

守护进程的编写和使用方法 04

如何杀死守护进程？ 如何杀死守护进程的子进程？

00 test4-1编写

注意要加入回收子进程的方法，这里使用信号机制。

```
// test4-1.c
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <errno.h>
#include <stdlib.h>
#include <signal.h>

static void sig_child(int signo);

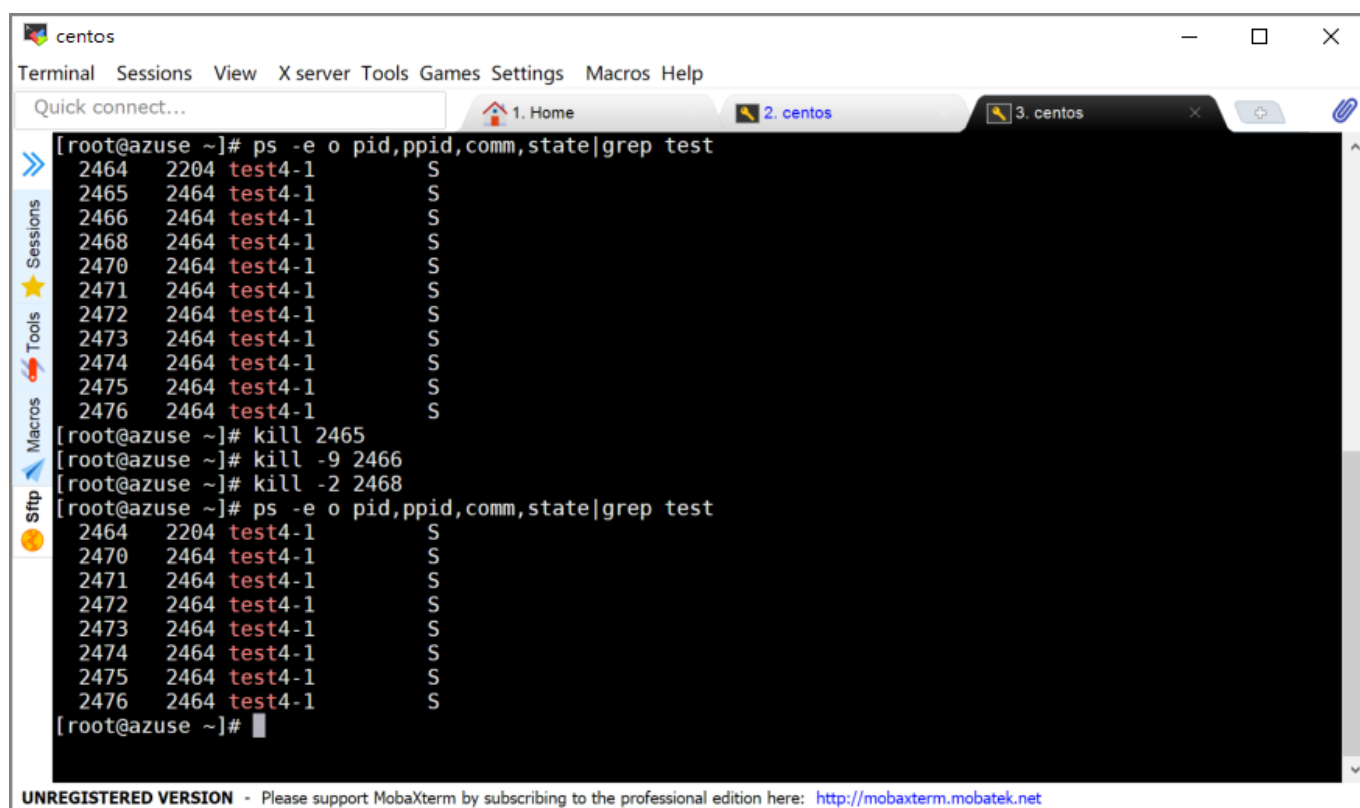
int main(){
    int pid;
    int i;
    signal(SIGCHLD,sig_child);
    for(i=0;i<10;i++){
        pid = fork();
        if(pid == 0)break;
        else sleep(3);
    }
    int counter = 0;
    while(1){

        if(pid == 0){
            printf("%d %d 1652238 sub\n",getpid(),getppid());
            fflush(stdout);
            sleep(15);
            counter++;
            // if(counter == 2){
            //     printf("sub %d exiting\n",getpid());
            //     fflush(stdout);
            //     break;
            // }
        }else{
            printf("%d %d 1652238 main\n",getpid(),getppid());
            fflush(stdout);
            sleep(5);
        }
    }
    return 0;
}
```

```
static void sig_child(int signo){
    pid_t pid;
    int stat;
    while((pid = waitpid(-1, &stat, WNOHANG)) > 0){
        printf("child %d exited\n",pid);
        fflush(stdout);
    }
}
}
```

01 如何杀死test4-1分裂出来的一个子进程？

kill 子进程pid



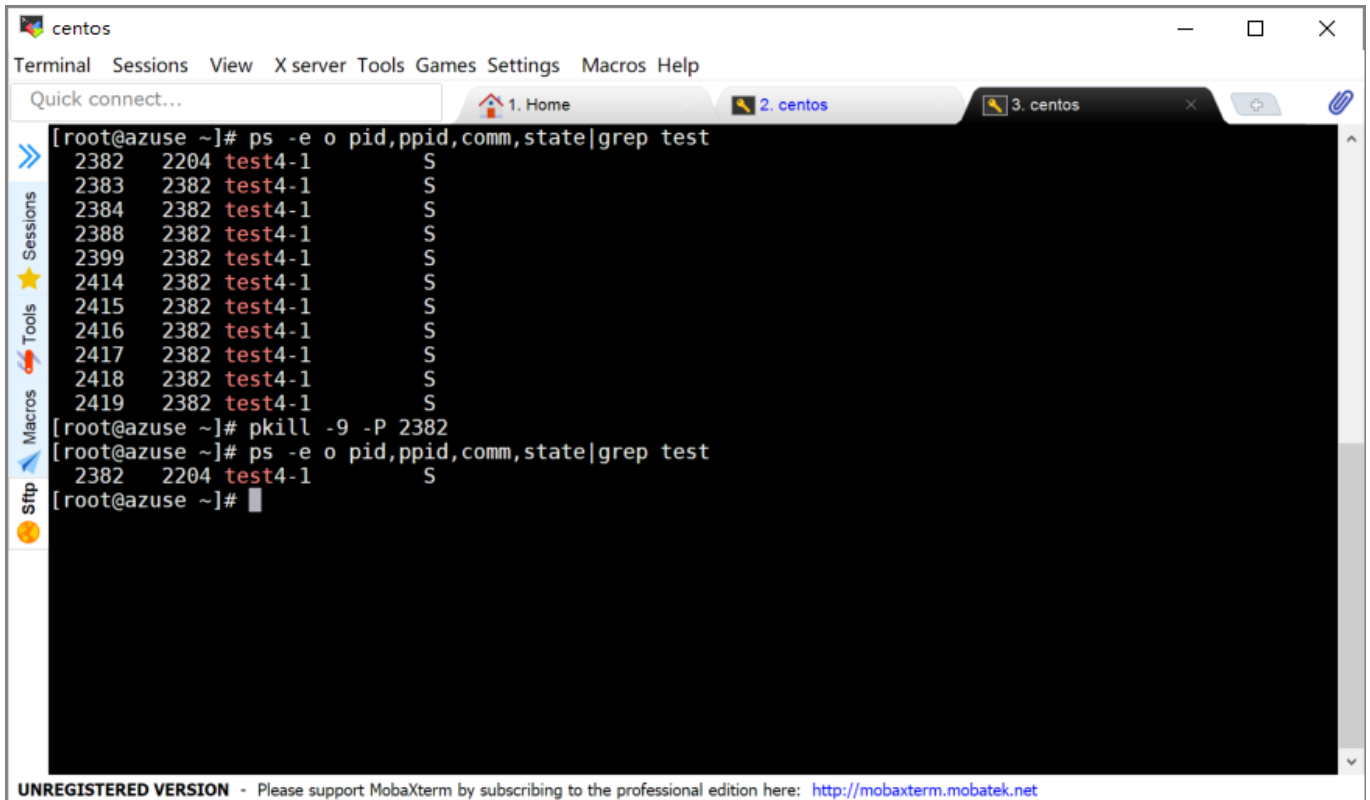
The screenshot shows a MobaXterm terminal window with the following content:

```
centos
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
1. Home 2. centos 3. centos
[root@azuse ~]# ps -e o pid,ppid,comm,state|grep test
2464 2204 test4-1 S
2465 2464 test4-1 S
2466 2464 test4-1 S
2468 2464 test4-1 S
2470 2464 test4-1 S
2471 2464 test4-1 S
2472 2464 test4-1 S
2473 2464 test4-1 S
2474 2464 test4-1 S
2475 2464 test4-1 S
2476 2464 test4-1 S
[root@azuse ~]# kill 2465
[root@azuse ~]# kill -9 2466
[root@azuse ~]# kill -2 2468
[root@azuse ~]# ps -e o pid,ppid,comm,state|grep test
2464 2204 test4-1 S
2470 2464 test4-1 S
2471 2464 test4-1 S
2472 2464 test4-1 S
2473 2464 test4-1 S
2474 2464 test4-1 S
2475 2464 test4-1 S
2476 2464 test4-1 S
[root@azuse ~]#
```

At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>"

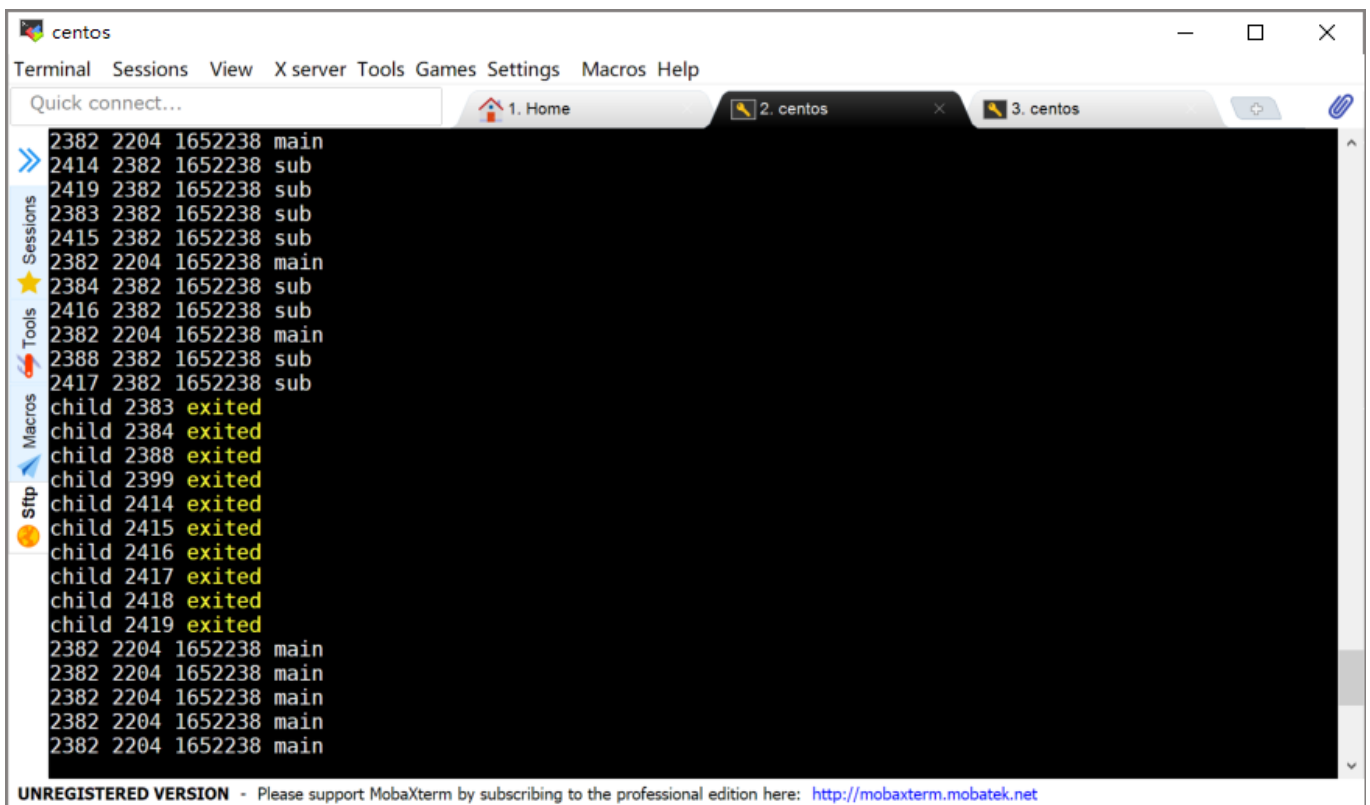
02 如何杀死test4-1分裂出来的全部子进程？

pkill -TERM -P 父进程pid



The screenshot shows a MobaXterm window with a terminal session on a CentOS system. The user has run the command `ps -e o pid,ppid,comm,state|grep test`, which displays a list of processes with PIDs 2382 through 2419, all having PPID 2204 and command `test4-1` in state `S`. Subsequently, the user runs `kill -9 -P 2382` to terminate the parent process. A second run of the `ps` command shows only the parent process (PID 2382, PPID 2204) remaining.

```
[root@azuse ~]# ps -e o pid,ppid,comm,state|grep test
2382  2204 test4-1      S
2383  2382 test4-1      S
2384  2382 test4-1      S
2388  2382 test4-1      S
2399  2382 test4-1      S
2414  2382 test4-1      S
2415  2382 test4-1      S
2416  2382 test4-1      S
2417  2382 test4-1      S
2418  2382 test4-1      S
2419  2382 test4-1      S
[root@azuse ~]# kill -9 -P 2382
[root@azuse ~]# ps -e o pid,ppid,comm,state|grep test
2382  2204 test4-1      S
[root@azuse ~]#
```



The screenshot shows the terminal after the parent process has been killed. The `ps` command output now shows that the child processes (PIDs 2414-2419) have changed their PPID to 1652238 and their state to `sub`. The parent process (PID 2382) is now in state `main`. Additionally, the status of the killed parent process is shown as `child 2383 exited` through `child 2419 exited`.

```
2382 2204 1652238 main
2414 2382 1652238 sub
2419 2382 1652238 sub
2383 2382 1652238 sub
2415 2382 1652238 sub
2382 2204 1652238 main
2384 2382 1652238 sub
2416 2382 1652238 sub
2382 2204 1652238 main
2388 2382 1652238 sub
2417 2382 1652238 sub
child 2383 exited
child 2384 exited
child 2388 exited
child 2399 exited
child 2414 exited
child 2415 exited
child 2416 exited
child 2417 exited
child 2418 exited
child 2419 exited
2382 2204 1652238 main
2382 2204 1652238 main
2382 2204 1652238 main
2382 2204 1652238 main
2382 2204 1652238 main
```

03 如果杀死test4-1，其他子进程会发生什么变化？

`kill -9 2272` 杀死父进程

发现子进程会变成孤儿进程，之后被init（1号进程）收养

```

[root@azuse ~]# clear
[root@azuse ~]# ps -e o pid,ppid,state,comm|grep test
 2272    2215  S test4-1
 2273    2272  S test4-1
 2274    2272  S test4-1
 2322    2272  S test4-1
 2324    2272  S test4-1
 2325    2272  S test4-1
 2326    2272  S test4-1
 2327    2272  S test4-1
 2328    2272  S test4-1
 2333    2272  S test4-1
 2334    2272  S test4-1
[root@azuse ~]# kill -9 2272
[root@azuse ~]# ps -e o pid,ppid,state,comm|grep test
 2273      1  S test4-1
 2274      1  S test4-1
 2322      1  S test4-1
 2324      1  S test4-1
 2325      1  S test4-1
 2326      1  S test4-1
 2327      1  S test4-1
 2328      1  S test4-1
 2333      1  S test4-1
 2334      1  S test4-1
[root@azuse ~]#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

04 如何让test4-2杀死后，全部子进程自动退出？

使用`prctl(PR_SET_PDEATHSIG, SIGHUP)`;设置在父进程终止时向子进程发送SIGHUP信号

`man prctl 2`

PR_SET_PDEATHSIG (since Linux 2.1.57)

Set the parent death signal of the calling process to arg2 (either a signal value in the range 1..maxsig, or 0 to clear).

This is the signal that the calling process will get when its parent dies. This value is cleared for the child of a fork(2) and (since Linux 2.4.36 / 2.6.23) when executing a set-user-ID or set-group-ID binary, or a binary that has associated capabilities (see capabilities(7)). This value is preserved

across `execve(2)`.

Warning: the "parent" in this case is considered to be the thread that created this process. In other words, the signal will be sent when that thread terminates (via, for example, `pthread_exit(3)`), rather than after all of the threads in the parent process terminate.

test4-2编写:

```
// test4-2.c
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <errno.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/prctl.h>

static void sig_child(int signo);

int main(){
    int pid;
    int i;
    signal(SIGCHLD,sig_child);
    for(i=0;i<10;i++){
        pid = fork();
        if(pid == 0){
            prctl(PR_SET_PDEATHSIG, SIGHUP);
            break;
        }
        else sleep(3);
    }
    int counter = 0;
    while(1){

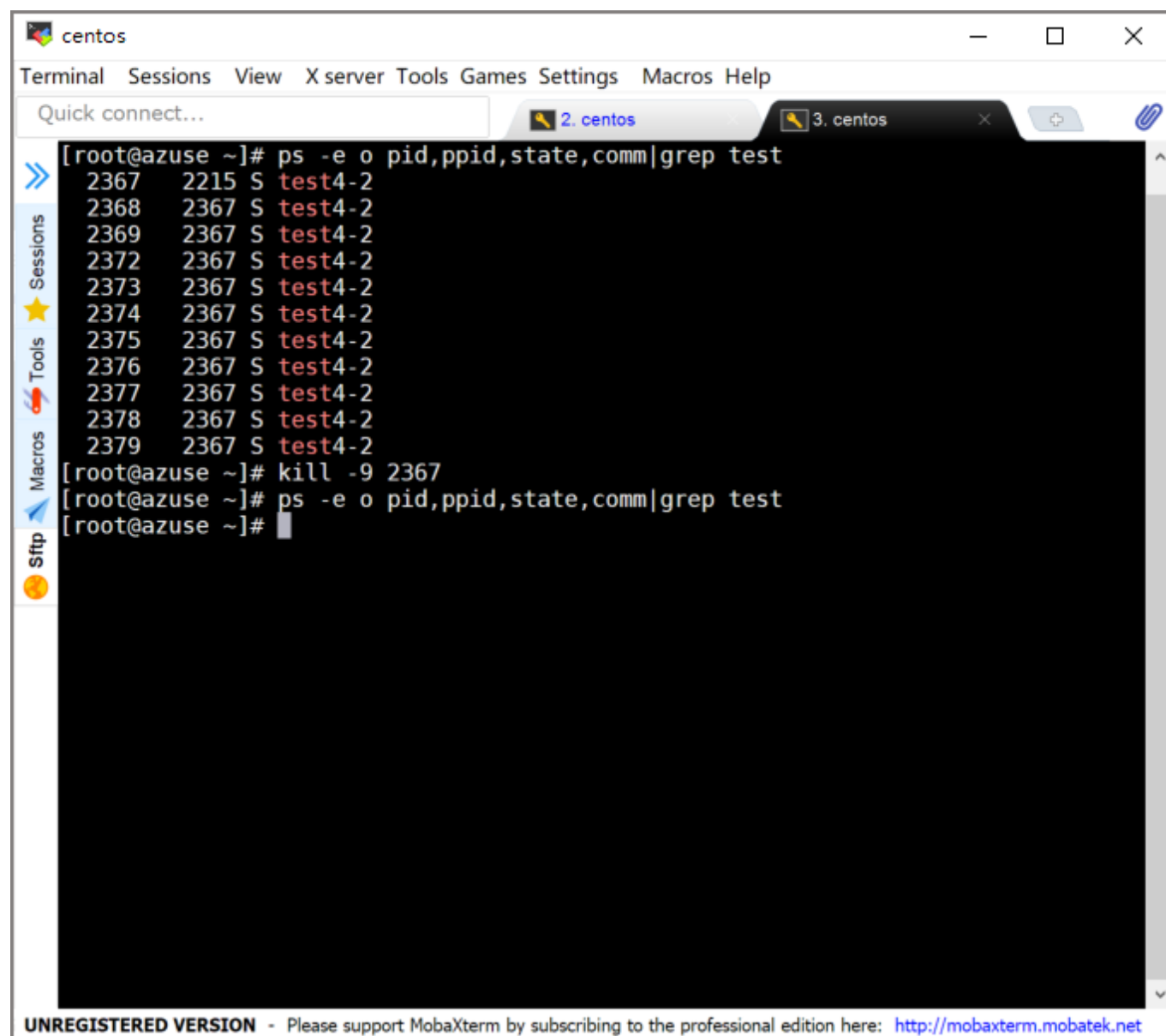
        if(pid == 0){
            printf("%d %d 1652238 sub\n",getpid(),getppid());
            fflush(stdout);
            sleep(15);
            counter++;
            // if(counter == 2){
            //     printf("sub %d exiting\n",getpid());
            //     fflush(stdout);
            //     break;
            // }
        }else{
            printf("%d %d 1652238 main\n",getpid(),getppid());
            fflush(stdout);
        }
    }
}
```

```
        sleep(5);
    }
};
return 0;
}

static void sig_child(int signo){
    pid_t pid;
    int stat;
    while((pid = waitpid(-1, &stat, WNOHANG)) > 0){
        printf("child %d exited with signal %d\n", pid, signo);
        fflush(stdout);
    }
}
```

测试结果:

kill -9 2367后子进程自动退出



The screenshot shows a MobaXterm terminal window with the following content:

```
[root@azuse ~]# ps -e o pid,ppid,state,comm|grep test
2367  2215 S test4-2
2368  2367 S test4-2
2369  2367 S test4-2
2372  2367 S test4-2
2373  2367 S test4-2
2374  2367 S test4-2
2375  2367 S test4-2
2376  2367 S test4-2
2377  2367 S test4-2
2378  2367 S test4-2
2379  2367 S test4-2
[root@azuse ~]# kill -9 2367
[root@azuse ~]# ps -e o pid,ppid,state,comm|grep test
[root@azuse ~]#
```

The terminal window has a sidebar with icons for Sessions, Tools, Macros, and Sftp. The top bar shows the window title 'centos' and standard window controls. The bottom of the window displays a message: 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>'.