**[循环结构中break、continue、return和exit的区别](http://blog.csdn.net/hunanchenxingyu/article/details/8101795)**

2012-10-23 10:43 22922人阅读 [评论](http://blog.csdn.net/hunanchenxingyu/article/details/8101795#comments)(4) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/hunanchenxingyu/article/details/8101795#report)

本文章已收录于：

http://static.blog.csdn.net/images/category_icon.jpg分类：

linux编程*（49）* http://static.blog.csdn.net/images/arrow_triangle%20_down.jpghttp://static.blog.csdn.net/images/arrow_triangle_up.jpg

[作者同类文章](http://blog.csdn.net/hunanchenxingyu/article/category/1170915)*X*

**1. break**

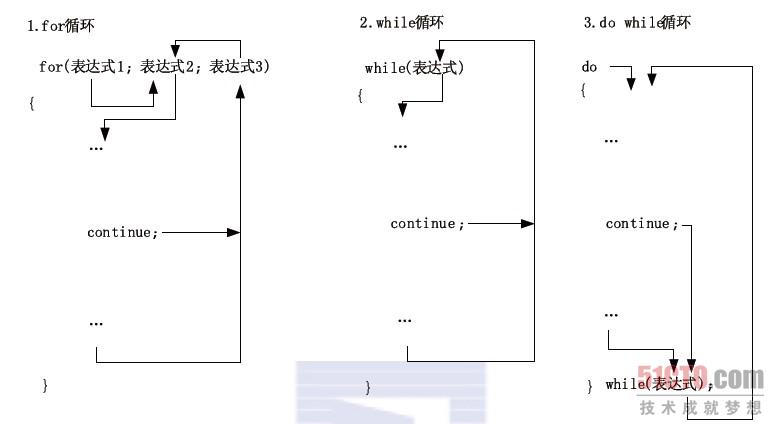
break语句的使用场合主要是switch语句和循环结构。在循环结构中使用break语句，如果执行了break语句，那么就退出循环，接着执行循环结构下面的第一条语句。如果在多重嵌套循环中使用break语句，当执行break语句的时候，退出的是它所在的循环结构，对外层循环没有任何影响。如果循环结构里有switch语句，并且在switch语句中使用了break语句，当执行switch语句中的break语句时，仅退出switch语句，不会退出外面的循环结构。通过图3-17，读者可以很直观地了解break语句的使用。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20120819/132805379.jpg> |
| （点击查看大图）图3-17　break语句 |

**2. continue**

continue语句是这5种结束循环的方式中最特殊的，因为它并没有真的退出循环，而是只结束本次循环体的执行，所以在使用continue的时候要注意这一点。图3-18为各种循环结构中continue语句的使用。

在for循环中，首先执行表达式1（注意表达式1在整个循环中仅执行一次），接着执行表达式2，如果满足条件，那么执行循环体，如果在循环体中执行了continue语句，那么就跳转到表达式3处执行，接下进行下一次循环，执行表达式2，看是否满足条件；在while循环中，如果执行了continue语句，那么就直接跳转到表达式处，开始下一次的循环判断；在do while循环体中如果执行了continue语句，那么就跳转到表达式处进行下一次的循环判断，这一点前面已经验证过了。

[](http://images.51cto.com/files/uploadimg/20120819/132825304.jpg) （点击查看大图）图3

**3. return语句**

如果在程序中遇到return语句，那么代码就退出该函数的执行，返回到函数的调用处，如果是main()函数，那么结束整个程序的运行。图3-20为return语句的使用。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20120819/133050589.jpg> |
| （点击查看大图）图3-20　return语句 |

如果是在自定义的函数中执行，那么执行return之后就返回到函数的调用处继续往下执行。

**4. exit()函数**

exit()函数与return语句的最大区别在于，调用exit()函数将会结束当前进程，同时删除子进程所占用的内存空间，把返回信息传给父进程。当exit()中的参数为0时，表示正常退出，其他返回值表示非正常退出，执行exit()函数意味着进程结束；而return仅表示调用堆栈的返回，其作用是返回函数值，并且退出当前执行的函数体，返回到函数的调用处，在main()函数中， return n和exit(n)是等价的。图3-21为exit()函数的使用。

接下来通过两段代码对return语句和exit()函数进行简单的对比，先来看return语句的使用。

1. #include **<stdio.h>**
2. #include **<stdlib.h>**
3. int print()
4. {
5. int n;
6. n = 0;
7. printf("使用return来结束循环\n");
8. while(1)
9. {
10. if(9==n)
11. return n;
12. n++;
13. }
14. return 0;
15. }
17. void main(void)
18. {
19. int ret;
20. printf("调用print()函数之前\n");
21. ret = print();
22. printf("print()函数的返回值ret=%d\n",ret);
23. printf("调用print()函数之后\n");
25. return ;
26. }

运行结果：

1. 调用print()函数之前
2. 使用return来结束循环
3. print()函数的返回值ret=9
4. 调用print()函数之后

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20120819/133154355.jpg> |
| （点击查看大图）图3-21　exit( )函数 |

在上面的代码中，用return语句来退出while死循环，在main()函数中print()函数的调用处将返回值赋给ret，打印输出后可以看到使用return语句成功地返回了9。

下面来看exit()函数的使用。

1. #include **<stdio.h>**
2. #include **<stdlib.h>**
3. void print()
4. {
5. int n;
6. n = 0;
7. printf("使用exit来结束循环\n");
8. while(1)
9. {
10. if(9==n)
11. exit(1);
12. n++;
13. }
14. return ;
15. }
17. void main(void)
18. {
19. int ret;
20. printf("调用print()函数之前\n");
21. print();
22. printf("调用print()函数之后\n");
24. return ;
25. }

运行结果：

1. 调用print()函数之前
2. 使用exit来结束循环

从以上代码可以看出，如果执行exit()函数后能够返回到main()函数的调用处，那么可以打印出接下来的信息“调用print()函数之后”，但是运行结果表明在调用exit()函数之后没有任何输出，所以**执行exit()函数之后将直接结束整个程序的运行。**