文章编号: 1006-2475(2010)06-0166-04

基于 Linux 的 ACM 在线评测系统研究

杨志伟,曾艳姗

(仲恺农业工程学院计算科学系,广东 广州 510225)

摘要: ACM 竞赛是目前计算机水平最高的国际大学生程序设计竞赛,而 ACM 在线评测系统则是根据竞赛需求而提供的一个培训平台。由于 Linux 系统具有安全性高、稳定等优点,目前大部分服务器都是使用 Linux 系统。本文提出一个基于 Linux 的 ACM 在线评测系统,详述该系统中编译、测试、获取计算机消耗情况、返回测试结果这几种功能的具体实现。并分析该系统中可能出现恶意占用资源、恶意调用系统等不安全因素,提出相应的解决方案。该系统具有实用性强、安全性好等优点。

关键词: ACM 在线评测系统; Linux; 编译

中图分类号: TP311.52 **文献标识码:** A **doi:** 10.3969/j.issn.1006-2475.2010.06.047

Research on ACM Online Judge System Based on Linux

YANG Zhi-wei, ZENG Yan-shan

(Department of Computer Science, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China)

Abstract: ACM contest is regarded as the one with the largest scale and highest level for international collegiate programming contest in the computer world. To meet the requirement of this contest, ACM online judeg system provides a training platform. So far most servers adopt Linux as their operation system due to its advantages like high security, stabilization etc. This paper proposes a ACM online judeg system based on linux and specifies its concrete realization of functions, like compling, testing, acquiring the information of computer consumption and returning the testing results. What's more, the system can identify the hazards and threats leading to malicious resources occupation and malicious system call, then puts forward corresponding solutions. The advantages of this system include practicability, high security and so on.

Key words: ACM online judge system; Linux; compile

1 研究背景

ACM 国际大学生程序设计竞赛(ACM/ICPC: ACM International Collegiate Programming Contest)是由国际计算机界历史悠久、颇具权威性的组织 ACM 学会(Association for Computing Machinery,美国计算机协会)主办,是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛[3]。用来评测参赛选手程序正确性与高效性的网络平台称为 ACM 评测系统[4-6],评测系统对参赛队员的培训尤其重要。目前很多论文主要针对基于 Windows 系统的评测系统研究[7-9],而基于 Linux 系统的研究不多[0]。由于 Linux 系统具有安全性高、稳定等优点,目前大部分服务器都是使用 Linux 系统[1]。本文主要对基于 Linux 的 ACM 评测系统进行研究。

2 ACM 评测系统架构

ACM 评测系统核心是接受用户提交的程序,把程序的运行结果返回给用户,运行结果为程序正确或程序不正确。程序不正确包括:运行错误(run-time error)、运行超时(time-limit exceeded)、运行结果错误(wrong answer)以及运行结果输出格式错误(presentation error)。根据 ACM 评测系统功能需求,本系统架构如图 1 所示。

收稿日期: 2010-02-04

作者简介: 杨志伟(1982-),男,广东广州人,仲恺农业工程学院计算科学系助教,硕士,研究方向: 计算机应用; 曾艳姗(1978-),女,广东广州人,讲师,硕士,研究方向: 运筹学。

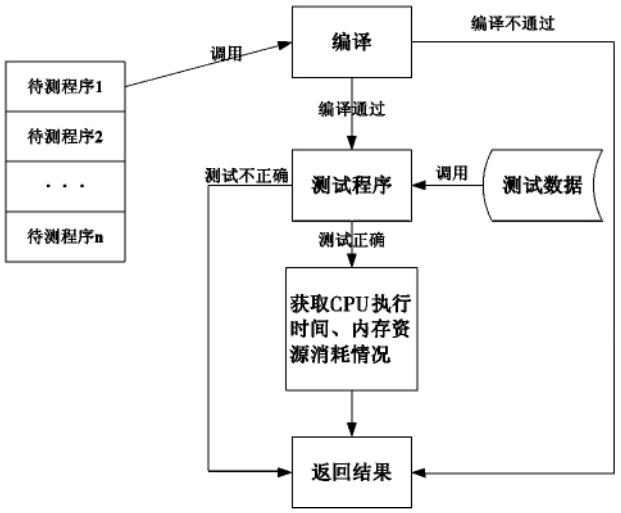

图 1　ACM 评测系统架构

2.1 编译

ACM 评测系统首先需要对提交的待测程序进行编译。对于 C、C ++ 程序的编译链接,采用 GNU 的旗舰产品 GCC。但是由于 GCC 没有提供应用程序接口,因而需要使用 system 函数。在本系统中,以调用 shell 命令的方式来调用 GCC 编译,链接编译用户提交的 C、C ++ 程序,然后将标准错误重定向到文件中,这样在编译出错的时候,能将错误提示返回客户。对于 C 程序,完整的调用类似 system("/usr/bin/gcc 1000.c -o 1000 -ansi -Wall -Wextra -lm 2 > /tmp/oj_gcc_err.txt"),使用的 GCC 选项如表 1 所示。

表 1　GCC 选项

-ansi	支持符合 ANSI 标准的 C 程序,关闭 GNU C 中某些不兼容 ANSI C 的特性
-Wall -Wextra	让编译器尽量提供警告信息,这样能尽早发现一些隐藏的错误
-lm	由于在线评测系统的题目往往设计数学运算,所以默认动态链接数学库 libm

对于 C ++ 、Java 程序,使用相应的编译器 g ++ 和 javac 即可。

2.2 执行测试程序

对于编译通过的程序,需要调用测试数据进行测试程序结果是否正确,因而需要执行测试程序。本系统按照经典的终端 shell 执行命令的方式: 父进程 fork 一个子进程,然后使用 exec 函数装载测试程序,用测试程序完全替换当前的进程,从测试程序的 main 函数开始执行,exec 仍然保持进程的父子关系。这样,父进程就能取得 exec 所执行的测试程序的结束状态、CPU、内存等资源使用情况。

由于不可能采用从键盘输入测试数据的方式,因而需要在 fork 出的子进程调用 exec 装载测试程序之前,使用 freopen 将要执行的测试程序的标准输入重定向到准备好的测试数据文件中,由内核读取测试数据文件,完成输入过程。

父子进程之间需要通讯,文献 [2] 使用 Linux 操作系统进程间通信的系统 VIPC 的队列方式实现。本系统使用共享内存的方式在父子进程之间进行通讯,由于父进程不需要往共享内存中写入数据,只有子进程往共享内存中写入其输出,所以不需要考虑同步、加锁的问题。父进程阻塞到子进程执行完毕,然后取得共享内存中子进程的输出结果,再与正确结果进行比较。本系统使用 mmap 函数创建共享内存,mmap 函数创建内存与指定的文件关联映射,对该内存的读写也就相当于对文件的读写,该内存在 fork 出来的子进程中依然有效,所以 mmap 函数可以用来创建有亲缘关系的进程间的共享内存。

由于提交的测试程序的结果都是输出到标准输出 stdout 中,为了记录测试程序的输出,首先使用 dup2 将测试程序的标准输出 stdout 指向指定的文件描述符中,再通过 mmap 函数将该文件描述符与一块共享内存进行关联映射。这样,当对文件写时,也即是对该共享内存进行写,共享内存中的数据即是测试程序的执行输出结果。mmap 函数 [3] 功能如图 2 所示。

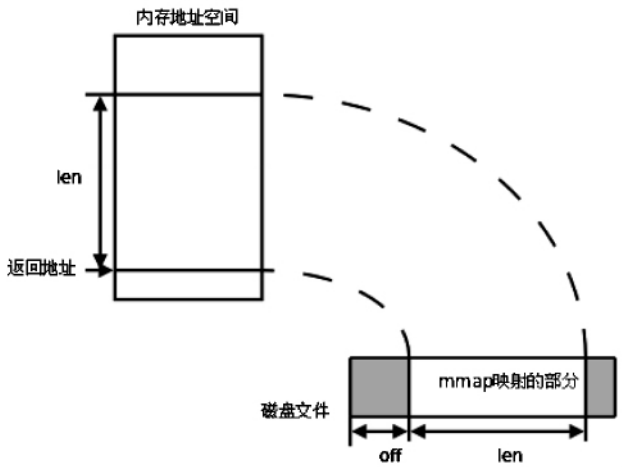

图 2　mmap 函数功能

2.3 获取 CPU 执行时间、内存资源消耗情况

在程序测试结果正确的前提下,算法优劣的最重要表现就是 CPU 执行时间的长短以及内存占用的多少,也就是空间复杂度与时间复杂度,因而 ACM 测评系统需要返回这两个结果。在 Linux 内核中,对进程各种资源的使用情况都有统计,包括用户 CPU 时间总量、系统 CPU 时间总量、页面出错次数、接收到信号的次数等。这些信息可以通过函数 int getrusage(int who, struct rusage * rusage) [4] 取得,其中参数意义如下:

第一个参数为 RUSAGE_SELF 或 RUSAGE_CHILDREN,指明是获取本进程或子进程的资源使用情况;

第二个参数指向结构体 rusage 的指针,rusage [4] 各成员如下:

```
struct rusage{
struct timeval ru_utime; /* user time used * /
struct timeval ru_stime; /* system time used * /
long ru_maxrss; /* max resident set size * /
long ru_ixrss; /* integral shared text memory size * /
long ru_idrss; /* integral unshared data size * /
long ru_isrss; /* integral unshared stack size * /
long ru_minflt; /* page reclaims * /
long ru_majflt; /* page faults * /
long ru_nswap; /* swaps * /
long ru_inblock; /* block input operations * /
long ru_oublock; /* block output operations * /
long ru_msgsnd; /* messages sent * /
long ru_msgrcv; /* messages received * /
long ru_nsignals; /* signals received * /
long ru_nvcsw; /* voluntary context switches * /
long ru_nivcsw; /* involuntary context switches * /
};
```

其中,有关 CPU 的参数包括 user time used 和 system time used,将用户 CPU 时间加上系统 CPU 时间相加得到测试程序总的执行时间。

有关内存参数包括 max resident set size 和 integral shared text memory size。因为进程的内存分为代码段、数据段、堆、栈、引用的动态链接库等,因而要精确地统计进程执行期间的内存使用量很困难。本系统使用 max resident set size 来对测试程序的内存使用情况进行度量。

在本系统中,父进程必须等待子进程执行完毕才能去计算子进程的 CPU 时间及内存使用量。因而使用 wait 函数阻塞等待子进程执行完毕,然后使用 getrusage 获得子进程的资源使用情况。这两个动作也可以使用 wait4 一起来完成。关键代码实现如下:

```
wait4( pid, &status, WUNTRACED, &res_usage) ;
//执行时间,以毫秒算,累计用户、系统时间
elapsed_time = ( res_usage. ru_utime. tv_sec + res_usage. ru_stime. tv_sec) * 1000;
elapsed_time + = ( double) ( res_usage. ru_utime. tv_usec + res_usage. ru_stime. tv_usec) /1000;
printf( "max resident set size: %ld\n",res_usage. ru_maxrss) ; //内存使用
```

3 系统安全性

对于待测程序,由于系统函数接口、文件系统以及网络资源对用户保留,因而存在安全性隐患,需要通过相应的措施来实现测评系统安全性,方式有:(1) 资源限制;(2) 限制某些系统调用。

3.1 资源限制

由于在线评测系统是直接编译链接用户提交的程序然后执行,有可能会出现用户编写恶意程序来影响性能,可能出现的恶意情况如下:

(1) 使用 malloc 不断地分配大内存,耗尽系统可用内存;

(2) 使用无意义的 while(1) 死循环,使 CPU 一直忙碌;

(3) 不停地向硬盘写大文件,耗尽硬盘存储空间;

(4) 不停地调用 fork 产生子进程,消耗大量的系统资源;

(5) 设置很大的套接字缓冲区,消耗可用内存。

针对以上恶意情况,需要对各种资源进行相应限制。Linux 内核提供的 setrlimit [4] 函数,可以对进程使用的各种资源进行限制设置:

```
int setrlimit( int resource,const struct rlimit* rlptr)
```

第一个参数指定可以限制的资源类型如表 2 所示。

表 2 第一个参数指定可以限制的资源类型

RLIMIT_AS	进程可用存储区的最大总长度,按字节计,会影响 sbrk 和 mmap 函数,也即是对 malloc 所分配的内存大小进行限制,因为 malloc 函数最终还是通过 sbrk 或 mmap 系统调用来分配内存的。
RLIMIT_CORE	进程崩溃的时候,Core 文件的最大字节数为 0,则阻止创建 Core 文件。
RLIMIT_CPU	CPU 时间的最大量值,按秒计,当超过限制时,引发 SIGXCPU 信号。
RLIMIT_DATA	数据段的最大字节长度。
RLIMIT_FSIZE	可以创建的文件的最大字节数,当超过限制时,发送 SIGXFSZ 信号。
RLIMIT_LOCKS	一个进程可持有的文件锁的最大数。
RLIMIT_MEMLOCK	一个进程使用 mlock 能够锁定在存储器中的最大字节长度。
RLIMIT_NOFILE	每个进程能够打开的最大文件数。
RLIMIT_NPROC	每个进程可以创建的最大子进程数。
RLIMIT_RSS	最大的驻内存集的字节长度,当内存不够用时,内核从进程处取回超过限制的部分。这也可以对进程的内存使用情况进行限制。
RLIMIT_SBSIZE	可以占用的套接字缓冲区的最大长度。
RLIMIT_STACK	栈的最大长度。

第二个参数:

```
struct rlimit{
rlim_t rlim_cur; /* 软限制* /
rlim_t rlim_max; /* 硬限制* /
};
```

其中硬限制是指操作系统所设定的限制,除了 root 用

户,普通用户是不能更改的。软限制是应用程序设定的,但是必须小于等于硬限制。

在本系统中,除了 RLIMIT_CPU,其他资源的软限制都是硬限制。对于 RLIMIT_CPU,当测试程序的 CPU 时间到达软限制时,引发 SIGXCPU 信号,默认的处理是终止程序,但是恶意的测试程序会设置 SIGXCPU 的处理动作,对 SIGXCPU 信号进行拦截,在达到 RLIMIT_CPU 软限制之后,内核每隔 1 秒向进程发送 SIGXCPU 信号,到达 RLIMIT_CPU 硬限制后,发送 SIGKILL 杀掉进程。如果将测试程序的 RLIMIT_CPU 软硬限制设置相同,就只能得到 SIGKILL 信号。

当程序通过 pause( ) 挂起的时候不计算 CPU 时间,因而通过设置 RLIMIT_CPU 限制还是没办法防止恶意用户提交大量测试程序。每个程序都调用 pause( ) 挂起,长时间闲置最终可能导致系统无法创建新进程。为了解决该问题,在父进程中通过调用 alarm( ),设置一个时钟,它的值比 RLIMIT_CPU 稍大,当时钟值到达的时候,直接向子进程发送 SIGKILL 信号,终止子进程。由于 fork 创建的子进程是没有继承父进程的 alarm 时钟,所以这个设置是不影响正常测试程序的运行。

当测试进程超过某项限制时,内核向该进程发送指定信号,这些信号的默认动作是终止进程,父进程通过 wait4 中返回的状态,可以正确地判断出测试进程的终止原因,给用户以准确的错误提示。但是测试进程也有可能对这些信号设置处理函数,拦截这些信号。这种情况下,测试进程最终会在 alarm 的时钟到达时被父进程 KILL 掉,因为 SIGKILL 信号是无法拦截的。在这种情况下,本系统仅仅是无法精确判断测试进程超越了那项限制,但是测试进程最终还是在规定时间内被终止了,因而对系统并没有造成危害。

3.2 限制某些系统调用

由于在线评测系统只要是为了对用户程序的算法优劣进行评测,因而不需要使用创建目录、文件、建立 Socket 等在普通应用程序中比较常见的函数,只需要通过改变系统调用表 sys_call_table 中相应调用的函数指针,来达到限制指定系统调用的目的。Fedora8 linux 下,所有系统调用对应的编号列举在 /usr/include/asm/unistd.h 中:

```
#define__NR_restart_syscall 0
#define__NR_exit 1
…
#define__NR_socketcall 102
```

如果要避免恶意用户调用 rmdir,只需要将 sys_call_table [_NR_rmdir] 指向一个空函数即可。因为系统调用都是通过 0x80 中断来进行的,故可以通过查找 0x80 中断的处理程序来获得 sys_call_table 的地址,步骤如下:

(1) 获取中断描述符表的地址;

(2) 从中查找 0x80 中断的服务例程;

(3) 搜索该例程的内存空间,从其中获取所有 sys_call_table 的地址。

4 结束语

ACM 竞赛是水平最高的国际大学生程序设计竞赛,而 ACM 评测系统对于 ACM 竞赛培训至关重要。本文提出了一个基于 Linux 的 ACM 评测系统,详述该系统的架构以及各个功能的实现,并针对该系统可能出现的安全性问题,提出相应的解决方案。

参考文献:

[1] 琚生根,廖勇,周刚,等. ACM 竞赛与实验教学创新[J]. 实验技术与管理,2009,26(5):125-126,131.

[2] 郭嵩山,王磊,张子臻. ACM/ICPC 与创新性 IT 人才的培养[J]. 实验室研究与探索,2007,26(12):181-185,189.

[3] 丁永生,张桂新,赵超. 基于 Web 的 ACM 国际大学生程序设计竞赛教学系统开发[J]. 东华大学学报(自然科学版),2001,27(4):93-97,100.

[4] 何静雯. ACM/ICPC 评测系统综述[C]//中南六省自动化学会第二十三届年会. 2005:405-409.

[5] 尤枫,史晟辉. ACM 在线测评在编译原理实践教学中的应用探讨[J]. 计算机教育,2009(20):113-115.

[6] 王辉,胡新华,张广泉. 集群式程序设计竞赛评测系统设计与开发[J]. 计算机应用与软件,2009,26(9):119-122.

[7] 赵惊,李玮,汪维富. 基于网格计算的 ACM 系统构架研究[J]. 科技咨询,2008(24):234.

[8] 秦越磊,彭四伟. 源代码在线评测系统的设计与实现[J]. 福建电脑,2008(6):156,137.

[9] 秦越磊,彭四伟. 源代码在线评测系统性能提升方法的研究[J]. 中国科技信息,2008(9):42-44,49.

[10] 惠敏顺,朱国进. 基于 SOA 的分布式程序设计竞赛系统的研究[J]. 计算机技术与发展,2008,18(10):123-126,161.

[11] Maurice J Bach. The Design of the UNIX Operating System[M]. 北京: 机械工业出版社,2000.

[12] 郑传生. 基于 B/S 结构的程序设计竞赛自动评测系统[J]. 计算机与现代化,2007(12):109-111.

[13] [美] Richard W. UNIX 网络编程第 2 卷: 进程间通信(第 2 版)[M]. 杨继张译. 北京: 清华大学出版社,2000.

[14] [美] Richard Stevens W, Stephen A Rago. Unix 环境高级编程(第 2 版)[M]. 尤晋元,等译. 北京: 人民邮电出版社,2006.