POSIX Threads

Introduction

Modern Multiprocessing: Processes & Threads

- System (kernel) opportunity for *processor* or *hardware thread* utilization (*migration*)
  - *Parallel* $\leftrightarrow$ *Concurrent*
- Memory: *Isolated* $\leftrightarrow$ *Shared* (Safe/Faster)
- Communication: Standard *Inter Process Communication (IPC)* $\leftrightarrow$ Direct
  - e.g. Named pipes, Message queues, Sockets vs Shared pointers, Semaphores
- Management overhead: Normally about the same but varies a lot
  - More setup on creation for processes, cache pollution, threadpools
- Lower end and older systems: Poor or non-existent thread exception handling

Typical Thread Usage Scenarios

- Parallel execution: Apportion work, gather results
- Defer blocking[1] and long running[2] calls
  - [1] e.g. I/O to disk, peripherals, or network
  - [2] e.g. audio mixing thread parallel to main execution
- UI and window applications obliged to respond rapidly to dispatched messages
  - Offload work from main thread to worker threads
- Run parts of programs with different scheduling priorities

pthread Library Interface

- A standardized (C) interface for creating and managing threads, mutexes and associated functionality
    - Native or close-to-native support on *nix systems, simulated or suboptimal in others (most notably Windows)

```
int pthread_create(pthread_t *thread,
            const pthread_attr_t *attr,
            void *(*start_routine)(void*),
            void *restrict arg)
```

```
void pthread_exit(void *value_ptr)
```

```
int pthread_join(pthread_t thread,
                 void **value_ptr)
```

Example Listing

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

#define NUM_CHARS 1000

void *print(void *arg)
{
    int i;
    const int c = *((int *) arg);

    for (i = 0; i < NUM_CHARS; i++)
        putchar(c);

    return NULL;
}


int main(void)
{
    pthread_t t1, t2;
    char a = '*', h = '#';

    printf("\nCreating threads\n");
    pthread_create(&t1, NULL, print, &a);
    pthread_create(&t2, NULL, print, &h);

    printf("\nJoining threads\n");
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("\nExiting\n");
    return 0;
}
```

Listing Output

```
$ gcc -Wall tt.c -o tt
$ ./tt

Creating threads
**************************************************
**********************
Joining threads
*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#***#*#*#*#*#*#*#*
*#*#*#*#*#*#*#*#*#*#*#*#*#*#*###*#*#*#*#*#*#*#*#*
*#*#############################################
###########################
Exiting
```