1.进程创建：

int main()

{

pid\_t pid;

int i;

for (i=0; i<3; i++)

{

/\* fork another process \*/

pid = fork();

if (pid < 0) { /\*error occurred \*/

fprintf(stderr, “Fork Failed”);

exit(-1);

}

else if (pid == 0) { /\* child process \*/

fprintf(stdout, “i=%d, pid=%d, parent pid=%d\n”,i,

getpid() ,getppid());

}

}

wait(NULL);

exit(0);

}

请问该程序最终一共生成几个进程？假设当前进程PID为1，生成的进程PID依次加1，请将生成进程关系图画出来。

i = =0时，pid==1的进程拷贝出pid==2的子进程。查阅资料的时候发现，父进程先运行还是子进程线运行取决于具体的操作系统。这里不妨先设父进程优于子进程运行。

pid==1中，i==1时，拷贝出子进程pid==3；i ==2时，拷贝出子进程pid==4，随后进入wait队列。

pid==4开始执行，因为i==3，所以没有fork，直接打印了pid==4的信息。随后退出。

pid==3开始执行，i==2时拷贝出pid==5的子子进程，随后退出。

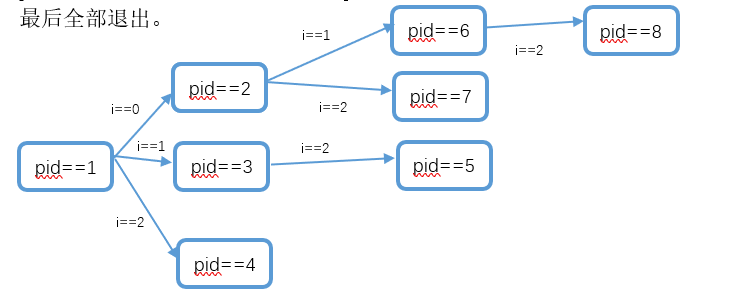
pid==2开始执行，i==1时拷贝出pid==6的子子进程，i==2时拷贝出pid==7的子子进程，随后进入wait队列。

pid==5开始执行，因为i==3，所以没有fork，直接打印了pid==5的信息。随后退出。

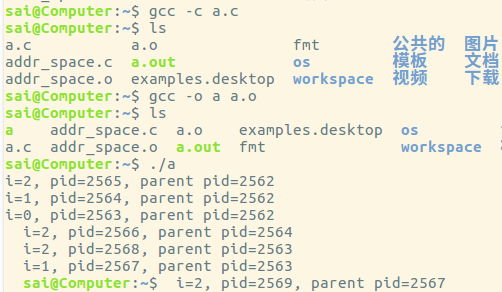
pid=**=**7开始执行，因为i==3，所以没有fork，直接打印了pid==7的信息。随后退出。

pid==6开始执行，i==2时拷贝出pid==8的子子进程，随后退出。

pid=**=**8开始执行，因为i==3，所以没有fork，直接打印了pid==8的信息。随后退出。



打印结果：



Modern Operating Systems, 3th Edition, Tanenbaum and Bos

(现代操作系统 第3版) 课后习题：

36. Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5, and X. In what order should they be run to minimize average response time? (Your answer will depend on X.)

A: Order will be decided by the value of X

if(X < 3): X, 3, 5, 6, 9

if(3 < X <= 5): 3, X, 5, 6, 9

if(5 < X <= 6): 3, 5, X, 6, 9

if(6 < X <= 9): 3, 5, 6, X, 9

else 3, 5, 6, 9, X

37. Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process **turn-around time**. Ignore process switching overhead.

(a) Round robin.

(b) Priority scheduling

(c) First-come, first-served (run in order 10, 6, 2, 4, 8).

(d) Shortest job first.

For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d) assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

A:

1. During the first 10 minutes, each task gets 2 minutes. At the end of the 10 minutes, task C is completed. Next 8 minutes, each task get 2 minutes, and task D is completed. Another 6 minutes, each task get 2 minutes, and taks B will be completed. Next 4 minutes, task E is completed and another 2 minutes later, all tasks are completed.

Task A: 30 minutes Task B: 24 minutes Task C: 10 minutes Task D: 18 minutes Task E: 28 minutes

Average: 22 minutes

1. B -> E -> A -> C -> D

Task A: 24 minutes Task B: 6 minutes Task C: 26 minutes Task D: 30 minutes Task E: 14 minutes

Average: 18.8 minutes

1. Task A: 10 minutes Task B: 16 minutes Task C: 18 minutes Task D: 22 minutes Task E: 30 minutes

Average: 19.2 minutes

d) C -> D -> B -> E -> A

Task A: 2 minutes Task B: 6 minutes Task C: 12 minutes Task D: 20 minutes Task E: 30 minutes

Average: 14 minutes

43. A real-time system needs to handle two voice calls that each run every 5 msec and consume 1 msec of CPU time per burst, plus one video at 24 frames/sec, with each frame requiring 20 msec of CPU time. Is this system schedulable?

A: Each voice call will be run 200 times per second. That means it will costs 200 msec per second. Two voice calls stand for 400 msec per second.

Video run 24 times per second, and it uses 480 msec per second.

They total use 880 mesc per second, so that is okay.