

操作系统 实验报告

实验名称: 实验一进程的创建实验

姓名: 孙肖冉

学号: 16340198

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. 理解 fork() 函数，学会如何编写程序创建子进程。
2. 了解子进程与父进程的关系，认识它们执行顺序的特点

三、实验过程：

Task1

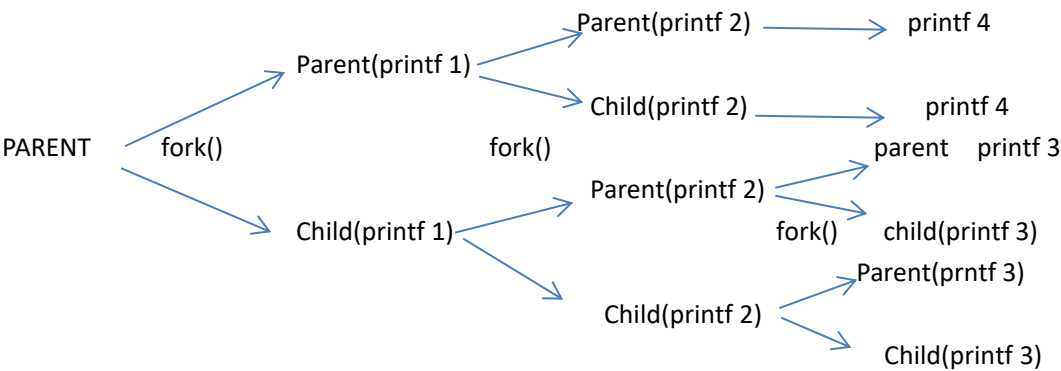
将下面的程序编译运行，并解释现象。

```
#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;
}
```

实验截图及结果解释：

```
终端
yali@yali-Lenovo-TianYi-310-14IKB: ~/桌面
yali@yali-Lenovo-TianYi-310-14IKB:~$ cd 桌面
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ gcc test.c -o test.o
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./test.o
**1**
**2**
**4**
**1**
**2**
**4**
**2**
**2**
**3**
**3**
**3**
**3**
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./test.o
**1**
**1**
**2**
**4**
**2**
**2**
**4**
**3**
```

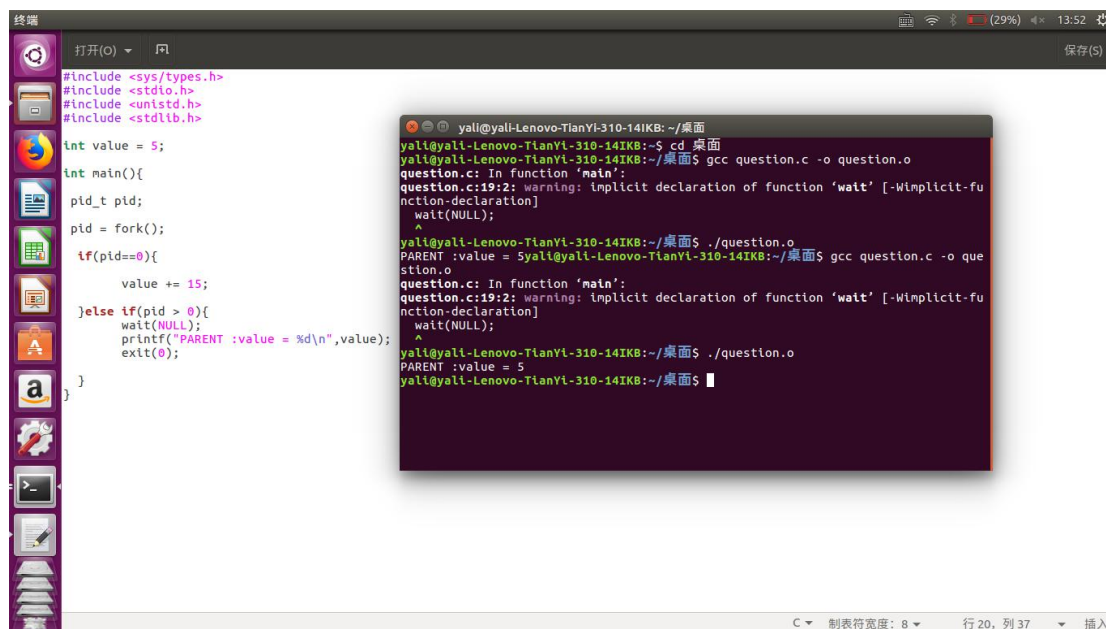
根据输出结果的统计，每次都会输出 12 个数字，虽然输出的数字顺序不同，但是经过统计每次都会出现 2 次“1”，4 次“2”，4 次“3”，2 次“4”。
可以用以下的图进行解释这一结果



在 fork 函数创建成功后会产生两个进程：父进程和子进程，在图中分别用 parent 和 child 表示。这两个进程在时间上并无先后之分，两者同时进行，所以数字显示的先后顺序并不一定。

Task2

通过实验完成第三章习题 3.4。

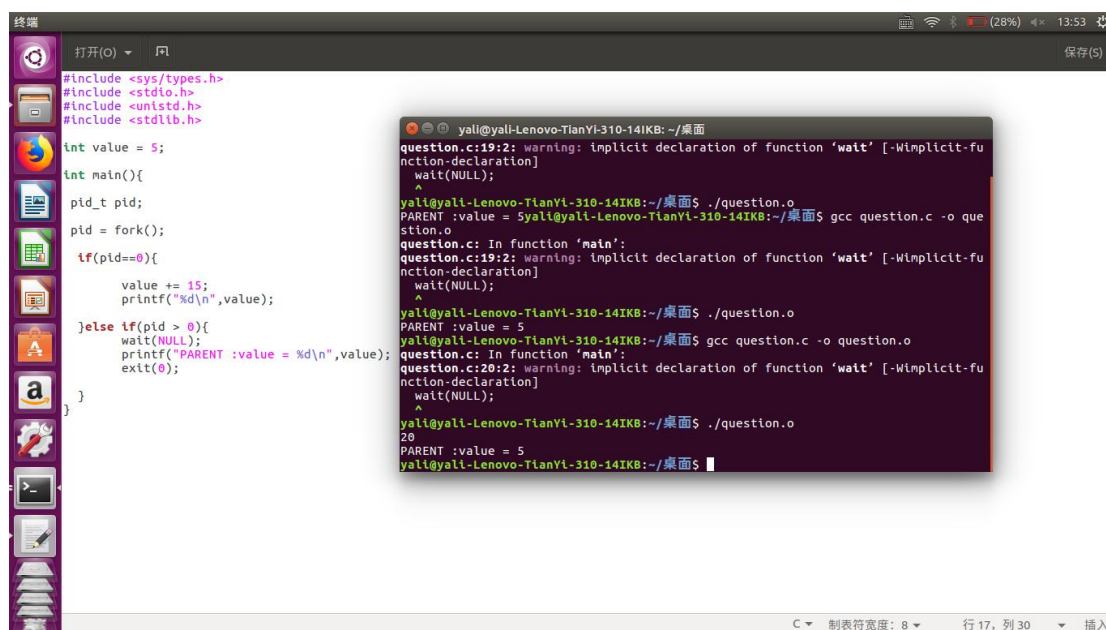


```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int value = 5;

int main(){
    pid_t pid;
    pid = fork();
    if(pid==0){
        value += 15;
    }else if(pid > 0){
        wait(NULL);
        printf("PARENT :value = %d\n",value);
        exit(0);
    }
}
```

```
yali@yali-Lenovo-TianYi-310-14IKB: ~/桌面
yali@yali-Lenovo-TianYi-310-14IKB:~$ cd 桌面
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ gcc question.c -o question.o
question.c: In function 'main':
question.c:19:2: warning: implicit declaration of function 'wait' [-Wimplicit-fu
nction-declaration]
    wait(NULL);
    ^
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./question.o
PARENT :value = 5yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ gcc question.c -o que
stion.o
question.c: In function 'main':
question.c:19:2: warning: implicit declaration of function 'wait' [-Wimplicit-fu
nction-declaration]
    wait(NULL);
    ^
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./question.o
PARENT :value = 5
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$
```



```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int value = 5;

int main(){
    pid_t pid;
    pid = fork();
    if(pid==0){
        value += 15;
        printf("%d\n",value);
    }else if(pid > 0){
        printf("PARENT :value = %d\n",value);
        exit(0);
    }
}
```

```
yali@yali-Lenovo-TianYi-310-14IKB: ~/桌面
question.c:19:2: warning: implicit declaration of function 'wait' [-Wimplicit-fu
nction-declaration]
    wait(NULL);
    ^
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./question.o
PARENT :value = 5yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ gcc question.c -o que
stion.o
question.c: In function 'main':
question.c:19:2: warning: implicit declaration of function 'wait' [-Wimplicit-fu
nction-declaration]
    wait(NULL);
    ^
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$ ./question.o
PARENT :value = 5
20
yali@yali-Lenovo-TianYi-310-14IKB:~/桌面$
```

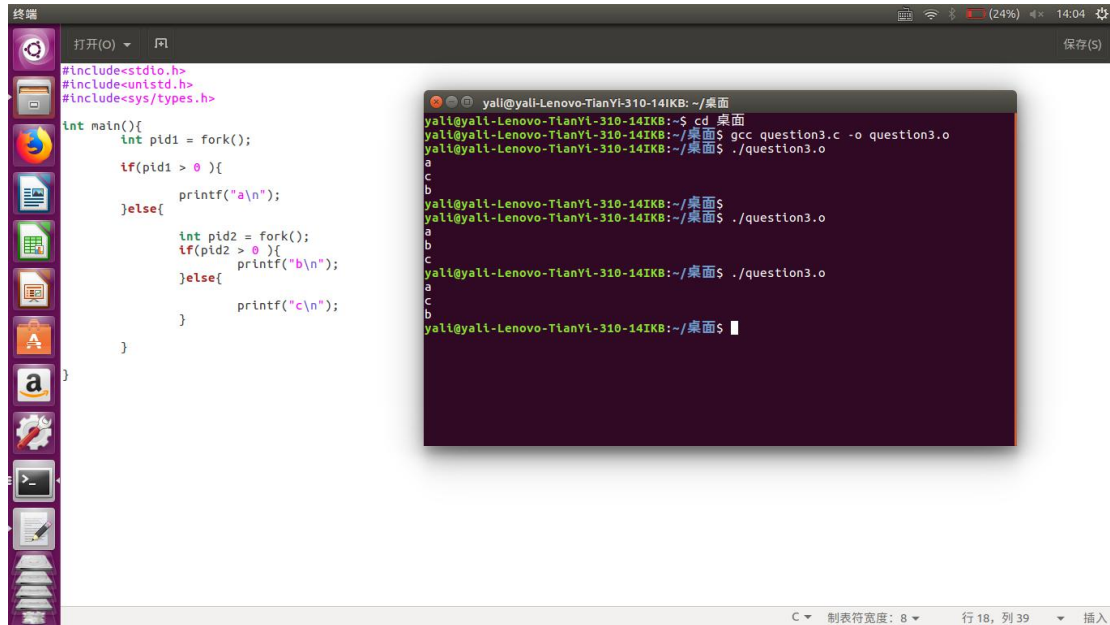
如图一所示，最终的输出结果是 **PARENT : value = 5;**

我又进行了一次额外的实验，查看子进程中的 value 值，此时为 20。

虽然按照程序的运行模式，是先对子进程的 value 值进行修改，但是子进程只是父进程的一个拷贝，子进程对父进程中的操作并无影响。所以最终父进程中的 value 仍然是 5。

Task3

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



源代码:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
```

```
int main(){
    int pid1 = fork();
    if( pid1 > 0 ){
        printf("a");
    }else{
        int pid2 = fork()
        if(pid2 = 0){
            printf("b");
        }else{
            printd("c");
        }
    }
}
```

图中出现了“abc”和“acd”两种结果，a 始终是最先输出，但 bc 的顺序不定，查阅资料后了解到结果始终符合拓扑顺序。