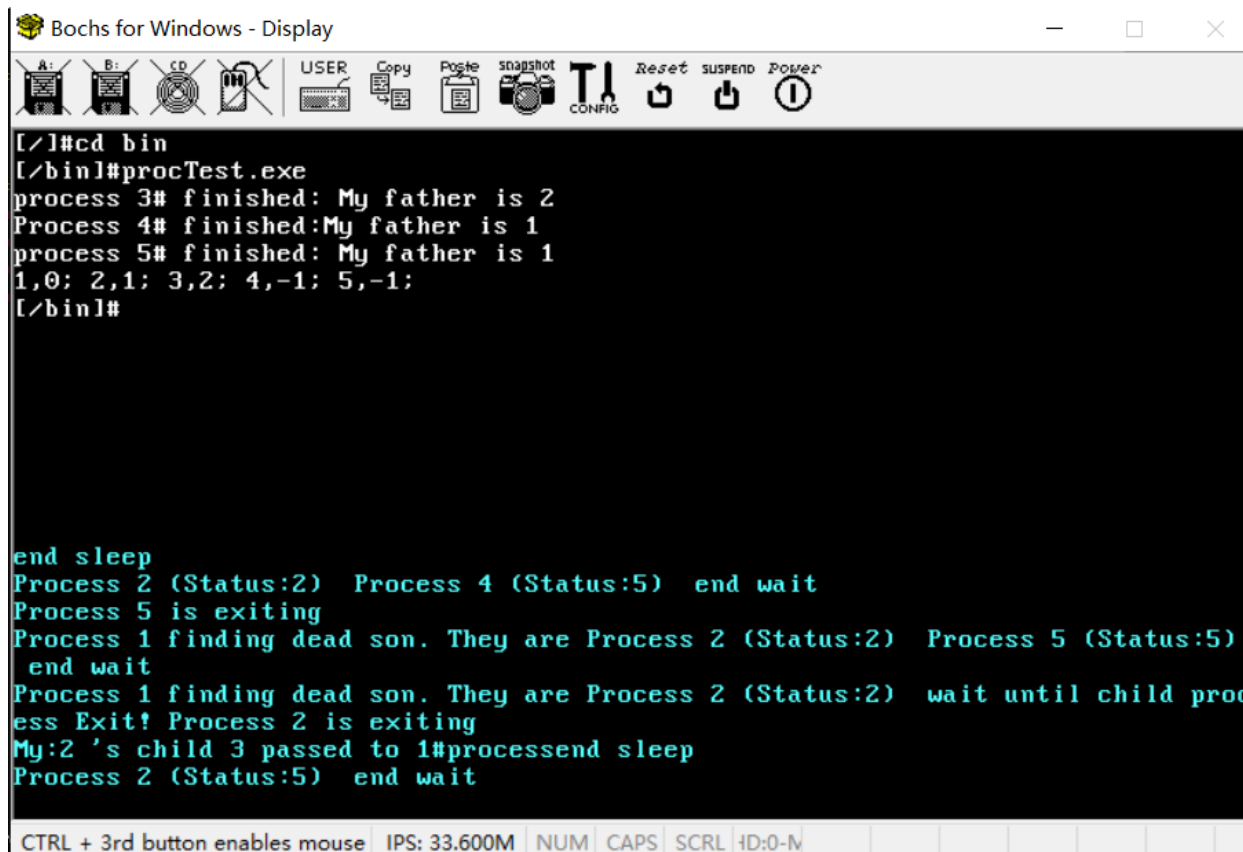



```

12         for(k=1;k<6;k++){
13             printf("%d,%d; ",k,getppid(k));
14         }
15         printf("\n");
16     }
17     else{
18         //3#
19         if(fork())
20         {
21             if(fork())
22             {
23                 //#3
24                 pid=getpid();
25                 ppid=getppid(pid);
26                 for(k=0;k<ws;k++)
27                 {
28                     i=wait(&j);
29                     printf ("process %d#:My child %d is finished with
exit status %d\n",pid, i, j) ;
30                 }
31                 printf("process %d# finished: My father is
%d\n",pid,ppid);
32                 exit(ppid);
33             }
34             else{
35                 //#5
36                 pid=getpid();
37                 ppid=getppid(pid);
38                 printf("process %d# finished: My father is
%d\n",pid,ppid);
39                 exit (ppid);
40             }
41         }
42     }
43     else{
44         //4#
45         pid=getpid();
46         ppid=getppid(pid);
47         printf("Process %d# finished:My father is %d\n",pid,ppid);
48         exit(ppid);
49     }
50 }
51 }
52

```

4.2父进程先于所有子进程结束



Bochs for Windows - Display

Process execution output:

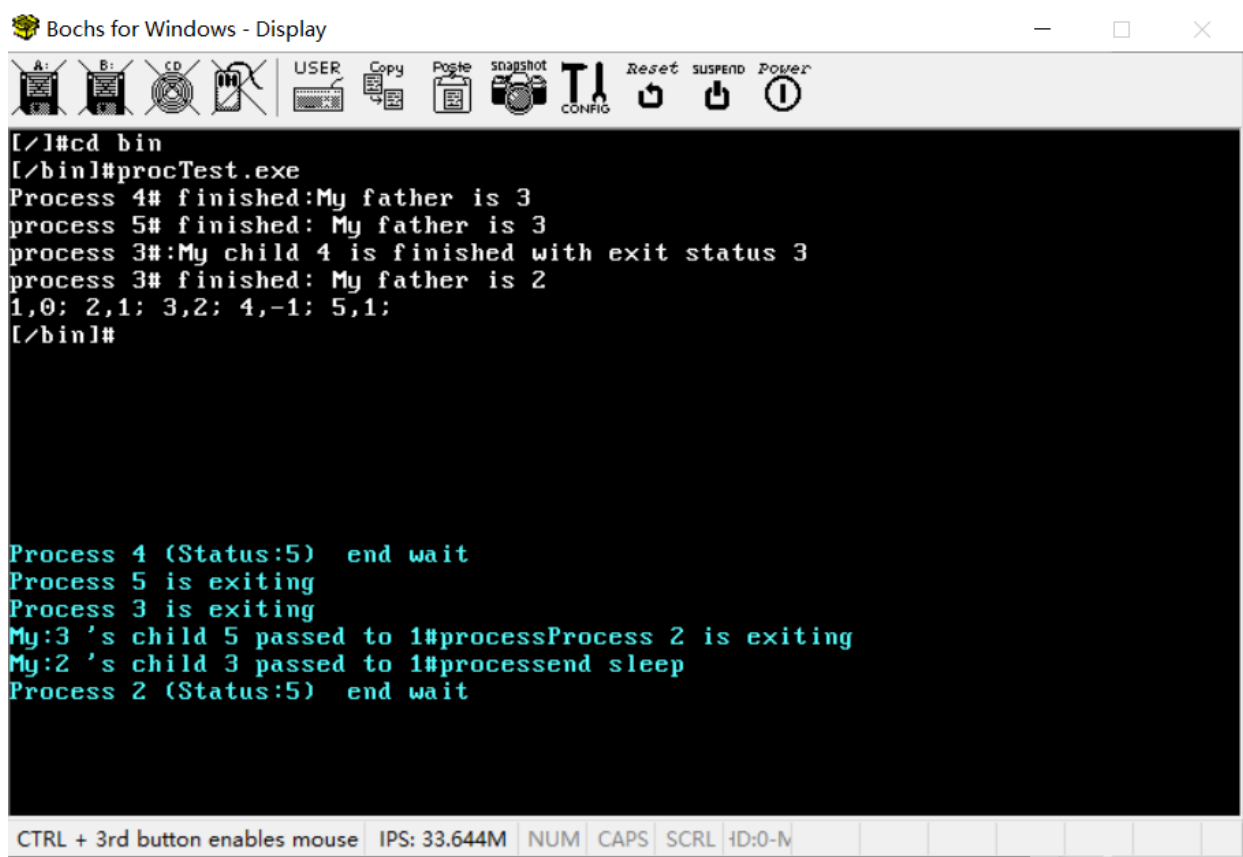
```
[/]#cd bin
[/bin]#procTest.exe
process 3# finished: My father is 2
Process 4# finished:My father is 1
process 5# finished: My father is 1
1,0; 2,1; 3,2; 4,-1; 5,-1;
[/bin]#

end sleep
Process 2 (Status:2) Process 4 (Status:5) end wait
Process 5 is exiting
Process 1 finding dead son. They are Process 2 (Status:2) Process 5 (Status:5)
end wait
Process 1 finding dead son. They are Process 2 (Status:2) wait until child proc
ess Exit! Process 2 is exiting
My:2 's child 3 passed to 1#processend sleep
Process 2 (Status:5) end wait
```

CTRL + 3rd button enables mouse IPS: 33.600M NUM CAPS SCRL ID:0-N

如果我们希望3#进程先结束，那么就不需要让3#进程执行wait等待4、5子进程，所以只需要把代码中的ws改成0，执行如上

4.3 父进程先于部分子进程结束



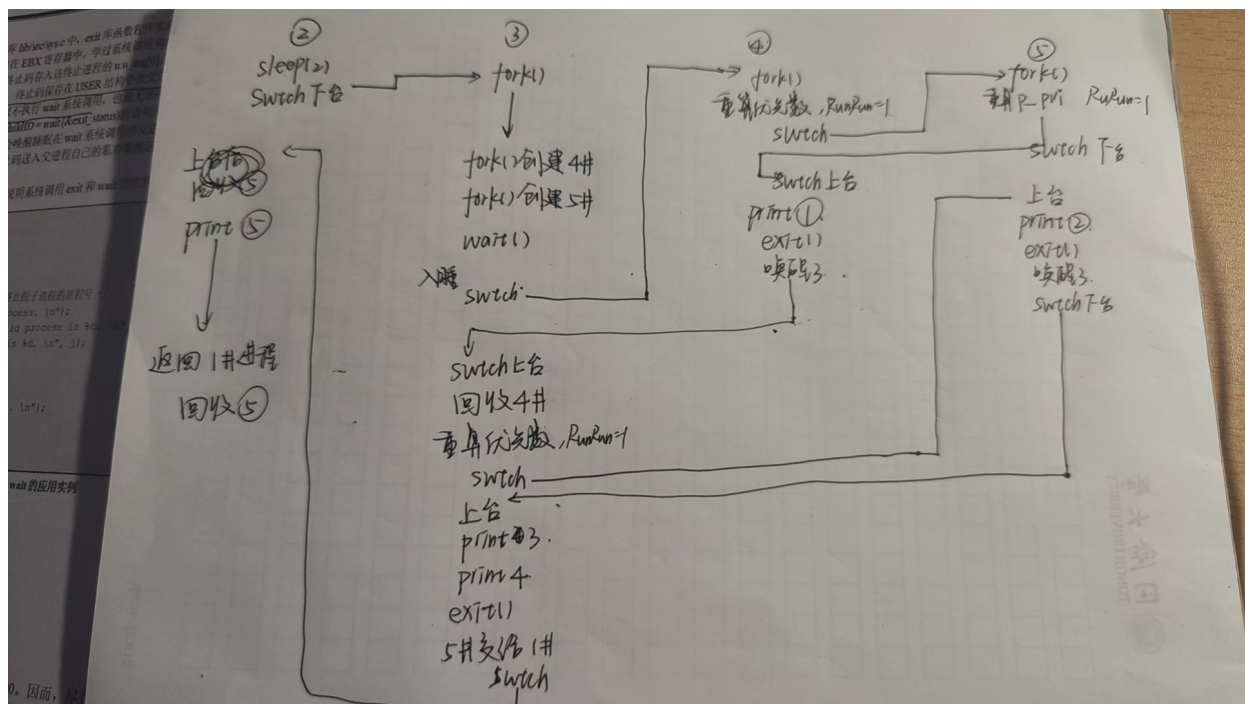
Bochs for Windows - Display

Process execution output:

```
[/]#cd bin
[/bin]#procTest.exe
Process 4# finished:My father is 3
process 5# finished: My father is 3
process 3#:My child 4 is finished with exit status 3
process 3# finished: My father is 2
1,0; 2,1; 3,2; 4,-1; 5,1;
[/bin]#

Process 4 (Status:5) end wait
Process 5 is exiting
Process 3 is exiting
My:3 's child 5 passed to 1#processProcess 2 is exiting
My:2 's child 3 passed to 1#processend sleep
Process 2 (Status:5) end wait
```

CTRL + 3rd button enables mouse IPS: 33.644M NUM CAPS SCRL ID:0-N



首先2#进程执行main1fork出3#进程，然后自己入睡2s，3#进程又fork出4、5进程并执行一次wait入睡，此时执行swtch，由于4、5优先数一样，根据id顺序所以4#先上台，在 fork 返回用户态前由于重算优先数被5#抢占。而5#在 fork 返回用户态前，同样由于重算优先数被4#抢占。然后4#执行print1打印自己的ID和父进程ID，终止自己并唤醒3进行回收，然后5#在3#执行wait返回用户态前抢占上台，pirnt2打印自己ID和父进程ID，结束自己，唤醒父进程3，可是没有执行wait等待，唤醒无效。

因为4、5进程中止时，3进程还没有中止。首先4进程结束时，3#进程未终止，并且由于父进程之前执行了wait等待子进程，此时4#唤醒父进程3#，父进程完整回收4#进程图像。而对于5#进程来说，父进程3#没有执行wait等待5#，所以5#直接print后中止，但是也会执行唤醒父进程操作，不过唤醒无效。此后会将4、5号进程的父进程改为1#，但是4#进程和5#进程终止时，父进程还是3。

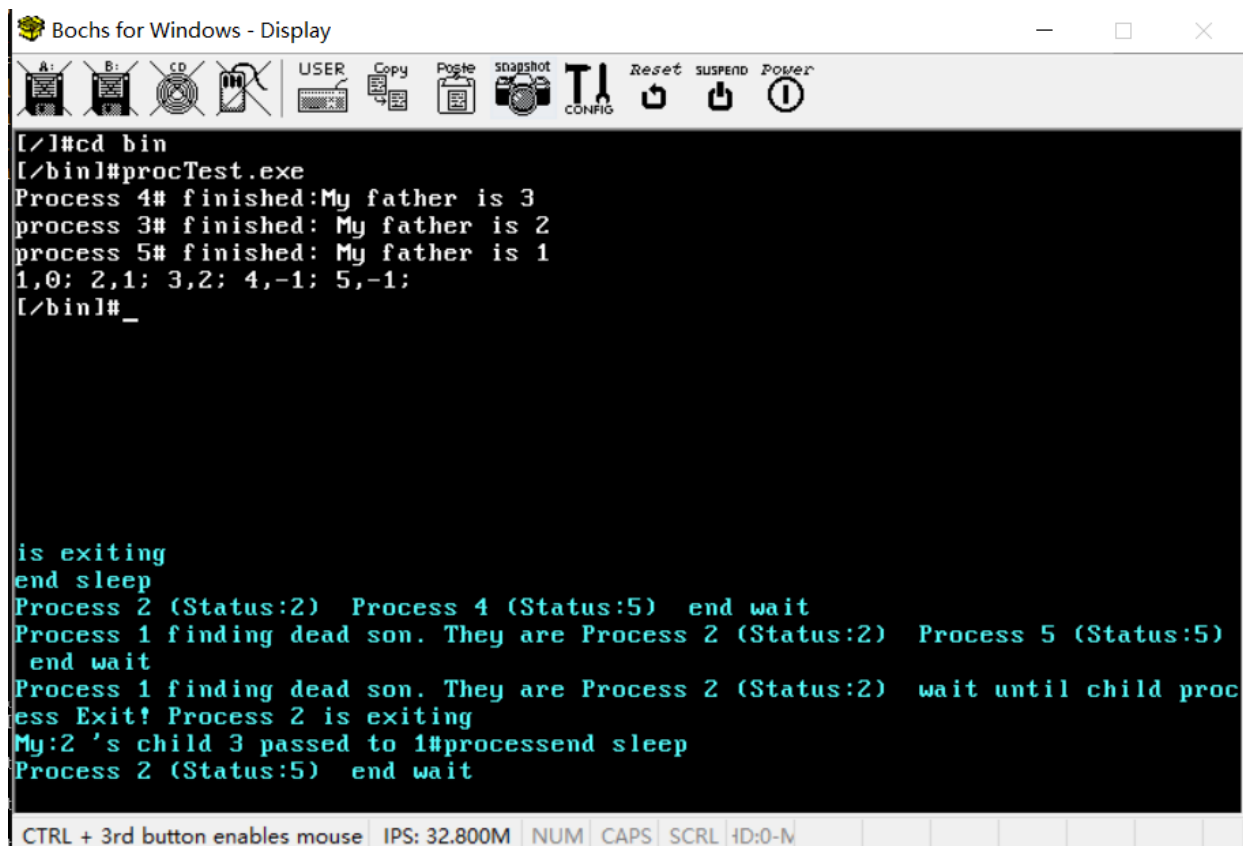
此后3#执行print3和4然后终止自己，唤醒父进程2#，并且把所有子进程（5#，因为此时4#已经被回收）的父进程改为1#，然后swtch下台，2#上台。

最后的打印输出是由2#进程执行的，此时3#进程已经中止，3#进程将没有回收的5#进程交给1#进程，等待1#进程处理。此时5#进程确实是存在的，是因为5#进程还没有被彻底回收，等待2#进程执行完成后返回用户态时1#进程上台，扫描到这个孤儿进程并回收。

4.4 抢占父进程

4.1-4.3的实验中，3#进程始终没有被抢占，因为直到创建完5#进程，一直执行核心态代码，而由于核心态不抢占策略，直到执行完fork（）代码，除非3#进程自己放弃处理器，子进程是没有机会抢占的。

4.4.1 父进程创建5#之前被抢占



The image shows a screenshot of the 'Bochs for Windows - Display' window. The window has a title bar with standard Windows window controls (minimize, maximize, close). Below the title bar is a toolbar with icons for A: drive, B: drive, CD-ROM, floppy disk, USER, Copy, Paste, Snapshot, CONFIG, Reset, Suspend, and Power. The main area is a black terminal window with white text. The text shows the execution of a program named 'procTest.exe' in the 'bin' directory. The output indicates that three processes (1, 2, and 5) have finished, each reporting their father's ID. The output also shows process 1 finding dead sons (processes 2 and 5) and waiting for them to finish. The output ends with 'Process 2 (Status:5) end wait'.

```
[/]#cd bin
[/bin]#procTest.exe
Process 4# finished:My father is 3
process 3# finished: My father is 2
process 5# finished: My father is 1
1,0: 2,1: 3,2: 4,-1: 5,-1:
[/bin]#_

is exiting
end sleep
Process 2 (Status:2) Process 4 (Status:5) end wait
Process 1 finding dead son. They are Process 2 (Status:2) Process 5 (Status:5)
end wait
Process 1 finding dead son. They are Process 2 (Status:2) wait until child proc
ess Exit! Process 2 is exiting
My:2 's child 3 passed to 1#processend sleep
Process 2 (Status:5) end wait
```

```
else{
    //3#
    for(t=1;t<1000000;t++)
    {
        double m=0.11111999*9.9999999*t/0.29999873;
    }
    if(fork())
    {
        if(fork())
        {
            //3#

            pid=getpid();
            ppid=getppid(pid);
            for(k=0;k<ws;k++)
            {
                i=wait(&j);
                printf ("process %d#:My child %d is finished with exit status %d\n",pid, i, j) ;
            }
            printf("process %d# finished: My father is %d\n",pid,ppid);
            exit(ppid);
        }
    }
}
```

要想3号进程在5进程创建之前被抢占，可以让3进程在创建4进程之前做大量的浮点运算，以占用处理机，降低优先级，以便在fork返回的例行调度中被抢占，这样3进程先被4进程抢占，然后4进程输出并终止，此时3进程是就绪状态，4进程唤醒无效，没有被完全回收，然后5进程被创建出来后，3号进程输出并结束，把自己的子进程4和5交给1进程，然后例行调度5进程上台，执行完代码后终止，唤醒1进程，1进程上台回收所有没有回收的进程（4,5）所以返回2进程的时候，4、5进程已经被回收，父进程id为-1

4.4.2父进程创新5#之后被抢占

```

d[/]#cd bin
[/bin]#procTest.exe
Process 4# finished: My father is 3
process 5# finished: My father is 3
process 3# finished: My father is 2
1,0; 2,1; 3,2; 4,1; 5,1;
[/bin]#

My:3 's child 4 passed to 1#process
My:3 's child 5 passed to 1#process
Process 2
is exiting
My:2 's child 3 passed to 1#process
end sleep
Process 2 (Status:5) end wait

```

CTRL + 3rd button enables mouse IPS: 32.800M NUM CAPS SCRL ID:0-N

```

else{
    //3#

    if(fork())
    {
        for(t=1;t<1000000;t++)
        {
            double m=0.11111999*9.9999999*t/0.29999873;
        }
        if(fork())
        {
            //3#

            pid=getpid();
            ppid=getppid(pid);
            for(k=0;k<ws;k++)
            {
                i=wait(&j);
                printf("process %d#:My child %d is finished with exit status %d\n",pid, i, j) ;
            }
            printf("process %d# finished: My father is %d\n",pid,ppid);
            exit(ppid);
        }
        else{

```

与上一部分同理，要想在创建5#之后抢占3#，只需要在fork5#之前让3#进程做大量运算，这样在fork5返回时，进行例行调度，3进程优先级很低，被4进程抢占，4执行完后终止，唤醒3进程无效，5进程上台执行完后终止，唤醒3进程也无效，最后3进程执行完终止，将45进程交给1#进程处理，所以返回2#进程时，45进程还没有被1#进程回收，父进程为1。