Lab 8 - Multithreading

Multithreading

- We can have concurrency within a single process using threads: independent execution sequences within a single process.
- Threads let us run multiple functions in our program concurrently
- Multithreading is very common to parallelize tasks, especially on multiple cores
- In C++: spawn a thread using thread() and the thread variable type and specify what function you want the thread to execute (optionally passing parameters!)

```
void myFunc(int& x, int& y) {...}
thread myThread(myFunc, ref(arg1), ref(arg2));
```

- Notes:
    - myFunc: the function the thread should execute asynchronously
    - args: a list of arguments (any length, or none) to pass to the function upon execution
    - Once initialized with this constructor, the thread may execute at any time!
    - Thread function's return value is ignored (can pass by reference instead)
- Thread manager switches between executing threads like the OS scheduler switches between executing processes
- Each thread operates within the same process, so they share a virtual address space (!) (globals, text, data, and heap segments)
- The processes's stack segment is divided into a "ministack" for each thread.
- Many similarities between threads and processes; in fact, threads are often called *lightweight processes*.
- To wait on a thread to finish, use the `.join()` method:

```
thread myThread(myFunc, arg1, arg2);
... // do some work
// Wait for thread to finish (blocks)
myThread.join();
```

Note: For multiple threads, we must wait on a specific thread one at a time:

```
thread friends[5];
...
for (size_t i = 0; i < 5; i++) {
friends[i].join();
}
```

Threads vs. Processes

- Processes:
    - isolate virtual address spaces (good: security and stability, bad: harder to share info)
    - can run external programs easily (fork-exec) (good)
    - harder to coordinate multiple tasks within the same program (bad)
- Threads:
    - share virtual address space (bad: security and stability, good: easier to share info)
    - can't run external programs easily (bad)
    - easier to coordinate multiple tasks within the same program (good)

Programming Exercises

1. simple_thread.cpp : This program creat a simple thread
2. thread_array.ccp : This program spawns threads to each greet you, and we wait for all threads to finish before terminating.