实验3 进程控制与线程创建

实验目的

1. 掌握Linux中如何加载子进程自己的程序；

2. 掌握父进程通过创建子进程完成某项任务的方法；

3. 掌握系统调用exit()和\_exit()调用的使用；

4.掌握Linux线程的基本概念和使用。

实验内容

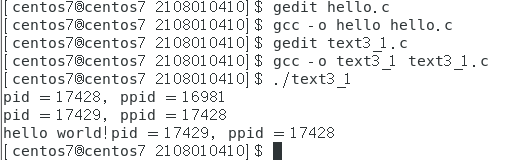
1. 设计两个程序hello.c和load.c，hello.c的功能是输出“hello world！”；load.c完成的功能是：创建一个子进程，子进程执行hello.c编译后的程序，父进程输出一行字符串（内容自定）后等待子进程结束（wait函数）。使用execl（）加载子进程的程序。子进程在execl前后各输出一次进程的pid，分析结果。

**要求：1.写出程序代码及运行截图**

1. **总结exec系列函数的作用**







hello程序在打出hello后会打出当前hello程序所在的pid和ppid

Text3\_1程序17428的子程序17429运行时候先输出pid和ppid再用execl运行hello程序，hello先输出hello之后输出pid和ppid，可以发现子程序运行了hello，hello的pid和ppid和text3\_1的子进程的pid和ppid一样，还可以发现子进程在运行hello程序后就不运行execl后面的输出语句error了

2．运行以下程序，注意程序的执行结果和printf函数（以行为单位进行缓冲的）的关系。

#include<stdlib.h>

#include <stdio.h>

#include <unistd.h>

int main()

{ printf("this is first line ");

sleep(2);

printf("this is the second line ");

sleep(2);

printf("this is the third line");

return 0;

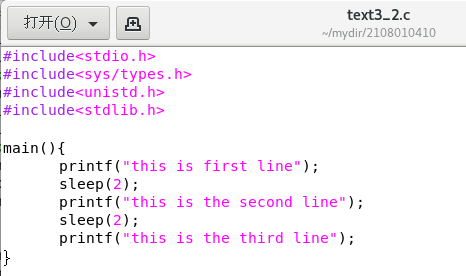
}

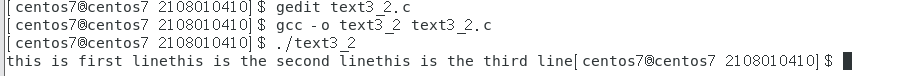
实验步骤：

（1）运行以上程序，查看运行结果

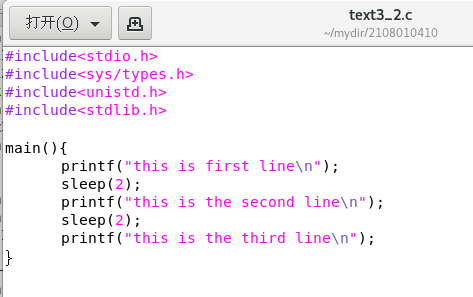
（2）在每条printf语句后都加上\n，再运行以上程序，观察两种结果的不同，分析printf缓冲的作用。

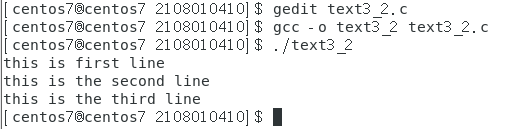
**要求：写出程序代码及运行截图**





printf先把这三行全都放入缓冲区，在进程结束的时候会把缓冲区全部输出





printf把一行字符串打入缓冲区，缓冲区在遇到\n的时候会把缓冲区的所有字符串都打出来，运行结果可以看到每输出一行等待2s后下一行输出

3. 运行以下程序，分析程序执行结果。

#include<stdlib.h>

#include <stdio.h>

#include <unistd.h>

main()

{ int p;

printf("This is the fisrt line");

p=fork();

if (p>0)

{ wait(0);

printf("this is parent ");

}

else{

printf("this is child first\n");

printf("this is child second");

\_exit(0);

}

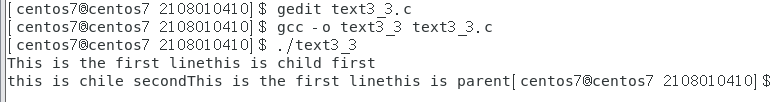
}

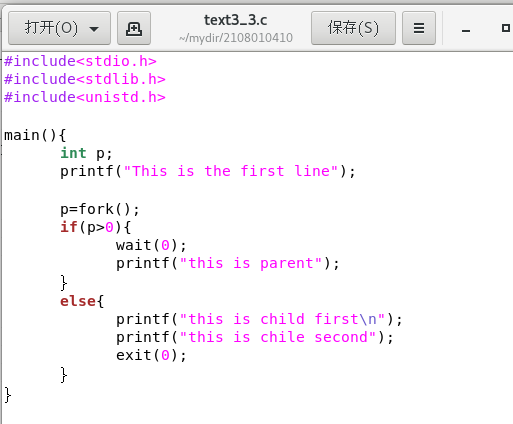
实验步骤：

（1）分析程序的执行结果。若第一条printf语句后加上\n结果如何？

（2）若exit(0)换为\_exit(0)结果会怎样？

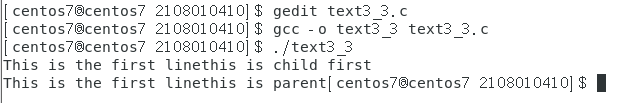
（3）分析printf和fork函数，printf和\_exit函数一起使用时特点。

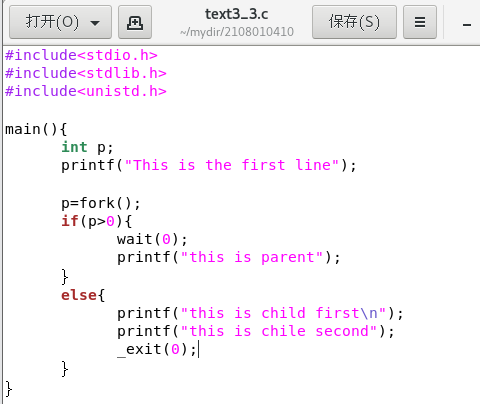




第一个printf把字符串放到缓存区，父进程创建子进程，子进程复制父进程的缓存区，子进程再printf的时候把childfirst放到缓存区，缓存区遇到\n后把子进程的缓冲区中从父进程继承来的first line和刚刚的child first一起输出，之后printf把chile second放到缓冲区，exit把缓冲区的内容写回磁盘中io输出文件并结束程序，io设备输出chile second，发送信号软中断SIGCHILD给父进程告诉父进程自己是一个僵尸程序了。

父进程的缓冲区在第一个printf后有first line，父进程等子进程结束后，运行第二个printf，父进程结束的时候，正常结束缓冲区的内容会被写入磁盘中io输出文件，io设备进行输出





父进程创建子进程后，子进程继承父进程的缓冲区，子进程的child first后加了\n缓冲区的内容会被输出，chile second后没有加\n，该字符串被写入缓冲区，\_exit不会把缓冲区的内容写到io输出对应的文件，程序直接结束，chile second不会输出。

父进程的缓冲区在第一个printf后有first line，父进程等子进程结束后，运行第二个printf，父进程结束的时候，正常结束缓冲区的内容会被写入磁盘中io输出文件，io设备进行输出

**要求：写出三种情况运行截图及结果分析**

1. 运行以下程序，回答问题。

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

void \*run(void \*ptr){

pid\_t pid;pthread\_t tid;

pid = getpid(); //获取进程id

tid = pthread\_self(); //获取线程id

printf("In thread,process id:%lu,thread id:%lu\n",(unsigned long)pid,(unsigned long)tid);

return 0;

}

int main(){

pthread\_t tid[3],mtid;

pid\_t pid;int i;

int ret=0;

for (i=0;i<3;i++){

ret=pthread\_create(&tid[i],NULL,run,NULL);

if(ret) {

printf("create thread failed ");

return 0;

}

}

for(i=0;i<3;i++) pthread\_join(tid[i],NULL);

pid = getpid(); //获取进程id

mtid = pthread\_self(); //获取线程id

printf("In main process id:%lu,thread id:%lu\n",(unsigned long)pid,(unsigned long)mtid);

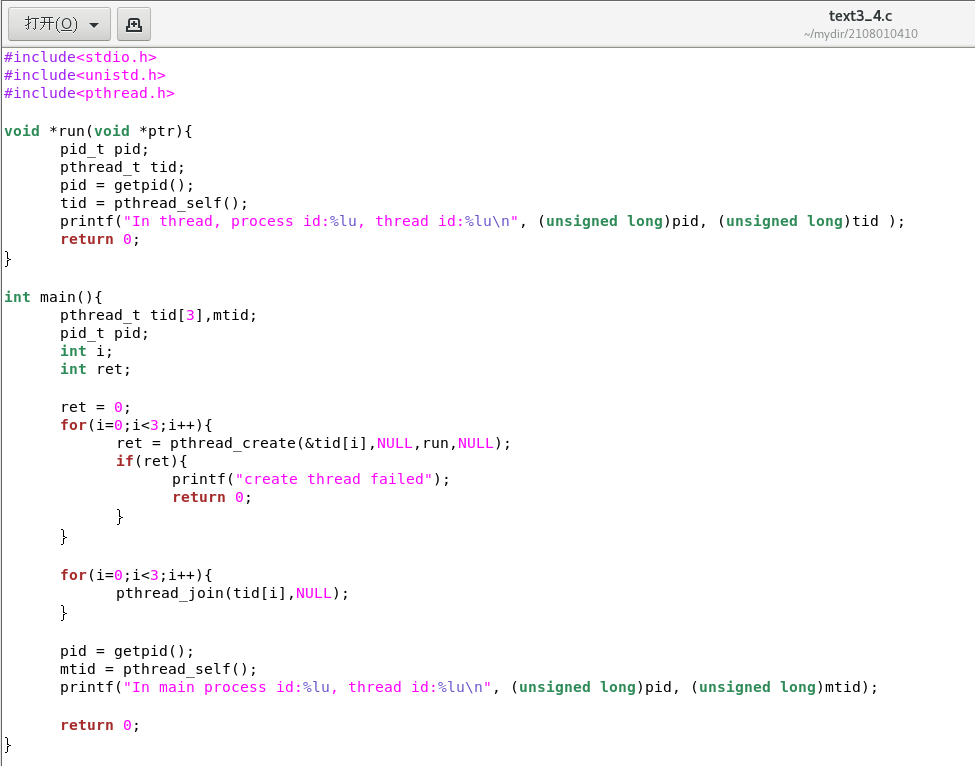
return 0;

}

1. 编辑程序thread.c，编译运行，运行结果是？

编译时使用如下命令：gcc -o thread thread.c -pthread

1. 根据程序结果分析，程序中的进程和线程有何关系？



**要求：写出运行截图及结果分析**