillip Koopman 是卡耐基梅隆大学计算机工程系的教授，他是嵌入式系统安全领域的专家，著有《Better Embedded System Software》一书。他的工作也包括为私人企业做软件设计审查，其客户中就有很多是汽车制造企业。他也对丰田的工程安全过程给出了证词。

这两位都使用了程序员们惯用的讽刺词汇——「一大坨代码」（译注：原文是 Spaghetti code，即通心粉代码，形容代码结构像通心粉一样绕成一坨，互相纠缠，根本就理不清楚，这是很明显的讽刺用语。），暗指丰田的代码无论是在写法上还是结构上都是一团乱麻。

Barr 的证词：

> （丰田的源代码中）有大量的函数，并且都过于复杂。以标准工业的尺度来衡量，这些方法中的一部分根本无法测试，也就是说他们的代码构成过于复杂，因此根本找不到一个可靠的测试工具或者方法来测试代码可能产生的所有情况。
>
> 其中有一部分代码甚至复杂到了无法维护的地步，具体来说就是如果你尝试对这类型的代码进行除bug或者做任何的修改，那么很可能就会有新的错误产生。你的汽车下载了最新版本的固件——也就是我们所说的嵌入式软件——并不意味着比以前更安全……并且能够得到结论故障保护机制不足。他们（丰田）的故障保护机制有瑕疵和漏洞。
>
> 总的来说，他们（丰田）的安全架构就像纸牌屋一样摇摇欲坠。因此故障保护机制失效同时油门控制失灵事件发生的概率是很高的。

Barr 还从资料中了解到，2007年10月，甚至一位来自丰田的程序员都将引擎控制应用程序代码描述为“像一坨”代码。他把这一点引入了他的证词中。

Koopman 则重点关注丰田的计算机工程过程管理。业界的首个工业编码标准是汽车工业软件可信度协会（MISRA）于 1995 年设立的，这个标准虽然只是厂家出于自身目的设立，却成为了业界普遍认可的行业标准。这个标准把一系列规则当成衡量的标尺，并将破坏规则的程度等同于破坏规则的数量：每违反 30 条规则，你可以认为会有 3 个小的 bug 和 1 个大的 bug 出现。而丰田，按照工业标准的尺度，算是犯了很严重的错误，Koopman 如是说。

2010年，因为与美国高速公路安全管理局（NHTSA）有合同关系，NASA（美国国家航空航天局）的软件工程师对丰田的部分源代码进行了审查，结果他们针对 MSIRA-C 的第 35 条规则，在可审查的源代码中找

到了 7134 处违反点。Barr 以 2004 年版 MISRA 标准为依据，在丰田的源代码中发现了 81514 处违反点。

丰田有自己的软件过程标准，并且和工业标准多少有一些重叠。即使如此，丰田的程序员还是屡屡违反自己制定的标准。他们没能有效地跟踪他们偏离标准的程度，并且会为这种偏离行为进行辩解，认为这样做是符合实际标准的。Koopman 在证词中提到：如果不是一开始就把软件安全纳入到产品的研发中去，那么再往后即使想加也加不进去了。

“在从事安全相关的软件工作时，你必须要学会万分小心谨慎。糊弄是不行的。丰田确实关心过一些安全的事情，但是他们没能达到设计安全相关软件所能接受的水平。”他说。

丰田所违反的最大的一条安全标准是：在系统中允许单点失灵（single point failures）机制存在。（单点失灵是指将整个系统安全与否的控制权赋予单个软件或硬件，就类似于单引擎飞机一样）Koopman 在证词中提到。

“如果是采用单点失灵的机制，在我见过的任何一种安全标准里面，都是属于不安全定义范围的，并且没有什么反制措施，没有多少故障保护机制能解决这个问题。所有的方法只能降低失灵发生的概率，但是不可能完全根除问题。我们有数以百万计的车在路上跑，（对于这么庞大的数字）出问题的可能性真的是千奇百怪，而且问题真的是会发生的。”

另一个非常糟糕的背离标准的行为是：在系统中大量使用全局变量。（变量代表内存中的一个位置，这个位置存储了一个数。全局变量则意味着系统中任一地点的任一软件都可以对其进行读写。）理想主义的全局变量个数应该是 0。丰田则（在代码中）使用了超过10000 个全局变量。

“在工程实践中，总共采用 5 个或 10 个全局变量，这都是 OK 的。10000 个就不行，那我们就完蛋了。这是不安全的，我可不想一次性查看 10000 个全局变量以后才知道哪里出了问题。”Koopman在作证时这么说。

Barr 和 Koopman 在软件设计中发现的其他错误还有：缺乏平级代码审查（peer code review），还有就是丰田使用了电装（Denso）公司的副 CPU，但是却没能对其源代码进行检查——而丰田公司董事会则信誓旦旦地告诉美国国会和 NHTSA，说这次“突然加速”事故不可能是由引擎的软件引起。

Barr 作证说，很多机动车的行为故障都是由 CPU 中的任务失灵（death of task）造成的，所以可以推断 Bookout 的“突然加速”事故也很可能是由 CPU 中某个不知名的任务突然失灵造成的，在庭审中这个任务被称为“X任务”。Barr 还给这个任务取了个形象的名字，叫“厨房多功能水槽”任务。因为这个任务同时控制着汽车的多项功能，包括油门控制、巡航控制、转向控制、定速和熄火控制等等——其中很多功能的故障防护功能逻辑都运行在主 CPU 上。

Barr 对丰田的“看门狗”监测程序的设计提出了批评，“看门狗”程序是专门用来探测任务失灵的软件。他在证词中说，丰田的“看门狗”检测程序“对主要任务的失灵现象向来就无法探测。这个程序的唯一目的就是探测任务失灵，但是它却无法做到。（因为）它就没按照正确的目的来设计。“

“相反，丰田把监测程序设计来监测CPU是否过载，并且”，Barr作证到：“他们（丰田）其实连监测CPU过载也没做对。CPU过载是指在瞬间有大量任务迸发，并要在一段时间内把这些任务都处理掉。如果这种场景持续的时间过长，那么也会将汽车置于险境，因为任务太多造成很多任务根本就没有机会在CPU

上执行，这和临时任务失灵的效果是一样的。”

Barr 还作证说，丰田的软件还抛弃了操作系统产生的错误代码，直接对任务产生的错误码置之不理。Barr 在庭审上说：

> 对于任务失灵的问题，尽管我的关注点是在 X 任务上，因为这个任务控制了油门，负责执行故障保护，它真的很重要，但是当任务 X 和其他的任务以不同的组合执行时，总会导致任务失灵的现象。比如任务 3 和任务 X 一起执行，或者任务 3 和任务 7 以及任务 X 一起执行，或者只有任务9 执行，（都可能产生失灵现象）。这样就让汽车的失常表现的不可预测性大大增加。而“突然加速”只是众多失常表现中最危险的一种而已。

你很可能想否定两位软件专家的结论，因为他们不过是原告方聘请的技术专家，自然要为原告的利益说话，但 Koopman 和 Barr 关于软件错误的评估，以及“突然加速”可能原因的解释却揭示了更多事实：丰田的系统如何会出问题后不留下任何系统痕迹；为什么在丰田其他车型中也出现过“突然加速”的问题，为什么丰田无法通过“地板垫和刹车踏板召回”来解决问题，以及丰田长期以来如何通过隐藏某些“突然刹车”根本原因来逃脱责任。

根据两位软件专家的描述，丰田软件系统的复杂程度令人难以置信，这也解释了为什么NHTSA会做出那样的反应，以及为什么NASA没有能找到丰田汽车存在“引擎马力全开，忽略用户刹车指令，以及没有错误代码”等严重瑕疵的相关证据。比如，Barr作证说，NASA的工程师时间有限，并且没有查看所有代码的权限。他们要完全依赖丰田公司给他们进行汇报——在某些情况下，丰田成功地误导了NASA。例如，NASA就错误地相信了丰田已经为汽车设计了针对”位反转”的硬件保护机制 EDAC（错误侦测和纠正编码）。2005版的丰田凯美瑞其实根本就没有 EDAC，Barr在证词中说，但是丰田的邮件却告诉NASA是有的。在庭审的时候他说：

> NASA 根本就不知道2005版凯美瑞没有 EDAC（保护机制）。2005版丰田凯美瑞根本就没有EDAC。所以当位反转发生的时候，肯定就不会有任何硬件机制去侦测。那么当位反转在特定情况下发生时，系统也就无法反应这一故障，那么软件防护措施自然也就无从对系统进行保护。所以可以得出结论，（在现实中）特定情况下，位反转故障就是有可能发生的。

软件专家们的证词解释了，为什么 NHTSA 不太可能查出丰田被软件问题掩盖的电气故障。在对丰田汽车的大多数调查过程中，NHTSA 下属的故障调查办公室团根本就没有软件工程师参与。他们也没有真正的专家，能够对现代汽车的关键安全问题的复杂度有足够的了解。NHTSA 下属的这些故障调查办公室的工程师就像远古人一样工作，工具原始，效率低下。然后人们就见识到了这个政府机构的固执程度翻了两倍，翻了三倍，翻了四倍，一直像一个老妇人喋喋不休，把“突然加速”的问题归咎于汽车的脚垫。

不过即使是 NHTSA 配备了专家，由于丰田的软件实在是过于复杂，因此故障调查办公室仍然不可能有足够的时间或者预算来评估丰田的源代码。这也就是为什么我们不停地强调 NHTSA 需要编写功能安全规范的原因——不管是出于他们自身考虑，还是国会强制要求，他们都应该写。

我们在网上贴出了 Koopman 的初步调查结果草案（部分1 和 部分2），以及 Barr 的 庭审证词 和 Barr 编写的幻灯片材料 ，这些材料很长，但是绝对值得一读，因为它不但能满足你对此次事件的所有兴趣，还能让你了解怎么才能把一个汽车行业的嵌入式系统搞砸，以及 NHTSA 的调查为什么会误入歧途，还有丰田的电气架构上的软件是如何脆弱得令人难以置信。

通常情况下，人们会把“公司保守商业机密”和“公司保护有高价值资产”这两件事顺理成章地联系起来。而在这个案件里，我们认为，所谓“有价值的资产”正是技术本身，也就是丰田公司在生产制造汽车产品时所采用的“秘密配方”。我们不能把汽车制造业的“保护资产秘密”和“可口可乐保护其配方”等同起来（译注：据说传统可口可乐汽水有99%的成分都是公开的，无非就是水，糖分，咖啡因等。但是可口可乐有1%的“秘密成分”是不公开的，其配方是可口可乐口味独特的决定性因素，也是公司的“最高机密”。据传这1%成分的配方在整个可口可乐集团只有不超过10人可以有权接触，估值上亿美元。），Koopman和Barr在法庭证言中指出，丰田公司想要隐藏的，其实是一种会制造灾难的“配方”。根据2007年9月在丰田公司内部员工之间的电子邮件的内容：

“’说实话，研发故障保护机制其实从来就不是丰田公司工程部门的风格，‘”，Barr在法庭上念出了邮件的原文，“接下来邮件又写道，‘不过如果（这种风格）被解读为：丰田公司在工程控制领域实力强劲，那么也不失为一件好事。’（译注：邮件的意思是，丰田公司工程部门实力很牛，产品质量很高，所以并不需要在软件方面再搞一个故障保护机制）。”，而后，Barr又专门标出了另一段邮件中的文字：“长此以往可不是什么好事情。”

网友评价：

@天天天降大大大圣：一万个全局变量。。。告诉我这种代码怎么做重构？怎么维护？

@小灰笔记_Grey：回复@douzigui:没有挑战，比如德国博世，他们的变量比丰田还多。通常，首先拆分成一百多个模块，命名方式再用模块名+物理衡量因子+物理含义+存储属性（可选）的方式，命名简单而自然。

@ysjynkpgmw：汽车代码的标定量必须是全局变量，所以一万个全局变量很正常，标定区给到512K的都有。行业的特殊情况。日本人这样，德国人美国人也这样。

@小灰笔记_Grey：博世的似乎是25000个，还不是欧六。汽车电子就是这样不是？否则，靠什么机制进行耦合模块间的信息交互呢？难不成写25000个函数？

@haitao深圳：一万个全局变量？如果都是需求必需的：必须全局随时使用。。。那只能尽量分层分类存储了，或者设计分级的访问接口

@重装旗舰：嵌入式开发不都是这样,大惊小怪，你以为一个8位的单片机，ram不足2m能给你多大空间写类来封装

@肖寒_THU：全局变量很多时候可以起到加速的作用。为了极致的速度可能会这么做。

@hentai悠：不要把全局变量搞得跟不能用似的。。。多少新人被吓得一个都不敢用！尽量按文件划分好模块使用static全局变量就比较明显了啊！

@市场街的私奔诺莎：嵌入式系统这样写很正常吧，没有MMU没有堆没有线程没有系统，甚至c编译器也不一定有成熟可靠的

@小灰笔记_Grey：回复@__ill__:汽车电子有一个很重要的特点，即要求实时性又要求安全性。一般只能选C这样的语言，速度必须考虑。同时，内存的动态分配一般是不要不要滴！//@__ill__: 用JS写的吗？

@逻辑引擎：由于行业壁垒，许多非信息技术行业的软件开发还处于石器时代，却还要把那些乱麻般难维护难重用高度平台依赖的垃圾代码当个宝，唯恐被竞争对手偷去了。事实上这些垃圾的重用成本太特么高了，除非直接把完整的产品模块的软硬件设计全搬过来，否则真没什么人稀罕那些破代码。

@逻辑引擎:Spaghetti code应译成乱麻般的代码，还拥有上万全局变量。丰田尚且如此，我现在严重怀疑整个汽车行业嵌入式控制代码的安全性。该行业代码量巨大，但跟汽车的机电结构不同，对外根本不可见。行车安全严重依赖这些对外不可见的代码，我们感觉的安全完全是不明觉厉的朦胧安全

@庞拟：代码写烂点会死人啊？！会！

@sumtec：你们不要笑丰田了，真的，有些汽车厂他们的汽车零部件标牌数据什么的还是一个CSV或者Excel文档。还有些车企把自己的VIN号码印制的事情外包以至于所有车型全部都在一个VIN车型下面无法区分。

鹏程 105

很早以前有个说法，用全局变量，函数不传参数就不会把变量值压入堆栈读取也不用出栈，高速频繁的运算会有明显优势

2天前

黄一 104

日系貌似都是这种风格，所有东西都定死，coder只要填代码。

2天前

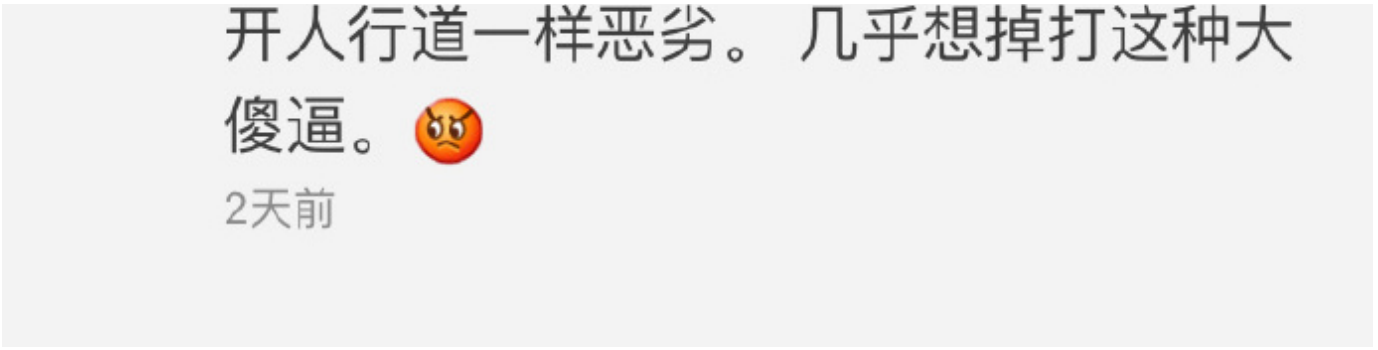
bowman han 103

全局变量指针一飞肯定崩溃

2天前

临峰不畏 101

最可恨的就是写出全局变量一大堆代码的程序员。 这跟图快闯红绿灯、图快捷

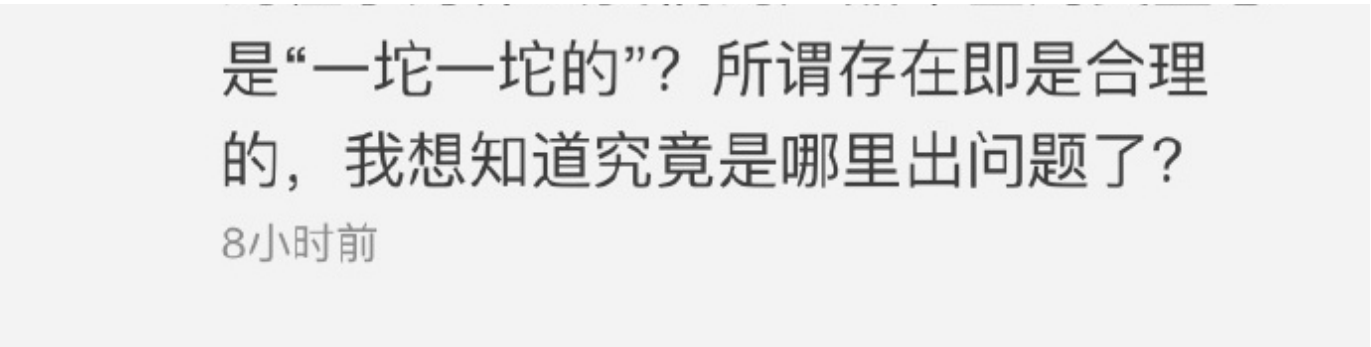

开人行道一样恶劣。几乎想掉打这种大
傻逼。
2天前


●●●●● 中国联通
19:14
78%
返回
Python开发者
李伟
98
欧洲文化追究真相。每次都会进步。
2天前
Orange
89
谁能讲讲从根本意义上全局变量是怎么出问题的?
2天前
Mandelbrot
83
全局变量极大地增加了程序模块之间的耦合性，某一个函数调用这个变量，尤其是修改这个变量的值之后，要想让其他函数调用这个变量时还保持正确的值，会让程序变得异常复杂，复杂就容易出乱子，这就是所谓的根本性原因
昨天
木不尤
70
全局变量污染命名空间，不利于维护。

这个文件几个全局变量，那个里面几个，我去你妹的，一坨

昨天


●●●●● 中国联通
19:14
78%
返回
Python开发者
我就是个坚果
48
记得以前有次在别人写过的程序上修改写程序，全局变量和局部变量名称重复了，运行的时候出现了莫名其妙的问题。还好代码少，容易发现问题。
昨天
孤灯残云
43
全局变量是导火索，栈崩溃才是炸药桶。
昨天
回家
23
多任务系统中全局变量的后果是不可预测的
10小时前
刘龙
19
大家都对全局变量嗤之以鼻，那我疑惑的在于为什么我们的产品中全局变量总



> 打赏支持译者翻译更多好文章，谢谢！