**进程的创建和终止**

同济大学计算机系 操作系统作业 2023-12-7

学号 姓名

四、下面的这个程序会输出几个整数？请写出程序的输出，并请在代码中标出父进程执行的所有语句和子进程执行的所有语句。

L1: #include <stdio.h>

L2: void main (void )

L3: {

L4: int i = 10, x ;

L5: if ( x = fork() )

L6: {

L7: i += 10；

L8: printf ( “%d\t”, i );

L9: }

L10: else

L11: printf ( “%d\t”, i );

L12: printf ( “%d\t”, i );

L13: }

参考答案：这个程序会输出4个整数。这是因为语句L12执行了2次。

父进程输出2个20，子进程输出两个10。

程序的输出：

可能是： 10 20 10 20

（子进程输出1行，父进程输出1行，子进程再输出1行，父进程再输出1行）

可能是： 10 20 20 10

（子进程输出1行，父进程输出2行，子进程再输出1行）

还可能是： 20 10 20 10

（父进程输出1行，子进程输出1行，父进程再输出1行，子进程再输出1行）

或： 20 10 10 20

（父进程输出1行，子进程输出2行，父进程再输出1行）

或： 20 20 10 10

（父进程输出2行，子进程输出2行）

或： 10 10 20 20

（子进程输出2行，父进程输出2行）

最后2种情况是常见的。

其余非常罕见。可能的原因是，这个应用程序是和其它应用程序并发执行的，进程有可能输出一条语句后因响应中断放弃CPU。

# 习题部分：

## 阅读程序，回答问题

代码 1.1。假定父进程的PID是 007，子进程的PID是008。写出程序的输出。

#include <stdio.h>

#include <sys.h>

main( )

{

int i=10, j =20;

if( i=fork( ) )

{

printf("It is parent process. PID = %d, i = %d\n",getpid( ), i);

i=wait(&j);

printf("The finished child process is %d. \n", i);

printf("The exit status is %d. \n", j);

}

else

{

printf("It is child process. PID = %d, i = %d\n",getpid( ), i);

exit(1);

}

}

有两种情况：

第一种：父进程先上台运行，然后入睡，子进程上台

It is parent process. PID = 7, i = 8

It is child process. PID =8, i = 0

The finished child process is 8.

The exit status is 256.

第二种：子进程先上台运行

It is child process. PID =8, i = 0

It is parent process. PID = 7, i = 8

The finished child process is 8.

The exit status is 256.

T0时刻，子进程终止。终止后，子进程的PCB何时回收，由谁来回收。

T0时刻，父进程007回收子进程008的PCB。

代码 1.2。假定父进程的PID是 007，子进程的PID是008。

#include <stdio.h>

#include <sys.h>

main( )

{

int i=10, j =20;

if( i=fork( ) )

{

printf("It is parent process. PID = %d, i = %d\n",getpid( ), i);

}

else

{

sleep(100);

printf("It is child process. PID = %d, i = %d\n",getpid( ), i);

exit(1);

}

}

1. 写出程序的输出。

It is parent process. PID = 7, i = 8

It is child process. PID = 8, i = 0（100s后）

1. T0时刻，父进程创建子进程。子进程何时终止？终止后，子进程的PCB何时回收，由谁来回收。

子进程的终止时刻 T0+100s。

子进程终止后，唤醒1#进程。PCB很快由1#进程回收（这是因为父进程007很早就终止了。007终止的时候，将子进程008 的 ppid 改成了 1）。

代码 1.3。假定父进程的PID是 007，子进程的PID是008。

#include <stdio.h>

#include <sys.h>

main( )

{

int i=10, j =20;

if( i=fork( ) )

{

printf("It is parent process. PID = %d, i = %d\n",getpid( ), i);

sleep(100);

}

else

{

printf("It is child process. PID = %d, i = %d\n",getpid( ), i);

exit(1);

}

}

1. 写出程序的输出。

输出分为两种情况：

第一种：父进程先上台，sleep入睡，子进程上台。

It is parent process. PID = 7, i = 8

It is child process. PID = 8, i = 0

第二种：子进程先上台，终止，父进程上台。

It is child process. PID = 8, i = 0

It is parent process. PID = 7, i = 8

1. T0时刻，父进程创建子进程。printf耗时忽略。子进程的PCB何时回收，由谁来回收。

子进程终止的时候，exit 叫不醒父进程\*。所以，无论哪种情况，子进程的PCB一定要等100s以后，父进程终止时，由1#进程回收。

注：

exit。wakeupall（父进程008的process对象的起始地址）

父进程008 sleep入睡，睡在 闹钟变量tout上

代码 1.4。假定父进程的PID是 007，第一个子进程的PID是008，第二个子进程的PID是009。

#include <stdio.h>

#include <sys.h>

main( )

{

int i=10, j =20;

if( i=fork( ) )

{

printf("It is parent process. PID = %d, First Son: %d\n", getpid( ), i);

if( i=fork( ) ) {

printf("It is parent process. PID = %d, Second Son: %d\n", getpid( ), i);

i = wait( &j );

printf("Exit Son: %d. Exit Status= %d\n", i, j);

}

else {

printf("It is child process. PID = %d, i = %d\n", getpid( ), i);

exit( 2 );

}

}

else

{

sleep(100);

printf("It is child process. PID = %d, i = %d\n", getpid( ), i);

exit(1);

}

}

1. 写出程序的输出。

第1行： It is parent process. PID = 7, First Son: 8

第2行： It is parent process. PID = 7, Second Son: 9

第3行： It is child process. PID = 9, i = 0

第4行： Exit Son: 9. Exit Status= 2

第5行： It is child process. PID = 8, i = 0（100s后输出）

注，输出序列中，第2行、第3行次序有可能颠倒。

1. T0时刻，父进程创建子进程。printf耗时忽略。子进程的PCB何时回收，由谁来回收。

子进程009的PCB，父进程007回收。T0时刻（之后的一小会）回收。

子进程008的PCB，1#进程回收。T0+100s回收。

（父进程007终止时，将未终止子进程的ppid改成1，子进程008终止的时候，1#进程回收其PCB）。

代码1.5 执行这个程序，系统需要使用几个进程？画与这个应用程序执行相关的进程树。

L1: #include <stdio.h>

L2: void main(void)

L3: { int i;

L4: printf ("%d %d \n", getpid ( ), getppid ( ) );

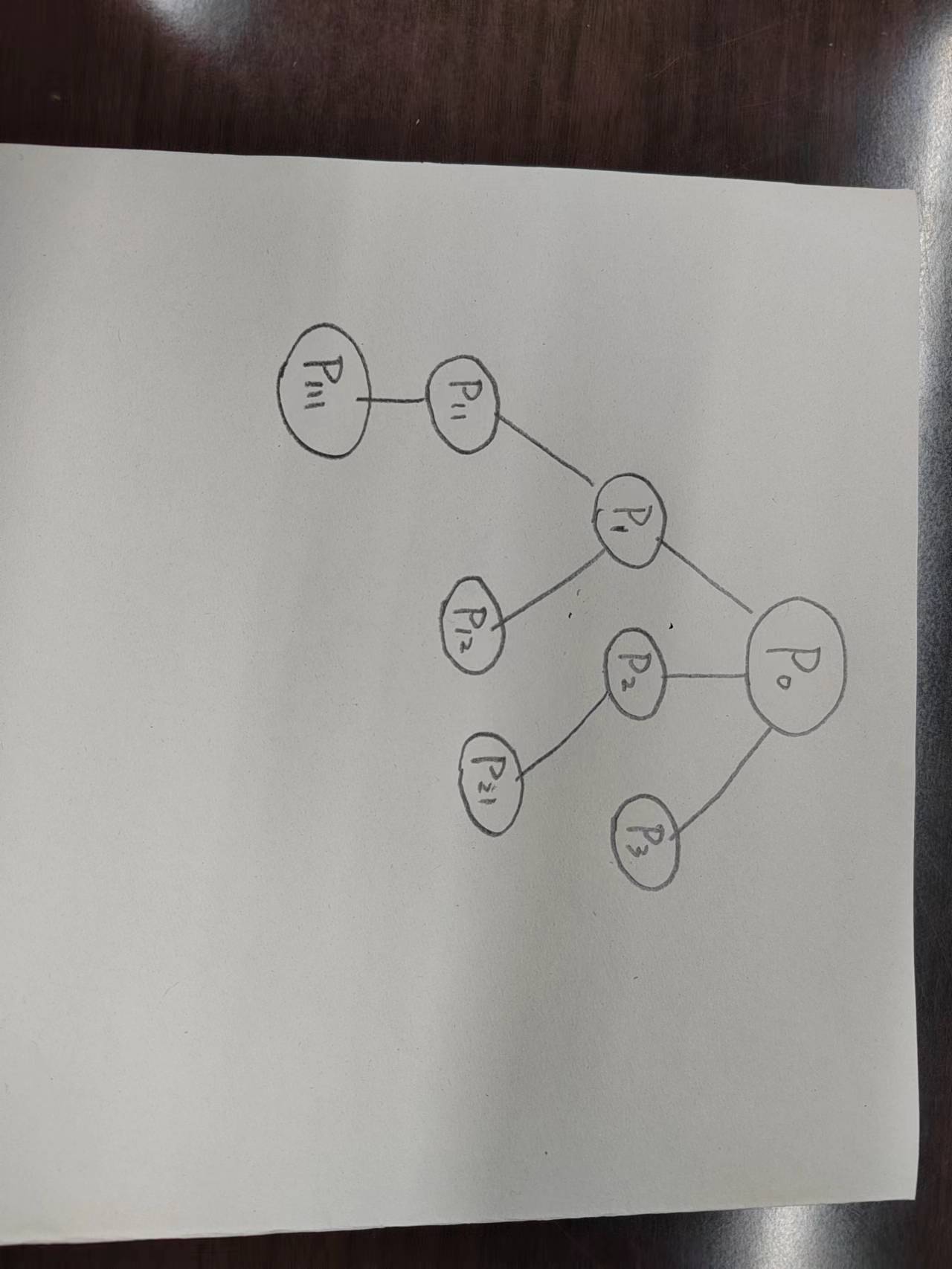
L5: for (i = 0; i < 3; ++i)

L6: if ( fork( ) == 0 )

L7: printf ("%d %d \n", getpid ( ), getppid ( ) );

L8: }

答：执行这个程序，系统需要使用8个进程。以Unix V6++系统为例，相关进程树如图所示：



第一个进程p0，从main函数入口开始执行，for循环3次，创建3个子进程p1,p2,p3。

子进程复制父进程图像，继承 i 变量的值。

p1，i变量初值是0，加1变1。p2，i变量初值是1，加1变2。p3，i变量初值是2，加1变3。只要i变量值小于3，子进程还会创建子进程。所以，p1会创建2个子进程p11,p12。P2会创建1个子进程p21。其中，p11还会再创建1个子进程p111。

为表达清晰，以进程名代pid号，程序输出如下：

（1）如果子进程输出前，父进程未终止

p0 1(shell进程的pid号)

p1 p0

p2 p0

p3 p0

p11 p1

p12 p1

p21 p2

p111 p11

（2）如果子进程输出前，父进程已终止。输出的ppid#会是1，不准了。

注，上面的8行输出，除第一行一定先输出外，其余各行，输出顺序是不确定的，依赖于哪个进程先执行。另外，本例子进程创建不会导致进程入睡，所有进程会接连创建自己的所有子进程，之后终止。所以，子进程输出的ppid，很多是1。

继续思考

* 怎样防止子进程输出ppid=1？
* 将程序输出重定向至磁盘文件，输出会变为20行。解释这个现象。

提示：屏幕输出是行缓存的，碰到回车就输出，写进tty输出缓存。磁盘文件是块缓存的（4096字节），不写满不执行write系统调用，数据还在进程的用户空间。