**操 作 系 统**

**实 验 报 告**

实验名称：**实验一 进程的创建实验**

姓名：**陈亚楠**

学号：**16340041**

**实验名称：进程的创建实验**

**一、实验目的：**

**1.** 加深对进程概念的理解，明确进程和程序的区别，进一步认识并发执行的实质。

**2.** 认识进程生成的过程，学会使用fork( )生成子进程，并知道如何使子进程完成与父进程不同的工作。

**二、实验要求：**

**1.** Linux下编辑程序；

**2.** Linux下编译和运行程序；

**3.** Linux下编译和调试程序：断点设置、断点管理、gdb应用。

**三、实验过程：**

**1.** 将下面的程序编译运行，并解释现象：  
 #include < sys/types.h >  
 #include < stdio.h >  
 #include < unistd.h >

int main( ) {

int pid1 = fork( );

printf ( “\*\*1\*\*\n” );

int pid2 = fork( );

printf ( “\*\*2\*\*\n” );

if ( pid1==0 ) {

int pid3 = fork( );

printf ( “\*\*3\*\*\n” );

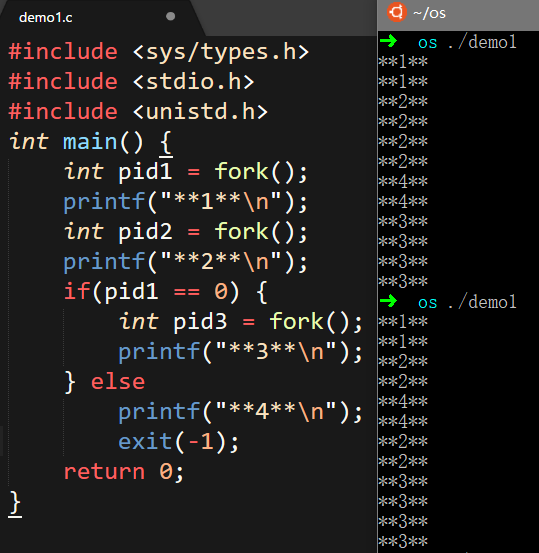
} else

printf ( “\*\*4\*\*\n” );

return 0;

}

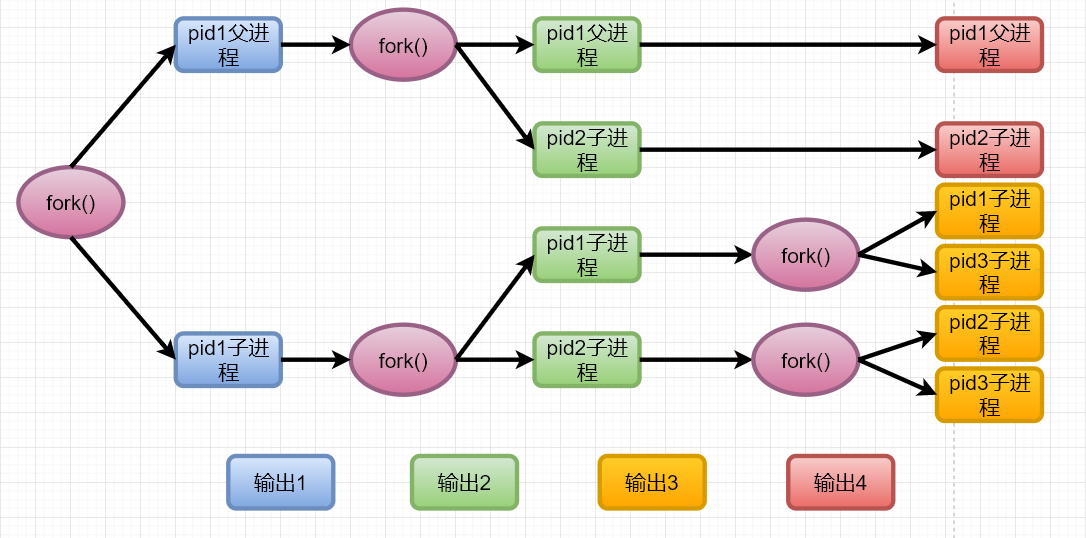
1. **实验结果：**



1. **结果分析：**

本例中，系统调用fork()后，进程没有使用系统调用exec()，父进程与子进程并发执行，子进程的地址空间复制父进程，子进程具有与父进程相同的程序与数据。

本例总计创建了四个进程，他们之间具有如下的关系：



由实验的结果我们可以看到，进程输出结果的数量总是一定的，但执行的顺序可能不同，这是因为当系统中有多个进程时，操作系统会进行进程调度，以提高CPU的利用率，而这种调度的顺序取决于调度队列的顺序。

1. 使用如图所示的程序，说明LINE A可能输出什么:

#include < sys/types.h >

#include < stdio.h >

#include < unistd.h >

int value = 5;

int main( ) {

pid\_t pid;

pid= fork( );

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

value += 15;

} else if ( pid > 0 ) {/\* parent process\*/

wait (NULL);

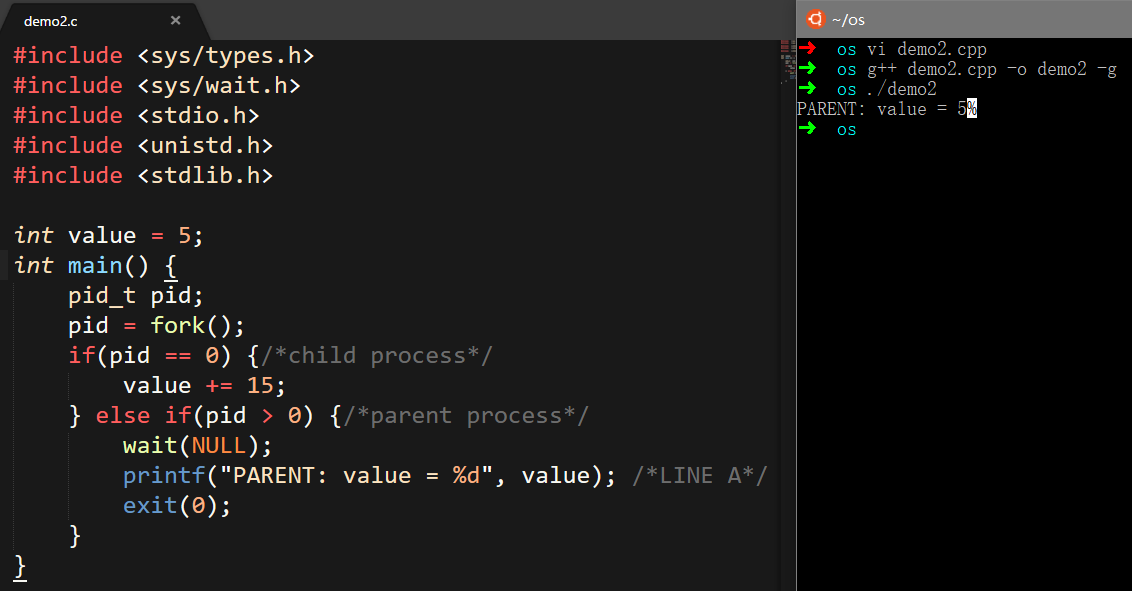
printf ("PARENT: value = %d", value); /\*LINE A\*/

exit();

}

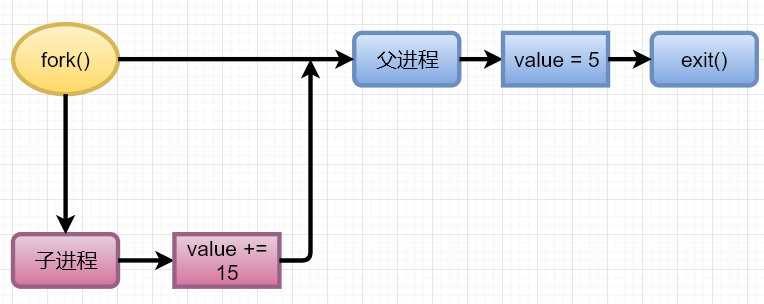
}

1. 实验结果：

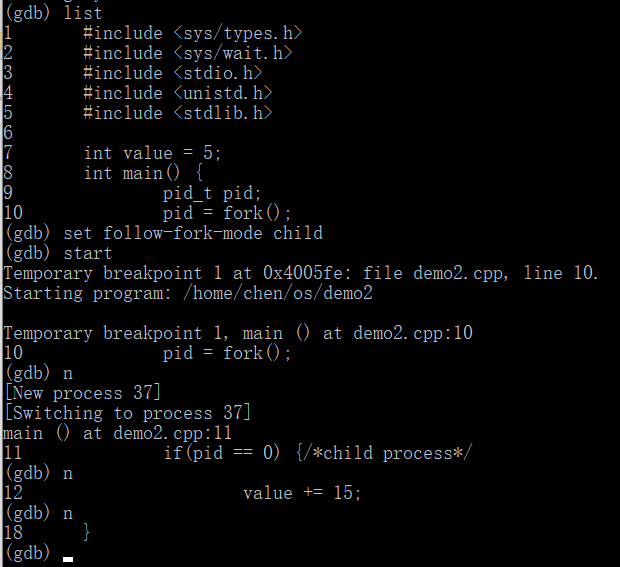


1. 实验分析：

该例中，实验过程如图：



使用gdb追踪子进程：



系统调用fork()后，创建了一个新的子进程。在子进程运行时，父进程通过系统调用wait()等待子进程的完成。子进程完成后，控制回到父进程，父进程从wait()调用处开始继续执行。尽管在子进程中对value的值进行了修改，但当时父进程已经退出就绪队列，父进程的值始终保持在value = 5。之后父进程调用系统调用exit()表示结束。

1. 编写一段程序，使用系统调用fork( )创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符；父进程显示字符“a”；子进程分别显示字符“b”和字符“c”。试观察记录屏幕上的显示结果，并分析原因。
2. 程序代码：

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

int main() {

pid\_t pid1, pid2;

pid1 = fork();

if (pid1 == 0) {

printf("b\n");

} else {

pid2 = fork();

if (pid2 == 0) {

printf("c\n");

} else {

printf("a\n");

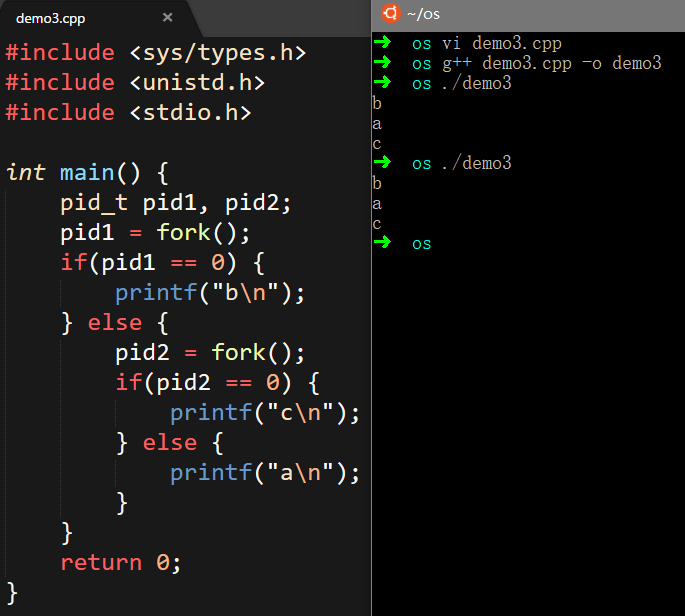
}

}

return 0;

}

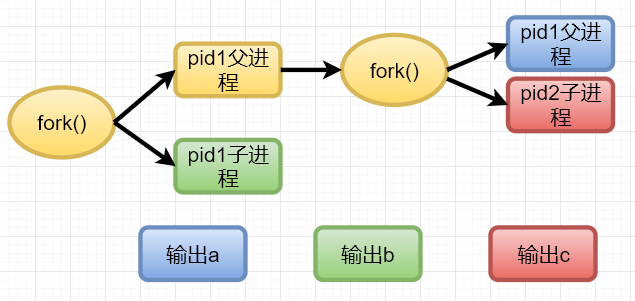
1. 实验结果：



1. 实验分析：

本例中，系统调用fork()后，进程没有使用系统调用exec()，父进程与子进程并发执行，子进程的地址空间复制父进程，子进程具有与父进程相同的程序与数据。

本例总计创建了三个进程，他们之间具有如下的关系：



本例与实验一类似，进程输出结果的数量总是一定的，但执行的顺序可能不同。当系统中有多个进程时，操作系统会进行进程调度，以提高CPU的利用率，而这种调度的顺序与调度队列的顺序有关。