实验一 线程创建

预备知识：

进程相对线程的缺点：

1. fork是昂贵的。内存映像要从父进程拷贝到子进程，所有描述字要在子进程中复制等等。

2. 进程运行需要独立的环境中进行、进程切换需要做大量的数据交换会占用大量存储空间及CPU时间，需要一个程序运行切换时代价小的单位完成并行。因此提出“线程”。

引入线程后的好处：

* 改进程序的实时响应能力
* 可以有效的发挥多处理器的功能
* 减少对系统资源的频繁调度与切换

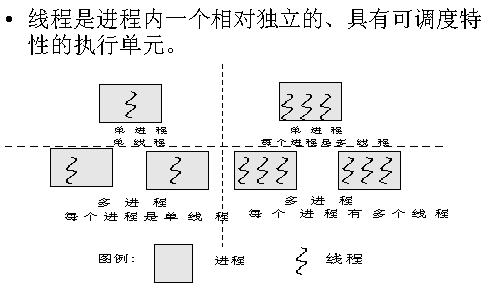
线程的优点与特征：

1. 创建线程比创建进程快10-100倍。
2. 一个进程中的所有线程共享相同的全局内存，这使得线程很容易共享信息
3. 一个进程中的所有线程不仅共享全局变量，而且共享：进程指令、大多数数据、打开的文件（如描述字）、信号处理程序、当前工作目录、用户ID和组ID。

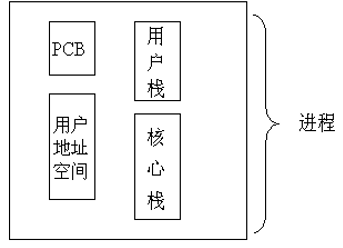
4.不同的线程可以使用进程内的变量来实现线程间数据的传递，而不需要复杂的IPC机制实现。

5.在同一个进程中的线程将并发的运行，这些线程将具有自己的“个性”。每个线程有自己的线程ID、寄存器集合（包括程序计数器和栈指针）、栈（用于存放局部变量和返回地址）、error、信号掩码、优先级。

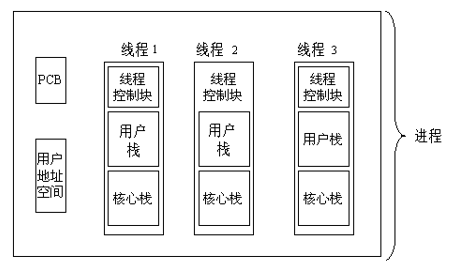
线程的进程模型：（进程与线程的对应关系）



单线程机制中的进程描述：



多线程机制中的进程描述：



Linux下的线程编程：

* linux的线程使用API位于libpthread.a和libpthread.so库中。
* 程序是只要使用头文件<pthread.h>即可。
* 编译使用，gcc －o threadName threadName.c －lpthread即可

或者 gcc threadName.c –lpthread –o threadName

一、实验目的： 1.学习Linux中创建线程的方法以及学习多线程编程

2.学习函数pthread\_create()的功能和使用方法

3.学习函数pthread\_attr\_init()的功能使用方法

4.学习函数pthread\_join ()的功能使用方法

5. 学习函数pthread\_exit ()的功能使用方法

6.学习atoi()函数的功能和用法

7.学习主函数带参数以及命令行传递参数程序设计

8.理解多线程并发执行

二、实验内容

1.将以下代码输入，在linux中运行，观察输出结果，理解程序功能和多线程并发执行过程。

#include <pthread.h>

#include <stdio.h>

int sum; /\* this data is shared by the thread(s) \*/

void \*runner(void \*param); /\* the thread \*/

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

{

pthread\_t tid; /\* the thread identifier \*/

pthread\_attr\_t attr; /\* set of attributes for the thread \*/

if (argc != 2) {

fprintf(stderr,"usage: a.out <integer value>\n");

/\*exit(1);\*/

return -1;

}

if (atoi(argv[1]) < 0) {

fprintf(stderr,"Argument %d must be non-negative\n",atoi(argv[1]));

/\*exit(1);\*/

return -1;

}

/\* get the default attributes \*/

pthread\_attr\_init(&attr);

/\* create the thread \*/

pthread\_create(&tid,&attr,runner,argv[1]);

/\* now wait for the thread to exit \*/

pthread\_join(tid,NULL);

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

}

/\*\*

\* The thread will begin control in this function

\*/

void \*runner(void \*param)

{

int i, upper = atoi(param);

sum = 0;

if (upper > 0) {

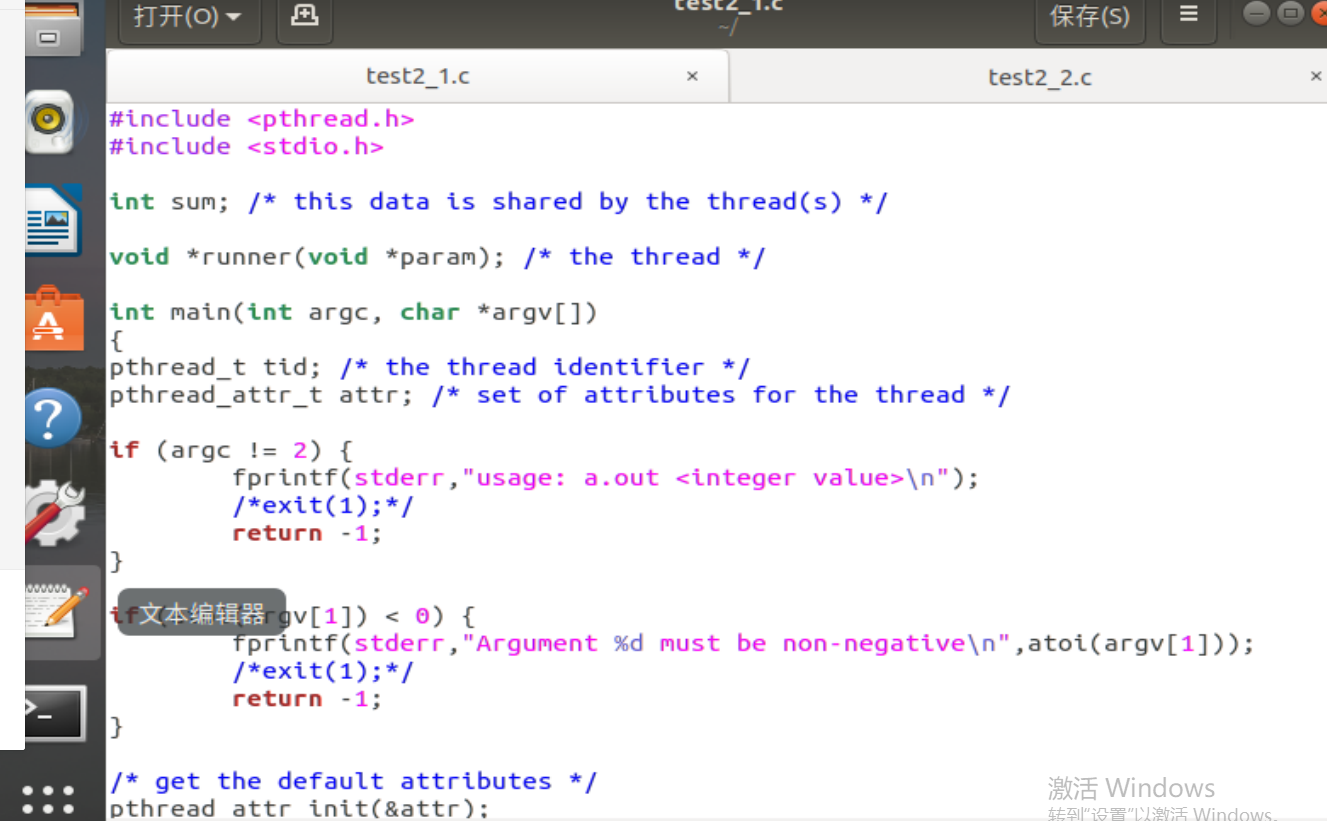
for (i = 1; i <= upper; i++)

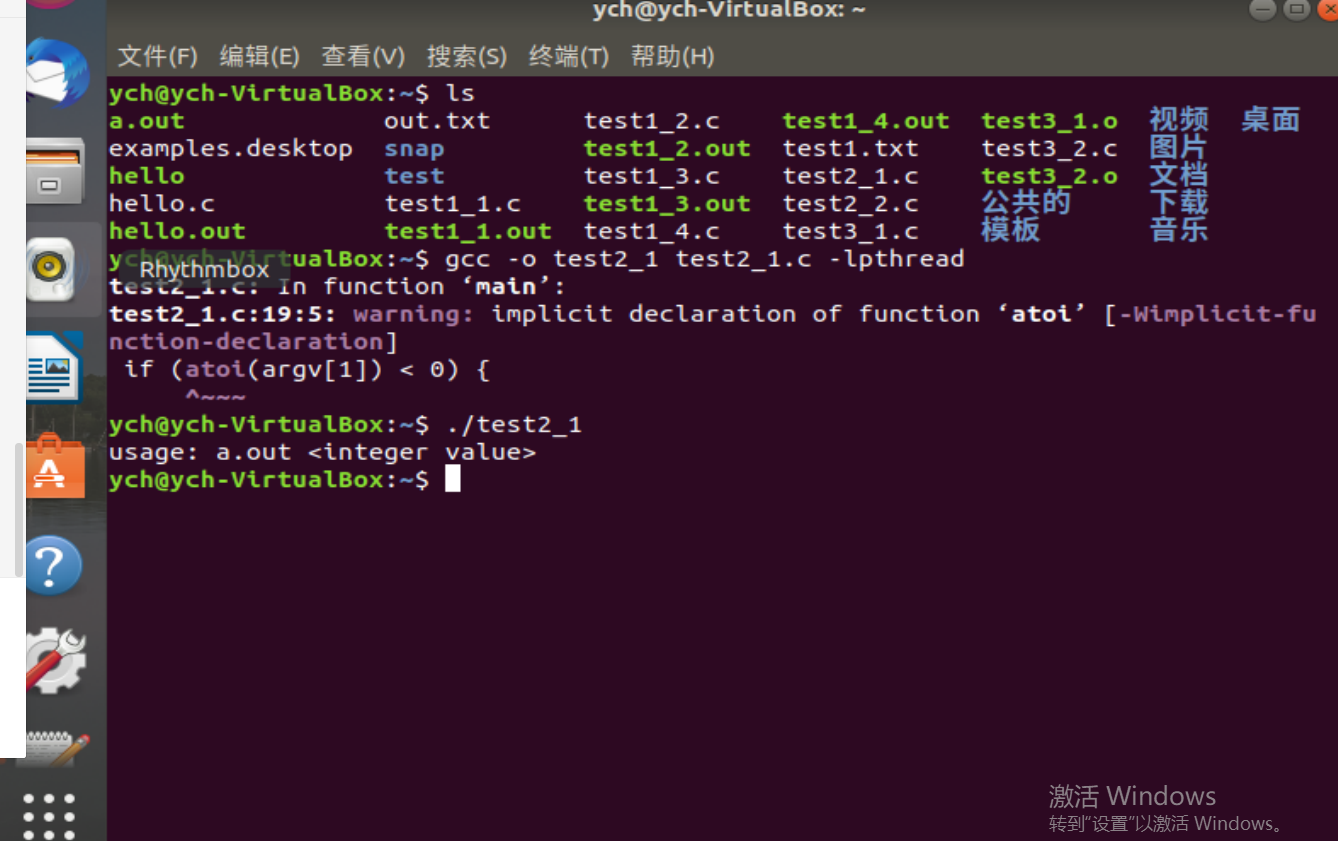
sum += i;

}

pthread\_exit(0);

} }





2.将以下代码输入，在linux中运行，观察LINE C与LINEP的输出结果，理解程序功能和多线程并发执行过程。

#include <pthread.h>

#include <stdio.h>

#include <sys/types.h>

int value = 0;

void \*runner(void \*param); /\* the thread \*/

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

{

pid\_t pid;

pthread\_t tid; pthread\_attr\_t attr;

pid = fork();

if (pid == 0) { /\* child process \*/

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,runner,NULL);

pthread\_join(tid,NULL);

printf("CHILD: value = %d",value); /\* LINE C \*/

}

else if (pid > 0) { /\* parent process \*/

wait(NULL);

printf("PARENT: value = %d",value); /\* LINE P \*/

}

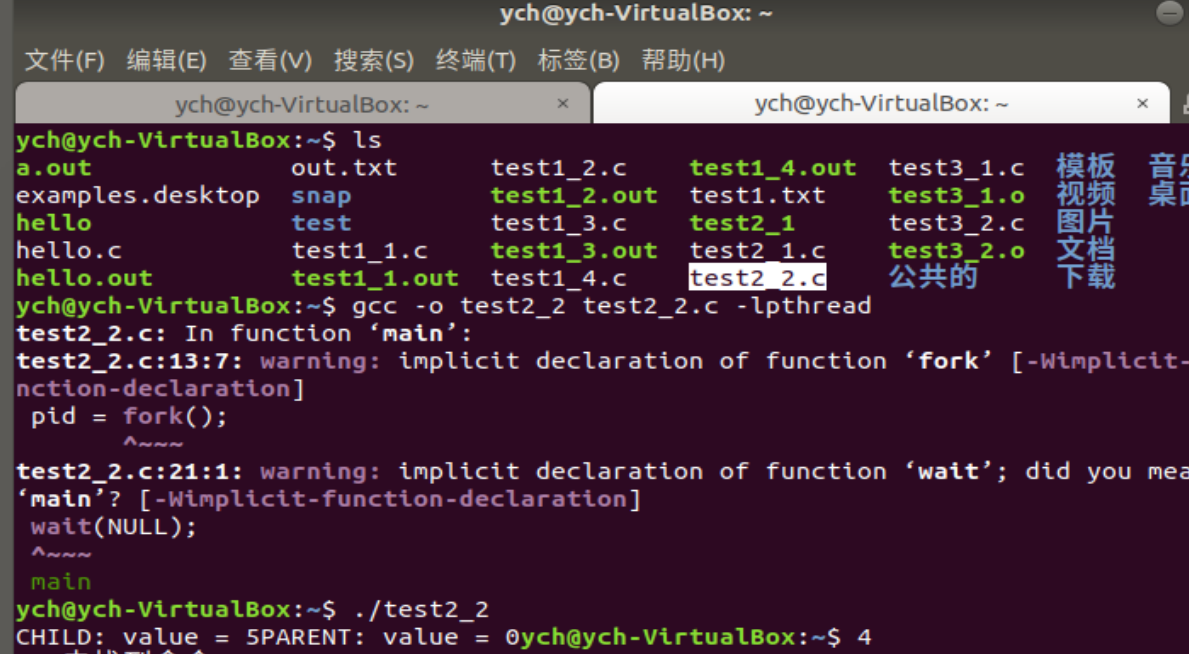
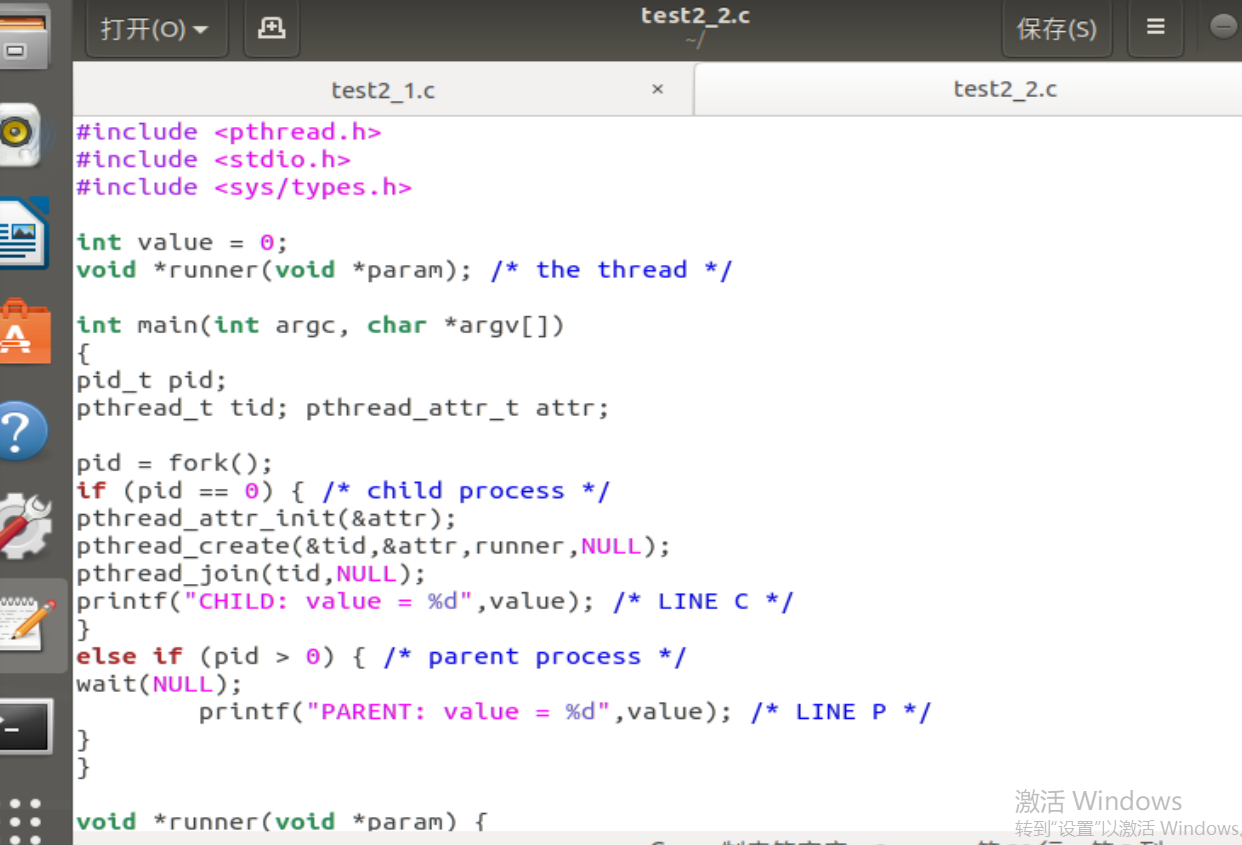
}

void \*runner(void \*param) {

value = 5;

pthread\_exit(0);

}



3.编写一个多线程程序，计算一组数字的多种统计值。要求：在父线程中定义用数组定义10个整数，把最小值、最大值和平均值变量作为全局变量，创建三个单独工作线程（子线程），第一个线程求数字的平均值，第二个线程求最大值。第三个线程求最小值。当工作线程退出时，父线程将输出这些值。

4.编写一个多线程来生成Fibonacci序列。这个程序工作如下：在命令行输入需要生成Fibonacci序列的长度；然后创建一个新的线程来产生Fibonacci数，把这个序列放到线程可以共享的数据结构中。父线程等待子线程完成后输出Fibonacci数列。

三、实验要求

1.独立完成

2.按模板撰写实验报告

3.于规定时间提交校网络教学平台