Subject: Shutdown procedure by clicking red x
Posted by nlneilson on Thu, 02 Dec 2010 14:17:27 GMT
View Forum Message <> Reply to Message

I have an app created with Upp that works great most of the time.

I am wondering about the shutdown procedure when clicking on the red x to end the app. The code has:

```
 void Exit0() {
  if(CommPort.IsOpened()) CommPort.Close();
  if(CommPort2.IsOpened()) CommPort2.Close();
  Thread::ShutdownThreads();
  Break(); // Closes window with 'Cancel'
 }

GUI_APP_MAIN{
 GPSv2().Run();
 if(CommPort.IsOpened()) CommPort.Close();
 if(CommPort2.IsOpened()) CommPort2.Close();
 Thread::ShutdownThreads();
}
```

When the app is closed with "void Exit0()" it explicitly closes any open ports and shuts down any threads.

When clicking the red x to end the app will this close the ports and end the threads if that is under _MAIN?

Subject: Re: Shutdown procedure by clicking red x
Posted by nlneilson on Thu, 09 Dec 2010 20:37:13 GMT
View Forum Message <> Reply to Message

After searching found this that gets deep into the red X to close:
http://www.codeguru.com/columns/vb/article.php/c15579

Also a thread wait would not work as the threads will continuously run unless shut down, which was the problem that seldom happened that required taskmanager or the MS pop up to terminate the app.

With some semi intuitive reasoning to end the threads that have in a while loop with Sleep(980) added this:

```
// A global bool end;

  while (CommPort.ReadDataWaiting() ) {
     if(end) break;
```

```
            ...
        Sleep(980);
        ....

 void Exit0() {
        end = true; Sleep(2000);
   if(CommPort.IsOpened()) CommPort.Close();
   if(CommPort2.IsOpened()) CommPort2.Close();
   Thread::ShutdownThreads();
   Break(); // Closes window with 'Cancel'
 }
void endX(){end = true; Sleep(2000);}

GUI_APP_MAIN{
   GPSv2().Run();
   endX();
   if(CommPort.IsOpened()) CommPort.Close();
   if(CommPort2.IsOpened()) CommPort2.Close();
   Thread::ShutdownThreads();
}
```

Setting end = true; and the Sleep(2000); gives the while loop time to break.
Sometimes if a GPS receiver is not closed properly pulling the receiver plug is about the only way.

This works when clicking the red x (or at least I have not had the error again) as well as the explicit Exit.

Neil

http://www.nlneilson.com/gpsv2.html

Subject: Re: Shutdown procedure by clicking red x
Posted by alendar on Fri, 24 Dec 2010 22:45:25 GMT
View Forum Message <> Reply to Message

Hi Neil,

I used this technique to interrupt a thread at closing time:

```
enum ListLoaderEnum {LL_RESET, LL_KILLLOAD};

class MainWin : public WithMainLayout<TopWindow> {
protected:

 StaticMutex loadListLock;
```

```
 Thread listLoaderThread;
 volatile Atomic stopFetchingTags;

public:
 MainWin() : stopFetchingTags(0) {

 virtual void Close() {
  // While we are in the Close event, no threads can continue, so a Wait will lock up the system
  // Notify threads to stop, then trigger a callback
  int x = listLoaderThread.GetCount();
  if (x) {
   loadListLock.Enter();
   stopFetchingTags = LL_KILLLOAD;
   loadListLock.Leave();
   this->ProcessEvents();
   Sleep(100);
   PostCallback(THISBACK(Close));
  } else {
   TopWindow::Close();  // Won't close unless this is called
  }
 }

 void EnrichDbFromTags() {
  listLoaderThread.Run(THISBACK(EnrichDbFromTagsThread));
 }

 void EnrichDbFromTagsThread() {
  stopFetchingTags = LL_RESET;

   ...

  while(lst.Fetch()) {

   if (stopFetchingTags == LL_KILLLOAD) {
    break;
   }
  ...
  }
 }
```

I tried the Thread::IsShutdownThreads() but it just locked up tight. The StaticMutex is probably overkill.

Jeff

Subject: Re: Shutdown procedure by clicking red x
Posted by nlneilson on Sat, 25 Dec 2010 07:44:08 GMT
View Forum Message <> Reply to Message

I had a few times when the app locked also on
Thread::ShutdownThreads(); or Thread::IsShutdownThreads()
I don't remember which but in debug just jumped from there and then stopped.
In a release it just hung and MS closed it.

This is what I changed to:

```
  while (CommPort.ReadDataWaiting() ) {
      if(end) break;
      try{
                    ....
void endX(){end = true; Sleep(2000);}

GUI_APP_MAIN{
 GPSx2().Run();
 if(CommPort.IsOpened()) CommPort.Close();
 if(CommPort2.IsOpened()) CommPort2.Close();
 endX();
}
```

This is in a GPS tracking app that has Sleep(1000) waiting for the next sentence, if not it checks every (500) to see if the signal is regained.
The if(end) break; closes the thread with the break inside the thread.
bool end; and the void endX() are global (if that is the right term).
I did it about the same way in a TrackReplay app, triggering a break inside the thread to end it.

Maybe a hack but no problem closing since.

Your code is above me just looking at it, will study it later.
It does look like you have a break inside the thread to close it also.
StaticMutex, volatile Atomic, PostCallback, are all new to me.

Neil

Subject: Re: Shutdown procedure by clicking red x
Posted by nlneilson on Mon, 27 Dec 2010 03:13:01 GMT
View Forum Message <> Reply to Message

mutex ~ mutual exclusion
No variables are shared or assessable between the two threads so that is not an issue.
I see you have StaticMutex but only one thread other than the main thread, I guess they could be both using the same variable at the same time.

PostCallBack ~ this thread gets into it but as mentioned above the data is not shared between the two worker threads
http://www.ultimatepp.org/forum/index.php?t=msg&goto=105 45&

I did look at this:
"I am not expert in threads, but it seems you use gui inside your threads. It would be better to be in main program."
http://www.ultimatepp.org/forum/index.php?t=msg&goto=298 57&
The two threads each use data from seperate GPS receivers and modify the content of seperate EditFields. The main thread can modify the format it is displayed in.

```
Sigq<<=Sigq;
Location<<=Lat + "," + Lon + "," + fAlt;
```

This works OK changing the data in the threads rather than in main.

atomic ~ concurrent reads and writes ~ a thread in my case does not read/write the other threads variables.

note: As far as error handling after about an hour the count for each GPS receiver may vary, I don't use the checksum for each sentence but as long as the Lat, Lon and Alt are OK it is used, otherwise it is skipped. It will not exit the app if a sentence is bad/skipped or missing.

Neil

Subject: Re: Shutdown procedure by clicking red x
Posted by crydev on Sun, 11 Nov 2012 19:32:21 GMT
View Forum Message <> Reply to Message

I have the same problem and I am looking for a solution. I tried the things posted here but I cannot solve the problem. My problem is that my program will always be using only 1 thread for the operation needed. (meaning GUI + 1 extra thread) Besides that, my thread performs tasks located in another class which are called by one main function. I don't have any blocking threads so using loops and volatile variables will not get me any further I think.

My code is as following:

```
void ButtonClicked() // This function is the button event
{
    Thread().Run(THISBACK1(RunImdbSearchAsync, row));
}
void RunImdbSearchAsync(int row) // Thread tasks
{
    ImdbManager im;
    im.WhenImdbMovie = THISBACK(ImdbMovieFound);
    im.ImdbSearch(mMovieList.Get(row, 0)); // This is a function that wraps around three tasks:
                                           // 1. Retrieving HTML using HttpRequest;
                                           // 2. Parsing HTML using Regular Expression;
```

```
                                    // 3. Creating an object containing the parsed data and returning it.
    PostCallback(THISBACK(AfterImdbSearch));
}
```

What would help for my situation? When I click the red cross to close my application while the thread above is still running I get an access violation.

Subject: Re: Shutdown procedure by clicking red x
Posted by nlneilson on Sun, 11 Nov 2012 22:58:21 GMT
View Forum Message <> Reply to Message

Try Thread::ShutdownThreads(); (or a global bool) with a Sleep time greater than any in your thread/s.

```
void endX(){
// clean up code here
 ...
 ...
 Sleep(2000); // This is required
}

GUI_APP_MAIN{
 NLNe().Run();
 Thread::ShutdownThreads();
 endX();
}
```

Then in each of your threads something like this with break; to close that thread from INSIDE that thread.

```
  while(!in.IsEof()){
      if(Thread::IsShutdownThreads()){ // or you can use a global bool
          in.Close();
          break;
      }
```

The code I posted in 2010 uses a global bool 'end' and in this code 'Thread::IsShutdownThreads'. The 'endX();' could be replaced with Sleep(2000); unless you want to clean other stuff up.

Clicking the red x is the same as Break(); // Closes window with 'Cancel' AFAIR.

Subject: Re: Shutdown procedure by clicking red x
Posted by crydev on Mon, 12 Nov 2012 12:07:46 GMT
View Forum Message <> Reply to Message

The way using a while loop and break it will not work for me, since I am not using any loops inside my thread.

Anyway, I solved my problem this way. I really wonder why a solution this easy didn't come to me any earlier:

```
GUI_APP_MAIN
{
 MovieManager().Run();
 ExitProcess(0); // End every thread inside your program
}
```

Subject: Re: Shutdown procedure by clicking red x
Posted by nlneilson on Mon, 12 Nov 2012 14:24:35 GMT
View Forum Message <> Reply to Message

ExitProcess(0);

Interesting, I will try that.
