HAY QUE TENER EN CUENTA QUE EN ESTE CASO SE HA USADO COMO BUCLE FOR EL SIGUIENTE CODIGO:

volatile unsigned long long j;
for (j = 0; j < 370000000ULL; j++);

Y que el tiempo de computo es de +-0.5 segundos, considerando que fallo temporal es cuando el tiempo de ejecucion
es mayor que 0.9.

PREGUNTAS:

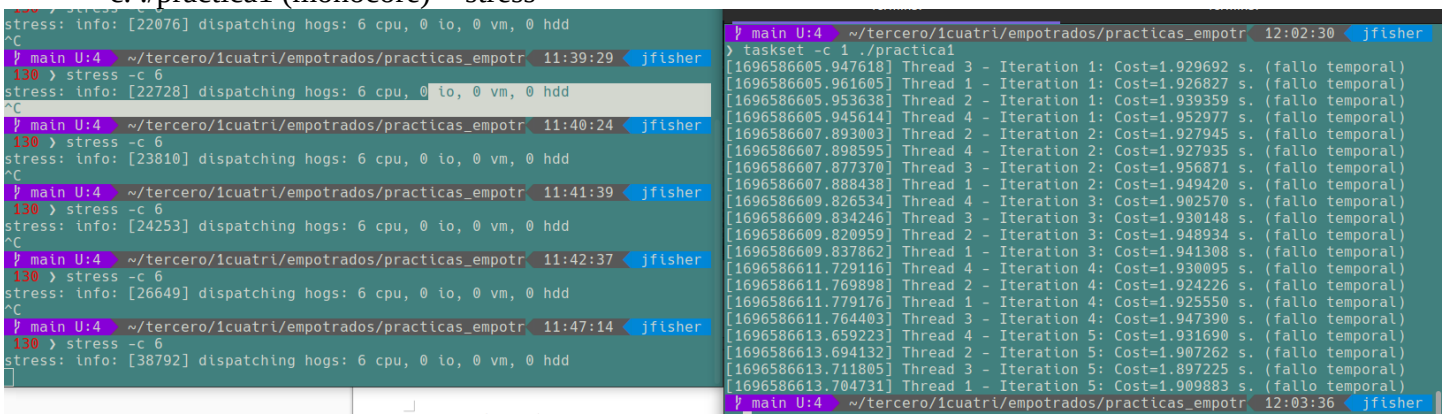
1. Ejecutar los siguientes casos y justificar su comportamiento:
   a. ./practica1 (multicore)

```
 main U:4  ~/tercero/1cuatri/empotrados/practicas_empotr  11:52:13  jfisher
❯ ./practica1
[1696586510.690970] Thread 2 - Iteration 1: Cost=0.474137 s.
[1696586510.691029] Thread 4 - Iteration 1: Cost=0.475053 s.
[1696586510.691075] Thread 3 - Iteration 1: Cost=0.475332 s.
[1696586510.690954] Thread 1 - Iteration 1: Cost=0.493090 s.
[1696586511.165222] Thread 2 - Iteration 2: Cost=0.478760 s.
[1696586511.166144] Thread 4 - Iteration 2: Cost=0.479858 s.
[1696586511.166506] Thread 3 - Iteration 2: Cost=0.480653 s.
[1696586511.184110] Thread 1 - Iteration 2: Cost=0.497004 s.
[1696586511.644044] Thread 2 - Iteration 3: Cost=0.478290 s.
[1696586511.646064] Thread 4 - Iteration 3: Cost=0.477757 s.
[1696586511.647222] Thread 3 - Iteration 3: Cost=0.478870 s.
[1696586511.681179] Thread 1 - Iteration 3: Cost=0.494506 s.
[1696586512.122397] Thread 2 - Iteration 4: Cost=0.478201 s.
[1696586512.123882] Thread 4 - Iteration 4: Cost=0.479087 s.
[1696586512.126150] Thread 3 - Iteration 4: Cost=0.477344 s.
[1696586512.175749] Thread 1 - Iteration 4: Cost=0.495439 s.
[1696586512.600659] Thread 2 - Iteration 5: Cost=0.484337 s.
[1696586512.603032] Thread 4 - Iteration 5: Cost=0.483409 s.
[1696586512.603568] Thread 3 - Iteration 5: Cost=0.483651 s.
[1696586512.671298] Thread 1 - Iteration 5: Cost=0.484394 s.
 main U:4  ~/tercero/1cuatri/empotrados/practicas_empotr  12:01:54  jfisher
```

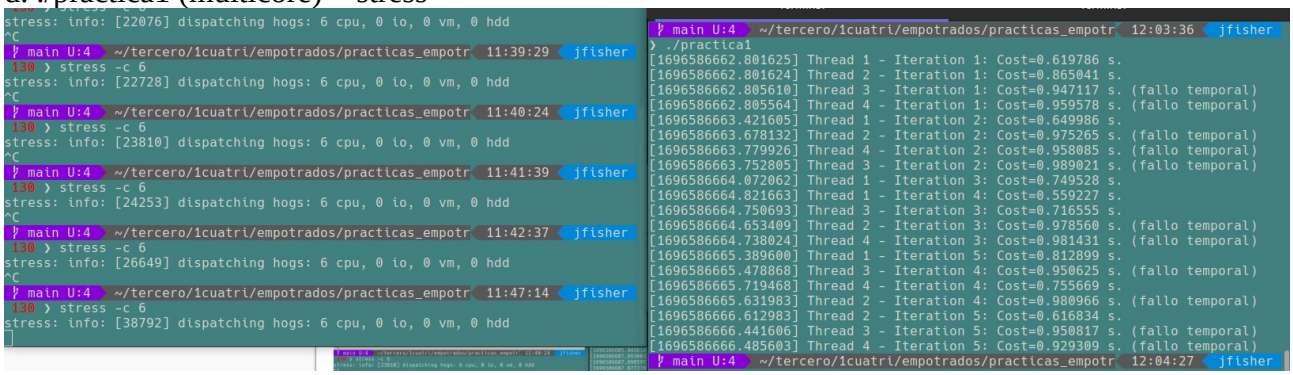
   b. ./practica1 (monocore)

```
 main U:4  ~/tercero/1cuatri/empotrados/practicas_empotr  12:01:54  jfisher
❯ taskset -c 1 ./practica1
[1696586540.553524] Thread 4 - Iteration 1: Cost=1.871105 s. (fallo temporal)
[1696586540.561604] Thread 3 - Iteration 1: Cost=1.871130 s. (fallo temporal)
[1696586540.581607] Thread 1 - Iteration 1: Cost=1.862556 s. (fallo temporal)
[1696586540.565614] Thread 2 - Iteration 1: Cost=1.883572 s. (fallo temporal)
[1696586542.424685] Thread 4 - Iteration 2: Cost=1.889093 s. (fallo temporal)
[1696586542.444170] Thread 1 - Iteration 2: Cost=1.878590 s. (fallo temporal)
[1696586542.449189] Thread 2 - Iteration 2: Cost=1.880558 s. (fallo temporal)
[1696586542.432737] Thread 3 - Iteration 2: Cost=1.902488 s. (fallo temporal)
[1696586544.313786] Thread 4 - Iteration 3: Cost=1.886855 s. (fallo temporal)
[1696586544.322764] Thread 1 - Iteration 3: Cost=1.886226 s. (fallo temporal)
[1696586544.329751] Thread 2 - Iteration 3: Cost=1.886350 s. (fallo temporal)
[1696586544.335229] Thread 3 - Iteration 3: Cost=1.887782 s. (fallo temporal)
[1696586546.200649] Thread 4 - Iteration 4: Cost=1.886807 s. (fallo temporal)
[1696586546.216106] Thread 2 - Iteration 4: Cost=1.879600 s. (fallo temporal)
[1696586546.208993] Thread 1 - Iteration 4: Cost=1.895233 s. (fallo temporal)
[1696586546.223017] Thread 3 - Iteration 4: Cost=1.888599 s. (fallo temporal)
[1696586548.111623] Thread 3 - Iteration 5: Cost=1.862992 s. (fallo temporal)
[1696586548.087466] Thread 4 - Iteration 5: Cost=1.889395 s. (fallo temporal)
[1696586548.095709] Thread 2 - Iteration 5: Cost=1.882155 s. (fallo temporal)
[1696586548.104232] Thread 1 - Iteration 5: Cost=1.875151 s. (fallo temporal)
 main U:4  ~/tercero/1cuatri/empotrados/practicas_empotr  12:02:30  jfisher
```

c. ./practica1 (monocore) + stress



d. ./practica1 (multicore) + stress



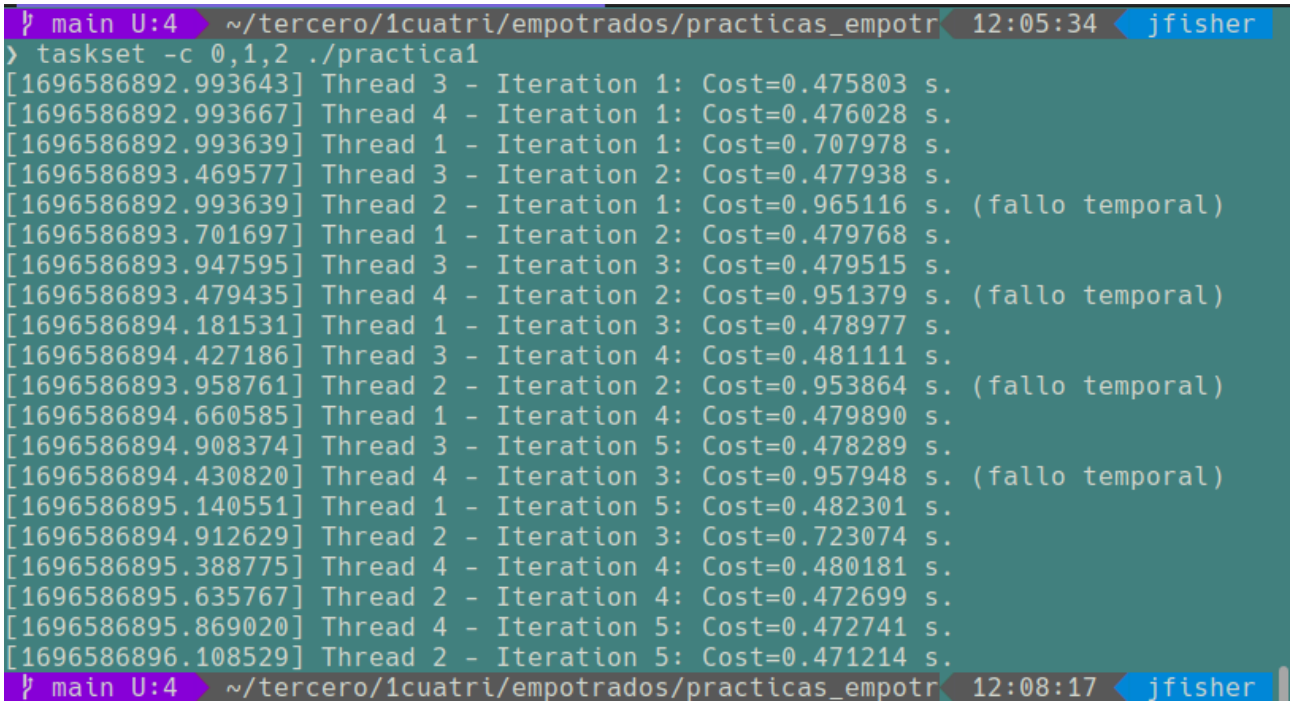
2. ¿En qué casos de ejecución (nombrados anteriormente) el sistema es capaz de cumplir las restricciones temporales (tanto tiempo de cómputo como periodicidad)?

Unicamente en el caso de ejecutar el programa en multicore sin stress, ya que en los demas casos el sistema no es capaz de cumplir las restricciones. En el resto de casos, el sistema no es capaz de cumplir las restricciones temporales, ya que el tiempo de ejecucion es mayor que el tiempo de computo. En el caso de stress con multicore y el monocore sin stress, el tiempo de ejecucion es mayor que el tiempo de computo pero en ese caso la periodicidad si se cumple. Al ejecutar tasket -c 0 ./practica1 (monocore) con stress, no es capaz de cumplir las restricciones temporales.

3. ¿Qué número mínimo de cpus se necesitan para que tu programa ejecute correctamente sin fallos de restricciones temporales? Usa el comando taskset para comprobarlo.

Para que el programa ejecute correctamente sin fallos de restricciones temporales, se necesitan 4 cpus (sin Stress). Para comprobarlo, se ejecuta el comando taskset -c 0,1,2,4 ./practica1 (multicore) y se comprueba que el programa se ejecuta correctamente sin fallos de restricciones temporales.

Si se ejecuta con Stress, aunque uses todas las cpus, el programa no cumple las restricciones temporales. En el caso de ejecutar el comando taskset -c 0,1,2 ./practica1 (multicore) sin stress el programa cumple casi sempre las restricciones temporales, pero en algun caso no las cumple.



4. ¿Qué solución se podría proponer para cumplir plazos estrictos temporales de periodicidad en la ejecución de los threads SIN cambiar la configuración actual que tienen los ordenadores del laboratorio?

Dando prioridad a los procesos que se ejecutan en el sistema, para que el proceso que se ejecuta en el sistema tenga prioridad sobre los demas procesos que se ejecutan en el sistema. Para ello, se ejecuta el comando nice -n -20 ./practica1 (multicore) y se comprueba que el programa se ejecuta correctamente sin fallos de restricciones temporales.