课程名称	操作系统	课程编号	A2130330
实验地点	C410/411		
实验名称	在 C 语言程序内使用系统调用		

一、实验内容

在 Linux 系统中编写 C 语言程序并通过系统调用输出计算机当前用户的标识，同时返回子进程与父进程的 pid。实验过程中应使用 fork(),getpid(),getppid(),execv()以及 shell 命令 whoami。

二、实验步骤及方案

1.使用 fork()创建子进程。

2.父进程中使用 getpid()输出自身 pid，并输出子进程 pid。输出完成后使用 wait()等待子进程结束。输出格式为：

Parent: My pid = pid-of-parent. My child has pid = pid-of-child

3.子进程中使用 getpid()输出自身 pid，并使用 getppid()输出父进程 pid。输出格式为：

Child: My pid = pid-of-child. My parent has pid = pid-of-parent

4.子进程中使用 execv()调用 shell 命令 whoami 输出当前用户名称。输出格式为：

Child: The current user is: name-of-the-user

三、结果及分析

实验源码：

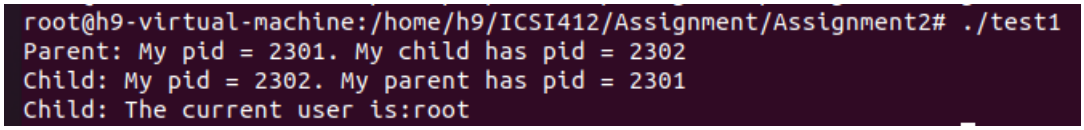
```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/types.h>
#include<sys/wait.h>

int main()
```

```
{
        pid_t pid;
        int statu = 0;
        pid = fork();
        char* argv[] = {"whoami", NULL};

        if(pid < 0)
        {
                perror("fork()");
        }
        else if(pid > 0)
        {
                printf("Parent: My pid = %d. My child has pid = %d\n", getpid(), pid);
                wait(&statu);
        }
        else
        {
                printf("Child: My pid = %d. My parent has pid = %d\n", getpid(), getppid());
                printf("Child: The current user is:");
                fflush(stdout);
                execv("/bin/whoami", argv);
        }
        return 0;
}
```

实验结果如下图所示：



结果分析：

1. 父进程成功输出父 pid-2301，子 pid-2302。
2. 子进程输出父 pid-2301，子 pid-2302，与父进程输出一致。
3. 子进程成功输出当前用户 id-root。
4. 注意：因使用 execv 执行 whomai 输出用户 id，而其之前的 printf 输出与其应保持同行输出，故应使用 fllush()刷新输出缓存，完成正常输出。

四、心得体会

1. 熟练掌握了在 linux 中使用 vim 编写 c 语言程序，并使用 gcc 编译 c 语言

程序。

2. 掌握了父子进程的关系，并通过相关实验加深了理解。
3. 掌握了 fork()，getpid()，getppid(), wait(), execv()等系统调用函数。
4. 掌握了 exec 函数家族，理解并区分了其内部的不同之处。
5. 熟悉了 shell 命令 whoami 等。
6. 巩固了 c 语言中 printf 与缓冲区，stdin 流和 stdout 流。