**1：**

1. 用户级线程：在用户空间内创建线程
2. 内核级线程：在内核空间内创建线程
3. 区别：操作系统不知道用户级线程的存在，一个进程只在一个处理器上执行，在用户级多线程中，一个线程阻塞，进程的其他线程也会阻塞。操作系统可以直接调度内核线程，一个进程上的多个线程可以分派到多个处理器上，并行执行。内核级线程的切换要进入内核态，用户级不用
4. 用户级线程更优：需要用户自定义调度算法，以及节约调度时间时
5. 内核级线程更优：交互式多个网页时，一个线程被阻塞了，其他线程还可以执行，快速响应请求

**2：**

跨内存共享：b,c

**3：**

创建了五个进程，两个线程

p->c1->c3->t2

->c5

->t1

->c4

->c2

**4：**P：PARENT: value = 0 //父子进程间有独立的内存和地址，则父进程的全局变量value

不被改变为0

C：CHILD: value = 5 //子进程传进线程函数的参数是变量value，但是在线程函数中，并未直接对传进来的参数进行操作，而是直接修改了全局变量value。如果对传进来的参数进行修改:首先：传进来的参数应该是指针类型，但是这里直接传了变量值。其次：如果传进来的参数确实能被修改，这里修改的应该只是线程拷贝的临时变量，因为并未传进value的地址

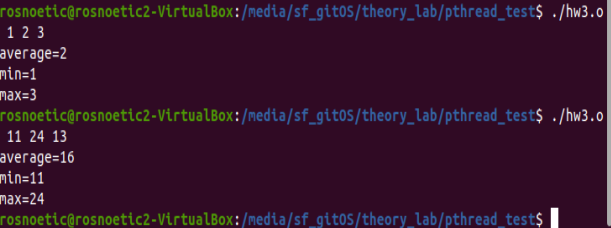
**5：**

A:用户线程>处理器>内核线程：此时线程调度只是内核线程上用户级别的线程调度，上下文切换速度快，但是用户级别的调度多

B:线程调度也只是用户级别上的调度，但是比A的内核线程多，能利用的处理器多，并行处理能力更强，用户级别线程的等待时间缩小

C：线程调度包括了内核级别和用户级别的线程调度，CPU利用率高，但是用时可能比B长，内核线程调度开销大

**选做：**执行效果如下：



#include <unistd.h>

#include<stdio.h>

#include<pthread.h>

#include <stdlib.h>

//全局变量//假设传入数据全为int型，且都为正数或者0，最大不超过1000

int aver=0;

int min=1000;

int max=-1;

void\* average(void\* data){ //求平均的线程

int num=\*((int\*)data);

int\* data\_int=(int\*)data;

//int sum=0;

int i=1;

for(i=1;i<num+1;i++){

aver+=data\_int[i];

}

aver/=num;

pthread\_exit(NULL);

}

void\* max\_my(void\* data){ //求最大值的线程

int num=\*((int\*)data);

int\* data\_int=(int\*)data;

//int sum=0;

int i=1;

int tmp;

for(i=1;i<num+1;i++){

tmp=data\_int[i];

if(tmp>max) max=tmp;

}

pthread\_exit(NULL);

}

void\* min\_my(void\* data){ //求最小值的线程

int num=\*((int\*)data);

int\* data\_int=(int\*)data;

//int sum=0;

int i=1;

int tmp;

for(i=1;i<num+1;i++){

tmp=data\_int[i];

if(tmp<min) min=tmp;

}

pthread\_exit(NULL);

}

续：

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

//从命令行输入数据

//传参最大数量100

int data[100];

data[0]=argc-1;//data[0]开头第一个放个数

int i=1;

for(i=1;i<argc;i++){

data[i]=atoi(argv[i]);//转为数字

}

pthread\_t tid;

//创建线程和执行线程函数

pthread\_create(&tid, NULL, average,(void\* )data);

pthread\_t tid2;

pthread\_create(&tid2, NULL, max\_my,(void\* )data);

pthread\_t tid3;

pthread\_create(&tid3, NULL, min\_my,(void\* )data);

pthread\_join(tid,NULL);

pthread\_join(tid2,NULL);

pthread\_join(tid3,NULL);

printf("average=%d\n",aver);

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

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

}