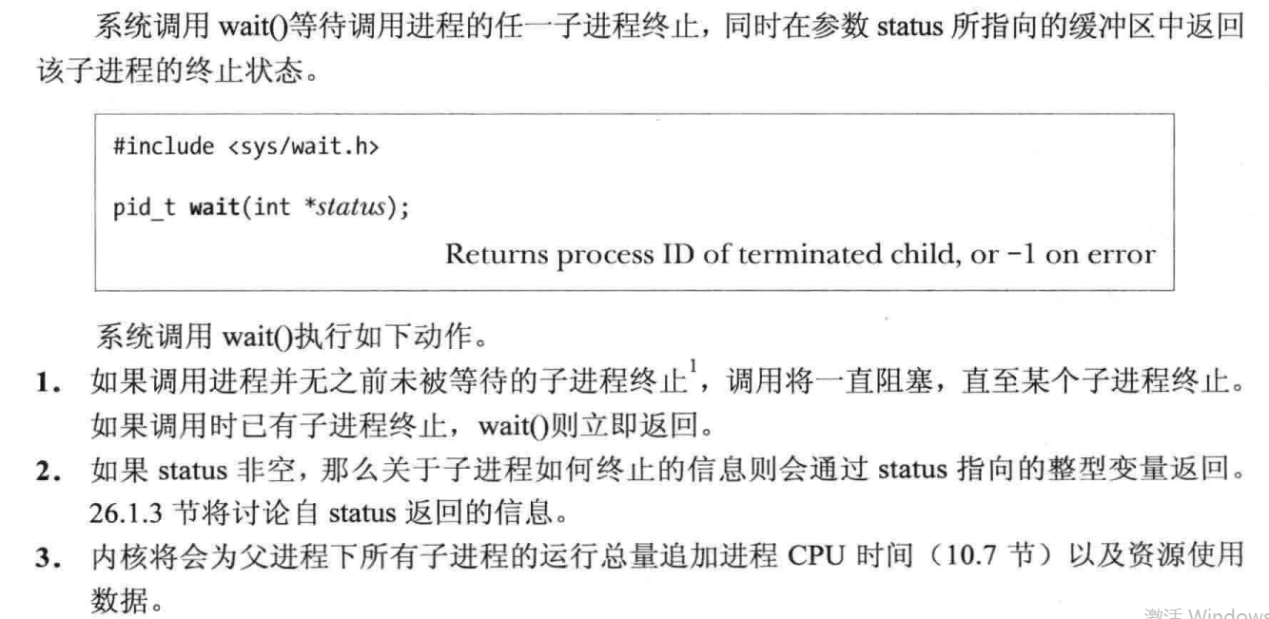
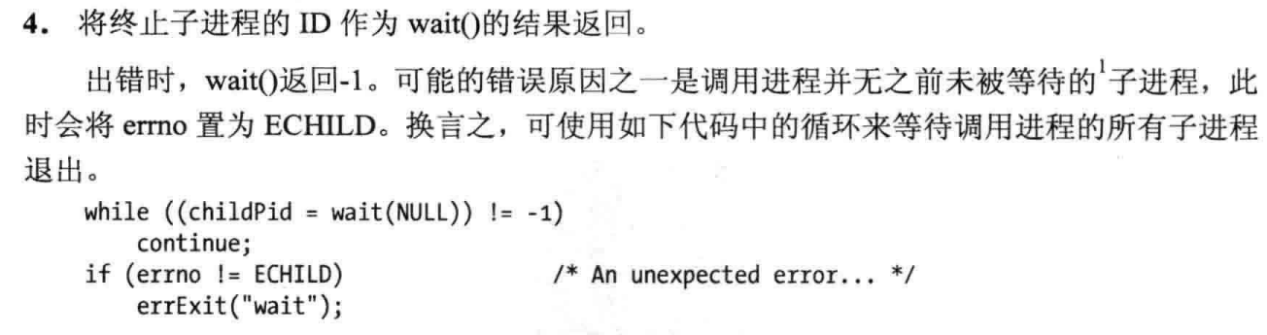
第二十六章

1. 等待子进程：
2. 系统调用wait（）：

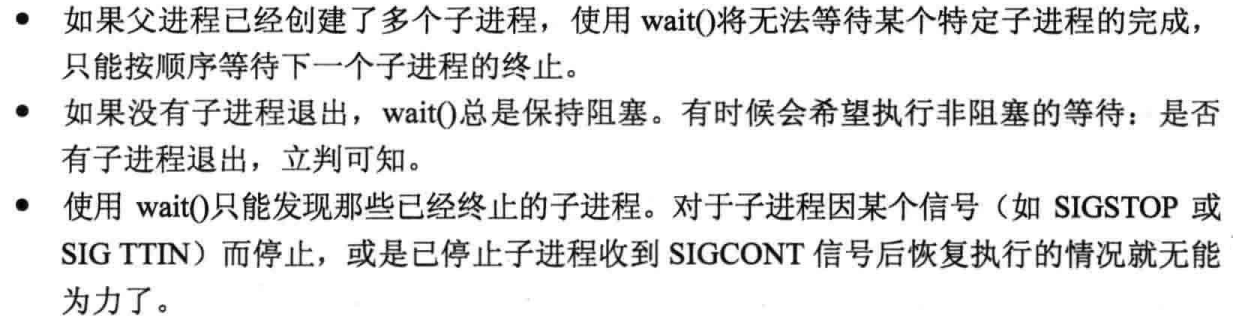


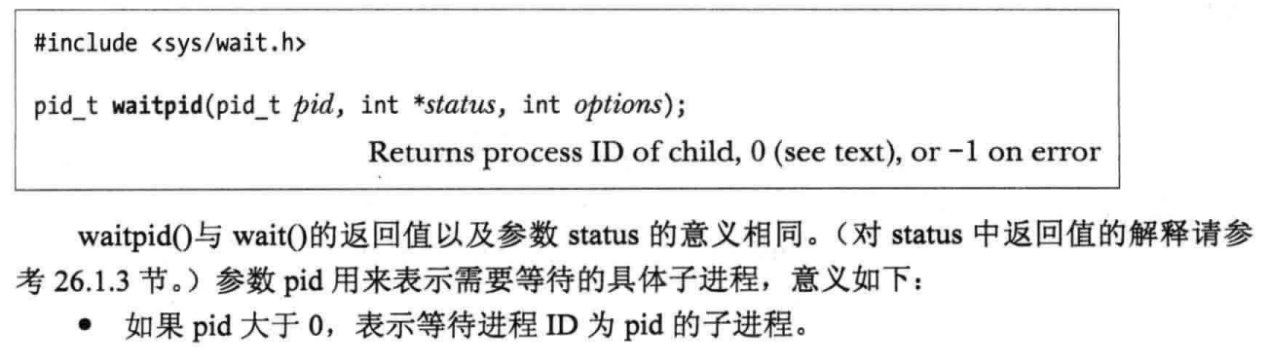


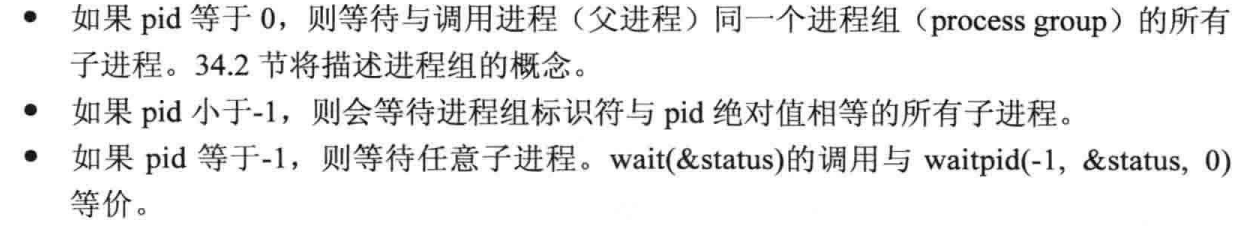
注意，子进程退出的顺序不能保证。

1. 系统调用waitpid：

系统调用wait的一些限制：



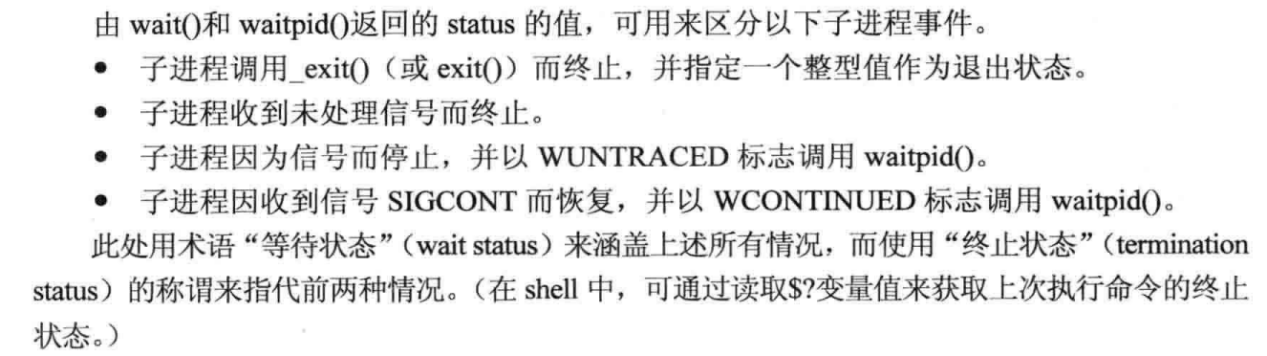




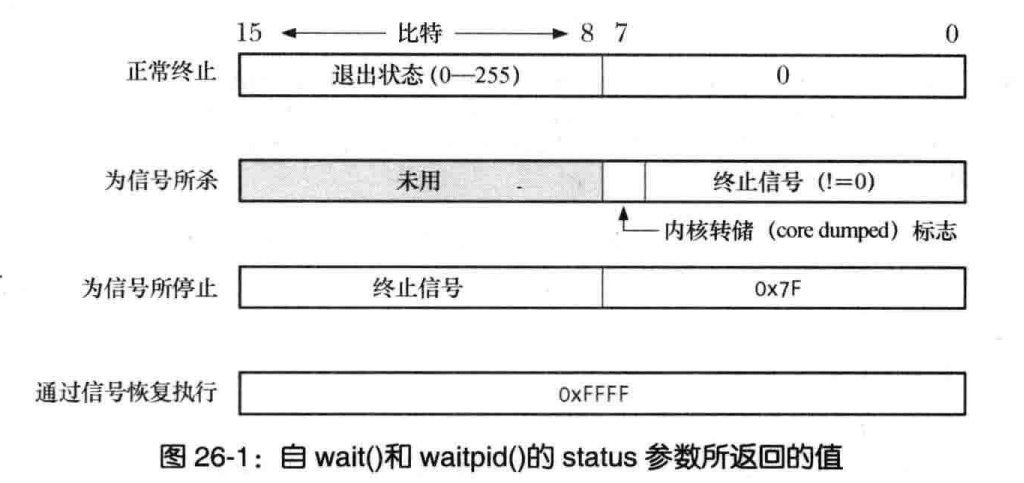
Option是一个位掩码，包括进程停止，继续，以及不阻塞。

1. 等待状态值：

状态值要分四种情况：



值得注意，在shell里面$?变量值获取的信号值是加上128，退出值则是原值。这与status里的值是不一样的。



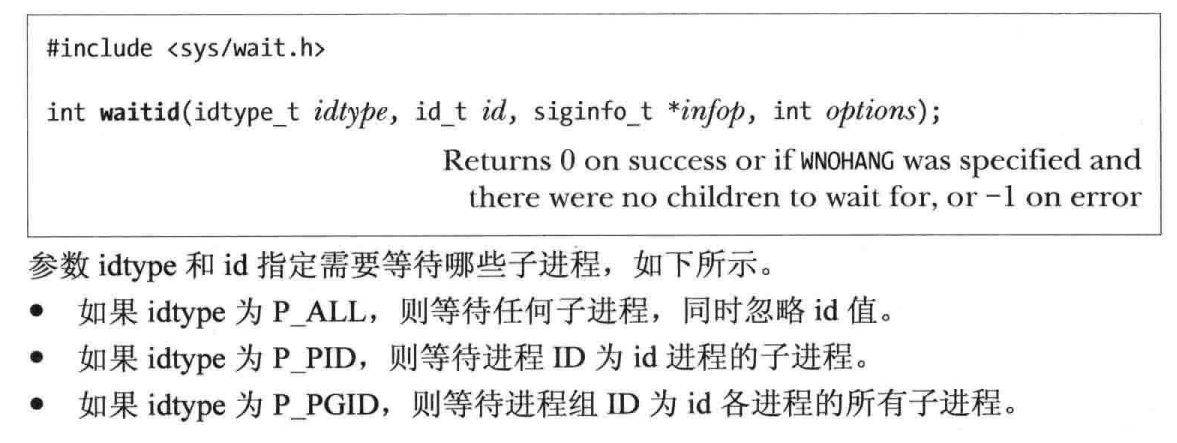
Status的返回值如上。一般使用宏去处理。

1. 从信号处理函数中返回：

通过系统调用依然可以捕获到的子进程在其信号处理函数中退出状态：子进程需要在信号处理函数返回\_exit或者废除信号再次给自己发送信号。

1. 系统调用waitid

与waitpid相比，waitid提供了更精确的控制。并且参数siginfo\_t \*infop提供了子进程相关的信息。



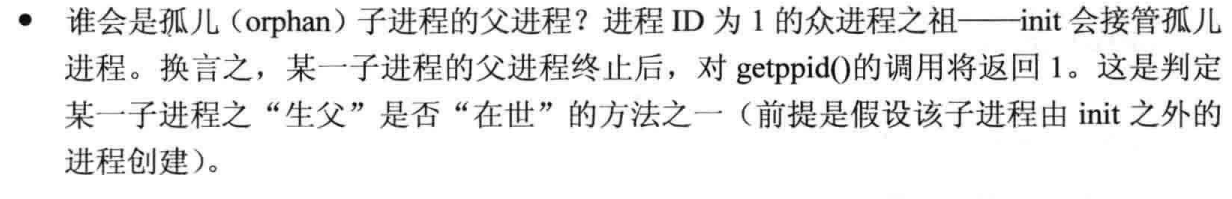
具体的参数信息查看手册。

1. 系统调用wait3和wait4：

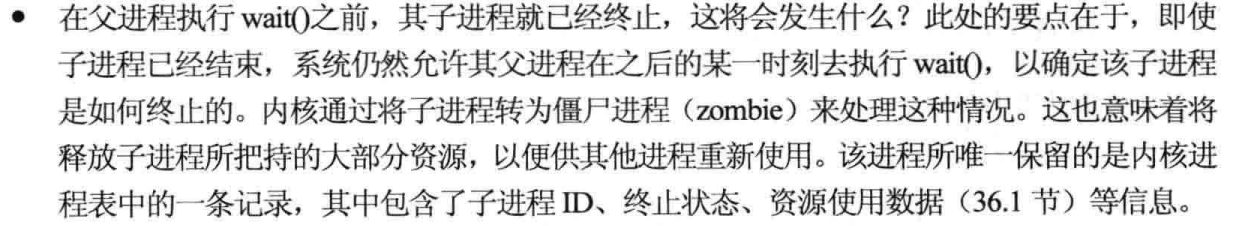
与waitpid相似，但在可移植性良好的程序里尽量避免使用这些函数。

1. 孤儿进程和僵尸进程：

孤儿进程：



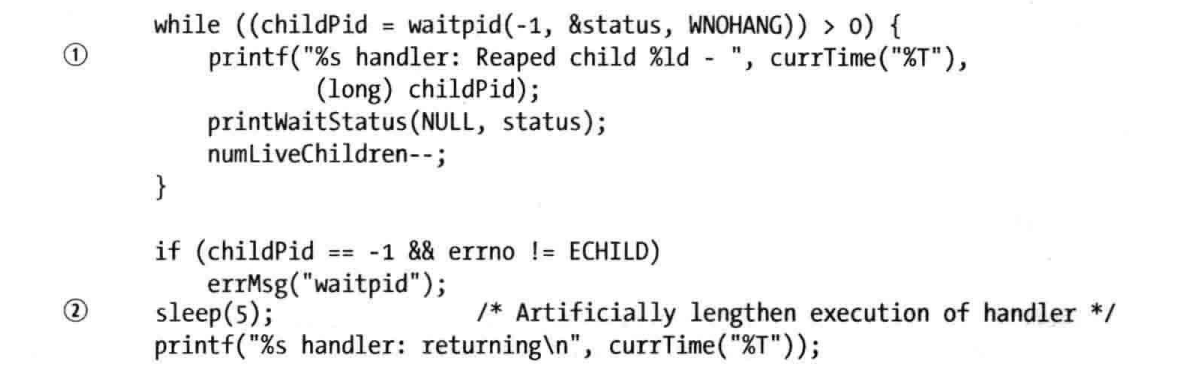
僵尸进程：



注意一点，处于僵尸进程的子进程，即使接收到了sigkill信号，也不会杀死它。

1. SIGCHLD信号：

由于子进程终止是异步事件，采用针对SIGCHLD信号处理程序和wait系列函数处理会比较好。



在信号处理函数里面，我一开始搞不懂上面那个if语句里面的判断是为何如此写。后来想了一下：为了防止waitpid函数接收完了子进程而信号处理函数仍然存在的原因。

1. 控制子进程停止与继续发送SIGCHLD的行为。
2. 显示设置SIGCHLD的SIG\_IGN的标志，可以在子进程终止后立即删除而不用转为僵尸进程。在此之前的僵尸进程仍旧保留。当然，不同系统有不同实现，具体查看书本。