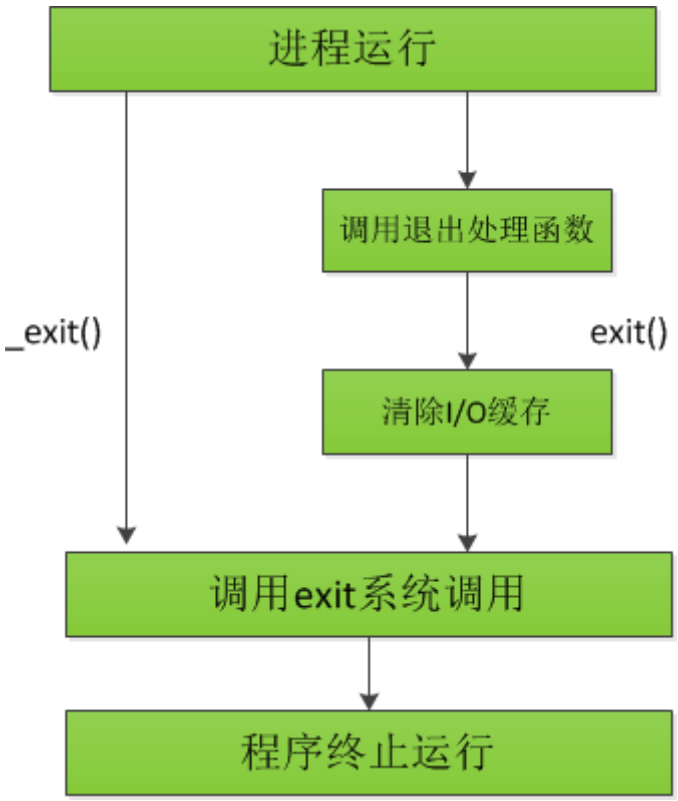

从图中可以看出，_exit 函数的作用是：直接使进程停止运行，清除其使用的内存空间，并清除其在内核的各种数据结构；exit 函数则在这些基础上做了一些小动作，在执行退出之前还加了若干道工序。exit() 函数与 _exit() 函数的最大区别在于exit()函数在调用exit系统调用前要检查文件的打开情况，把文件缓冲区中的内容写回文件。也就是图中的“清理I/O缓冲”。

所需头文件： exit: #include<stdlib.h>

_exit: #include<unistd.h>

函数原型：exit: void exit(int status)

_exit: void _exit(int status)

函数传入值：status 是一个整型的参数，可以利用这个参数传递进程结束时的状态。一般来说，0表示正常结束；其他的数值表示出现了错误，进程非正常结束。在实际编程时，父进程可以利用wait 系统调用接收子进程的返回值，从而针对不同的情况进行不同的处理。

exit()与_exit() 实例分析

printf(const char *fmt,...)函数使用的是缓冲I/O方式，该函数在遇到 "\n" 换行符时自动从缓冲区中将记录读出。

<代码示例>

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>

int main()
{
    pid_t result;
    result = fork();
    if(result<0)
    perror("fork");
    if(result == 0)
    {
        printf("This is _exit test\n");
        printf("This is the content in the buffer000");
        _exit(0);
    }
    else
    {
        printf("This is exit test\n");
        printf("This is the content in the buffer");
        exit(0);
    }
    return 0;
}
```

下面是运行结果：

```
[root@localhost process_test]# gcc exit.c -o exit
[root@localhost process_test]# ./exit
This is  _exit test
This is exit test
This is the content in the buffer[root@localhost process_test]#
```

结果分析：子进程中运行_exit(0)并未将Thisis the content in the buffer000 打印出来，而父进程中运行的exit(0)将Thisis the content in the buffer打印出来了。说明，exit(0)会在终止进程前，将缓冲I/O内容清理掉，所以即使printf里面没有 \n也会被打印出来，而_exit(0)是直接终止进程，并未将缓冲I/O内容清理掉，所以不会被打印出来。