## 目录

实验一...............................................................................................................................1

实验二................................................................................................................................

实验三................................................................................................................................

实验四................................................................................................................................

实验五................................................................................................................................

实验六................................................................................................................................

# 实验一 UNIX/Linux 入门 报告

**一、实验目的**

1、了解UNIX/Linux运行环境。

2、熟悉UNIX/Linux的常用基本命令。

3、熟悉和掌握UNIX/Linux下C语言程序的编写、编译、调试和运行方法。

4、掌握在C语言程序中使用**命令行参数**。

**二、实验内容**

1、熟悉UNIX/Linux的常用基本命令如ls、who、pwd、ps等。

2、练习UNIX/Linux的文本编辑器（如gedit或vi）的使用方法。也可以熟悉Linux中的C/C++集成开发环境（IDE）Netbeans的使用。

3、熟悉UNIX/Linux下C语言编译器cc/gcc的使用方法。

（1）用gedit或其他编辑器编写一个简单的C语言程序hello.c，功能是在标准输出设备上显示“Hello,World!”。

（2）在终端窗口中，用gcc编译hello.c，得到名为hello的可运行程序：

gcc hello.c **-o hello**

注意，上面一行的编译命令中，“hello.c -o hello”是gcc命令的**命令行参数**。其中，

* “-o hello”表示编译的输出结果（在本例中即可执行程序）命名为hello的文件，
* “hello.c”是被编译的源程序文件名。

（3）运行可执行程序hello，并观察运行结果：

./hello

4、掌握C语言程序**命令行参数**的使用方法。

（1）使用编辑器编辑下面的C语言源程序（假定源程序的文件名为argtest.c），**请特别注意main函数的参数**：

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\*\* argv) {

int i;

printf("argc = %d\n", argc);

for (i = 0; i <argc; i++) {

printf("Argument %d: %s\n", i, argv[i]);

}

return (EXIT\_SUCCESS);

}

（2）编译：

gcc argtest.c -o argtest

（3）在终端窗口中，分别以下列3种方式运行，观察输出结果，**请解释程序的功能**：

./argtest

./argtest hello

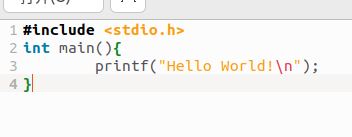
./argtest hello world

**三、实验要求**

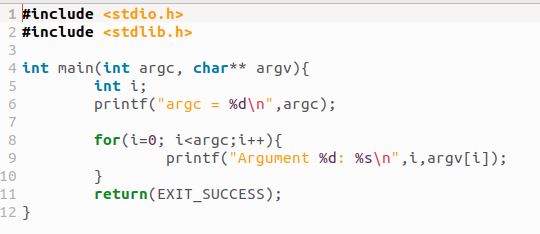
按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

1. **实验设计**

Hello world程序设计

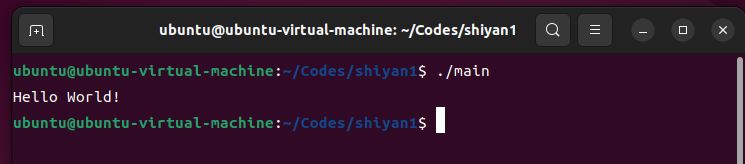
****

Argtest程序设计：

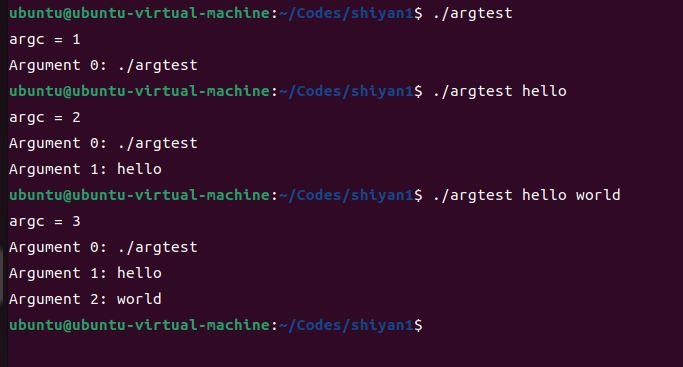
****

1. **实验测试结果及分析**

Hello world程序运行结果：

****

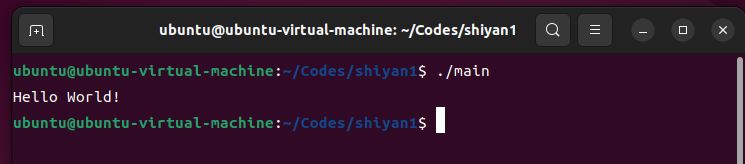
Argtest程序运行结果：

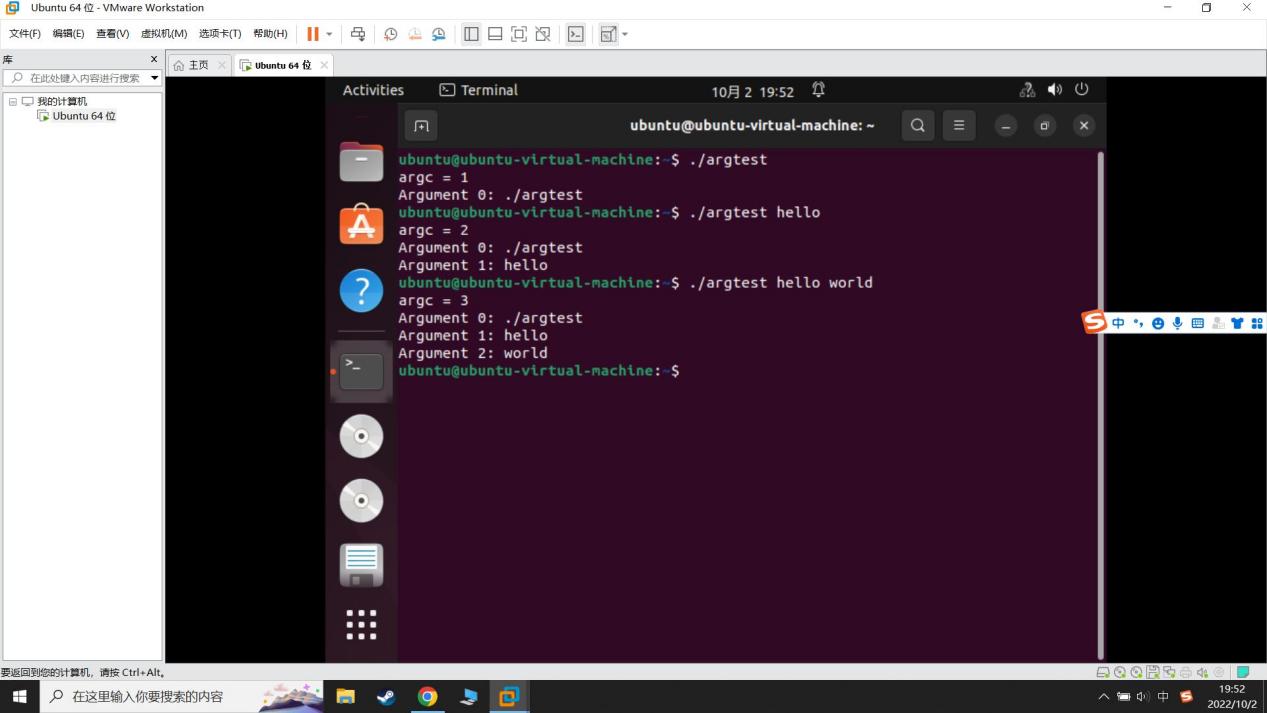
****

分析：argc为argument count，argv为argument vector，分别是提供给主函数的参数个数，

和存有参数的字符串数组的指针，首个参数默认为程序名字，存在argv[0]

1. **运行结果**

****

****

## 实验二 进程创建与管道通信

**一、实验目的**

加深对进程概念的理解，明确进程与程序的区别；进一步认识并发执行的实质。学习在Linux系统中创建子进程和进行管道通信的方法。

**二、实验内容**

1、进程创建

（1）阅读 程序 1，编译、执行该程序。解释输出结果。注意，getpid()和getppid()分别用于获得本进程、父进程的PID；wait(NULL)用于在父进程中等待子进程结束。

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <unistd.h>  #include <string.h>  #include <errno.h>  int main() {  pid\_t pid; // process ID  printf("Before fork, PID = %d, PPID = %d. ", getpid(), getppid());  printf("\n");    pid = fork(); // to create a child process  if (pid == -1) {  perror("Failed in calling fork");  exit(1);  } else if (pid == 0) {  /\* the child process \*/  printf("In child process, PID = %d, PPID = %d. ", getpid(), getppid());  printf("\n");    // ----- A -----  // ----- B -----  exit(0);  } else {  /\* the parent process \*/  printf("In parent process, child's PID = %d. ", pid);  printf("\n");    printf("In parent process, PID = %d, PPID = %d. ", getpid(), getppid());  printf("\n");    // ----- C -----  wait(NULL);  }  printf("Before return in main(), PID = %d, PPID = %d. ", getpid(), getppid());  printf("\n");    return (EXIT\_SUCCESS);  } |

程序 1 使用fork()创建子进程

（2）在“Terminal”窗口，执行 ps 命令，可以显示出 Shell （可能是bash）的进程PID以及ps命令执行所对应的进程的ID。记住Shell进程的ID，再次执行上面的程序，看看父进程的父进程ID与Shell进程ID的关系。

（3）在 程序 1 中“// ----- A -----”的下面，添加下面一条语句：

sleep(3);

这条语句的作用是“睡眠”3秒钟后再执行。重新编译、执行程序，解释现象。

（4）在（3）的基础上，将“// ----- B -----”下面的一条语句删去。重新编译、执行程序，解释现象。

（5）在（4）的基础上，将“// ----- C -----”下面的一条语句删去。重新编译、执行程序，解释现象。注意，在UNIX/Linux系统中，如果一个进程的父进程先于自己“结束”，系统将该进程的父进程设为1号进程。

（6）编写一段程序，使用系统调用 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**一行**字符：父进程显示 “aaaaa\n”；两个子进程分别显示 “bbbbb\n”和 “ccccc\n”。观察、记录屏幕上的显示结果，并分析原因。

2、绘制进程家族树

阅读 程序 2，编译、运行该程序。回答下列问题：

（1）一共创建了几个进程？它们各自的PID是什么？

（2）根据输出，绘制一棵进程家族树，表示父进程和被创建的进程之间的关系。

（3）根据输出，说明每个进程都执行了哪些内容（注意时间顺序）。

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <unistd.h>  #include <string.h>  #include <errno.h>  int main() {  pid\_t result1;  pid\_t result2;  printf("<A> -- pid = %d, ppid = %d\n", getpid(), getppid());  result1 = fork();  if (result1 < 0) {  perror("Failed to create process after <A>");  exit(1);  }  printf("<B> -- pid = %d, ppid = %d, result1 = %d.\n", getpid(), getppid(), result1);  result2 = fork();  if (result2 < 0) {  perror("Failed to create process after <B>");  exit(1);  }  printf("<C> -- pid = %d, ppid = %d, result2 = %d.\n", getpid(), getppid(), result2);    sleep(2);  return (EXIT\_SUCCESS);  } |

程序 2 进程家族树实例代码

3、进程的管道通信

编写程序实现进程的管道通信。具体要求：首先使用系统调用pipe( )建立一个管道；然后分别创建2个子进程，要求2个子进程分别向管道各写一句话：“Child 1 is sending a message!” 和 “Child 2 is sending a message!” 最后，父进程从管道中读出二个来自子进程的信息并显示。

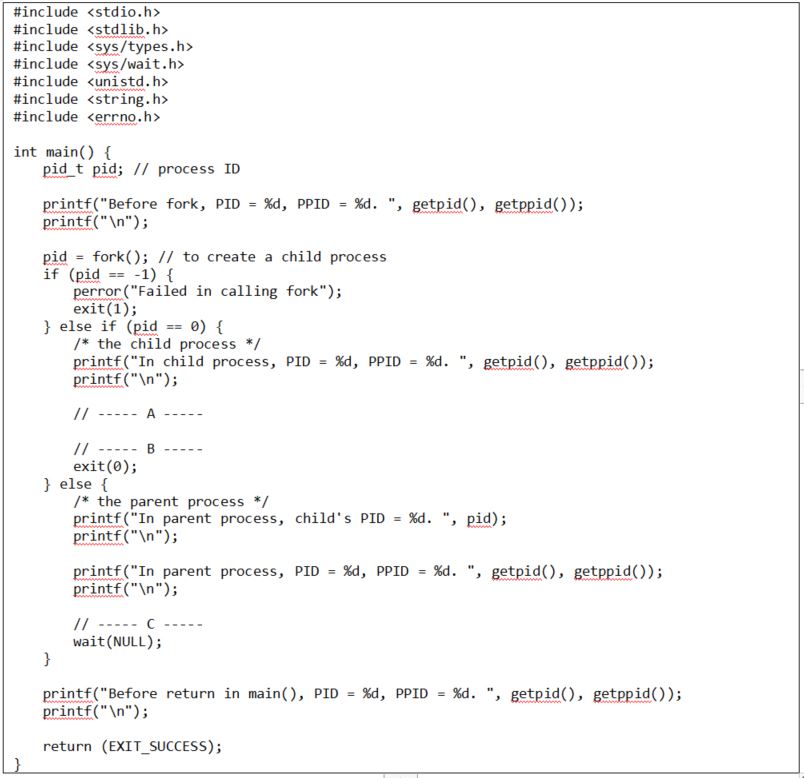
请参考“四、补充材料”。

三、**实验要求**

按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

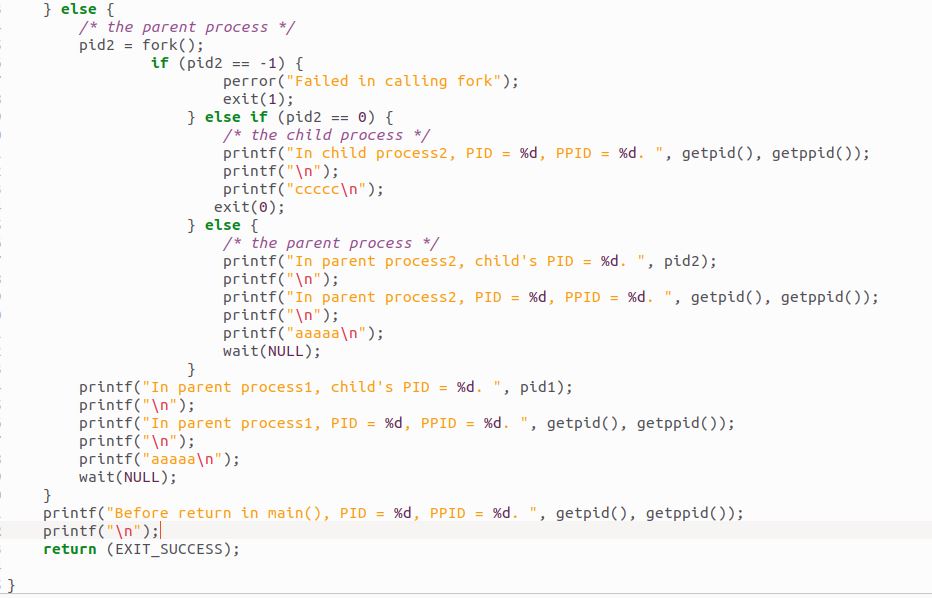
1. **实验设计**

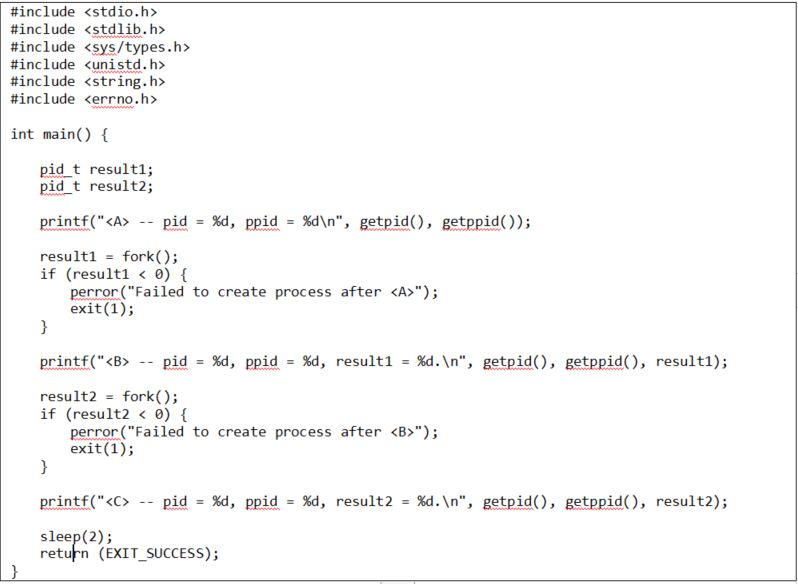
第一部分进程创建

****

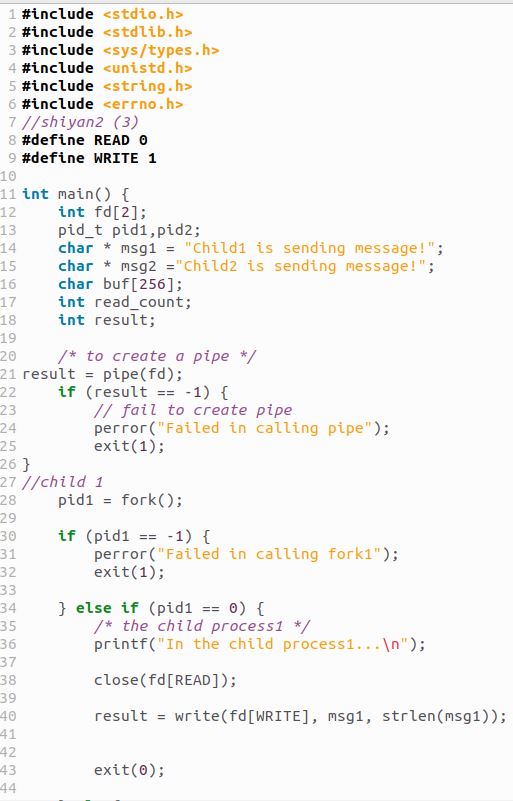
**三个子情况分别是A下添加sleep(3);在此基础上删去B下的代码;在此基础上删去C下的代码**

第一部分（6）****

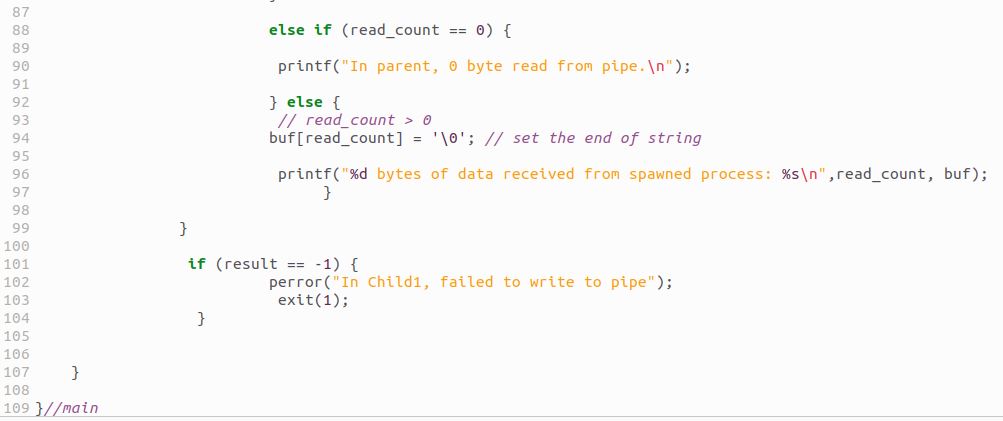
****

**第二部分**

**第三部分管道通信：**

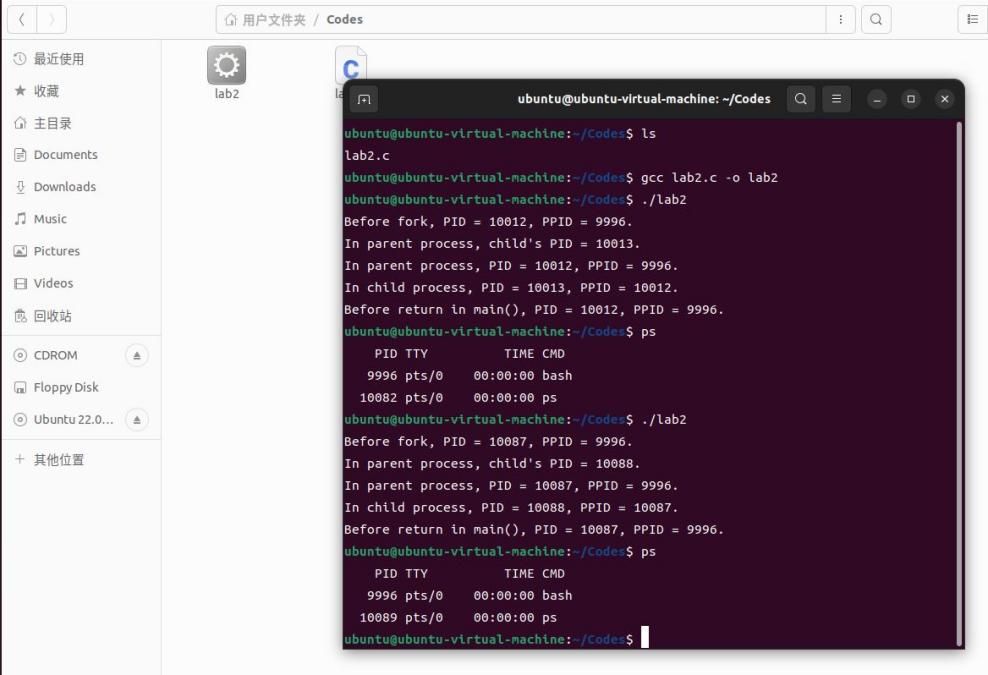
****

****

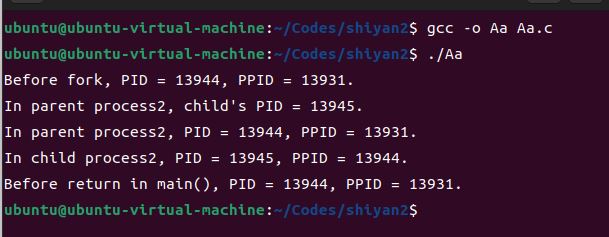
****

1. **实验测试结果及分析**

**第一部分**

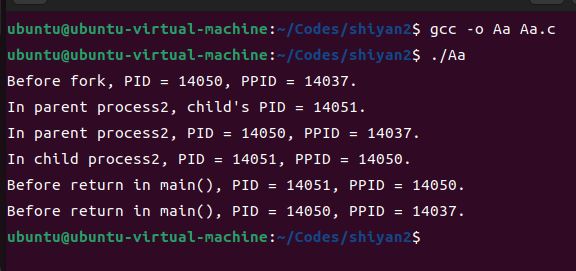
****

**A下添加sleep(3)：**

****

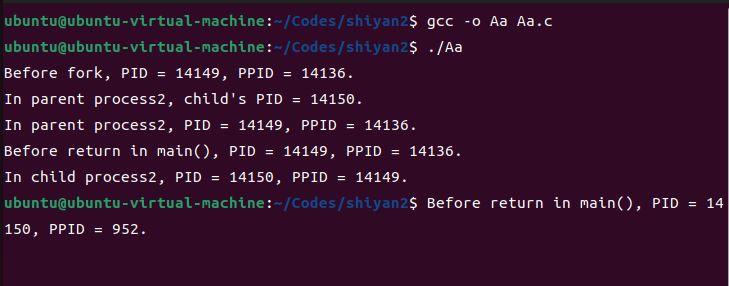
**相比于没有sleep的，添加后程序中父进程会等待子进程在3秒后exit以后才打印before return语句**

**B删除**

****

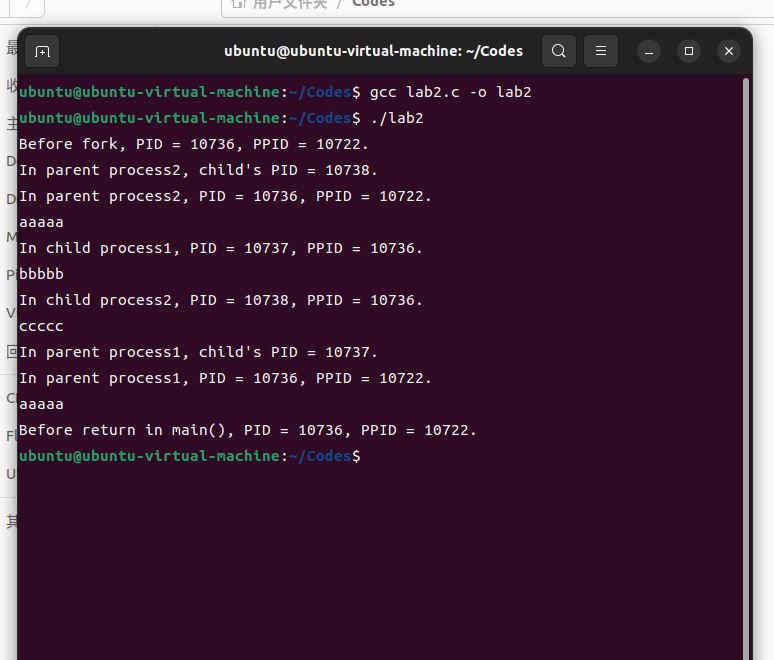
**子进程不exit，所以也会打印返回语句**

**C删除**

****

**父进程不等待，直接输出了返回语句，子进程没有exit，也进行了输出**

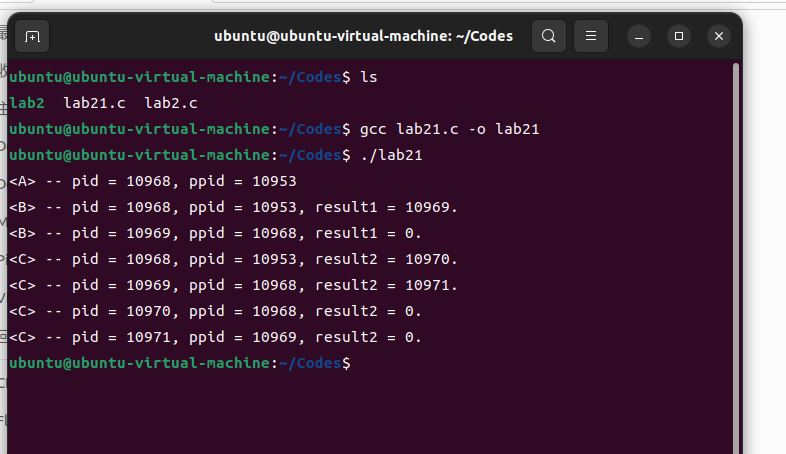
**第一部分（6）自己创建：**

****

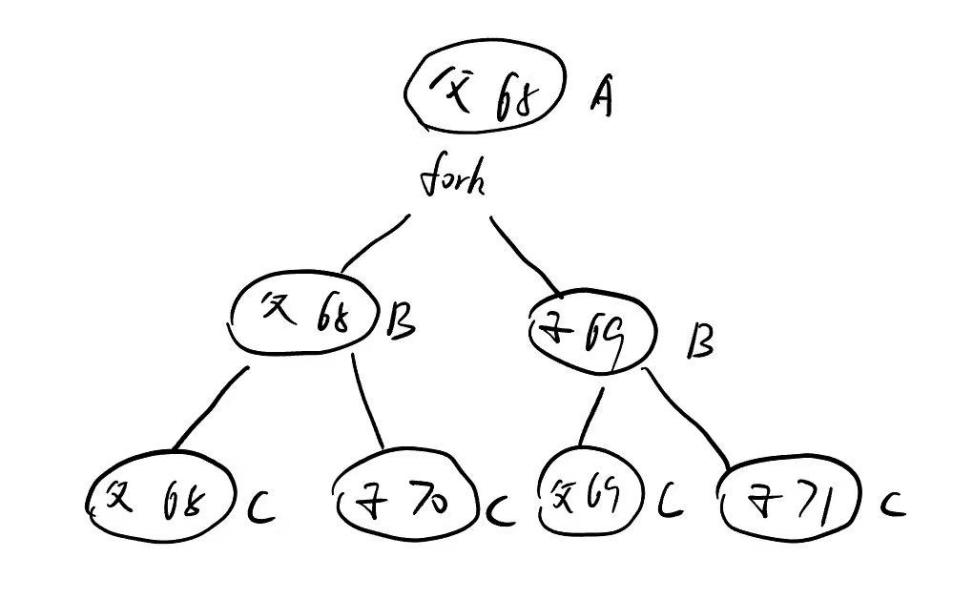
**可以看到由一个进程fork后再fork，共创建了2个进程。**

**首先打印before fork，然后执行了两次fork，非顺序挨个打印父进程和创建的两个子进程的内容，最后再回到父进程打印内容，最后打印before return**

**第二部分家族树**

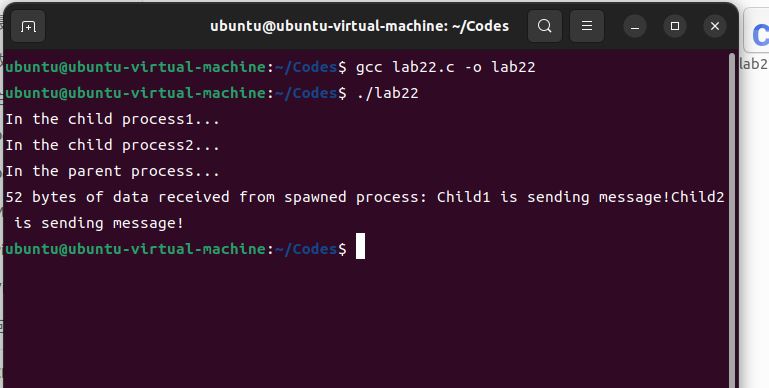
****

**（1）一共创建了三个子进程，PID分别为10969、10970、10971**

**（2）**

**（3）进程10968 打印A、B、C各一次，fork了两次；进程10969打印B、C各一次，fork一次；进程10970和10971各打印C一次**

**第三部分管道通信**

****

**创建子进程后关闭读，父进程关闭写，通过缓冲区buf通信**

## 实验三 一个进程启动另一个程序的执行

#### 一、实验目的

在Linux环境系统中，**execve系统调用**用于执行一个程序（可执行二进制文件或脚本）。**exec函数家族，包括execl、execlp、execle、execv、execvp**，是execve系统调用的前端。本实验要求学生学习在一个进程中启动另一个程序执行的基本方法，了解execve系统调用和exec函数家族的使用方法。

#### 二、实验内容

（一）初步认识“在一个进程中启动另一个程序的执行”。

1、编辑一个源程序dummy.c，并编译为可执行程序dummy。

// dummy.c

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

int main(int argc, char\*\* argv) {

int result;

printf("\nYou are now in a running program \"%s\". \n", argv[0]);

printf("My PID is %d. My parent's PID is %d.\n", getpid(), getppid());

printf("Please input an integer (0-255), which will be returned to my parent process:\n");

scanf("%d", &result);

printf("Goodbye.\n\n");

return (result & 0377);

}

2、再编辑一个源程序exec\_test.c，并编译为可执行程序exec\_test。

// exec\_test.c

#include <stdio.h>

#include <stdlib.h>

#include <error.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

int main(int argc, char\*\* argv) {

int result;

result = fork();

if (result < 0) {

perror("Failed to create child");

exit(1);

} else if (result == 0) {

// Child 1

char \*cmd = "./dummy";

printf("Child process's PID is %d. My parent's PID is %d.\n", getpid(), getppid());

printf("Child process is about to execute \"%s\"\n\n", cmd);

result = execlp(cmd, cmd, NULL);

if (result == -1) {

perror("In child process, failed to exec a program");

}

exit(0);

} else {

// parent

int status;

printf("Parent process's PID is %d.\n", getpid());

printf("Parent process is waiting ... \n");

wait(&status);

printf("In parent process, status = 0x%x, WEXITSTATUS(status) = %d (i.e. 0x%x)\n", \

status, WEXITSTATUS(status), WEXITSTATUS(status));

}

return (EXIT\_SUCCESS);

}

3、先执行dummy，观察、分析执行结果；然后执行程序exec\_test，观察、分析执行结果。（注意，两个可执行程序都在当前目录下）

（二）实现一个简单的命令解释外壳（Shell）。

**1、基本功能**：

（1）从标准输入读取一行字符串，其中包含欲执行的命令和它的命令行参数（如果有的话）。提示：需要将输入的一行字符串进行拆分，以**空格**、**制表符**（\t）作为分隔符，分解为命令、命令行参数（零个或多个）。如果用户输入的命令是“quit”，则退出执行。

（2）创建一个子进程。

（3）在子进程中，执行在（1）读入的命令，如果有命令行参数，也要传递。

（4）在父进程中，等待子进程结束，然后打印子进程的返回值。

（5）在父进程中，控制转移至（1）。

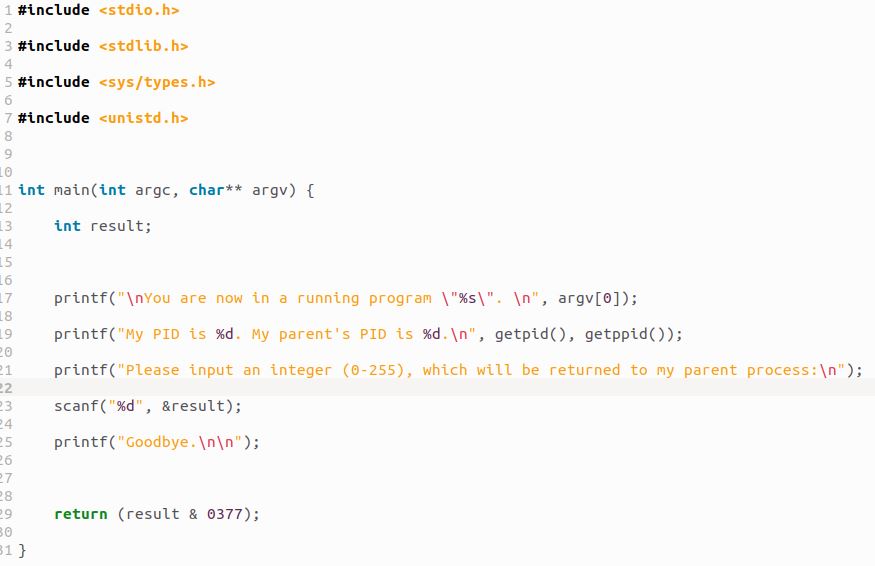
**2、扩展功能（选作）**：实现子进程的输入/输出重定向。

#### 三、实验要求

按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

1. **实验设计**

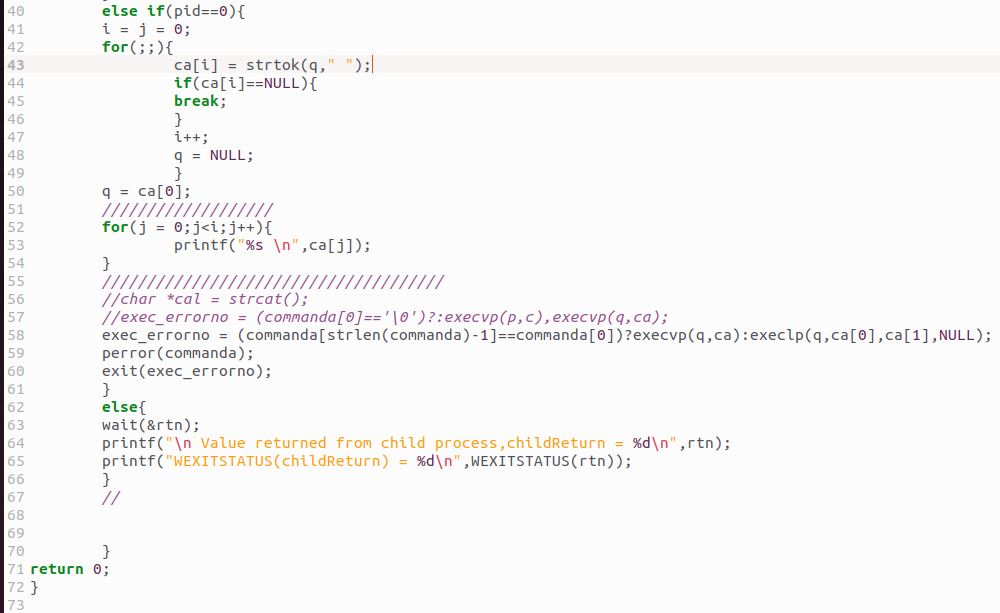
**第一部分dummy和exectest**

****

****

**myshell：**

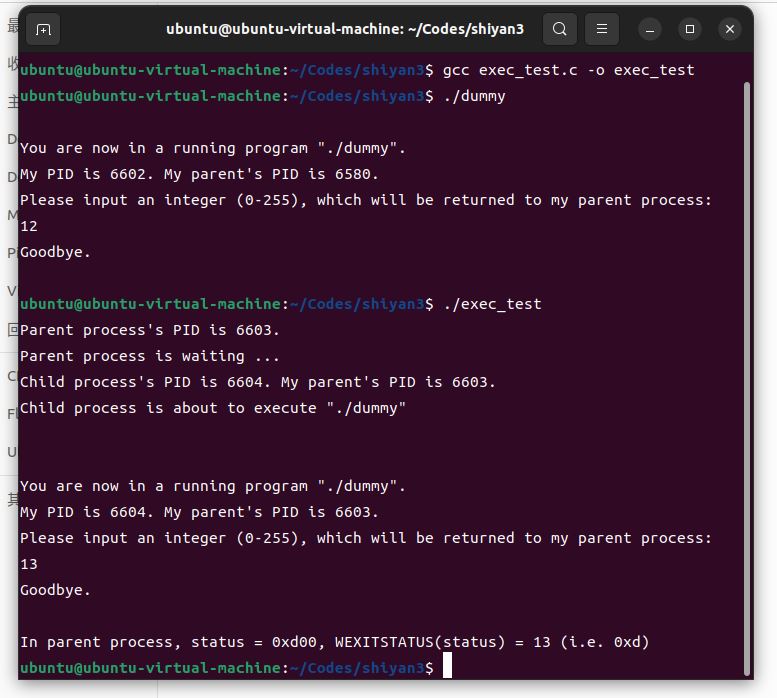
****

****

1. **实验测试结果及分析**

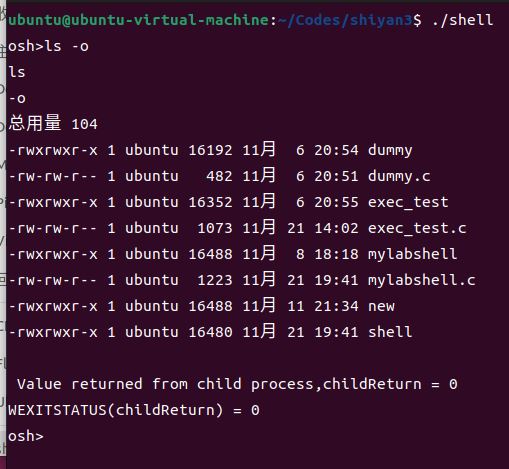
**dummy.c**

**Exec\_test.c**

****

**可以看到exec\_test里面调用了dummy.c并作了打印输出**

**Myshell.c**



本来是全部获取再传递数组execvp，后来发现不太好区分有无参数，看源码后发现linux参数缓冲空间有大小上限128，MAX\_ARG\_PAGES、BINPRM\_BUF\_SIZE

调用shell以后，启动了另一个进程，通过输入输出在另一个程序中找到可执行文件，替换原本进程的段

l传参为列举

v传参为数组

e传递环境变量

p自动查找环境变量

## 实验四 线程管理

#### 一、实验目的

编写Linux环境下的POSIX多线程程序，了解多线程的程序设计方法，掌握最常用的三个函数pthread\_create、pthread\_join和pthread\_exit的用法。

#### 二、实验内容

1、在主程序中创建两个线程，myThread1和myThread2。每个线程打印一句话。

提示：使用pthread\_create(&id, NULL, (void \*) thread, NULL)完成。先定义每个线程的执行体，然后在main( )中创建两个线程，最后主线程等待子线程结束后再退出。

2、创建三个线程，分别向线程传递如下三种类型的参数并在线程中打印：

* 传递整型值
* 传递字符
* 传结构体类型

*3、*（选作）利用多线程实现矩阵的乘法，思考在矩阵的乘法过程中，哪一部分适合用多线程。

#### 三、实验要求

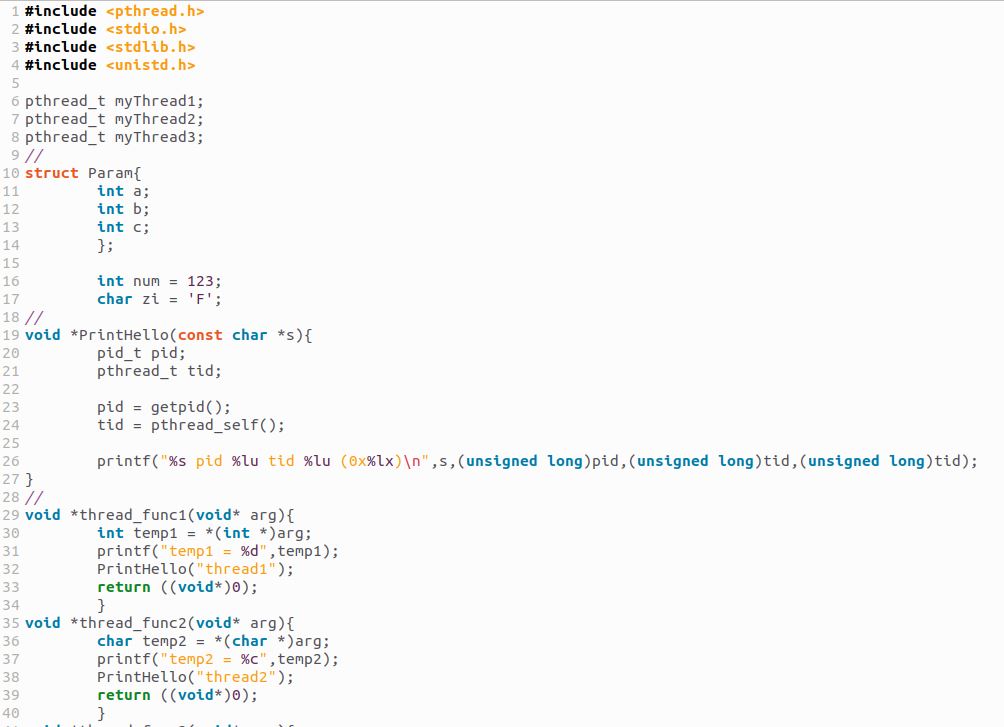
按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

**四、实验设计**

第一部分创建线程：



第二部分传递参数：

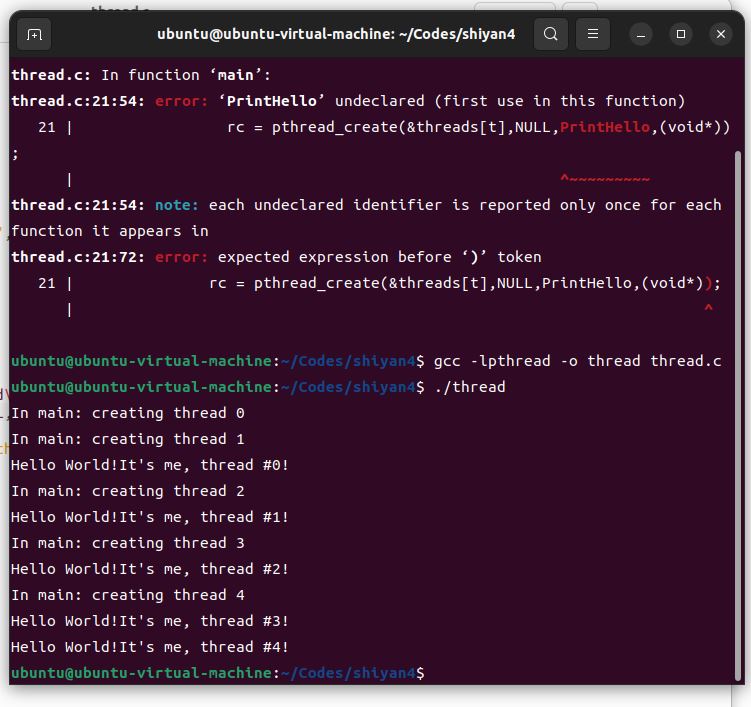




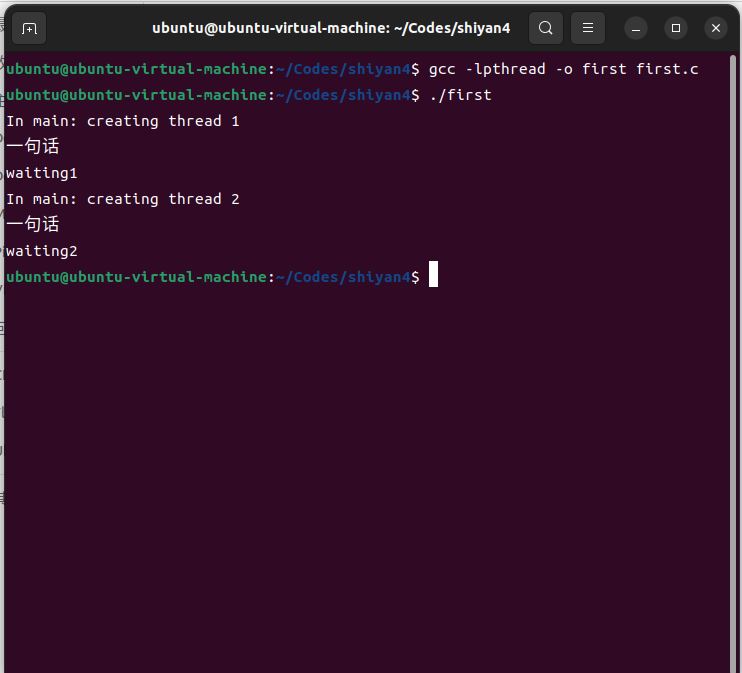


**五、实验测试结果及分析**

示例程序：



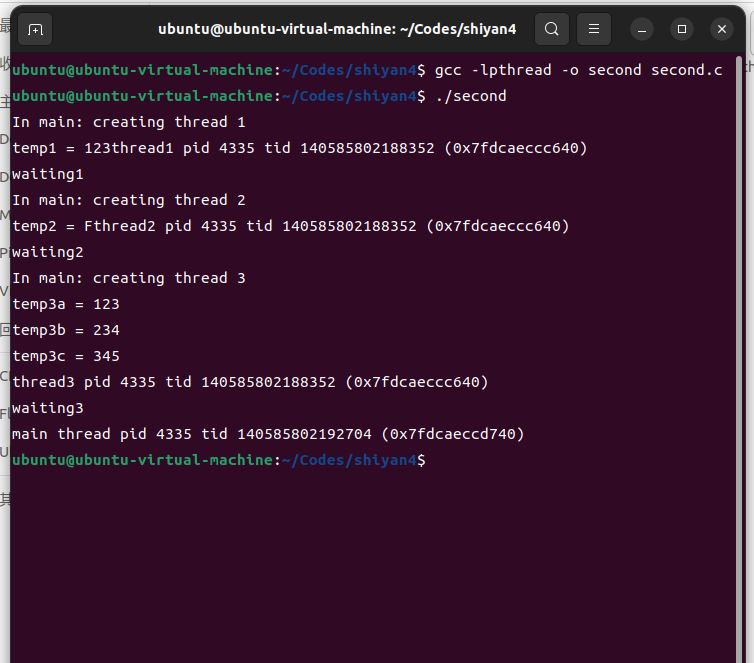
创建线程：



两次pthread\_create建立两个线程，分别触发相应的函数打印

主程序等待利用pthread\_join实现

传递参数：



与第一个部分类似，只是在各自pthread\_create创建线程时传入定义好的不同的参数

## 实验五 线程间的互斥与同步

实验学时：2学时

实验类型：验证、设计型

#### 一、实验目的

理解POSIX线程（Pthread）互斥锁和POSIX信号量机制，学习它们的使用方法；编写程序，实现多个POSIX线程的同步控制。

#### 二、实验内容

创建6个POSIX线程实现垃圾分类。其中2个线程(A和B)不停的随机产生4种垃圾（厨余垃圾、可回收垃圾、有害垃圾、其他垃圾）放入一个垃圾缓冲池. 缓冲池由n个缓冲区构成（n=5，并可以方便地调整为其他值），每个缓冲区可以存放一包垃圾。另外4个线程为垃圾分类线程，C、D、E、F，各从垃圾缓冲池中获取相应类型的垃圾，获得10包垃圾后，将垃圾打包，送到相应的垃圾处理厂（输出提示信息即可）。

提示：

（1）在创建6个线程当中，A和B是生产者，负责将垃圾送到公共的缓冲区，C、D、E和F是消费者，从缓冲区取得相应类型垃圾后进行累加，累加到10输出信息。使用互斥锁和信号量控制这些线程的同步。

（2）设计垃圾数据结构体。

在开始设计和实现之前，务必认真阅读下列内容：

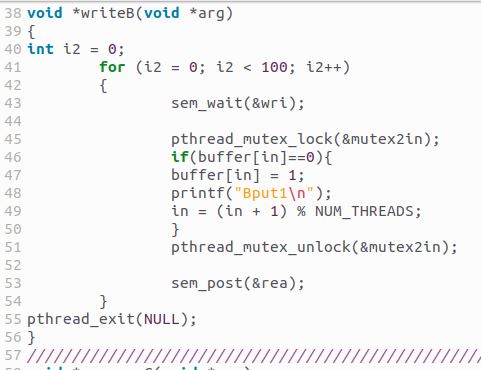
* **课本6.8.4节；**
* **讲义（课堂PPT）中关于“生产者-消费者问题”的部分；**
* **课本第6章后面的编程项目——生产者-消费者问题。**

#### 三、实验要求

按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

1. **实验设计**

****

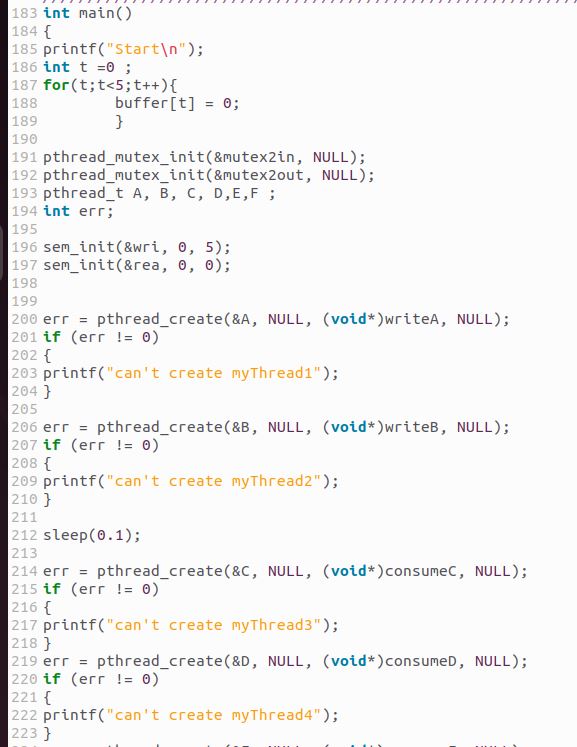
****

****

****

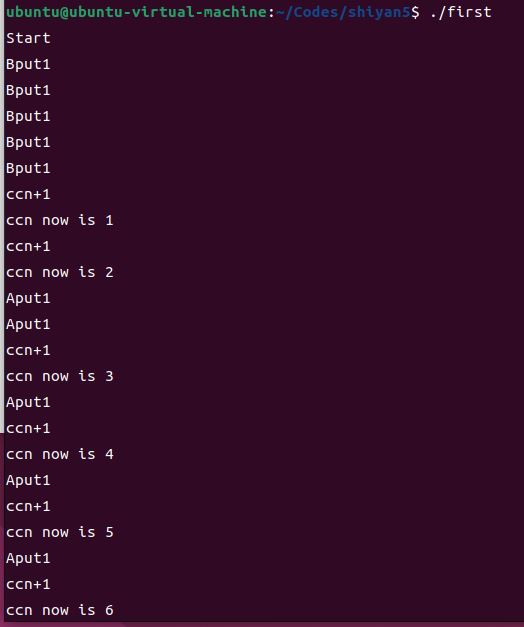
****

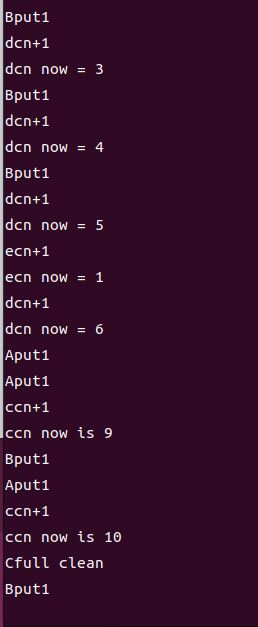
****

****

**设置两个互斥锁两个信号量，分别设置**

1. **实验测试结果及分析**

****

**此图中B收集够垃圾清空了，结果不唯一**

## 实验六 基于消息队列和共享内存的进程间通信

#### 一、实验目的

“系统V进程间通信”（System V IPC，System V interprocess communication）机制包括：消息队列、信号量集和共享存储区。本实验的目的是了解和熟悉**消息通信机制**、**共享存储区**的原理，以及它们的使用方法。

#### 二、实验内容

1、消息的创建、发送和接收：

使用系统调用msgget()、msgsnd()、msgrcv()、msgctl()编制长度为1K的消息的发送和接收程序，消息的内容是“A man is not old as long as he is seeking something. A man is not old until regrets take the place of dreams.”。

2、共享存储取得创建、附接和断接：

使用系统调用shmget()、shmat()、shmctl()，编制一个与上述功能相同的程序。

#### 三、实验要求

按照要求编写程序，放在相应的目录中，编译成功后执行，并按照要求分析执行结果，并写出实验报告。

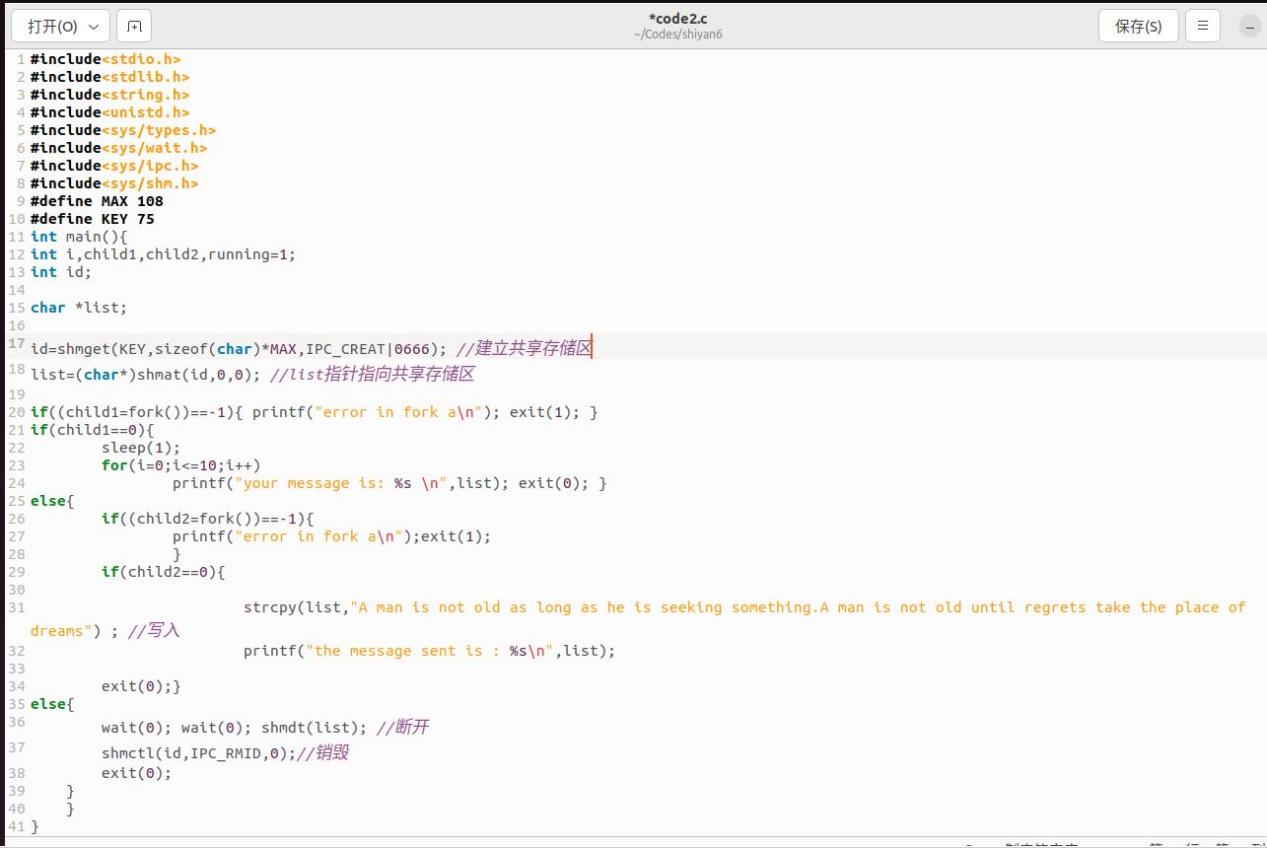
1. **实验设计**

第一部分：

****

****

第二部分：

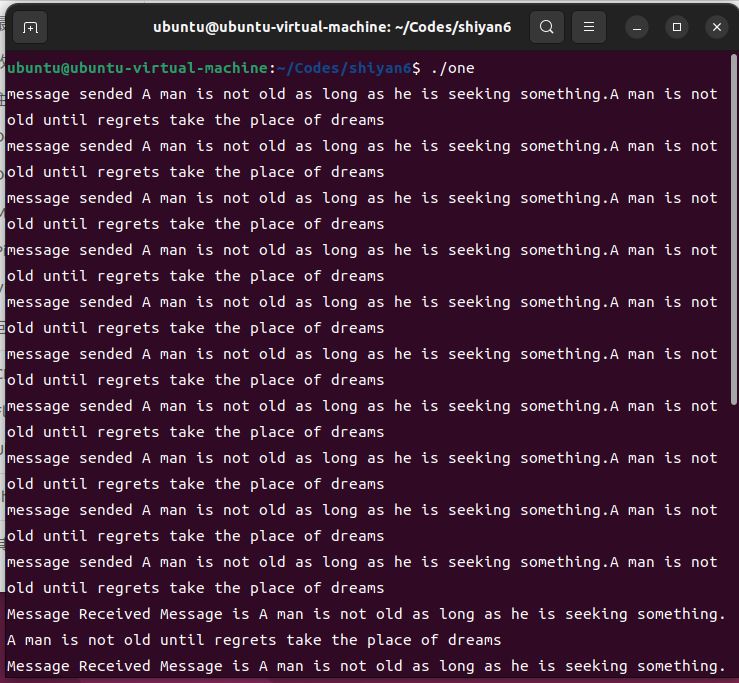


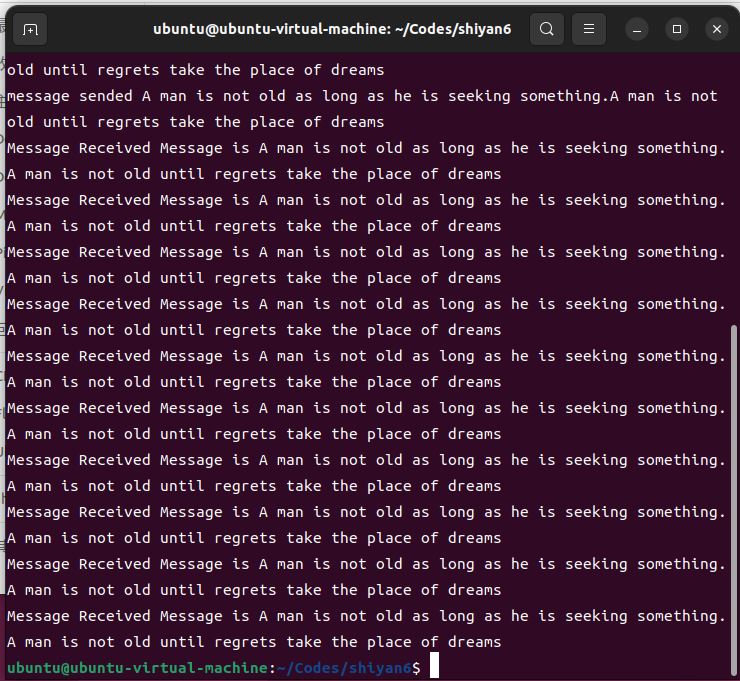
1. **实验测试结果及分析**

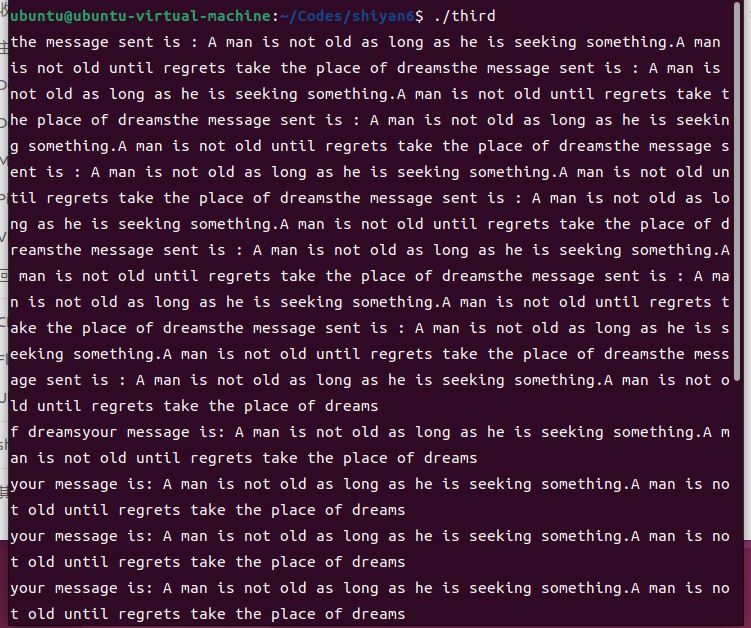
**第一部分：**

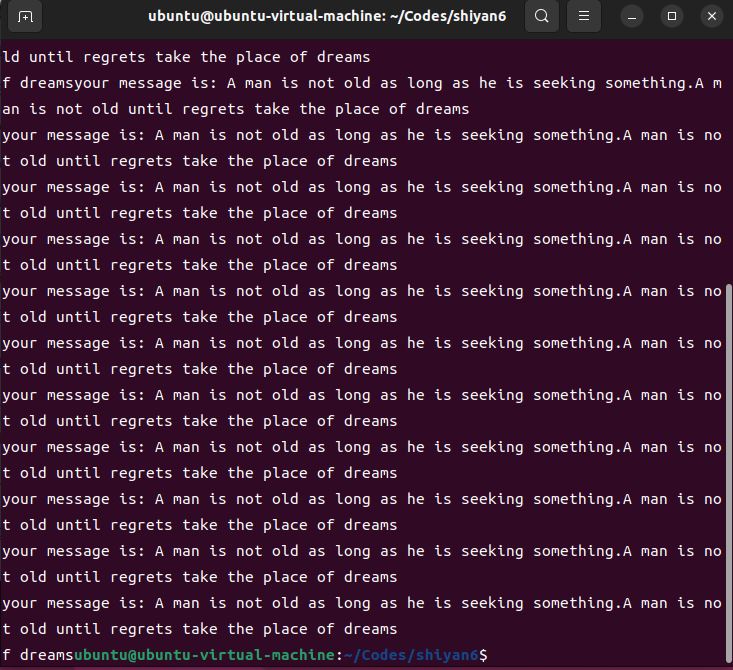
10个信息发送，receive接收到停止信号1时终止

**通过msgsnd和msgrcv发送接收，msgctl删除**

****

****

第二部分：



Shmget建立共享储存区，shmat使得指针指向共享储存区，shmctl删除