
# OPERATING SYSTEMS

## Threads and Concurrency

**Chandravva Hebbi**
Department of Computer Science

# OPERATING SYSTEMS

## Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

# OPERATING SYSTEMS

## Pthreads and Windows Threads

**Chandravva Hebbi**
Department of Computer Science

## Thread Libraries: Pthreads

- Provides the programmer with an API for creating and managing threads.
- Two primary ways of implementing a thread library.
  - provide a library entirely in user space with no kernel support.
    - All code and data structures for the library exist in user space.
  - Implement a kernel-level library supported
  - code and data structures for the library exist in kernel space.
  - Invoking a function directly by the operating system.

## Thread Libraries: Pthreads

- Three main thread libraries are in use today: (1) POSIX Pthreads, (2)Win32, and (3) Java.
- Pthreads, the threads extension of the POSIX standard, may be provided as either a user- or kernel-level library.
- The Win32 thread library is a kernel-level library available on Windows systems.
- The Java thread API allows threads to be created and managed directly in Java programs.

## Thread Libraries: Pthreads

- May be provided either as user-level or kernel-level
- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- ***Specification***, not ***implementation***
- API specifies behavior of the thread library, implementation is up to development of the library
- Common in UNIX operating systems (Solaris, Linux, Mac OS X)

# OPERATING SYSTEMS

## 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;
   }
```

## Pthreads Example

```
   /* 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);
}

/* 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);
}
```

# Pthreads Code for Joining 10 Threads

```
#define NUM_THREADS 10

/* an array of threads to be joined upon */
pthread_t workers[NUM_THREADS];

for (int i = 0; i < NUM_THREADS; i++)
   pthread_join(workers[i], NULL);
```

## Windows threads

- Windows implements theWin32 API as its primary API.
- A Windows application runs as a separate process, and each process may contain one or more threads.
- Windows uses the one-to-one mapping.
- Windows also provides support for a **fiber** library, which provides the functionality of the many-to-many model

## Windows threads

- The general components of a thread include:
  - A thread ID uniquely identifying the thread
  - A register set representing the status of the processor.
  - A user stack, employed when the thread is running in user mode, and a kernel stack, employed when the thread is running in kernel mode
  - A private storage area various run-time libraries and dynamic link libraries (DLLs)

## Windows Threads

- Threads are created in theWin32 API using the CreateThread() function the Win32 API.
- Using the WaitForSingleObject() function, which causes the creating thread to block until the child thread has exited.
- WaitForMultipleObjects(N,Thandles,TRUE,INFINITE) function is used to wait for multiple objects.
- The parameters to the function
  - The number of objects to wait for
  - A pointer to the array of objects
  - A flag indicating whether all objects have been signaled
  - A timeout duration( or INFINITE)
- Example: WaitForMultipleObjects(N,Thandles,TRUE,INFINITE)

# OPERATING SYSTEMS

## Windows Multithreaded C Program


```
#include <windows.h>
#include <stdio.h>
DWORD Sum; /* data is shared by the thread(s) */

/* the thread runs in this separate function */
DWORD WINAPI Summation(LPVOID Param)
{
   DWORD Upper = *(DWORD*)Param;
   for (DWORD i = 0; i <= Upper; i++)
      Sum += i;
   return 0;
}

int main(int argc, char *argv[])
{
   DWORD ThreadId;
   HANDLE ThreadHandle;
   int Param;

   if (argc != 2) {
      fprintf(stderr,"An integer parameter is required\n");
      return -1;
   }
   Param = atoi(argv[1]);
   if (Param < 0) {
      fprintf(stderr,"An integer >= 0 is required\n");
      return -1;
   }
```

# Windows Multithreaded C Program (Cont.)

```
   /* create the thread */
   ThreadHandle = CreateThread(
      NULL, /* default security attributes */
      0, /* default stack size */
      Summation, /* thread function */
      &Param, /* parameter to thread function */
      0, /* default creation flags */
      &ThreadId); /* returns the thread identifier */

   if (ThreadHandle != NULL) {
       /* now wait for the thread to finish */
      WaitForSingleObject(ThreadHandle,INFINITE);

      /* close the thread handle */
      CloseHandle(ThreadHandle);

      printf("sum = %d\n",Sum);
   }
}
```


# THANK YOU

**Chandravva Hebbi**
Department of Computer Science Engineering

**chandravvahebbi@pes.edu**