# Creating Threads

## Programming Details

COMP750 Distributed Systems

# Thread and Process Creation

- Processes can be created on Unix systems in C or C++ using the **`fork()`** function.
- Threads can be created in C and C++ under Unix or Windows using pthreads.
- Microsoft Visual Studio provides thread and process functions.
- Threads can be created in Java on all platforms.

# Creating a new Process

- In C or C++, you can create a new process with the fork function

```
int fork();
```

- When fork is called, the RAM of the program is copied to another location is RAM. A new process is created to run in the new program address space. The new process is put on the ready list.

# fork function

The return value of the fork function is:

- -1 = Error
- 0 = This is the process that was just created.
- number = This process is the parent that just executed the fork function. The number returned is the PID of the child process.

# Microsoft Windows Processes

- Application programs are processes in the Windows world.
- You can start a new process with the _spawn functions.
- The process calling _spawn can wait for the child to complete, continue running or be overlaid.

# POSIX Threads

- The POSIX operating system standard defines a thread library that is portable to any system supporting the POSIX standard.
- The **`pthread_create`** function creates a new thread.
- You can learn about POSIX threads from **`man pthread`**

# Pthread Programming in C

```
#include <pthread.h>
 int pthread_create(
        pthread_t               *thread,
        const pthread_attr_t  *attr,
        void *                  (*start_routine)(void *),
        void                    *arg);
```

compile as

```
cc myprog.c -lpthread
```

# Pthread Example

```
#include <pthread.h>
pthread_t threadObj;
int     aNumber = 47;

 pthread_create(&threadObj, NULL, myfunc,
                &aNumber);

void * myfunc(void *arg) { int x = *(int*)arg; }
```

# Pthread Example *(cont.)*

- The thread object of type **`pthread_t`** is used by the **`pthread`** system to store thread information. No initialization of this object is necessary.
- The **`pthread_create`** function calls the **`myfunc`** function with a pointer to **`aNumber`** as the function argument.
- **`myfunc`** executes in parallel with the calling function.

# Thread Termination

- Thread creation calls a function or method that will execute in parallel with the calling thread.
- The new thread will terminate when it would return to the calling function.
- Threads can also be terminated by:

```
void  pthread_exit(void value);
```

# Thread Creation with Visual Studio .NET

```
HANDLE CreateThread(
  LPSECURITY_ATTRIBUTES lpThreadAttributes, // SD
  SIZE_T       dwStackSize,                 // initial stack size
  LPTHREAD_START_ROUTINE lpStartAddress,    // function
  LPVOID       lpParameter,                 // thread argument
  DWORD        dwCreationFlags,             // creation option
  LPDWORD lpThreadId                        // thread identifier
);
```

# .NET parameters

- ThreadAttributes - security descriptor for the new thread or null
- dwStackSize – stack size or zero for default
- lpParameter - single parameter value passed to the thread
- dwCreationFlags - CREATE_SUSPENDED or 0 to start now
- lpThreadId – [out] returned thread ID or null

# Started Thread Function

```
DWORD WINAPI yourfunction(
  LPVOID lpParameter   // thread data
);
```

The return value indicates the thread's success or failure.

The parameter is the value passed to CreateThread

# Threads in Java

- Threads are built into Java as part of the standard class library.
- There are two ways to create threads in Java:
  - Extend the class **`Thread`**
  - Implement the interface **`Runnable`**

# run Method

- Both means of creating threads in Java require the programmer to implement the method:

```
public void run() { … }
```

- When a new thread is created, the **run** method is executed in parallel with the calling thread.

# Extending Java Thread Class

```
class Example extends Thread {
  int classData;  // example data
  // optional constructor
  Example(int something) {
     classData= something;
  }

  public void run() {
  // runs in parallel
  . . .
  }
}
```

# Starting a Thread Object

- Parallel execution of the run method of a Thread object is initiated by:

```
// create Thread Object
Example xyz = new Example(143);
// start execution of run method
xyz.start();
```

# Runnable Interface

- Java does not support multiple inheritance.  If a class extends Thread, it cannot extend another class.
- Programmers frequently want to use multiple threads and extend another class, such as Applet.
- The Runnable interface allows a program use multiple threads and inheritance.

# Implementing Runnable Interface

```
class Example implements Runnable {
  int classData;   // example data
  // optional constructor
  Example(int something) {
     classData= something;
  }

  public void run() {
  // runs in parallel
  . . .
  }
}
```

# Starting a Runnable Object

- Parallel execution of the run method of a Runnable object is initiated by:

```
// create Thread Object
Example xyz = new Example(143);
// start execution of run method
new Thread(xyz).start();
```

# Java Thread Termination

- Similar to pthreads, Java threads terminate when the run method returns.