1. 在多进程间通信中已经描述信号通信，这边重复一遍。

1. #include <signal.h>
3. **void** (\*signal(**int** signo,**void** (\*func)(**int**)))(**int**);
5. 返回之前注册的函数指针。

sigaction函数类似于signal函数，完全可以替代后者，也更加稳定。

1. #include <signal.h>
3. **int** sigaction(**int** signo,**const** **struct** sigaction \*act,**struct** sigaction \*oldact);
5. 成功返回0，失败返回-1
7. signo：传递的信号
8. act：对英语第一个参数的信号处理函数
9. oldact：获取之前注册的信号处理函数指针，若不需要则传递0
11. **struct** sigaction{
12. **void** (\* sa\_handler)(**int**);
13. sigset\_t sa\_mask;
14. **int** sa\_flags;
15. }

2. 利用信号销毁僵尸进程

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <signal.h>
5. #include <sys/wait.h>
7. **void** read\_childproc(**int** sig){
8. **int** status;
10. pid\_t id = waitpid(-1, &status, WNOHANG);
12. **if**(WIFEXITED(status)){
13. printf("Removed proc id : %d \n",id);
14. printf("child send : %d \n",WEXITSTATUS(status));
15. }
16. }
18. **int** main(**int** argc, **char** \*argv[]){
19. pid\_t pid;
20. **struct** sigaction act;
22. act.sa\_handler = read\_childproc;
23. sigemptyset(&act.sa\_mask);
24. act.sa\_flags = 0;
26. sigaction(SIGCHLD,&act,0);  //注册信号，触发SIGCHILD信号后将会调用act函数
28. pid = fork();
30. **if**(pid == 0){
31. //child
32. puts("Hi, I'm child process");
33. sleep(5);
34. **return** 12;  //五秒后将会触发SIGCHILD信号
35. }**else**{
36. printf("Child proc1 id : %d \n",pid);
38. pid = fork();
40. **if**(pid ==0){
41. puts("Hi, sec child");
42. sleep(5);
43. exit(24);
44. }**else**{
45. **int** i;
46. printf("child proc2 id : %d \n",pid);
47. **for**(i=0;i<10;i++){
48. puts("wait...");
49. sleep(1);
50. }
51. }
52. }
53. **return** 0;
54. }