**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

a. 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 >

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

c. 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/ ]

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen with text and numbers

Description automatically generated

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

A) The ps command shows the threads as individual lines for a process with the -L option. It lists threads by their thread ID (TID) under the same process. In contrast, top shows a total number of threads and may show threads under the same process but in a less granular way compared to ps.

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

A) \* In the ps command, use the -L option to see the threads, and the column showing the thread count is labeled NLWP (Number of Light Weight Processes, which refers to threads).

\* In top, the thread count is shown in the summary section at the top as Tasks and shows the number of threads.

f. Check the last message timestamp from the threads

A) After canceling the threads, check the last printed timestamp in the console for each thread. The output of the program should show when each thread last printed a message before cancellation.

g. Which thread was cancelled first and why?

A) \*Thread 1, which is detached and has cancellation disabled (PTHREAD\_CANCEL\_DISABLE), cannot be canceled until the program terminates, so it won't be affected immediately.

\*Thread 3, having asynchronous cancellation (PTHREAD\_CANCEL\_ASYNCHRONOUS), will likely be canceled first as it is immediately responsive to cancellation requests.

\*Thread 2, with deferred cancellation, may continue running until it hits a cancellation point.

h. Were all 3 threads cancelled? Justify the observation

A) \* Thread 1 will not be canceled because cancellation is disabled.

\*Thread 2 will be canceled after reaching a cancellation point (it is using deferred cancellation).

\* Thread 3 will be canceled immediately due to asynchronous cancellation.