

《高级操作系统Linux》实验报告



**实验【5】： Linux环境下的C语言编程**

**学 生：**  **白乐荣**

**学 号： 201606060101**

**学 院： 电子信息与人工智能学院**

**系 别： 计算机系**

**专 业： 计算机科学与技术**

**《高级操作系统Linux》**实验预习报告

学号 201606060101 姓名 白乐荣 班级 计本163

实验日期： 2019 年 6 月 5 日 成绩：

一、实验目的

1. 学会用系统调用和库函数进行编程
2. 加深对进程概念的理解，明确进程和程序的区别。
3. 进一步认识并发执行的实质。
4. 分析进程竞争资源现象，学习解决进程互斥的方法。
5. 了解Linux系统中进程通信的基本原理。

二、实验内容

1. 编写一段C语言程序使其完成：父进程创建两个子进程，每个进程都在屏幕上显示自己的进程ID号。

运行结果示例：

# ./forktest

this is child1:16602

this is child2:16603

this is parent:16601

1. 上机调试下面的程序，观察运行结果，分析原因。

#include <sys/types.h>

#include <unistd.h>

int glob=3;

int main()

{

pid\_t pid;

int loc=3;

printf("befor fork():glob=%d,loc=%d\n", glob, loc);

if ((pid=fork()) < 0)

{

printf("fork() error\n");

exit(0);

}

else if (pid == 0)

{

glob++;

loc--;

printf("child:glob=%d, loc=%d\n", glob, loc);

else

{

printf("parent:glob=%d, loc=%d\n", glob, loc);

exit(0);

}

return 0;

}

1. 利用两个管道进行双向通信，实现父子进程协作把整数x从1加到10。

运行结果示例：

# ./pipetest

child 5938 read: 1

parent 5937 read: 2

child 5938 read: 3

parent 5937 read: 4

child 5938 read: 5

parent 5937 read: 6

child 5938 read: 7

parent 5937 read: 8

child 5938 read: 9

parent 5937 read: 10

* 说明：5938和5937分别是子进程和父进程的pid

1. 利用消息队列实现父子进程的通信。父进程创建消息队列，并向消息队列发送消息；子进程接受消息；父进程等待子进程接收消息后删除消息队列。

运行结果示例：

# ./msgtest

Parent:Send to the message queue successfully!

The message sent is :This is the input!

Child:Receiving from the message queue:This is the input!

三、预习心得

通过预习，我简单的了解到了一些c语言的基础语法，以及父子进程之间的关系，进程间通信等相关知识。

**《高级操作系统Linux》**实验报告

学号 201606060101 姓名 白乐荣 班级 计本163

实验日期： 2019 年 6 月 5 日 成绩：

一、实验目的

1. 学会用系统调用和库函数进行编程
2. 加深对进程概念的理解，明确进程和程序的区别。
3. 进一步认识并发执行的实质。
4. 分析进程竞争资源现象，学习解决进程互斥的方法。
5. 了解Linux系统中进程通信的基本原理。

二、实验内容

1. 编写一段C语言程序使其完成：父进程创建两个子进程，每个进程都在屏幕上显示自己的进程ID号。

运行结果示例：

# ./forktest

this is child1:16602

this is child2:16603

this is parent:16601

1. 上机调试下面的程序，观察运行结果，分析原因。

#include <sys/types.h>

#include <unistd.h>

int glob=3;

int main()

{

pid\_t pid;

int loc=3;

printf("befor fork():glob=%d,loc=%d\n", glob, loc);

if ((pid=fork()) < 0)

{

printf("fork() error\n");

exit(0);

}

else if (pid == 0)

{

glob++;

loc--;

printf("child:glob=%d, loc=%d\n", glob, loc);

else

{

printf("parent:glob=%d, loc=%d\n", glob, loc);

exit(0);

}

return 0;

}

1. 利用两个管道进行双向通信，实现父子进程协作把整数x从1加到10。

运行结果示例：

# ./pipetest

child 5938 read: 1

parent 5937 read: 2

child 5938 read: 3

parent 5937 read: 4

child 5938 read: 5

parent 5937 read: 6

child 5938 read: 7

parent 5937 read: 8

child 5938 read: 9

parent 5937 read: 10

* 说明：5938和5937分别是子进程和父进程的pid

1. 利用消息队列实现父子进程的通信。父进程创建消息队列，并向消息队列发送消息；子进程接受消息；父进程等待子进程接收消息后删除消息队列。

运行结果示例：

# ./msgtest

Parent:Send to the message queue successfully!

The message sent is :This is the input!

Child:Receiving from the message queue:This is the input!

三、源程序清单、测试数据及结果。

1. 编写一段C语言程序使其完成：父进程创建两个子进程，每个进程都在屏幕上显示自己的进程ID号。

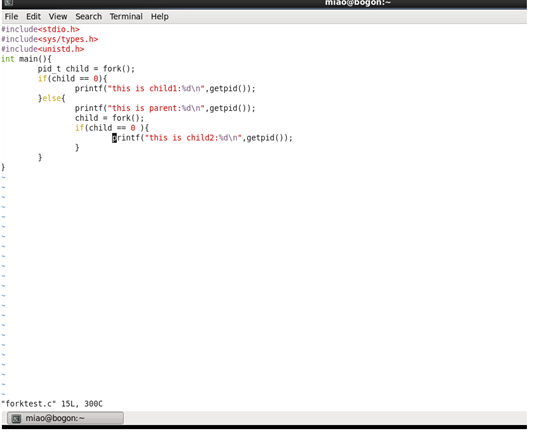
运行结果示例：

# ./forktest

this is child1:16602

this is child2:16603

this is parent:16601



1. 上机调试下面的程序，观察运行结果，分析原因。

#include <sys/types.h>

#include <unistd.h>

int glob=3;

int main()

{

pid\_t pid;

int loc=3;

printf("befor fork():glob=%d,loc=%d\n", glob, loc);

if ((pid=fork()) < 0)

{

printf("fork() error\n");

exit(0);

}

else if (pid == 0)

{

glob++;

loc--;

printf("child:glob=%d, loc=%d\n", glob, loc);

else

{

printf("parent:glob=%d, loc=%d\n", glob, loc);

exit(0);

}

return 0;

}



原因：子进程会复制父进程的所有资源，包括属性，且子进程的fork()函数返回值为0，执行了glob++和loc--，值分别变为4和2，在父进程中则仍为3。

1. 利用两个管道进行双向通信，实现父子进程协作把整数x从1加到10。

运行结果示例：

# ./pipetest

child 5938 read: 1

parent 5937 read: 2

child 5938 read: 3

parent 5937 read: 4

child 5938 read: 5

parent 5937 read: 6

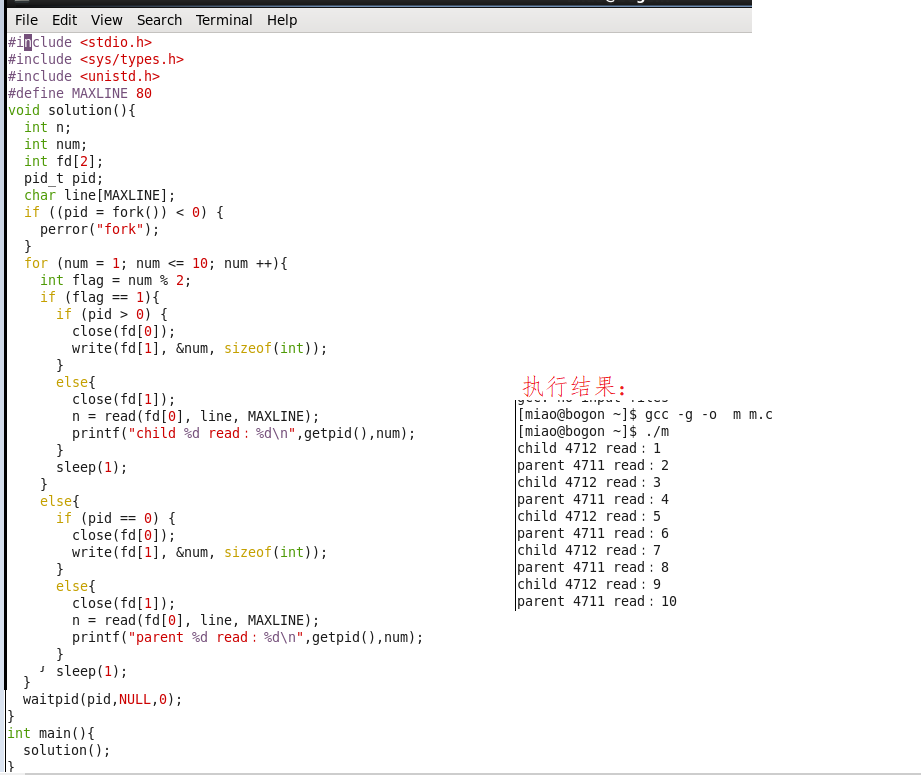
child 5938 read: 7

parent 5937 read: 8

child 5938 read: 9

parent 5937 read: 10

* 说明：5938和5937分别是子进程和父进程的pid



1. 利用消息队列实现父子进程的通信。父进程创建消息队列，并向消息队列发送消息；子进程接受消息；父进程等待子进程接收消息后删除消息队列。

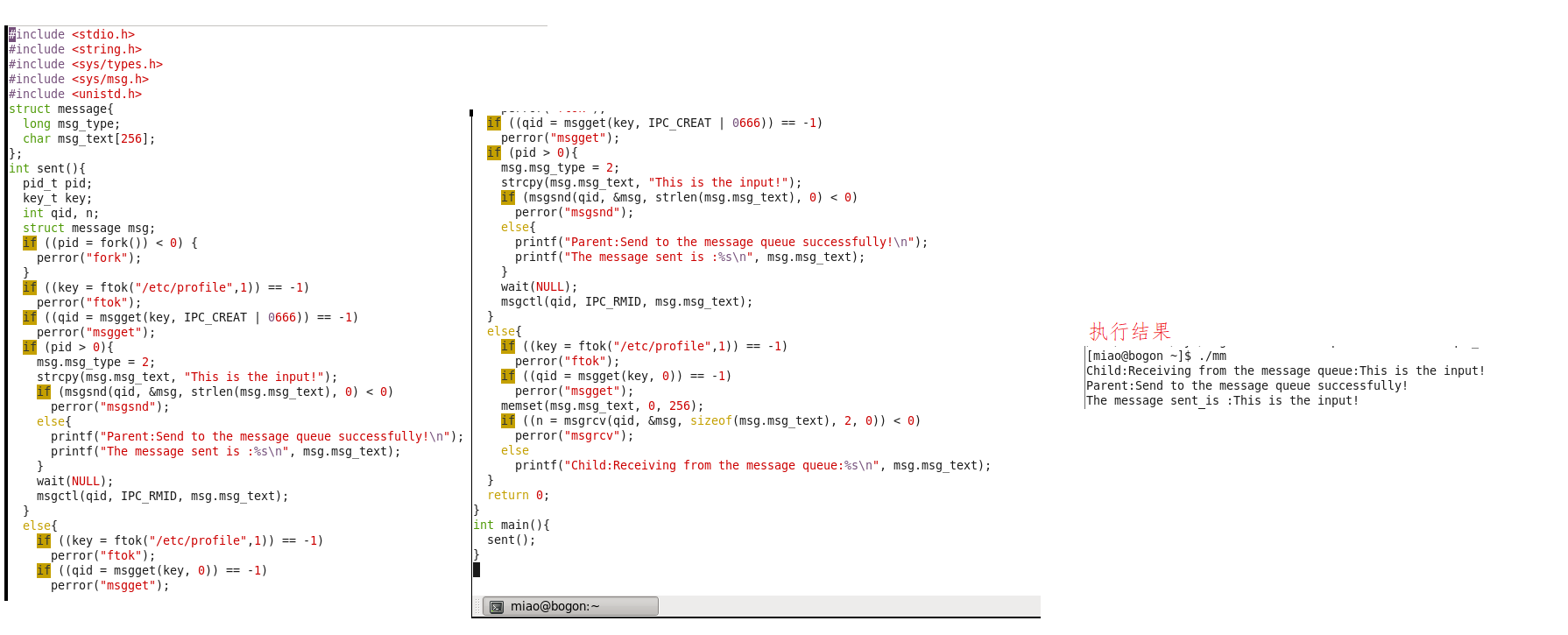
运行结果示例：

# ./msgtest

Parent:Send to the message queue successfully!

The message sent is :This is the input!

Child:Receiving from the message queue:This is the input!



四、实验小结

通过本次实验我在Linux学习到了一些基础的c语言编程，对进程间通信，父进程与子进程的关系以及区别都有了更深的认识，代码能力也有所提高。