Basics5 – Killing Threads

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: (•) Yes ( ) No

Name: Joey Domino

Date: 10/25/2022

Submission Details

Final ***Changelist*** number: 158942

Verified build: (•) Yes ( ) No

Required Configurations: 1: { Debug | x86 }

Discussion (What did you learn):

I learned how to use asynchronous threading. Crazy stuff!

Learned how to instance a thread counter system to track and handle locking and unlocking of multiple threads

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.suo,
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user
  - *.cpp, *.h
  - CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Seriously, I'm checking

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

Leave Project Settings

- Do NOT change the project or warning level
  - Any changing of level or suppression of warnings is an integrity issue

Leaking Memory

- If the program leaks memory
  - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
  - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
  - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
  - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
  - All files must be active to get credit.
  - Better to lose points for unit tests than to disable and lose all points
- Disable your debug printing otherwise you will lose points

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

Goals

- Learn
  - How to kill multiple threads cleanly
    - Create a coordinator thread that kills all the subordinate threads

Assignments

1. ***Spawn 4 or more threads…***
    a. Use the supplied threads in demo
        i. Threads A, B, C, D
    b. Create a Kill Thread (Controller)
        i. This keeps the SimulatedMain's thread nice and clean

2. ***Details***
    a. Spawn 4 worker threads & 1 controller thread
        i. Worker threads - All doing different work with different timing
            1. Thread A count up at 1 second intervals
                a. add sleep if you want
                b. Print the value using Debug::out()
            2. Thread B count down at 2 second intervals starting at 0x10000
                a. add sleep if you want
                b. Print the value using Debug::out()
            3. Thread C do something with strings...
                a. every 0.5s displaying a different string
                b. Use an array to hold 4 different strings and just play a different one each time
            4. Thread D print a string - long sentence
                a. every 0.75 seconds delete a char
                b. when string is 0, restore and repeat
        ii. Supplied project should help greatly
    b. Spawn a Controller thread
        i. All it does
            1. Waits for a key press from the main thread
            2. Triggers the kill all threads
        ii. Controller should kill everything cleanly and no leaking
            1. Using no globals
            2. If you need to pass data… pass it into the thread use SharedData

3. ***You need to rework the existing solution***
    a. Majority of work done in the controller
        i. You may need to modify the spawned threads {A,B,C,D}
        ii. You can add additional classes as you see fit
        iii. NO data can be global
            1. Shared data needs to be created in main() and then shared
    b. Use futures or shared futures
        i. You can use either… but future will do the work without overhead of shared futures
    c. The kill is initiated in the unit tests.. by a simulated keyboard press

i. All the “spawned threads” {A,B,C,D} should gracefully exit those potentially infinite loops.
ii. Any thread created should be effectively asynchronously and detached.

d. No join() anywhere
   i. Make sure all the threads have ended BEFORE leaving the controller

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** in all configurations without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
  - Is all the code there and compiles “as-is”?
  - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do many little check-ins
  - Iteration is easy and it helps.
  - Perforce is good at it.
- READ the book
  - Many good ideas in there.
- I had to do a lot of googling and web searching
  - Not make examples out there.
  - Dig into it you’ll get it
- Week 5 lecture
  - Was amazing for this basics