Operating system concepts

Programming Multi-core Systems
Slides Set #7

By Prof K R Chowdhary

JNV University

2023

Multi-core Programming

▶ Whether the cores appear across CPU chips or within a CPU chip, these systems are *multiprocessor systems*

▶ Multithreaded programming provides a mechanism for more efficient use of these multiple computing cores

▶ Difference between parallelism and concurrency?

▶ Multiprocessing systems are multicore architectures or SMP (symmetric multi-processing)

▶ "Questions:"
  - Is it possible to have concurrency without parallelism?
  - Is it possible to have parallelism without concurrency?
  - What is multiprocessing?
  - What is multiprogramming?

Programming Challenges

- ▶ The multicore systems demands from system designers and application programmers to make better use of cores.
- ▶ Designers of operating systems must write scheduling algorithms that use multiple processing cores to allow the parallel execution
- ▶ Challenges in programming for multicore systems:
  - Identifying (concurrent) tasks
  - Balance
  - Data splitting
  - Data dependency
  - Testing and debugging
- ▶ Question: Which algorithms can be parallelized?
  - Bubble sort? Insertion sort? Quick sort? Any Search?

Types of parallelism

▶ There are two types of parallelism:
- *data parallelism*
- *task parallelism*
- In most instances, applications use a hybrid of these two
▶ Question:
- Divide array a[0] ... a[999] among separate cores: a[0]-a[499], a[500]-a[999], for summing, is what parallelism?
- Two threads one finds *smallest* element in a[0]-a[999] and other find *largest* element in a[0]-a[999], is what parallelism?

Multithreading models

▶ Support for threads may be provided at:

- the *user level*, for *user threads*, or

- by the *kernel*, for *kernel threads*.

▶ *Thread libraries*: provides the programmer with an API for creating and managing threads.

▶ Main thread libraries: POSIX Pthreads, Windows, and Java.

▶ *pthreads* is threads extension of the POSIX standard, may be provided as either a *user-level* or a *kernel-level* library.

▶ Question:

- How the threads of Windows, Linux, and java are different?

- Is pthread user-level or kernel level thread?

- How java threads work?

Two categories of threads creation

▶ Two general strategies for creating multiple threads:

- *Asynchronous threading*: Once the parent creates a child thread, the parent resumes its execution, so that the parent and child execute concurrently.

- *Synchronous threading*: When the parent thread creates one or more children and then must wait for all of its children to terminate before it resumes.

- For example, the parent thread may combine the results calculated by its various children. Our examples use synchronous threading.

▶ Questions:

- Which thread type share data with other threads, synchronous or asynchronous?

- Which thread type uses fork-join?

Pthreads

- ▶ Pthreads refers to the POSIX standard (IEEE 1003.1c) defining an API for thread creation and synchronization.

  ```
  pthread_t tid; /* the thread identifier */
  pthread_attr_t attr; /* set of thread attributes */
  void *runner(void *param);
  ......
  /* get the default attributes */
  pthread_attr_init(&attr);
  /* create the thread */
  pthread_create(&tid,&attr,runner,argv[1]);
  /* wait for the thread to exit */
  pthread_join(tid,NULL);
  ```

- ▶ Question: Is above thread synchronous or asynchronous?

Pthreads....

- ▶ runner is name of thread function
- ▶ All Pthreads programs must include the *pthread.h* header file.
- ▶ Each thread has a *set of attributes*, including stack size, priority,... and scheduling information.
- ▶ The *pthread_attr_t attr* declares thread attributes
- ▶ We set the attributes in the function call pthread_attr_init(&attr).
- ▶ The above uses default attributes.
- ▶ At this point, the program has two threads: (main() + thread)
- ▶ Question:
  - What are the two threads in above?
  - How one or more or nil arguments are passed to thread function?

Signal Handling

▶ A standard UNIX function for delivering a signal to a process:

    kill(pid_t pid, int signal)

The above function specifies the process (pid) to which a particular signal is to be delivered.

▶ POSIX Pthreads provides following function, which allows a signal to be delivered to a specified thread (tid):

    pthread_kill(pthread_t tid, int signal)

▶ Questions:
- Can a thread block a signal sent to kill the thread?

Thread Cancellation

▶ Thread cancellation involves terminating a thread before it has completed.

▶ A thread to be canceled is called target thread. Cancellation may occur in two different ways:

- Asynchronous cancellation

- Deferred cancellation

▶ In pthreads, thread cancellation is initiated using the pthread_cancel() function.

```
pthread_t tid;
pthread_create(&tid, 0, worker, NULL); //create thread
. . .
pthread_cancel(tid); //cancels the thread
```

▶ Question:

- What may be the reason of cancelling a thread?

- What is difference between two cancellations?