Linux 内核函数鲁棒性关联测试

王立荣,何 炜

(中国船舶重工集团公司江苏自动化研究所,江苏 连云港 222006)

摘 要:在简要介绍软件鲁棒性基准程序测试方法的基础上,以 Linux 操作系统内核函数为例,通过对用于分析测试结果的维度模型进行分析,提出了软件鲁棒性的关联测试方法,并给出了相应的测试实例及测试结果,为 Linux 操作系统内核函数鲁棒性测试提供了更为直观、有效的方法。

关键词:鲁棒性;软件测试;关联测试;维度失效

中图分类号:TP311 **文献标识码**:A **文章编号**:1674-7720(2010)17-0004-03

Relationship testing of the Linux kernel function robustness

WANG Li Rong, HE Wei

(Jiangsu Automation Research Institute of CSIC, Lianyungang 222006, China)

Abstract: The article briefly introduces the software in the robustness of the benchmark test method based on the Linux operating system kernel to function as an example, for analysis dimensional model of the test results were analyzed by the relationship of software robustness testing, and the corresponding test cases and test results for the Linux operating system kernel function provides a more robust testing of intuitive, efficient method.

Key words: robustness; software testing; relationship testing; dimensions failure

软件(或软件构件)鲁棒性是衡量软件在异常输入和应力环境条件下保持正常工作能力的一种度量。鲁棒性测试主要用于测试操作系统、应用程序、COTS 软件、构件及服务协议等软件和协议的可靠性及健壮性。在操作系统和安全关键软件等一些重要软件的测试上尤为重要。对于系统鲁棒性的评价一般有基于测量的方法和基于故障注入的方法,近年来提出了鲁棒性基准程序方法(Robustness Benchmarking)[1]。鲁棒性基准程序(Robustness Benchmark)由一组健壮性测试用例组成。

实施软件鲁棒性测试的目的是发现所测代码的健壮性薄弱环节,并予以消除或增强抵抗异常情况的能力。增强代码健壮性的过程包括:(1)确定软件的激发健壮性失效的异常值参数,并进行测试;(2)分析测试结果,找出失效原因;(3)写保护代码屏蔽导致失效的异常值;(4)把保护代码与软件模块相连接[3]。

1 Linux 内核函数测试

Linux 操作系统体系结构从底层到顶部的顺序依次是:内核(包含内核函数)、系统调用、内建程序(操作系统的命令)。内核函数是内核代码的组成部分,其调用程序直接运行在内核空间。内核函数一旦出现异常,将立刻对整个操作系统产生影响。系统调用一般对内核函数进行封装,以此作为内核与用户空间的接口。当用户程序使用系统调用时会转到内核空间,调用结束后又会返回用户空间。内核函数的测试结果一般分类为:函数错误码返回、异常、内核挂起、工作负载夭折、工作负载结果不正确、工作负载完成[1]。

Linux 内核函数鲁棒性测试的最终目的是要提高系统的健壮性,需要根据测试结果生成相应的保护代码。这方面的研究目前处于初期阶段。

2 鲁棒性维度分析

典型的鲁棒性测试包括模块化基准测试和层次化测试两种主要方法。模块化基准测试是对一个系统进行分离测试。它把一个独立的系统看作是一系列组件的集合,如文件系统、内存系统、外部交互系统、锁机制和多道程序运作等,另外还通过一个监视器程序来监视和收集测试的结果。而层次化测试是通过定义一个清晰的交互层,使测试和对各种模块进行测试的执行细节相分离。一些测试可以适用于所有模块,而另一些可能只适

合一个模块子集。使用层次性结构是分解系统的好方法。通过层次化来对操作系统进行测试可以收到较好的效果[2]。

无论采用模块化基准测试还是层次化方法，最终都是对操作系统接口函数采用参数的组合测试。对鲁棒性测试结果进行分析的一种方法是使用维度(Dimensionality)模型。维度有两种定义：(1)参数维度，它指的是模块中参数的个数，对于一个软件模块而言，参数维度被定义为其变量的个数；(2)鲁棒性失效维度，对于引起鲁棒性失效的一组特殊参数，那些确实引起失效的参数的个数被定义为鲁棒性失效维数。

2.1 维度失效

维度失效分一维失效和多维失效。多维失效的参数一定都是符合条件的系统能够识别的值。一维失效和多维失效所引发的原因不同，一维失效是参数非法，多维失效是参数组合非法(每个参数都是合法的)。当一维失效用例被保护和屏蔽后，会不会跳转为多维失效，关键是看其参数是否构成组合关系。从对其参数的组合关系的判定上，可以判断该失效用例是真维失效用例还是变维失效用例[3]。

基于低维度优先的维度模型中失效维度不可能超过参数维度。可能属于一维失效的某个失效，同时也可能属于高维失效。对于这种情况，一般把这个失效作为低维失效来对待。同时激发软件鲁棒性失效所需考虑的最少因素取决于鲁棒性失效维数，当参数维度为失效维度时，测试结果的观察最为直观；当参数维度大于失效维度，测试结果的观察就不太直观了。失效维度也可以通过观察鲁棒性测试的响应模式得到。

2.2 失效状态分析

维度失效状态分为三类。(1)真维失效指状态失效条件被屏蔽后，测试用例跳转到正常状态；(2)同维失效指状态失效条件被屏蔽后，失效维度保持不变；(3)变维失效指相同条件下产生失效维度升高。由于基于低维度优先原则，所以由高维度向低维度的失效跃迁跳变不可能发生[3]。鲁棒性测试用例的失效维度状态转变如图1所示。

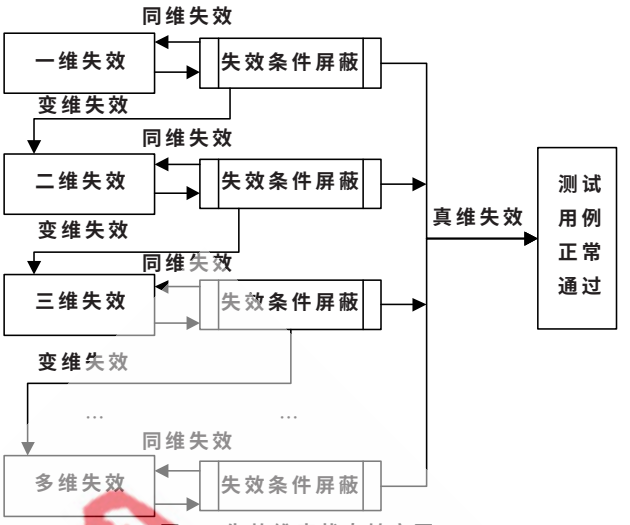

图1 失效维度状态转变图

现以Linux系统函数read(fd，buf，count)为实例进行分析，说明上述不同失效维度之间的转变问题。函数的三个参数取值如表1所示。

假设当参数fd取值errno file，buf取值Null时，均会产生一维失效。当fd取合法的值，并且buf分配空间小于count时产生一个二维失效。此时，对参数fd取值empty file进行保护屏蔽，则一些测试用例将会通过测试，如read(empty file，8，1)；而另一些用例则维持一维失效不变，如read(empty file，Null，1024)；还有一些用例将转化为多维(维度≥2)失效，如read(empty file，1，8)。

表1 read函数参数组合情况

int read(fd，buf，count)		
fd	buf	count
valid file-closed	0	0
valid file-read only	1	1
valid file-read write	8	8
empty file	Null	1024
errno file	Large buffer	Small buffer
…	…	…

3 鲁棒性关联测试

当参数维度等于失效维度时，很容易看出是哪些参数失效。而测试时维度的跳变，会给鲁棒性测试的分析带来困难，会影响测试覆盖率的问题，还牵扯到测试用例的增加[4]。在鲁棒性测试中可以利用参数的关联性进行测试。将传统的组合测试法分为两步：关联性测试和非关联性测试。鲁棒性关联测试的流程如图2所示。

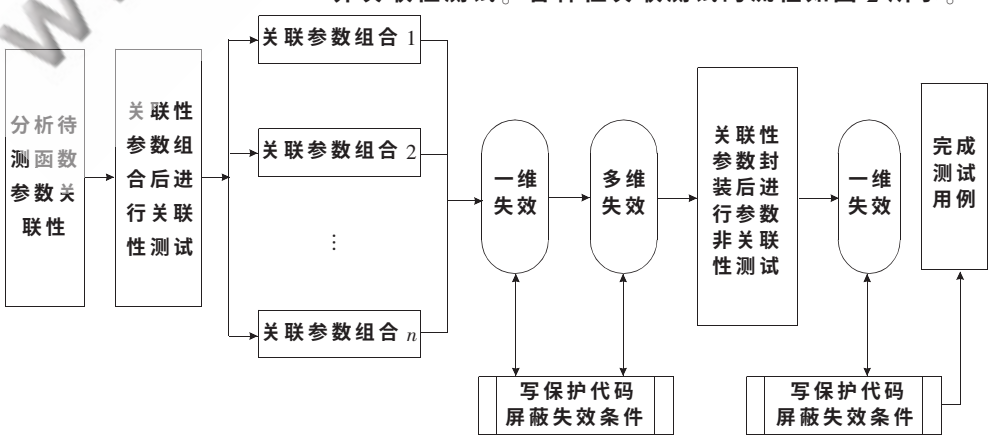

图2 鲁棒性关联测试流程图

在进行鲁棒性关联测试时，首先进行参数关联性测试，先把待测函数中有互相作用的参数进行包装，在测试中人为构造参数维度等于失效维度的情况。例如，函数$f(A,B,C,D,E)$中，参数A、B、C有关联。首先将参数E和参数F取合法输入值，然后测试参数A、B、C的所有组合。若有失效，必定是一维失效或者三维失效。由于参数维度等于失效维度，通过分析测试结果就可以写出保护代码。在对函数进行了充分的关联测试后，再进行参数非关联性测试。取出上例中参数A、B、C的一个合法组合，对参数E和参数F的所有用例分别进行测试。若有失效，必定是一维失效，这样也很容易分析测试结果和写出保护代码。

表2 函数read参数关联性测试表

buf	count				
	0	1	8	1024	Larger buffer
0	√	×	×	×	×
1	√	√	×	×	×
8	√	√	√	×	×
Null	×	×	×	×	×
Large buffer	√	√	√	√	×

表3 函数read参数非关联性测试表

参数fd	valid file-closed	valid file-read only	valid file-read write	empty file	errno file
测试结果	√	√	√	×	×

通过对函数的参数关联性进行测试可得出结论，只有当函数所有参数都发生关联作用时，鲁棒性关联测试所需用例的个数才会等于传统组合测试所需的用例个数[5]。所以，在覆盖率不变的情况下，若采用鲁棒性关联测试法，可以有效减少测试用例个数，并且还能够消除维度失效跳变带来的影响。

4 测试实例

实际测试中测试环境为DELL的DIMENSION 4700，操作系统为Redhat Linux 8.0，系统内核为2.2.24。实测以read()函数参数组合表为例，其表中组合测试用例的个数为5×5×5=125个。进行关联测试时先对其参数的关联性进行分析，通过分析可以得知它的三个参数中只有buf和count有关联。

实测中首先进行关联性测试，对read()函数的参数fd取正常值，测试参数buf和count的所有组合，测试结果如表2所示。共使用了25个测试用例。

在对上述测试结果进行屏蔽失效后，转入第二步，对参数fd进行非关联性测试，即针对fd与(buf+count)的组合进行测试。对buf与count的组合取合法值后，针对参数fd的所有取值分别测试，这时只会发生一维失效，测试用例个数是5个，其结果如表3所示。

由上述测试实例可见，传统组合测试法需要125个用例，而关联测试只需要30个用例，两者最终完成的函数测试覆盖率相同。由此可见，关联测试是对传统的组合测试的一种有效改进。传统的参数组合测试忽略了参数之间的关系，结果导致测试用例大量增加，覆盖率却有可能降低，同时还由于产生了维度跳转而给测试增加了困难[6]。采用关联测试可以避免上述问题的产生。使用关联测试时对参数之间关系进行分析，还有可能发现传统的组合测试没有测到的失效用例，这样关联测试的覆盖率相对于传统组合测试来说，只会提高而不会降低，这对于Linux内核函数的鲁棒性提升十分有效。

理论分析和实例应用的结果表明，在Linux内核函数的鲁棒性测试中采用关联测试来代替传统的组合测试，可以在保证测试覆盖率的同时，使所需的测试用例大大减少，而且函数中相关联的参数个数越少优势越明显。现实中Linux内核函数的参数之间关联性较少，因此在其鲁棒性测试中关联测试方法具有很好的实际应用价值。

参考文献

[1] 刘利枚，汪文勇，唐科.嵌入式软件测试方法与技术[J].计算机与现代化，2005(4):124-126.
[2] 赵则章，江建慧.操作系统健壮性测试方法研究[J].计算机工程与应用，2007，43(7):93-97.
[3] 刘洪涛，江建慧，赵则章.软件健壮性的包裹测试[J].计算机工程与科学，2005，27(4):19-21，24.
[4] 周章慧，王同洋，吴俊军，等.基于有限状态机的健壮性测试研究[J].计算机工程与科学，2009，31(5):93-97.
[5] 江建慧.嵌入式系统性能评估的基准程序方法[J].机械与电子，2002(4):43-48.
[6] 郑人杰.计算机软件测试技术[M].北京:清华大学出版社，1990.

(收稿日期:2010-05-15)

作者简介：
王立荣，女，1978年生，工程师，主要研究方向：软件测试。
何炜，男，1980年生，工程师，主要研究方向：嵌入式系统及软件。