

操作系统实验

实验 3 Linux进程控制 lockf()

一、实验目的及要求

- 1.了解进程与程序的区别，加深对进程概念的理解；
- 2.进一步认识进程并发执行的原理，理解进程并发执行的特点，区别进程并发执行与顺序执行；
- 3.分析进程争用临界资源的现象，学习解决进程互斥的方法。
- 4.了解fork()系统调用的返回值，掌握用fork()创建进程的方法；
- 5.熟悉lockf() 等系统调用。

二、实验内容

利用系统调用lockf (fd, mode, size) , 对指定区域 (有size指示) 进行加锁或解锁, 以实现进程的同步或互斥。其中, fd是文件描述字; mode是锁定方式, mode=1表示加锁, mode=0表示解锁; size是指定文件fd的指定区域, 用0表示从当前位置到文件结尾 (注: 有些Linux系统是 locking (fd, mode, size))

三、实验源码

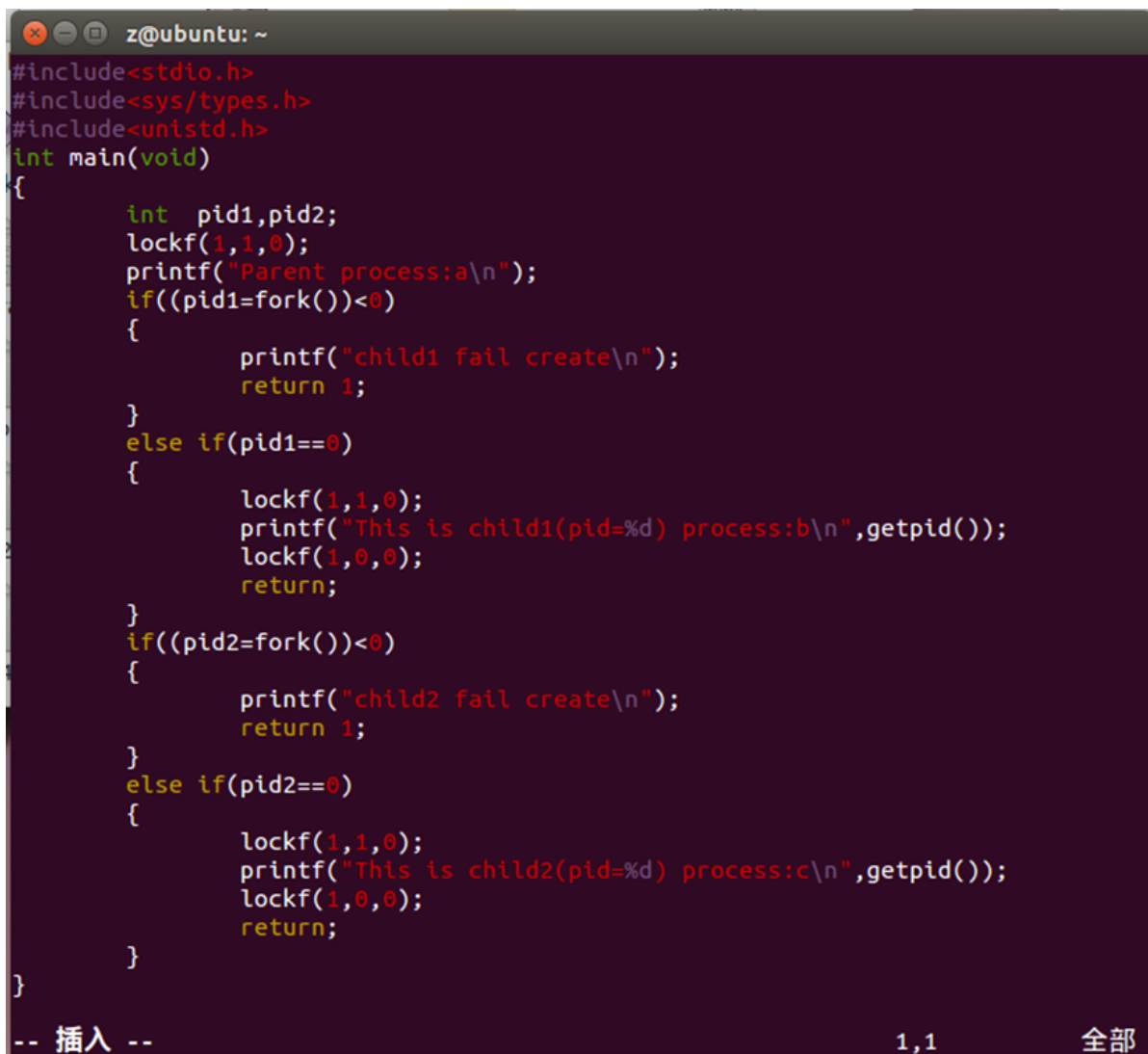
```
1  #include<stdio.h>
2  #include<sys/types.h>
3  #include<unistd.h>
4  int main(void)
5  {
6      int pid1,pid2;
7      lockf(1,1,0);
8      printf("Parent process:a\n");
9      if((pid1=fork())<0)
10     {
11         printf("child1 fail create\n");
12         return 1;
13     }
14     else if(pid1==0)
15     {
16         lockf(1,1,0);
17         printf("This is child1(pid=%d) process:b\n",getpid());
18         lockf(1,0,0);
19         return;
20     }
```

```

21     if((pid2=fork())<0)
22     {
23         printf("child2 fail create\n");
24         return 1;
25     }
26     else if(pid2==0)
27     {
28         lockf(1,1,0);
29         printf("This is child2(pid=%d) process:c\n",getpid());
30         lockf(1,0,0);
31         return;
32     }
33 }
34

```

四、实验结果



```

z@ubuntu: ~
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main(void)
{
    int pid1,pid2;
    lockf(1,1,0);
    printf("Parent process:a\n");
    if((pid1=fork())<0)
    {
        printf("child1 fail create\n");
        return 1;
    }
    else if(pid1==0)
    {
        lockf(1,1,0);
        printf("This is child1(pid=%d) process:b\n",getpid());
        lockf(1,0,0);
        return;
    }
    if((pid2=fork())<0)
    {
        printf("child2 fail create\n");
        return 1;
    }
    else if(pid2==0)
    {
        lockf(1,1,0);
        printf("This is child2(pid=%d) process:c\n",getpid());
        lockf(1,0,0);
        return;
    }
}
-- 插入 --
1,1 全部

```

```

z@ubuntu:~$ vi lockf.c
z@ubuntu:~$ gcc -o lockf lockf.c
z@ubuntu:~$

```

```
Parent process:a
This is child2(pid=5922) process:c
z@ubuntu:~$ This is child1(pid=5921) process:b
./lockf
Parent process:a
This is child1(pid=5924) process:b
z@ubuntu:~$ This is child2(pid=5925) process:c
```

五、结果分析

这段代码创建了一个父进程和两个子进程，分别称为child1和child2。每个进程都会尝试获取锁来输出内容，以避免多个进程同时输出导致混乱。然而，锁的获取和释放不是原子操作，可能导致进程之间的竞争和调度差异。

当代码运行时，父进程首先输出"a"。然后，根据操作系统的调度策略，父进程可能会先让child1执行，也可能会先让child2执行。

如果child1先执行，它尝试获取锁并输出"b"，然后释放锁。接着，child2再次尝试获取锁，获取成功后输出"c"，然后释放锁。

结果可能是： $a \rightarrow b \rightarrow c$

如果child2先执行，那么child2会尝试获取锁并输出"c"，然后释放锁。此时，child1再次尝试获取锁，获取成功后输出"b"，然后释放锁。

结果可能是： $a \rightarrow c \rightarrow b$

由于多个进程之间的调度是由操作系统负责的，因此无法确定每次运行程序的结果顺序。结果的不同顺序是由于进程调度的随机性造成的。