**Thread Management using POSIX Library - Detachment and Cancellation assignments**

1. Write a program to create 3 threads with the detach and cancel status as below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thread#** | **Detached (Y/N)** | **Cancel type** | **Cancel status** |
| 1 | Y | **PTHREAD\_CANCEL\_DISABLE** |  |
| 2 | N | **PTHREAD\_CANCEL\_ENABLE** | **PTHREAD\_CANCEL\_DEFERRED** |
| 3 | N | **PTHREAD\_CANCEL\_ENABLE** | **PTHREAD\_CANCEL\_ASYNCHRONOUS** |

1. Let all the threads read and display their detach, cancel type and status and then display thread specific message as below.

T1: Display message in the format as below every 2 secs

<timestamp> Health OK

T2: **Print numbers starting from 1000 in steps of 2 at an interval of 3 secs in format as below.**

<timestamp> <threadid> <countvalue>

T3: **Print numbers starting from 2000 in steps of 2 at an interval of 3 secs**

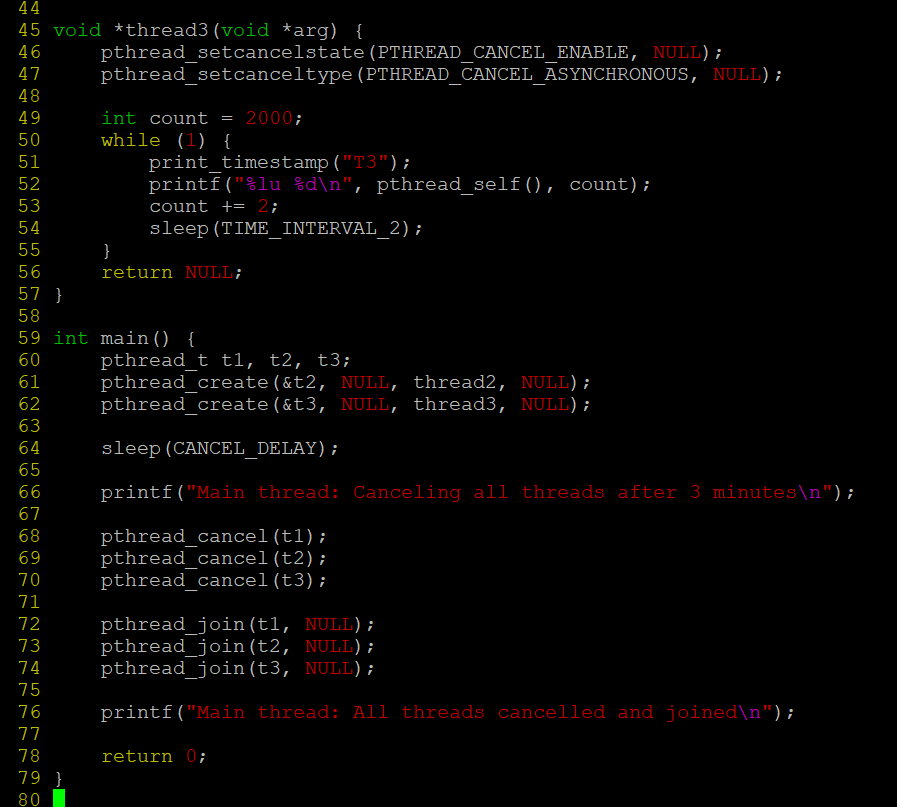
<timestamp> <threadid> < countvalue >



A screen shot of a computer program

Description automatically generated

1. **After creating threads, and after 3 minutes from main(), cancel all 3 threads**



1. **From an other terminal, use command below to view the thread count of your program**
   * + ps -eLF
     + top

[For top command usage to refer <https://www.golinuxcloud.com/check-threads-per-process-count-processes/> ]

1. **What difference did you observe between top and ps command?**

Ans: top should be used to see which processes are most active, ps could be used to see which processes are running currently.

1. **Which column shows the number of threads in ps and in top commands?**

Ans: In ps -eLF, the **NLWP** column shows the number of threads.

In top, the **THR** column shows the number of threads for each process.

1. **Which thread was cancelled first and why?**

Thread 3 is canceled first because it uses PTHREAD\_CANCEL\_ASYNCHRONOUS, meaning it is canceled immediately

1. **Were all 3 threads cancelled? Justify the observation**

Yes, all threads were canceled. However, Thread 3 was canceled immediately. Thread 2 was canceled after the sleep, and Thread 1 was canceled but since it was detached, it might not have been joined.