操作系统原理与设计

第4章 Threads2（线程2）

陈香兰

中国科学技术大学计算机学院

October 28, 2009

提纲

Outline

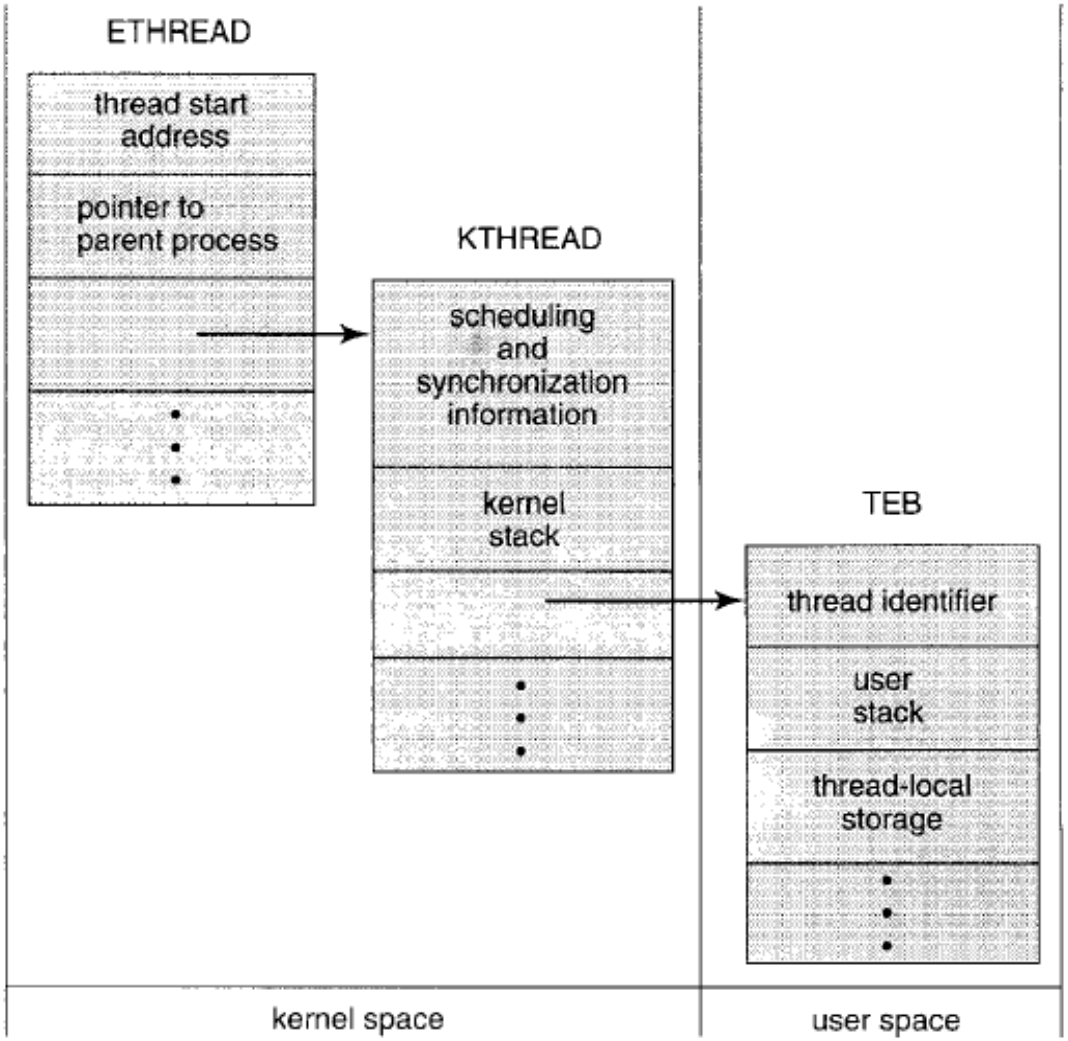
Windows XP Threads I

- An Windows XP application runs as a seperate process, and each process may contain one or more threads.
- Implements the **one-to-one mapping**
  - each user-level thread maps to an assotiated kernel thread
  - any thread belonging to a process can access the address space of the process
- Each thread contains
  - **A thread id**
  - **Register set**
  - **Separate user and kernel stacks**
  - **Private data storage area**
- The register set, stacks, and private storage area are known as the context of the threads

Windows XP Threads II

- The primary data structures of a thread include:
  - ETHREAD (executive thread block)
  - KTHREAD (kernel thread block)
  - TEB (thread environment block)

Windows XP Threads III



Linux Threads

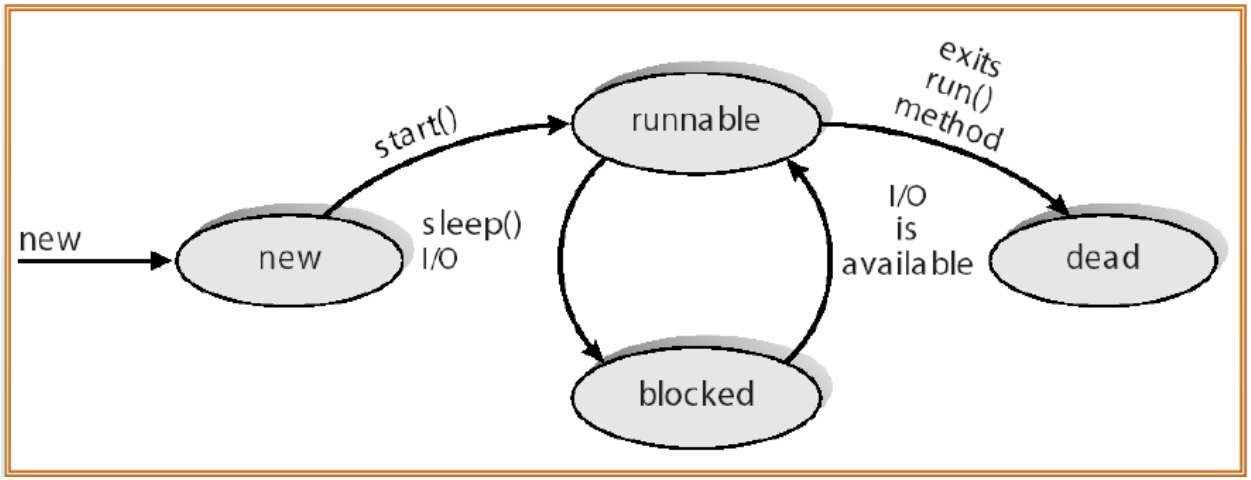
- Linux refers to them as **tasks** rather than threads
- Thread creation is done through **clone()** system call
- clone() allows **a child task to share the address space of the parent task (process)**
- clone() VS. fork()

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

Java Threads

- Java在**语言级**提供线程创建和管理支持功能
  - Java threads are managed by the JVM, not user-level library or kernel
- Java threads may be created by:
  - Extending Thread class
  - Implementing the Runnable interface Java

Thread States



Outline

Thread Scheduling

- user-level thread : kernel-level thread (or LWP)
- Local Scheduling – How the **threads library** decides which thread to put onto an available LWP
  - many-to-one, many-to-many models
  - **process-contention scope**, PCS
- Global Scheduling – How the **kernel** decides which kernel thread to run next
  - many-to-one, many-to-many & one-to-one models
  - **system-contention scope**, SCS

Pthread Scheduling API I

- POSIX Pthread API allows specifying either PCS or SCS during thread creation
  - PTHREAD_SCOPE_PROCESS, many-to-many
  - PTHREAD_SCOPE_SYSTEM, one-to-one
    - create and bind an LWP for each user-level thread
- example

Pthread Scheduling API II

```
#include <pthread.h>
#include <stdio.h>
#define NUM THREADS 5

int main(int argc, char *argv[])
{
    int i;
    pthread_t tid[NUM THREADS];
    pthread_attr_t attr;

    pthread_attr_init(&attr);       /* get the default attributes */

    /* set the scheduling algorithm to PROCESS or SYSTEM */
    pthread_attr_setscope(&attr, PTHREAD_SCOPE_SYSTEM);

    /* set the scheduling policy - FIFO, RT, or OTHER */
    pthread_attr_setschedpolicy(&attr, SCHED_OTHER);
```

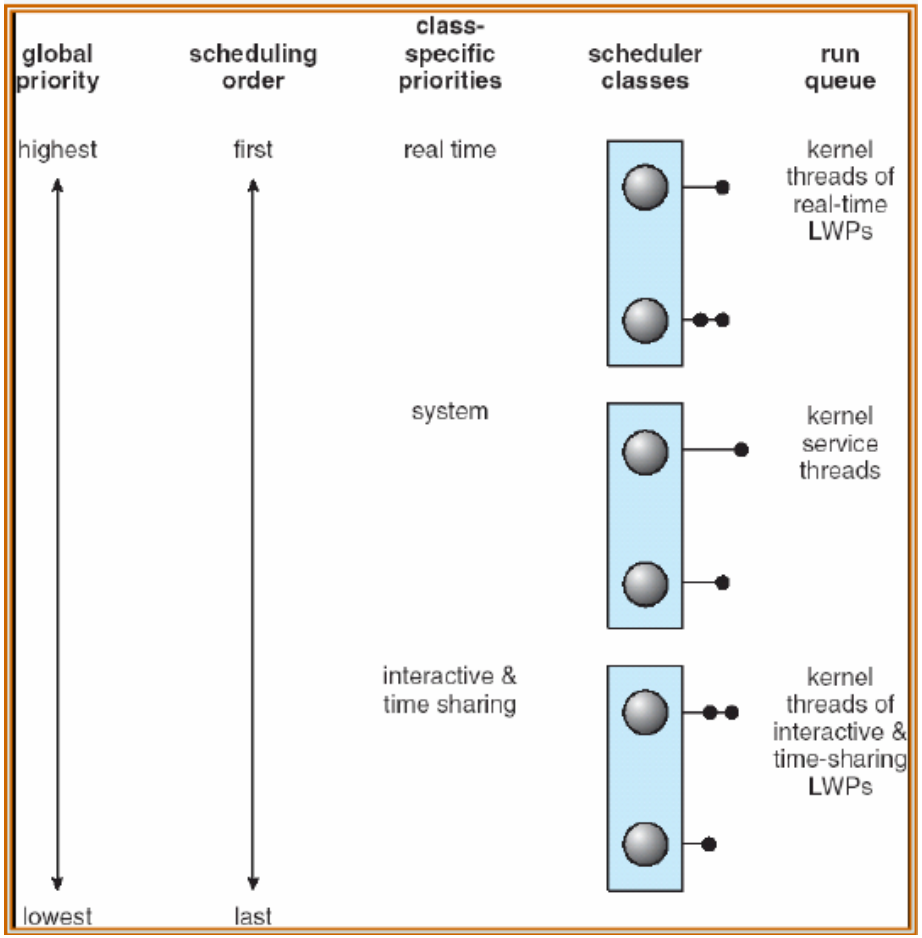
Pthread Scheduling API III

```
    for (i = 0; i < NUM_THREADS; i++)        /* create the threads */
        pthread_create(&tid[i],&attr,runner,NULL);

    for (i = 0; i < NUM THREADS; i++)        /* now join on each thread */
        pthread_join(tid[i], NULL);
}

/* Each thread will begin control in this function */
void *runner(void *param) {
    printf("I am a thread\n");
    pthread_exit(0);
}
```

Solaris scheduling I

- Solaris: **priority-based** thread scheduling
- 4 classes of scheduling, in order of priority
  - **Real time**
  - **System** (do not change the priority)
  - **Time sharing** (default, with a **multilevel feedback queue**)
  - **Interactive**, the same as time sharing, but higher priority
- Within each class there are different priorities and different scheduling algorithms.

Solaris scheduling II



Solaris scheduling III

- Solaris Dispatch Table

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Windows XP scheduling

- Dispatcher: priority-based, preemptive scheduling algorithm
- the dispatcher uses a-32 level priority scheme to determine the order of thread execution
  - 0: idle thread
  - 1~15: variable classes priorities
  - 16~31: real-time class
  - a queue for each priority
- Windows XP Priorities (**policy classes**, **relative priority**)

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2

Outline

小结

作业

- 非华夏班：5.1, 5.3
- 华夏班：4.4, 4.7

上机作业

- 所有：
  - 写一个多线程的程序，求素数。
  - 要求1：用户运行程序时在命令行输入一个素数，然后程序创建一个独立新的线程来输出小于或等于用户所输入数的所有素数。
  - 要求2：采用两种thread来实现
    - pthread或者Windows thread或者Java thread

谢谢！