#include<sys/types.h>

#include<unistd.h>

pid\_t getpid(void); 返回：调用进程的进程I D

pid\_t getppid(void); 返回：调用进程的父进程I D

uid\_t getuid(void); 返回：调用进程的实际用户I D

uid\_t geteuid(void); 返回：调用进程的有效用户I D

gid\_t getgid(void); 返回：调用进程的实际组I D

gid\_t getegid(void); 返回：调用进程的有效组I D

pid\_t fork()

#include<sys/types.h>

#include<unistd.h>

pid\_t vfork()

#include<vfork.h>

接收到SIGCHLD信号调用

#include <sys/types.h>

#include <sys/wait.h>

pid\_t wait(int \*status)

子进程终止前阻塞 返回状态给status

pid\_t waitpid(pid\_t pid,int \*status,int options)

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ID与pid相等的子进程。

pid == 0 等待其组ID等于调用进程的组ID的任一子进程。

pid < -1 等待其组ID等于pid的绝对值的任一子进程。

WNOHANG 若由pid指定的子进程并不立即可用，则waitpid不阻塞，此时其返回值为0

WUNTRACED 若某实现支持作业控制，则由pid指定的任一子进程状态已暂停，且其状态自暂停以来还未报告过，则返回其状态。

WIFSTOPPED宏确定返回值是否对应于一个暂停子进程

exec函数簇

#include <unistd.h>

int execve(const char \*filename, char \*const argv[],char \*const envp[]);系统调用

int execl(const char \*path, const char \*arg, ...);

int execlp(const char \*file, const char \*arg, ...);

int execle(const char \*path, const char \*arg, ..., char \* const envp[]);

int execv(const char \*path, char \*const argv[]);

int execvp(const char \*file, char \*const argv[]);

l为list p为根据环境变量查找程序名 e为传递环境变量

v为数组方式传递argv