《Linux 操作系统设计实践》
实验报告

实验 1：进程管理

院　　系：数学与计算机科学学院

专　　业：计算机科学与技术

年　　级：2016 级计算机 5 班

学　　号：031602507

姓　　名：陈俞辛

一、 实验环境

二、 实验内容

(一) 代码简介

题目要求实现的效果有两个：

1. 使用 fork( )函数；

2. 两个或多个进程之间有信息交互效果；

碰巧这两个要求在上学期的操作系统课上都有接触过，因此这次作业对我来说还是比较简单的。

首先，我在原先的进程中创建一个管道，然后再调用 fork()创建一个新的进程，最后父进程通过管道向子进程传递数据。

(二) 实验代码

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define BUFLEN    50
int main(int argc, char **argv)
{
        size_t data_len = 0;
        int filedes[2];
        const char data[] = "This is a message from parent process\n";
        char buffer[BUFLEN];
        pid_t pid;


        if (pipe(filedes) == 0)
        {
                // 创建管道成功
```

```
        // 调用 fork 创建子进程
        pid = fork();
        if (pid == -1)
        {
            fprintf(stderr, "fork failure\n");
            exit(EXIT_FAILURE);
        }

        if (pid == 0)
        {
            // 子进程
            // 读数据
            data_len = read(filedes[0], buffer, BUFSIZ);
            printf("I'm child porcess,Read %ld bytes: %s\n", data_len, buffer);
            exit(EXIT_SUCCESS);
        }
        else
        {
             // 父进程
            // 写数据
            data_len = write(filedes[1], data, strlen(data));
            printf("I'm parent process,Wrote %ld bytes: %s\n", data_len, data);

            // 休眠 2 秒, 主要是为了等待子进程结束
            sleep(2);
            exit(EXIT_SUCCESS);
        }
    }
    else
    {
        fprintf(stderr, "pipe failure\n");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

注意：如果在父进程中没有 sleep()语句，父进程可能在子进程结束前结束，这样可能将看到两个输入之间有一个命令提示符分隔。

(三) 实验结果

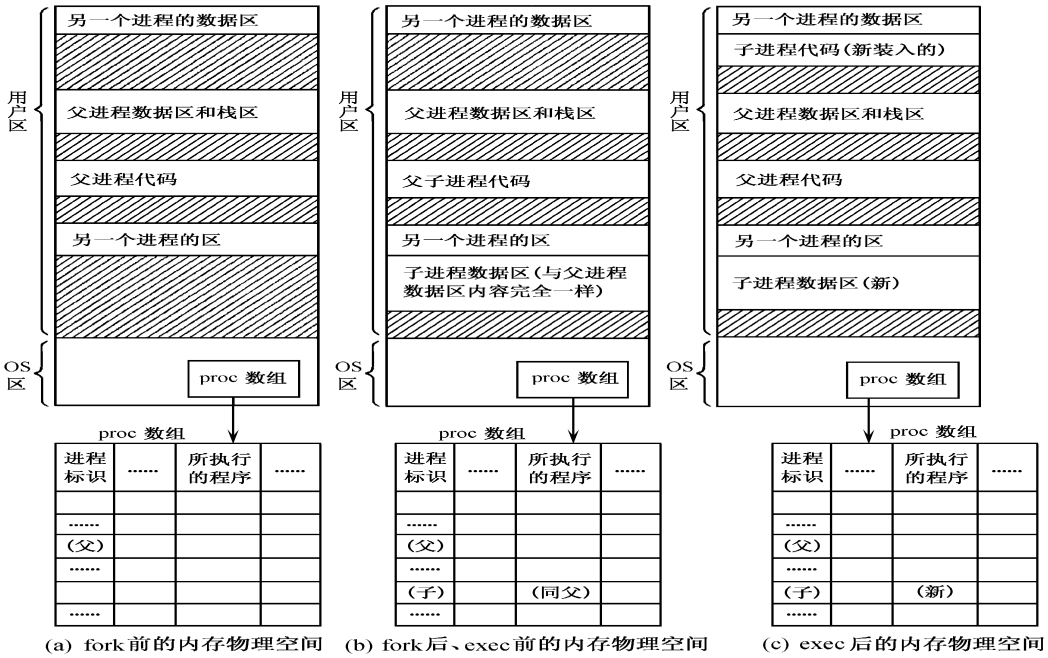
```
chenyx@chenyx-VirtualBox: ~/ex1
chenyx@chenyx-VirtualBox:~$ cd ex1
chenyx@chenyx-VirtualBox:~/ex1$ gcc code.c -o code.exe
chenyx@chenyx-VirtualBox:~/ex1$ ./code.exe
I'm parent process,Wrote 38 bytes: This is a message from parent process

I'm child porcess,Read 38 bytes: This is a message from parent process

chenyx@chenyx-VirtualBox:~/ex1$
```

三、 实验总结

1) fork()

作用是创建子进程。子进程得到父进程地址空间的一个复制。

返回值：成功时，该函数被调用一次，但返回两次，fork（）对子进程返回 0，对父进程返回子进程标识符（非 0 值）。不成功时对父进程返回-1，没有子进程。子进程是父进程的副本，它将获得父进程数据空间、堆、栈等资源的副本。 注意，父子进程间并不共享这些资源，它们只共享代码段。



(a) fork 前的内存物理空间　(b) fork 后、exec 前的内存物理空间　(c) exec 后的内存物理空间

2) 管道通信

```
#include <unistd.h>

int pipe(int file_descriptor[2]);
```

i. 作用：创建管道

ii. 返回值：成功返回 0，出错返回-1

iii. pipe 是基于文件描述符工作的，所以在使用 pipe 后，数据必须要用底层的 read()和 write()调用来读取和发送。

iv. 管道只允许具有血缘关系的进程间通信，如父子进程间的通信，这一点也是我这次作业选择管道通信实现的原因。

v. 管道是半双工的，数据只能向一个方向流动；需要双方通信时，需要建立起两个管道。本次实验只使用了父进程向子进程单向消息。

vi. 管道实现进程通信的步骤：