**[操作系统 第二次作业](https://bb.xjtu.edu.cn/webapps/assignment/uploadAssignment?content_id=_247348_1&course_id=_13345_1&group_id=&mode=cpview)**

1、下述程序，确定A、B、C、D四行中pid和pid1的值。（假设父进程和子进程的pid分别为2600和2603）

**#include <sys/types.h>**

**#include <stdio.h>**

**#include <unistd.h>**

**int main()**

**{**

**pid\_t pid,pid1;**

**pid=fork();**

**if (pid<0)**

**{**

**fprintf(stderr,"fork fail");**

**return 1;**

**}**

**else if (pid==0)**

**{**

**pid1=getpid();**

**printf("child:pid=%d",pid);      //A**

**printf("child:pid1=%d",pid1);     //B**

**}**

**else**

**{**

**pid1=getpid();**

**printf("parent:pid=%d",pid);     //C**

**printf("parent:pid1=%d",pid1);   //D**

**wait(NULL);**

**}**

**return 0;**

**}**

**A 0**

**B 2603**

**C 2603**

**D 2600**

**2. Discuss the difference between user-level thread and kernel level thread.**

**User-level thread s loaded entirely in user space, the kernel knows nothing about them. When threads are managed in user space, each process needs its own private thread table. The thread table consists of the information of the program counter, stack pointer, registers, state, etc. The thread table managed by the run-time system. When a thread is moved to a ready state or blocked state. The information needed to restart it is stored in the thread table.**

**In Kernel Level Thread, the kernel does total work of thread movement. There is no thread table in each process. The kernel has a thread table that keeps track of all the threads in the system. When a thread wants to create a new thread or destroy any existing thread, it makes a kernel call which takes the action.The kernel’s thread table holds each thread registers, state and other information. The information is the same as with user-level threads, but it is now in the kernel instead of user-space.**

**3.  Which of the following components of program state are shared across threads in a multithreaded process?**

**a. Register values**

**b. Heap memory**

**c. Global variables**

**d. Stack memory**

**BC**

**4. The program shown in Figure 4.11 uses the Pthreads API. What would be output from the program at LINE C and LINE P?**

**include <stdio.h>**

**#include <pthread.h>**

**int value=0;**

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

**int main()**

**{**

**int pid;**

**pthread\_t tid;**

**pthread\_attr\_t attr;**

**pid=fork();**

**if(pid==0)**

**{**

**pthread\_attr\_init(&attr);**

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

**pthread\_join(tid, NULL);**

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

**}else if(pid>0){**

**wait(NULL);**

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

**}**

**}**

**void \*runner(void \*param)**

**{**

**value=5;**

**pthread\_exit(0);**

**}**

**LINE C output: 5**

**LINE F output: 0**