**北京邮电大学软件学院**

**2016-2017学年第2学期实验报告**

**课程名称： 操作系统**

**实验名称： 进程管理**

**实验完成人：**

**姓名：刘浩博 学号：2015212086 成绩：**\_\_\_\_\_\_\_\_

**姓名：陈润泽 学号：2015212105 成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_ 陈晋鹏\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2017 年 4 月 16 日**

1. **实验目的**

（1）理解进程的概念，明确进程和程序的区别。

（2）理解并发执行的实质。

（3）掌握进程的睡眠、同步、撤销等进程控制方法。

1. **实验内容**

2.1. 进程的创建。

① 编写一段源程序，使系统调用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。

② 修改已编写的程序，将每个进程输出一个字符改为每个进程输出一句话，在观察程 序执行时屏幕出现的现象，并分析原因。

2.2. 进程的控制 ① 用fork()创建一个进程，再调用exec()用新的程序替换该子进程的内容。 ② 利用wait()来控制进程执行顺序。

1. **实验环境**

在Linux和MacOS下，采用以VSCode +GCC+GDB为开发环境的C语言实验环境。其中VSCode 作为编辑器，GCC作为编译器，GDB作为调试器。

1. **实验过程**

分析了两个小实验的目标和相关函数的使用方法，然后进行分工：

刘浩博负责2.1实验以及实验报告的编写；

陈润泽负责2.2实验以及调试修改实验代码。

1. **实验结果**

实验均能达到预想目标，与实验指导手册给出的结果基本吻合。

1. **附件**

**6.1 源代码**

**Lab1.c:**

**#include <unistd.h>**

**#include <stdio.h>**

**#include <time.h>**

**#include <stdlib.h>**

**#ifdef \_POSIX\_VERSION**

**#define LAB1\_1\_1 1/\*在此处修改便可查看四个小实验\*/**

**#define LAB1\_1\_2 0**

**#define LAB1\_2\_1 0**

**#define LAB1\_2\_2 0**

**#if LAB1\_1\_1**

**int main(const int argc, const char \*argv[]) {**

**pid\_t child\_process\_0;**

**pid\_t child\_process\_1;**

**if (child\_process\_0 = fork()) {**

**if (child\_process\_1 = fork()) {**

**putchar('c');**

**} else putchar('a');**

**} else putchar('b');**

**return 0;**

**}**

**#elif LAB1\_1\_2**

**int main(const int argc, const char \*argv[]) {**

**pid\_t child\_process\_0;**

**pid\_t child\_process\_1;**

**if (child\_process\_0 = fork()) {**

**if (child\_process\_1 = fork()) {**

**puts("I'M PROCESS FATHER");**

**} else puts("I'M PROCESS SON 2");**

**} else puts("I'M PROCESS SON 1");**

**return 0;**

**}**

**#elif LAB1\_2\_1**

**int main(const int argc, const char \*argv[]) {**

**puts("THIS IS THE ORIGIN PROGRAM");**

**switch (fork()) {**

**case -1:**

**puts("\*\*\*FORK FAILED!\*\*\*");**

**break;**

**case 0:**

**execl("./Lab1Exec1", "./Lab1Exec1");**

**break;**

**default:**

**switch (fork()) {**

**case -1:**

**puts("\*\*\*FORK FAILED!\*\*\*");**

**break;**

**case 0:**

**execl("./Lab1Exec2", "./Lab1Exec2");**

**break;**

**default:**

**puts("(CALLING ls)");**

**if (!fork()) {**

**execl("/bin/ls", "ls");**

**}**

**break;**

**}**

**break;**

**}**

**return 0;**

**}**

**#elif LAB1\_2\_2**

**int main(const int argc, const char \*argv[]) {**

**pid\_t child\_process\_1;**

**pid\_t child\_process\_2;**

**pid\_t child\_process\_3;**

**puts("THIS IS THE ORIGIN PROGRAM");**

**switch (child\_process\_1 = fork()) {**

**case -1:**

**puts("\*\*\*FORK FAILED!\*\*\*");**

**break;**

**case 0:**

**execl("./Lab1Exec1", "./Lab1Exec1");**

**break;**

**default:**

**wait(&child\_process\_1);**

**switch (child\_process\_2 = fork()) {**

**case -1:**

**puts("\*\*\*FORK FAILED!\*\*\*");**

**break;**

**case 0:**

**execl("./Lab1Exec2", "./Lab1Exec2");**

**break;**

**default:**

**wait(&child\_process\_2);**

**puts("(CALLING ls)");**

**if (!(child\_process\_3 = fork())) {**

**execl("/bin/ls", "ls");**

**}**

**wait(&child\_process\_3);**

**puts("(FINISHED)");**

**break;**

**}**

**break;**

**}**

**return 0;**

**}**

**#endif**

**Lab1Exec1.c:**

**#include <stdio.h>**

**int main(const int argc, const char \*argv[]) {**

**puts("\e[1;34;43mTHIS IS ANOTHER PROGRAM 1\e[0m");**

**return 0;**

**}**

**Lab1Exec2.c:**

**#include <stdio.h>**

**int main(const int argc, const char \*argv[]) {**

**puts("\e[1;35;46mTHIS IS ANOTHER PROGRAM 2\e[0m");**

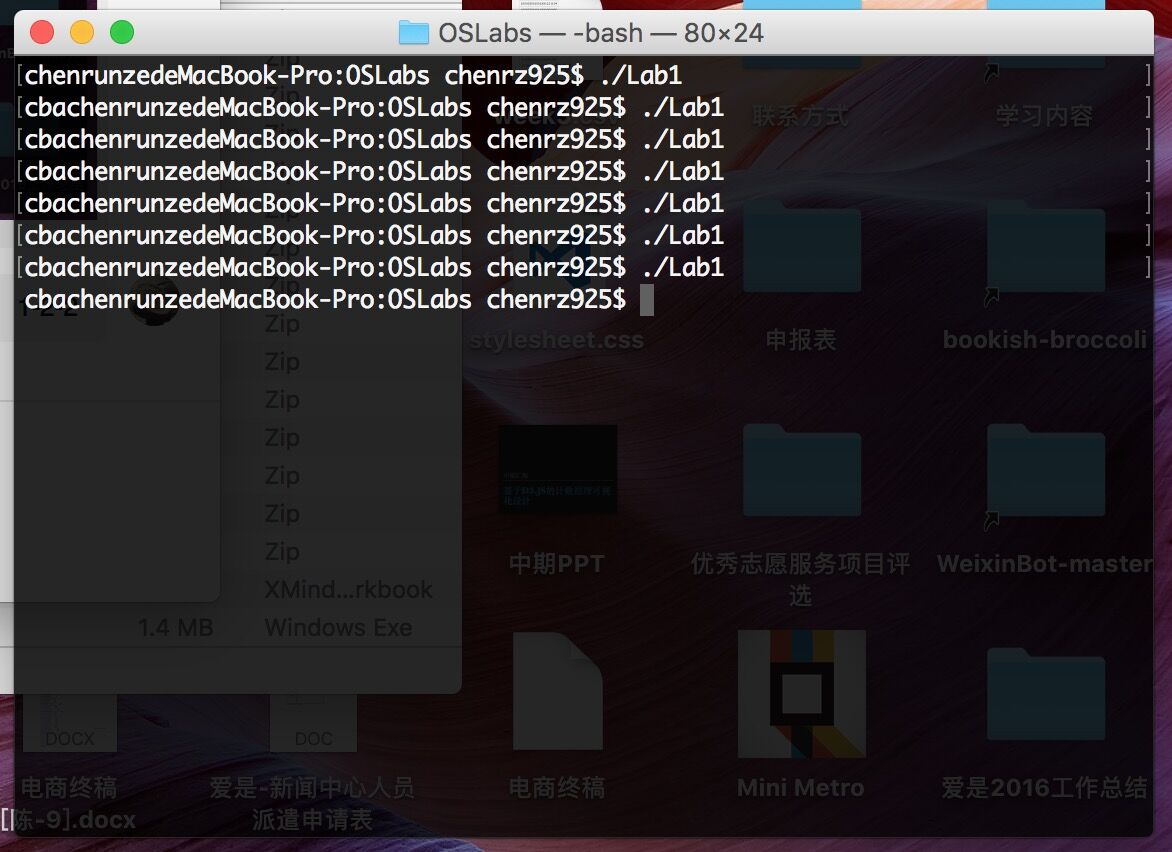
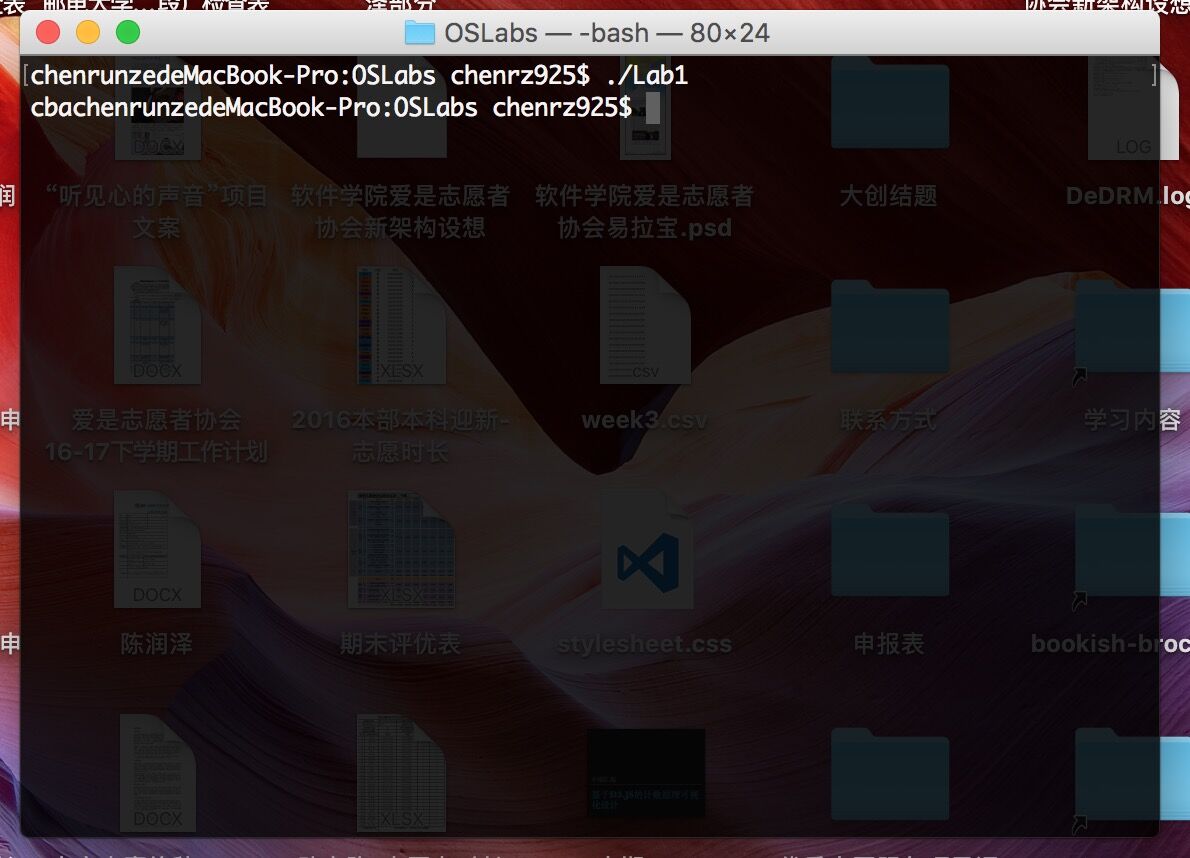
**return 0;**

**}**

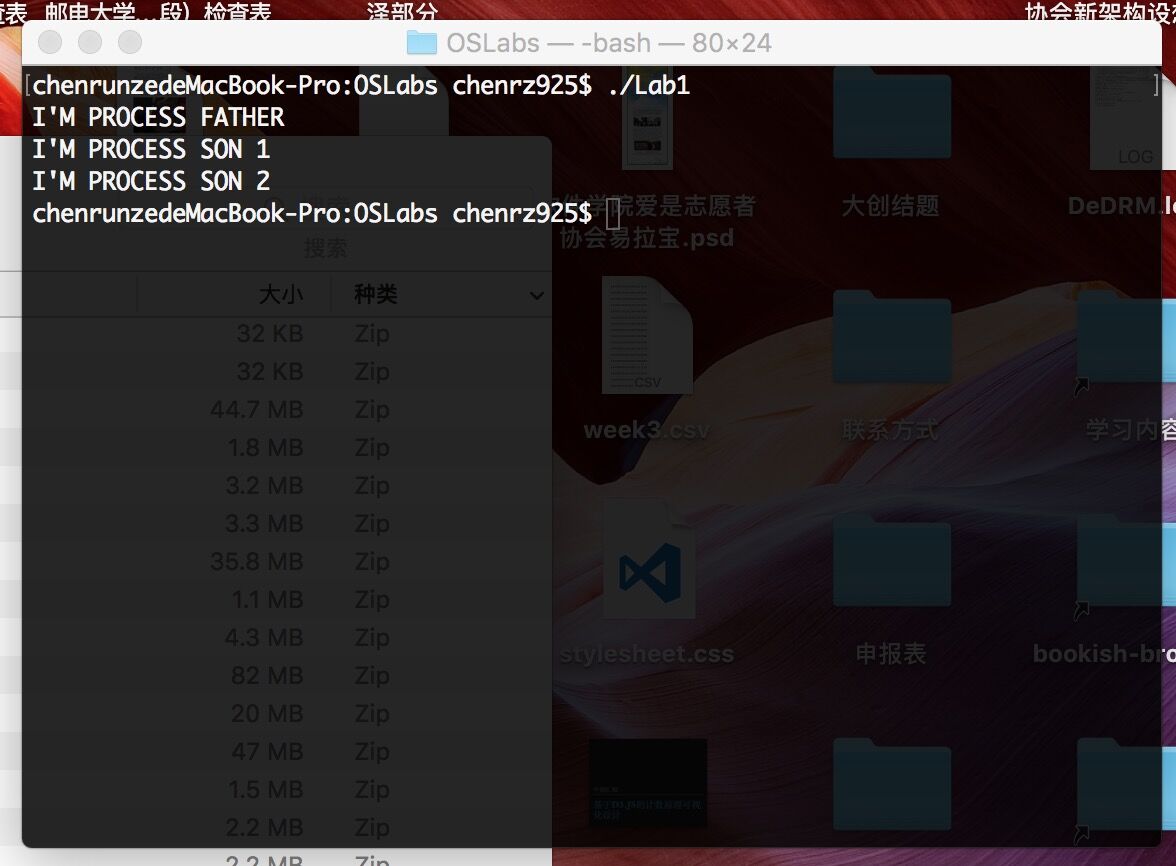
**#endif**

**6.2 实验结果截图**

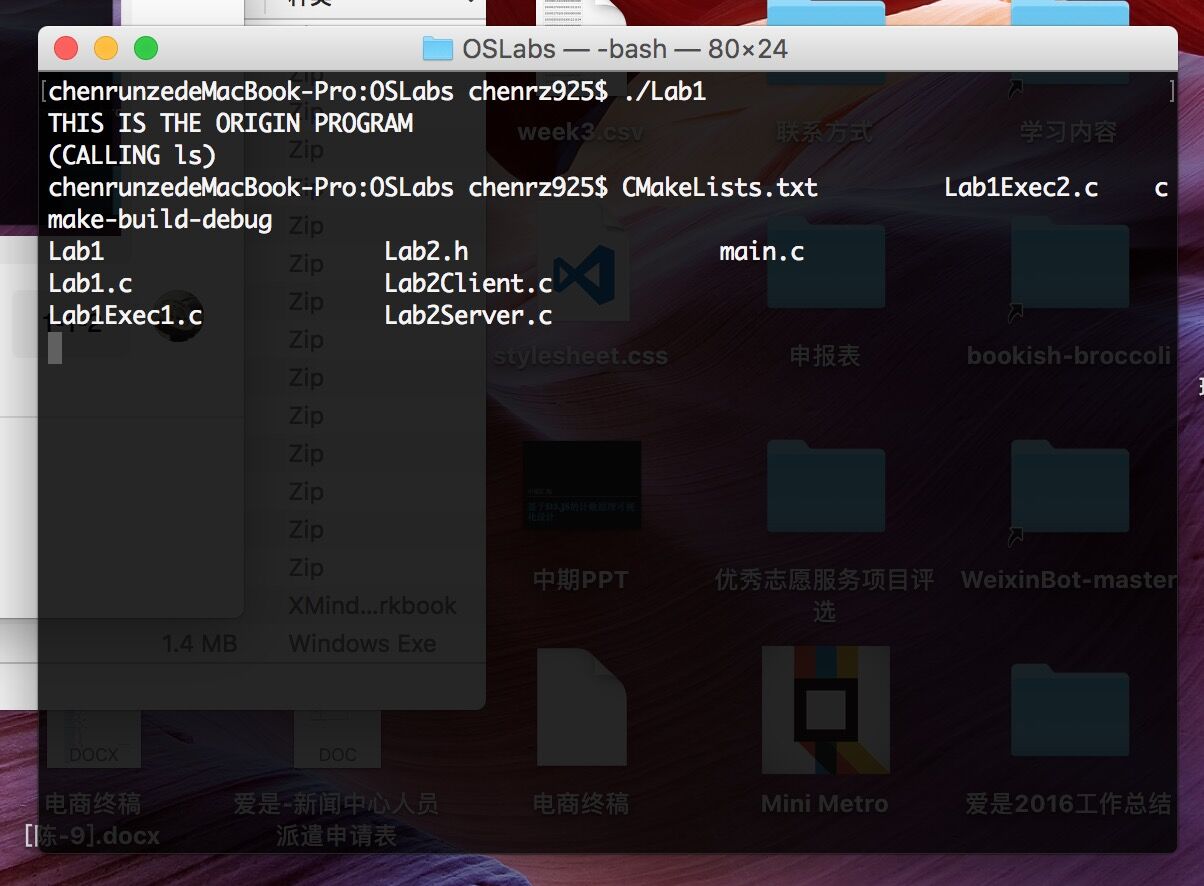
**2.1-1 输出结果为cba,多次运行结果相同。**



**2.1-2 父进程与子进程关系**



**2.2-1 调用exec1**



**2.2-2 调用exec2**

