1 wait()函数

#include <sys/types.h>

#include <sys/wait.h>

pid_t wait(int *status);

进程一旦调用了wait，就立即阻塞自己，由wait自动分析是否当前进程的某个子进程已经退出，如果让它找到了这样一个已经变成僵尸的子进程，wait就会收集这个子进程的信息，并把它彻底销毁后返回；如果没有找到这样一个子进程，wait就会一直阻塞在这里，直到有一个出现为止。

参数status用来保存被收集进程退出时的一些状态，它是一个指向int类型的指针。但如果我们对这个子进程是如何死掉的毫不在意，只想把这个僵尸进程消灭掉，（事实上绝大多数情况下，我们都会这样想），我们就可以设定这个参数为NULL，就象下面这样：

```
pid = wait(NULL);
```

如果成功，wait会返回被收集的子进程的进程ID，如果调用进程没有子进程，调用就会失败，此时wait返回-1，同时errno被置为ECHILD。

- wait系统调用会使父进程暂停执行，直到它的一个子进程结束为止。
- 返回的是子进程的PID，它通常是结束的子进程
- 状态信息允许父进程判定子进程的退出状态，即从子进程的main函数返回的值或子进程中exit语句的退出码。
- 如果status不是一个空指针，状态信息将被写入它指向的位置

可以上述的一些宏判断子进程的退出情况：

宏定义	描述
WIFEXITED(status)	如果子进程正常结束，返回一个非零值
WEXITSTATUS(status)	如果WIFEXITED非零，返回子进程退出码
WIFSIGNALED(status)	子进程因为捕获信号而终止，返回非零值
WTERMSIG(status)	如果WIFSIGNALED非零，返回信号代码
WIFSTOPPED(status)	如果子进程被暂停，返回一个非零值
WSTOPSIG(status)	如果WIFSTOPPED非零，返回一个信号代码

这些宏在sys/wait.h头文件里定义

2 waitpid()函数

#include <sys/types.h>

#include <sys/wait.h>

pid_t waitpid(pid_t pid, int *status, int options);

参数:

status:如果不是空，会把状态信息写到它指向的位置，与wait一样

options：允许改变waitpid的行为，最有用的一个选项是WNOHANG,它的作用是防止waitpid把调用者的执行挂起

The value of options is an OR of zero or more of the following constants:

WNOHANG return immediately if no child has exited.

WUNTRACED also return if a child has stopped (but not traced via ptrace(2)). Status for traced children which have stopped is provided even if this option is not specified.

WCONTINUED (since Linux 2.6.10) also return if a stopped child has been resumed by delivery of SIGCONT.

返回值：如果成功返回等待子进程的ID，失败返回-1

对于waitpid的p i d参数的解释与其值有关:

pid == -1 等待任一子进程。于是在这一功能方面waitpid与wait等效。

pid > 0 等待其进程I D与p i d相等的子进程。

pid == 0 等待其组I D等于调用进程的组I D的任一子进程。换句话说是与调用者进程同在一个组的进程。

pid < -1 等待其组I D等于p i d的绝对值的任一子进程

wait与waitpid区别:

- 在一个子进程终止前， wait 使其调用者阻塞，而waitpid 有一选择项，可使调用者不阻塞。
- waitpid并不等待第一个终止的子进程—它有若干个选择项，可以控制它所等待的特定进程。
- 实际上wait函数是waitpid函数的一个特例。waitpid(-1, &status, 0);