Today’s Papers


Concurrent vs. Parallelism

Concurrent: 2 queues, 1 waiting machine

Parallel: 2 queues, 2 waiting machines

Concurrency
Tasks start, run and complete in an interleaved fashion

Parallelism
Tasks run simultaneously

Microprocessor Trends

42 Years of Microprocessor Trend Data

Processes vs. Threads

Single Thread

Multi Threaded

Events
Types of Threads

Thread Models
Paper Discussion

- Why are threads cheaper than processes?
- How is IPC performed using threads?
- Why is synchronization between threads needed?
- Two creation approaches: create ALL threads or create only CALLING thread; difference?
- What is “thread-local storage”?
- What are bound threads and why are they useful?
- Why is signaling challenging?

Pthreads (POSIX 1003.1c)

```c
#include <stdio.h>
#include <pthread.h>

void printMsg(char* msg) {
    int status = 0;
    printf("%s\n", msg);
    pthread_exit(&status);
}

int main(int argc, char** argv) {
    pthread_t thrdID;
    int* status = (int*)malloc(sizeof(int));
    printf("creating a new thread\n");
    pthread_create(&thrdID, NULL, (void*)printMsg, argv[1]);
    printf("created thread %d\n", thrdID);
    pthread_join(thrdID, &status);
    printf("Thread %d exited with status %d\n", thrdID, *status);
    return 0;
}
```
Common Programming Models

Multi-threaded programs tend to be structured as:

– **Producer/consumer**
  Multiple producer threads create data (or work) that is handled by one of the multiple consumer threads

– **Pipeline**
  Task is divided into series of subtasks, each of which is handled in series by a different thread

– **Defer work with background thread**
  One thread performs non-critical work in the background (when CPU idle)

Threads vs. Events

- **What is biggest problem with threads (in reading assignment)?**
- Threads:
  – Independent execution streams
  – Preemptive scheduling
  – Synchronization
  – Deadlocks
  – Debugging
  – “Threads break abstraction”
  – Getting good performance
  – OS support of threads
Threads vs. Events

- Events:
  - No CPU concurrency
  - Callbacks; event handlers
  - No preemption
  - Long-running handlers
  - State across handler invocations
  - Debugging
  - Overheads
  - Portability