操作系统实验

实验 2 Linux进程控制wait、exit

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wait、exit等系统调用。

二、实验内容

> 修改程序，在父、子进程中分别使用wait、exit等系统调用"实现"其同步推进，多次反复运行改进后的程序，观察并记录运行结果。

三、实验源码

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
   pid_t pid;
   int status,i;
   if(fork()==0){
       printf("This is the child process .pid =%d\n",getpid());
       exit(5);
   }
   else{
       sleep(1);
       printf("This is the parent process ,wait for child...\n");
       pid=wait(&status);
       i=WEXITSTATUS(status);
       printf("child's pid =%d .exit status=%d\n",pid,i);
   }
}

```

四、实验结果

z@ubuntu: ~

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
    pid_t pid;
    int status,i;
    if(fork()==0){
        printf("This is the child process .pid =%d\n",getpid());
        exit(5);
    }
    else{
        sleep(1);
        printf("This is the parent process ,wait for child...\n");
        pid=wait(&status);
        i=WEXITSTATUS(status);
        printf("child's pid =%d .exit status=%d\n",pid,i);
    }
}
```

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

```
z@ubuntu:~$ ./we
This is the child process .pid =5906
This is the parent process ,wait for child...
child's pid =5906 .exit status=5
z@ubuntu:~$ ./we
This is the child process .pid =5908
This is the parent process ,wait for child...
child's pid =5908 .exit status=5
z@ubuntu:~$
```

五、结果分析

我发现运行出来的结果每次都是一样的。

父进程先创建了子进程，然后使用wait函数等待子进程的结束。子进程打印了自己的进程ID后就通过exit结束，并向父进程返回了退出状态值5。父进程在等待子进程结束后，通过wait函数获取子进程的进程ID和退出状态，并打印出来。

根据以上分析，每次运行的输出结果应该是一致的，不会有变化。这是因为父进程会在子进程结束之后才继续执行，而子进程执行的内容在每次运行时都是一样的，因此输出结果也是一致的。