

菜鸟程序员的成长历程

彪悍的人生可以没有妹，但是不能没有技术！

博客园 博问 闪存 首页 新随笔 联系 管理 订阅

C 语言中 setjmp 和 longjmp

在 C 语言中，我们不能使用 goto 语句来跳转到另一个函数中的某个 label 处；但提供了两个函数——setjmp 和 longjmp来完成这种类型的分支跳转。后面我们会看到这两个函数在处理异常上面的非常有用。

setjmp 和 longjmp 使用方法

我们都知道要想在一个函数内进行跳转，可以使用 goto 语句（不知怎么该语句在中国学生眼中就是臭名昭著，几乎所有国内教材都一刀切地教大家尽量不要使用它，但在我看来，这根本不是语言的问题，而是使用该语言的人，看看 Linux 内核中遍地是 goto 语句的应用吧！），但如果从一个函数内跳转到另一个函数的某处，goto 是不能完成的，那该如何实现呢？

函数间跳转原理

我们要实现的一个 GOTO 语句（我自己定义的），能实现在函数间进行任意跳转，如下例，在函数 g() 中有条语句GOTO Label;可以跳转到 f() 函数的 Label: 标签所指向的位置，那么我们应该如何实现呢？

```
void f()
{
    //...
    Label:
    //...
}

void g()
{
    //...
    GOTO Label;
    //...
}
```

首先我们要知道，实现这种类型的跳转，和操作系统中任务切换的上下文切换有点类似，我们只需要恢复 Label 标签处函数上下文即可。函数的上下文包括以下内容：

- **函数栈帧**，主要是栈帧指针BP和栈顶指针SP
- **程序指针PC**，此处为指向 Label 语句的地址
- **其它寄存器**，这是和体系相关的，在 x86 体系下需要保存有的 AX/BX/CX 等等 callee-regs。

这样，在执行 GOTO Label; 这条语句，我们恢复 Label 处的上下文，即完成跳转到 Label 处的功能。

如果你读过 Linux 操作系统进程切换的源码，你会很明白 Linux 会把进程的上下文保存在 task_struct 结构体中，切换时直接恢复。这里我们也可以这样做，将 Label 处的函数上下文保存在某个结构体中，但执行到 GOTO Label 语句时，我们从该结构体中恢复函数的上下文。

这就是函数间进行跳转的基本原理，而 C 语言中 setjmp 和 longjmp 就为我们完成了这样的保存上下文和切换上下文的工作。

函数原型

```
#include <setjmp.h>
int setjmp(jmp_buf env);
```

setjmp 函数的功能是将函数在此处的上下文保存在 jmp_buf 结构体中，以供 longjmp 从此结构体中恢复。

- 参数 env 即为保存上下文的 jmp_buf 结构体变量；
- 如果直接调用该函数，返回值为 0；若该函数从 longjmp 调用返回，返回值为非零，由 longjmp 函数提供。根据函数的返回值，我们就可以知道 setjmp 函数调用是第一次直接调用，还是由其它地方跳转过来的。

```
void longjmp(jmp_buf env, int val);
```

longjmp 函数的功能是从 jmp_buf 结构体中恢复由 setjmp 函数保存的上下文，该函数不返回，而是从 setjmp 函数中返回。

我的Github

我的CSDN博客

我的sina微博

个人简历（to 20

站长统计·当前在

Visit From 2012.

Visitors

26,443

938

611

224

198

94

FLAG

昵称：hazir

园龄：3年1个月

粉丝：76

关注：12

+加关注

| | | |
|----|----|----|
| < | | > |
| 日 | 一 | 二 |
| 27 | 28 | 29 |
| 3 | 4 | 5 |
| 10 | 11 | 12 |
| 17 | 18 | 19 |
| 24 | 25 | |
| 31 | 1 | |

分享到：

- 常用链接
- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签
- 更多链接

- 最新随笔
1. 浅谈 C++ 中的
2. 【译】什么是
3. 【译】Getting
4. 例解 Linux 下
5. Hash 函数及
6. 例解 Linux cd
7. 破解 Windows
- ad 2
8. Linux 内核版
9. C 语言中 setj
10. GCC 中零长

- 我的标签
- Linux(12)
- kernel(5)
- command(3)
- C语言(3)
- array(2)

- 参数 env 是由 setjmp 函数保存过的上下文。
- 参数 val 表示从 longjmp 函数传递给 setjmp 函数的返回值，如果 val 值为0， setjmp 将会返回1，否则返回 val。
- longjmp 不直接返回，而是从 setjmp 函数中返回，longjmp 执行完之后，程序就像刚从 setjmp 函数返回一样。

简单实例

下面是个简单的例子，虽然还只是函数内跳转，但足以说明这两个函数的功能了。

```
#include < setjmp.h >
main() {
    jmp_buf env;
    int i;

    i = setjmp(env);
    printf("i = %d\n", i);
    if (i != 0) exit(0);

    longjmp(env, 2);
    printf("This line does not get printed\n");
}
```

Records the state at this point. Specifically, the PC points here. By default, setjmp returns 0

When we call longjmp, control jumps back to the point where the last setjmp was called, supplying 2 as the value assigned to i.

运行该程序得到的结果为：

```
i = 0
i = 2
```

C 语言异常处理

Java、C# 等面向对象语言中都有异常处理的机制，如下就是典型的 Java 中异常处理的代码，两个数相除，如果被除数为0抛出异常，在函数 f() 中可以获取该异常并进行处理：

```
double divide(double to, double by) throws Bad {
    if(by == 0)
        throw new Bad ("Cannot / 0");
    return to / by;
}

void f() {
    try {
        divide(2, 0);
        //...
    } catch (Bad e) {
        print(e.getMessage());
    }
    print("done");
}
```

在 C 语言中虽然没有类似的异常处理机制，但是我们可以使用 setjmp 和 longjmp 来模拟实现该功能，这也是这两个函数的一个重要的应用：

```
static jmp_buf env;

double divide(double to, double by)
{
    if(by == 0)
        longjmp(env, 1);
    return to / by;
}

void f()
{
    if (setjmp(env) == 0)
        divide(2, 0);
    else
        printf("Cannot / 0");
    printf("done");
}
```

如果复杂一点，可以根据 longjmp 传递的返回值来判断各种不同的异常，来进行区别的处理，代码结构如下：

```
switch(setjmp(env)){
    case 0: //default
```

process(2)

Python(2)

syscall(2)

systemtap(1)

top500(1)

更多

随笔分类(64)

C/C++编程(16)

JAVA咖啡馆(3)

Python(2)

编程心得(3)

操作系统(8)

读书笔记(9)

考研日记(4)

梦幻乌托邦

内核代码艺术(3)

嵌入式入门(3)

算法初体验(4)

细说生活(5)

智能车竞赛(4)

随笔档案(66)

2014年3月 (2)

2014年1月 (3)

2013年12月 (1)

2013年11月 (3)

2013年10月 (4)

2013年9月 (5)

2013年8月 (1)

2013年7月 (3)

2013年6月 (1)

2013年5月 (3)

2012年12月 (1)

2012年11月 (2)

2012年6月 (1)

2012年5月 (6)

2012年4月 (15)

2012年3月 (2)

2011年8月 (2)

2011年7月 (3)

2011年5月 (8)

积分与排名

积分 - 45332

排名 - 3531

最新评论

1. Re:破解 WindwnPad 2 @luiswin 楼主 加油啊 把

2. Re:破解 WindwnPad 2 修改后，确实fal:npad时，就会弹还是没有开启，动更新版本，该无效。

3. Re:破解 WindwnPad 2 这个用ildasm和改16进制的方式

4. Re:浅谈 C++ 写得非常好，赞

5. Re:破解 WindwnPad 2 @hazir 跳第三步，搜索2

```
    //...
    case 1:           //exception 1
    //...
    case 2:           //exception 2
    //...
    //...
```

关于使用 C 语言来处理异常，可以参见[这篇文章](#)，介绍了更多复杂的结构，但无外乎就是 setjmp 和 longjmp 的应用。

参考资料

- <http://en.wikipedia.org/wiki/Longjmp>
- www.cs.purdue.edu/homes/cs240/lectures/Lecture-19.pdf



本博客的内容如果没有标注转载字样，均属个人原创！欢迎学习交流，如果觉得有价值，欢迎转载，转载请注明出处，谢谢！

邮箱：[#->@](mailto:haifenglinying@yahoo.cn)

个人主页：www.hazirguo.com



海风林影

: 链接给错了，猛戳这个：http://t.cn/RPlaQB1(转)海风林影：空指 (图)

8月4日 08:40 | 微博

 weibo.com/haifenglinying

分类: [C/C++编程](#)

标签: [setjmp](#), [longjmp](#)

绿色通道:


[好文要顶](#)

[关注我](#)

[收藏该文](#)

[与我联系](#)



 [hazir](#)

[关注 - 12](#)

[粉丝 - 76](#)

[+加关注](#)

30

(请您对文章做出评价)

- « 上一篇: [GCC 中零长数组与变长数组](#)
- » 下一篇: [Linux 内核版本命名](#)

posted @ 2013-11-03 15:54 [hazir](#) 阅读(590) 评论(6) [编辑](#) [收藏](#)

发表评论

- #1楼 2013-11-03 16:13 | [飞鸟_Asuka](#)

从一个函数"跳转"到另一个函数内部，或者从流程外调到流程内本来就违反了结构化程序设计方法的原则。而且会造成程序的逻辑混乱，所以是不建议使用goto或者类似的直接跳转指令的。

支持(0) 反对(0)
- #2楼[楼主] 2013-11-03 16:28 | [hazir](#)

@飞鸟_Asuka
Linus 说过: "It's made more horrible by the fact that a lot of substandard programmers use it", 不是 goto 语句破坏了原则，是写程序的人逻辑混乱，破坏原则！

支持(0) 反对(0)
- #3楼 2013-11-03 17:12 | [悠闲的小众人](#)

代码你试过了吗?我昨天刚看另外一本书说到这个,但是写过在codeblock下无法正常运行

支持(0) 反对(0)
- #4楼[楼主] 2013-11-03 17:42 | [hazir](#)

@悠闲的小众人
在 Linux 下用 gcc 编译测试通过~~~ 我不知道你说的 codeblock 是用的是什么编译器，在什么平台下运行的~~~

支持(0) 反对(0)
- #5楼 2013-11-03 18:25 | [garbageMan](#)

我觉得与其说longjmp类似goto
倒不如说更类似一种超级return

支持(0) 反对(0)
- #6楼 2013-11-04 09:58 | [求道于盲](#)

解释得很清楚 谢谢博主分享~~~

支持(0) 反对(0)

阅读排行榜

1. 大学总结之影
2. 破解 Windows
3. C语言中结构
4. 内核探测工具
5. 进程与线程的

评论排行榜

1. 大学总结之影
2. 破解 Windows
3. Linux吃掉我的
4. GCC 中零长数
5. Linux Kernel

推荐排行榜

1. 大学总结之影
2. 浅谈 C++ 中的
3. Hash 函数及
4. Linux Core Di
5. Linux Kernel

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

[博客园首页](#) [博问](#) [新闻](#) [闪存](#) [程序员招聘](#) [知识库](#)



最新IT新闻:

- WP平台Lumia系列一统天下局面将结束
 - 马云：中国为什么需要足球？
 - 淘宝数据：女生胸围越大越是有钱
 - 体热充电宝两小时给iPhone充满电？
 - 科学家详解宇航员如何在太空上厕所
- » [更多新闻...](#)

最新知识库文章:

- 父子页面之间跨域通信的方法
 - Android开发在路上：少去踩坑，多走捷径
 - 从用户行为打造活动交互设计闭环——2014年世界杯竞猜活动设计总结
 - 如何通过一个问题，完成最成功的技术面试
 - 我所理解的技术领导力
- » [更多知识库文章...](#)

Copyright ©2014 hazir

一个代码可以创造一个世界，也可以毁灭一个世界！

