守护进程的编写和使用方法 05

守护进程再次分裂子进程，极限测试

00 test5-1编写，可以分裂参数个进程

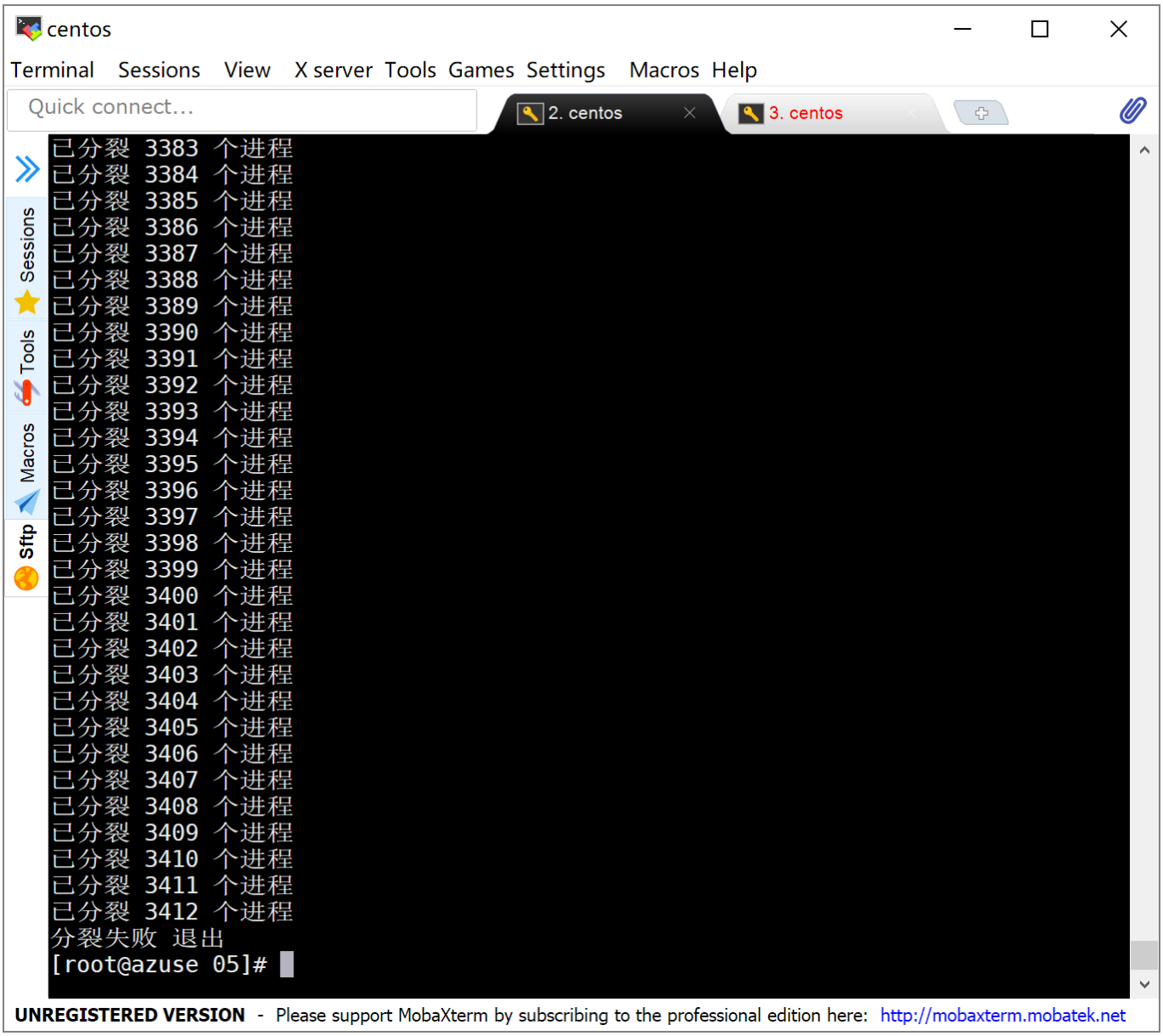
```
// test5-1.c
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <errno.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/prctl.h>

static void sig_child(int signo);

int main(int argc, char* argv[]){
        int pid;
        int i;
        if(argc == 1){
                printf("缺少参数 退出\n");
                exit(0);
        }
        int times = atoi(argv[1]);
        printf("循环生成%d个子进程\n",times);
        fflush(stdout);
        signal(SIGCHLD,sig_child);
        for(i=0;i<times;i++){
                pid = fork();
                if(pid == 0){
                        prctl(PR_SET_PDEATHSIG, SIGHUP);
                        break;
                }
                else{
                        printf("已分裂 %d 个进程\n",i+1);
                        fflush(stdout);
                }
        }
        int counter = 0;
        while(1){

                if(pid == 0){
                        char str[1024];
                        int j;
                        for(j=0;j<1024;j++)
                                str[j]='0';
                        while(1)
                                sleep(1);
```

```
                        //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);
        }
}
```

01 512MB 1024MB 2048MB内存，分裂数量测试

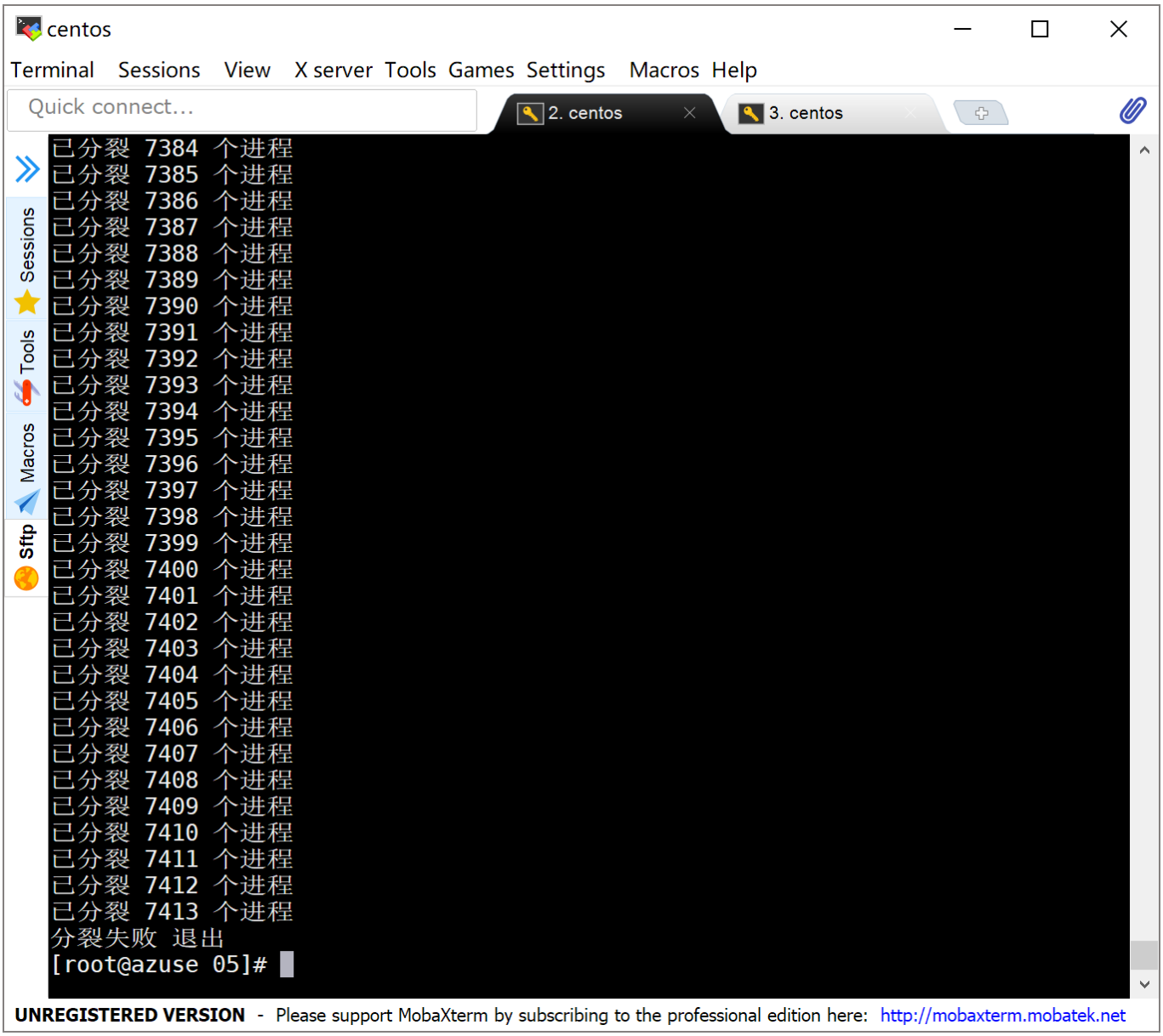
512MB内存：

3412个子进程



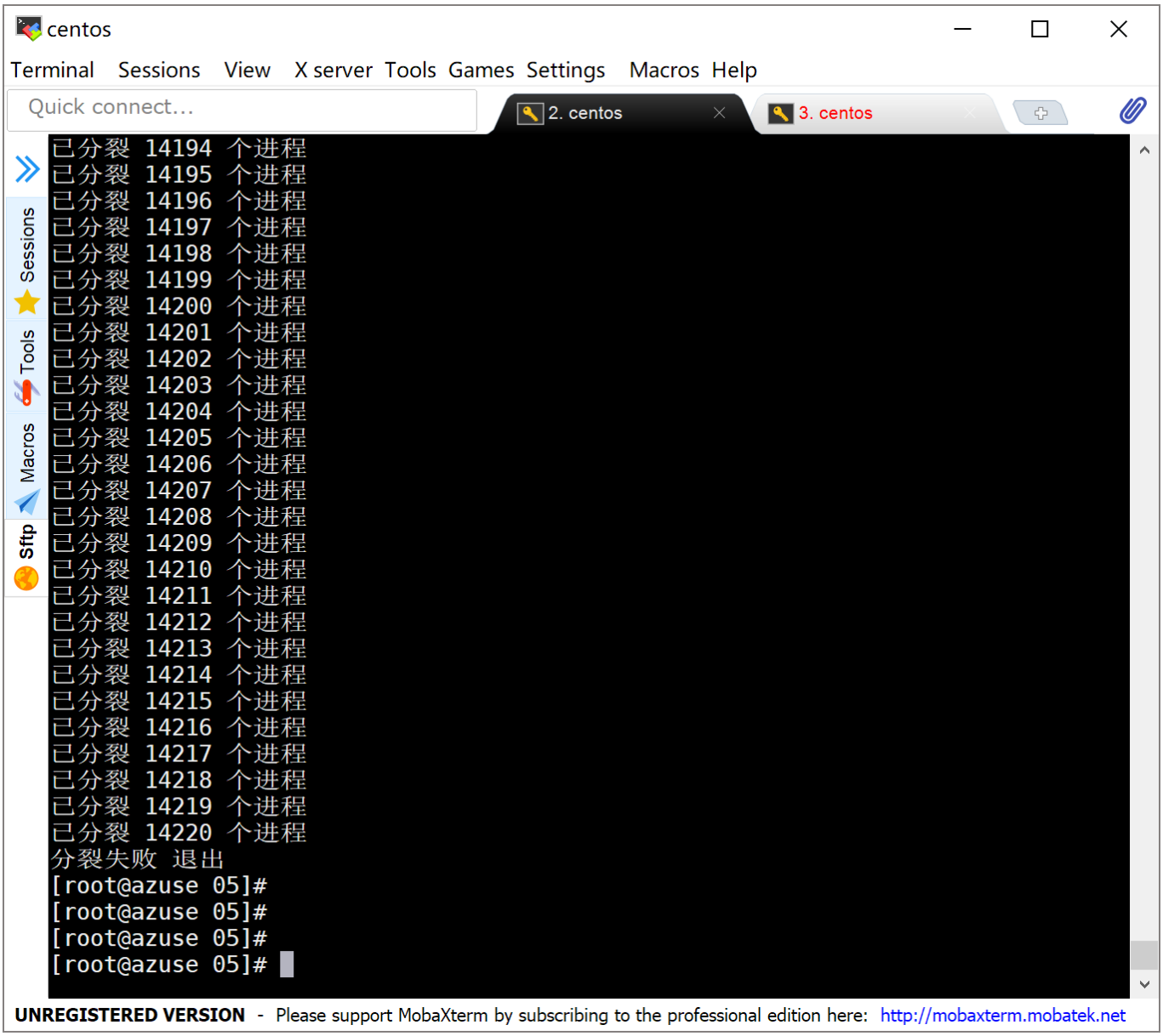
1024MB内存:

7413个子进程



20148MB内存：

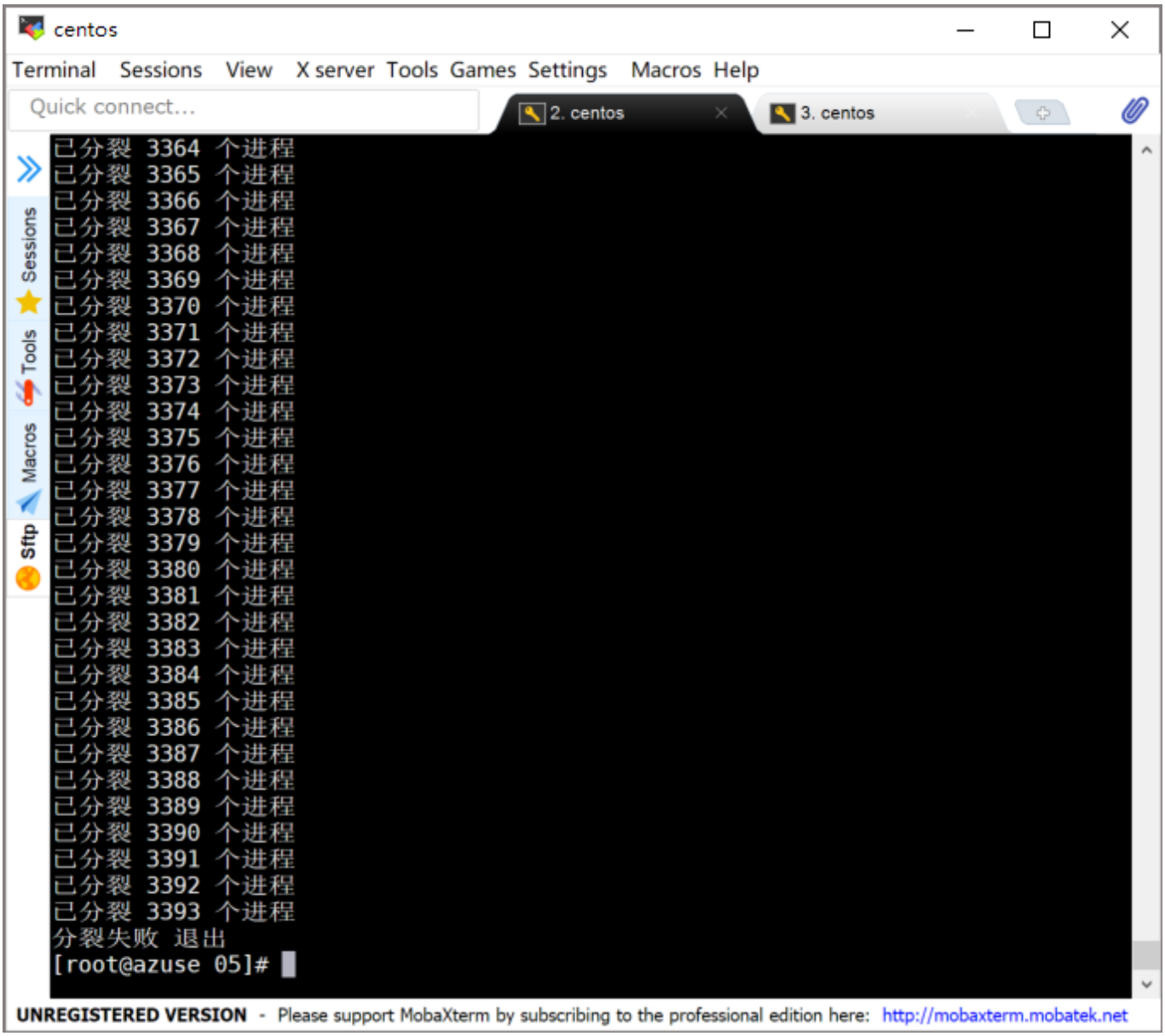
14220个子进程



02 char str[1024*10]的场合

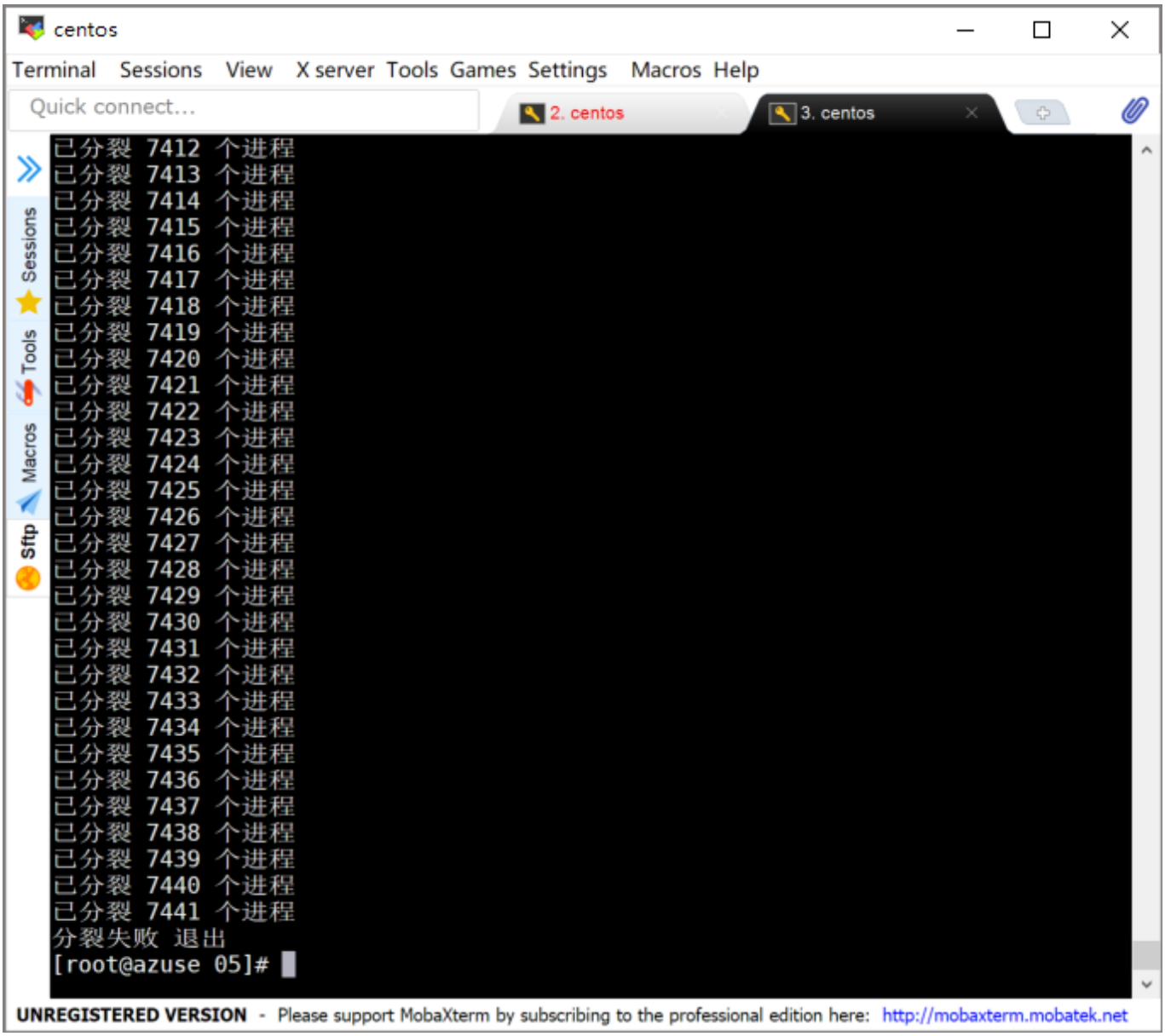
512MB

3393个子进程



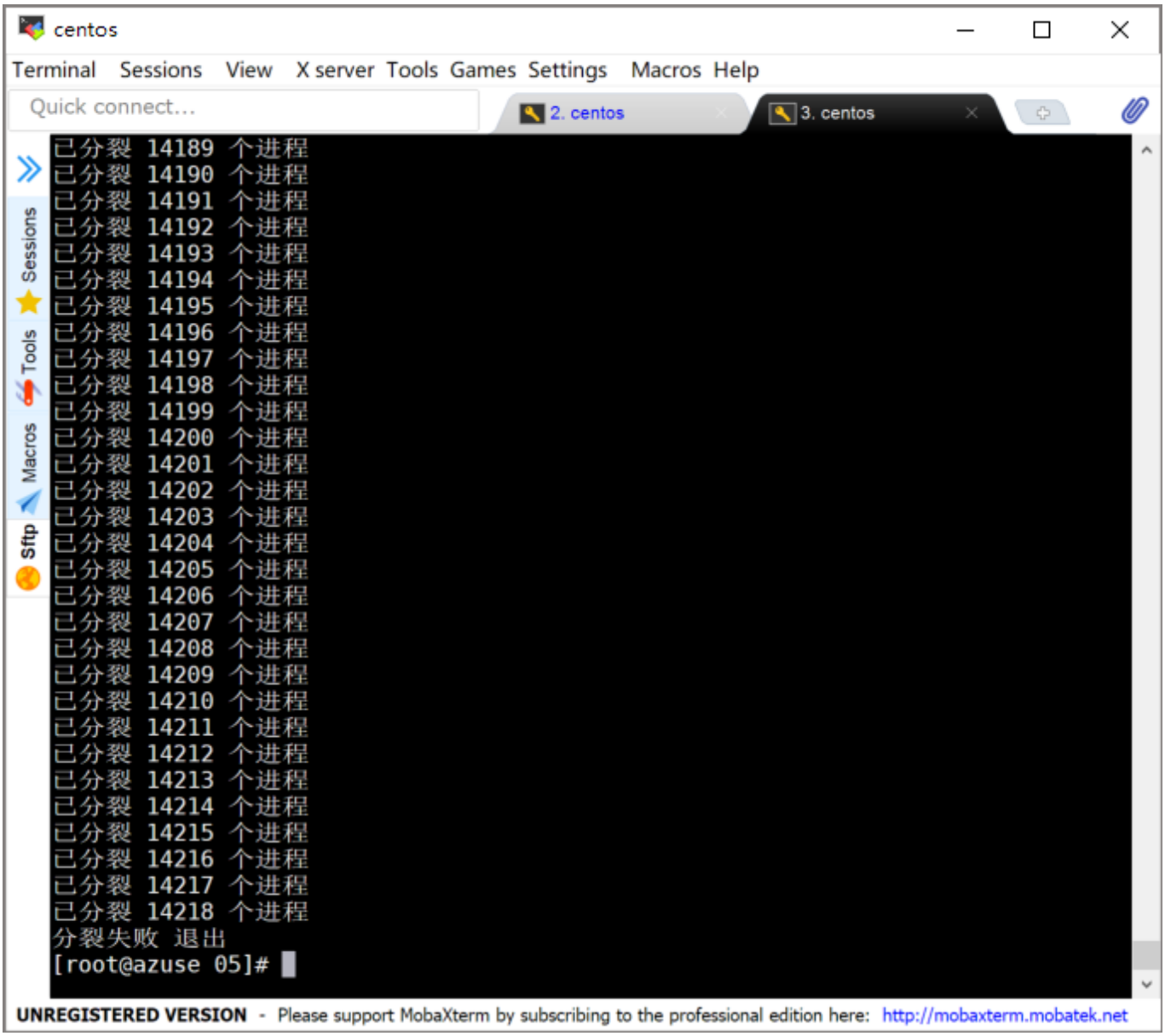
1024MB

7441个子进程



2048MB

14218个子进程



03 test5-2编写，分裂超过10w个子进程

```c
// test5-2.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 argc, char* argv[]){
        int pid;
        int pid_max = 1;
        long int i;
        if(argc == 1){
                printf("缺少参数 退出\n");
```

```
                exit(0);
        }
        int times = atoi(argv[1]);
        if(times == -1)
                printf("循环生成无限个子进程\n",times);
        else
                printf("循环生成%d个子进程\n",times);

        fflush(stdout);
        signal(SIGCHLD,sig_child);
        for(i=0;i<times||times==-1;i++){
                pid = fork();
                if(pid > pid_max)
                        pid_max = pid;
                if(pid == 0){
                        prctl(PR_SET_PDEATHSIG, SIGHUP);
                        break;
                }
                else if(pid == -1){
                        printf("无法分裂 等待\n");
                        fflush(stdout);
                        sleep(1);
                        //return 0;
                }
                else{
                        printf("已分裂 %d 个进程\n",i+1);
                        fflush(stdout);
                }
        }
        int counter = 0;
        while(1){

                if(pid == 0){
                        char str[1024];
                        int j;
                        for(j=0;j<1024;j++)
                                str[j]='0';
                        sleep(1);
                        return 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("pid=%d ppid=%d 1652238 main biggest pid is
%d\n",getpid(),getppid(),pid_max);
                        fflush(stdout);
                        sleep(5);
```

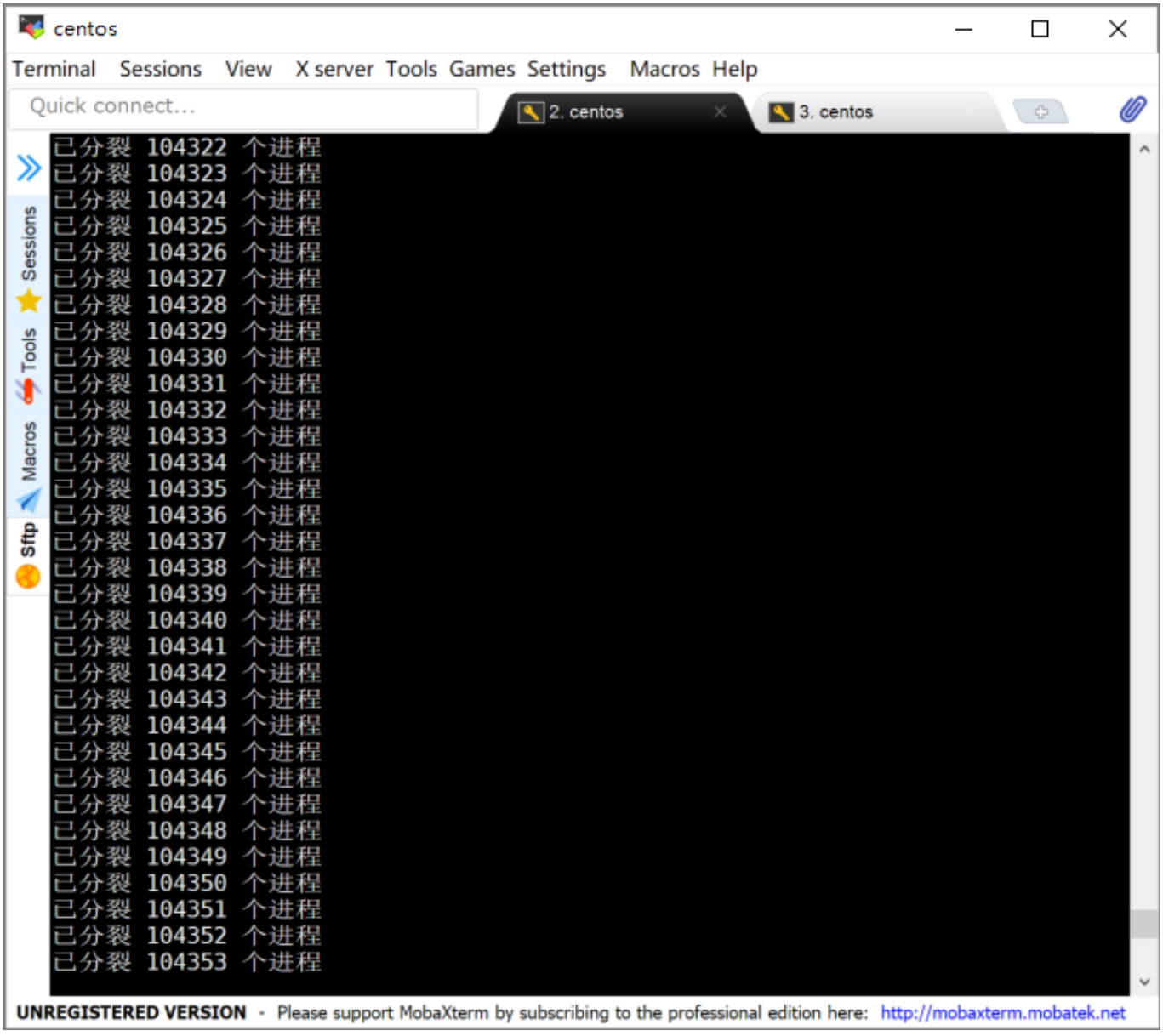
```
                }
        };
        return 0;
}

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

分裂测试:

```
./test5-2 -1
```

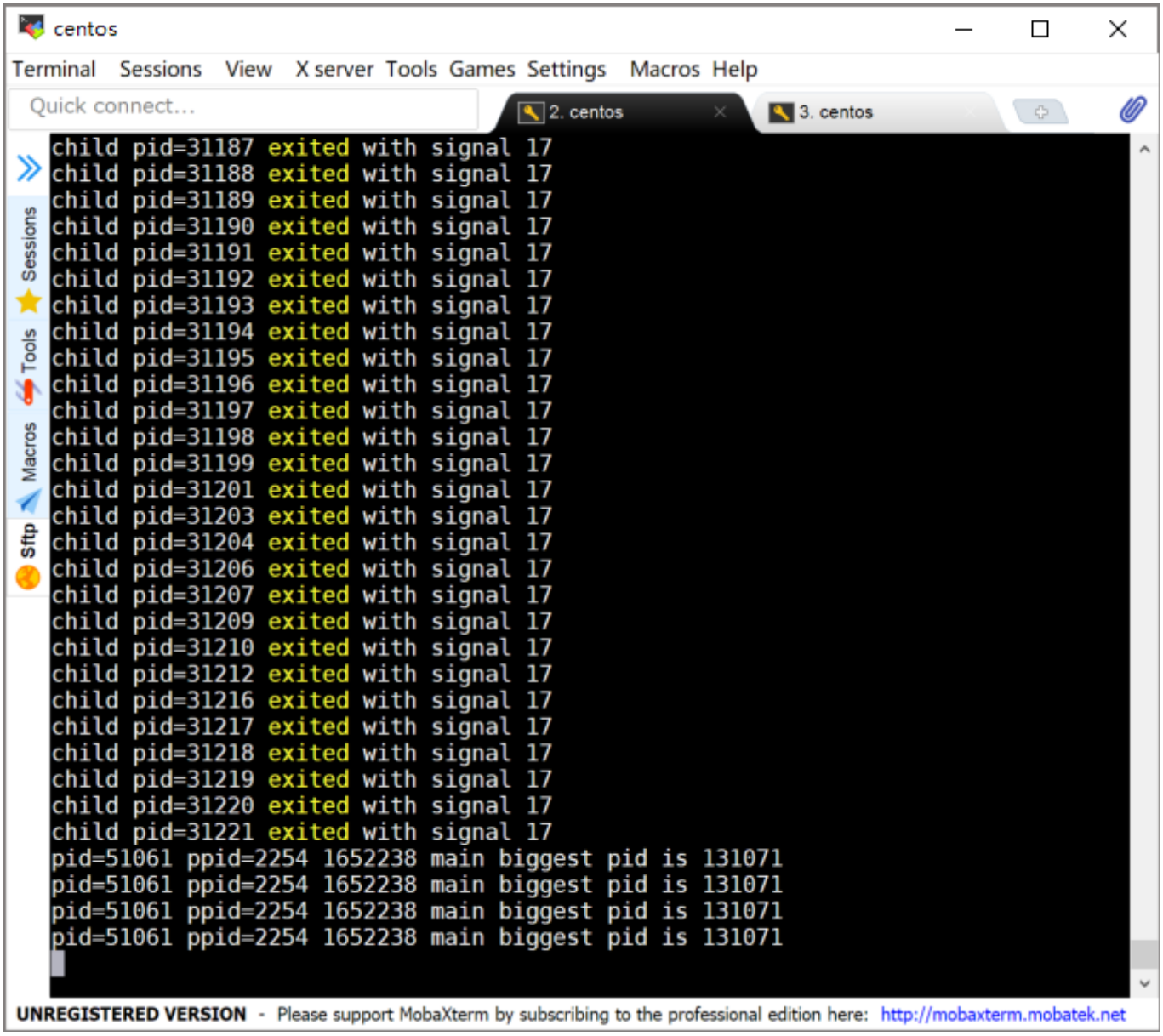
结果:

在512MB机器上可以超过10w个



04 分裂子进程的最大进程号是多少

分裂110000个进程测试出最大子进程号是131071



05 编写test5-3.c，记录生成的子进程数和回收的子进程相同

```c
// test5-3.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 pid_created = 0;
int pid_extracted = 0;
```

```
int main(int argc, char* argv[]){
        int pid;
        int pid_max = 1;
        long int i;
        if(argc == 1){
                printf("缺少参数 退出\n");
                exit(0);
        }
        int times = atoi(argv[1]);
        if(times == -1)
                printf("循环生成无限个子进程\n",times);
        else
                printf("循环生成%d个子进程\n",times);

        fflush(stdout);
        signal(SIGCHLD,sig_child);
        for(i=0;i<times||times==-1;i++){
                pid = fork();
                if(pid > pid_max)
                        pid_max = pid;
                if(pid == 0){
                        prctl(PR_SET_PDEATHSIG, SIGHUP);
                        break;
                }
                else if(pid == -1){
                        printf("无法分裂 等待\n");
                        fflush(stdout);
                        sleep(1);
                        //return 0;
                }
                else{
                        pid_created++;
                        printf("已分裂 %d 个进程\n",i+1);
                        fflush(stdout);
                }
        }
        int counter = 0;
        while(1){

                if(pid == 0){
                        char str[1024];
                        int j;
                        for(j=0;j<1024;j++)
                                str[j]='0';
                        sleep(1);
                        return 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("pid=%d ppid=%d 1652238 main biggest pid is
%d\n",getpid(),getppid(),pid_max);
                        fflush(stdout);
                        sleep(5);

                }
        };
        return 0;
}

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

测试200000个子进程

`created=200000, extracted=200000`生成的进程数等于回收的进程数，全部回收。