Thread Safety

Anyone who thinks there's safety in numbers hasn't looked at the stock market pages.

—Irene Peter

Threading vs. Processes

- Overhead
  - Startup
  - Context switching
- Communication
  - IPC vs. Shared memory
- Containment
  - Memory
  - I/O
  - Privileges
  - Integrity
- Location
  - Threads: Same CPU
  - Processor: Anywhere
    - Memory
    - I/O Bandwidth
    - CPU

Threading in CASA

- 3rd Party Libraries (e.g., FFTW)
- OpenMP
- Asynchronous I/O
- GUIs: Viewer's worker threads
- IPC message handles (future)

Current Thread Support

- Qt
- OpenMp
- Pthreads (replace with C++11)
- Boost (replace with C++11)

C++11 Thread Support

- Thread Class
- Mutex
- Condition Variables
- Futures
- Atomic Class
  - Limited to primitive data types

CasaCore and Threading

- Most data structures based on CountedPtr
  - CountedPtr will be implemented using C++11 shared_ptr which supports thread safety
- Other reference-counted implementations
  - To be reworked to support thread safety
- Allows user to use CasaCore data structures safely
  - Read-only access between threads is OK.
  - RW access requires user-built mechanism

CasaCore and Threading (cont'd)

- Need to be aware of copy vs. reference semantics
  - Array<T> a2 (someArray); // by ref
  - a2 = anotherArray; // copies
  - N.B.: Array<T> a2 = someArray;
    - Same as copy constructor example, not assignment!

CasaCore and Threading (cont'd)

- Some data structures inherently not thread safe
- Example: Table Objects
  - When attached to same MS always share an underlying object (BaseTable-derived plug in).
  - Below that a file descriptor is shared.

Terminating Threads

- Thou shalt not kill !!!!
  - Design thread objects so that they periodically check for terminate flag.
- Thread failure takes out the whole process.