

Threads and Concurrency 04

Nitin V Pujari Faculty, Computer Science Dean - IQAC, PES University

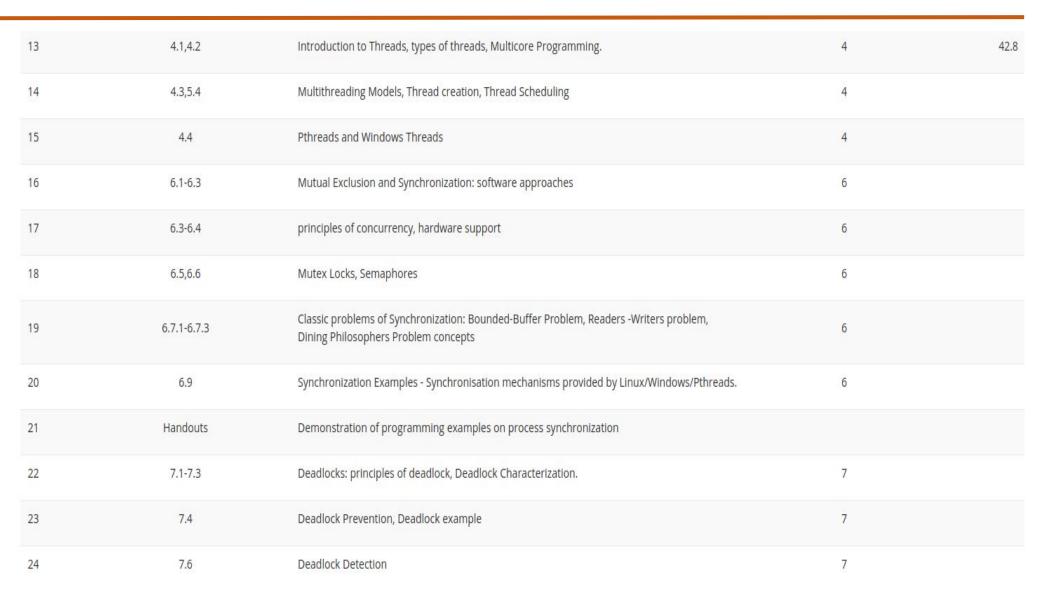
Course Syllabus - Unit 2



UNIT 2: Threads and Concurrency

Introduction to Threads, types of threads, Multicore Programming, Multithreading Models, Thread creation, Thread Scheduling, PThreads and Windows Threads, Mutual Exclusion and Synchronization: software approaches, principles of concurrency, hardware support, Mutex Locks, Semaphores. Classic problems of Synchronization: Bounded-Buffer Problem, Readers -Writers problem, Dining Philosophers Problem concepts. Synchronization Examples - Synchronisation mechanisms provided by Linux/Windows/Pthreads. Deadlocks: principles of deadlock, tools for detection and Prevention.

Course Outline - Unit 2







Threads and Concurrency

■ Thread Creation using Pthreads - An Example

Thread Libraries



Three main thread libraries are in use today

POSIX Pthreads

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

Thread Creation



- Two general strategies for creating multiple threads
 - Synchronous Threading.
 - Synchronous threading occurs when the parent thread creates one or more children and then must wait for all of its children to terminate before it resumes using the fork-join strategy.
 - Typically, synchronous threading involves significant data sharing among threads.

Thread Creation - Pthreads



- Pthreads refers to the POSIX standard (IEEE 1003.1c) defining an API for thread creation and synchronization.
 - This is a specification for thread behavior, not an implementation.
 - Operating-system designers may implement the specification in any way they wish.
 - Numerous systems implement the Pthreads specification; most are UNIX -type systems, including Linux, Mac OS X, and Solaris.
 - Although Windows doesn't support Pthreads natively, some third party implementations for Windows are available.

Pthreads Example

```
#include <pthread.h>
#include <stdio.h>
int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* threads call this function */
int main(int argc, char *argv[])
  pthread_t tid; /* the thread identifier */
  pthread_attr_t attr; /* set of thread attributes */
  if (argc != 2) {
     fprintf(stderr, "usage: a.out <integer value>\n");
     return -1:
  if (atoi(argv[1]) < 0) {
     fprintf(stderr, "%d must be >= 0\n", atoi(argv[1]));
     return -1:
  /* 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);
  printf("sum = %d\n",sum);
```

Pthreads Example



```
/* The thread will begin control in this function */
void *runner(void *param)
  int i, upper = atoi(param);
  sum = 0;
  for (i = 1; i <= upper; i++)
     sum += i;
  pthread_exit(0);
```



Threads and Concurrency

■ Thread Creation using Pthreads - An Example

Topics Uncovered in this Session



- Pthreads With Proper Synchronisation
 - First thread will find the sum of first N nos out of M nos
 - Second Thread will find the sum of next M-N nous
 - Third Thread will check whether the sum returned by Thread 1 and Thread 2 are Prime numbers
 - Fourth Thread will check whether the sum returned by Thread and Thread 2 are golden numbers If n os is prime and sum of individual digits of that number is also prime.



THANK YOU

Nitin V Pujari Faculty, Computer Science Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com