

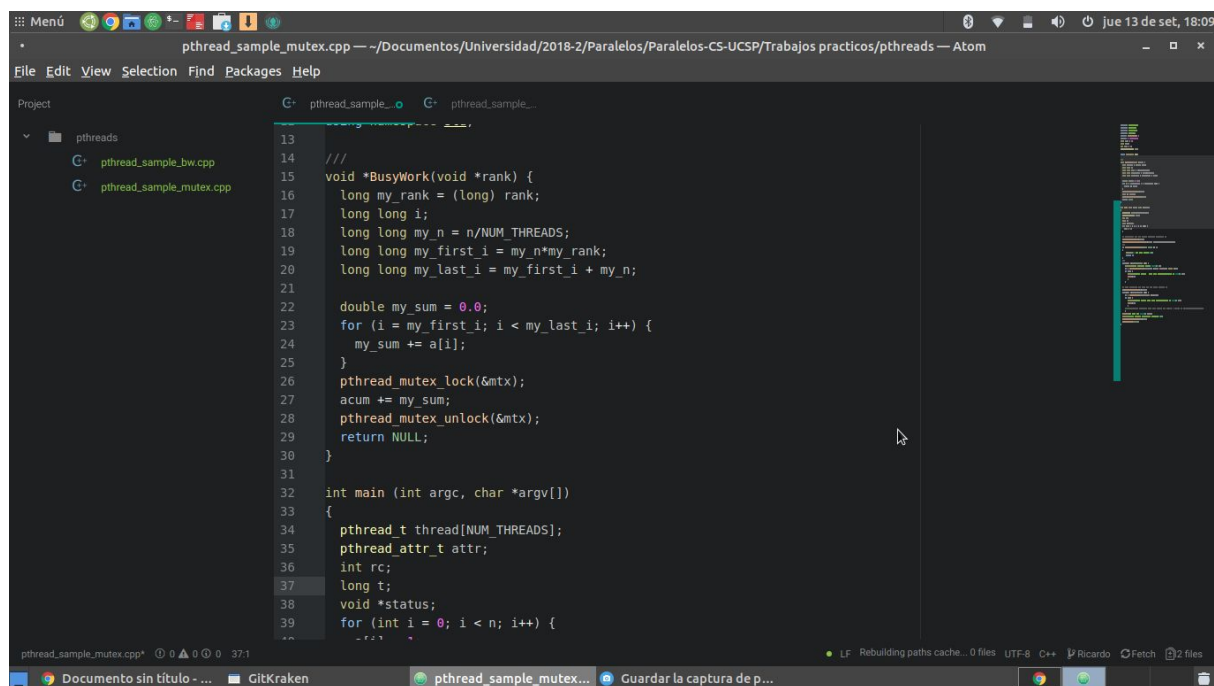
Informe de Pthreads: mutex vs busy waiting

Conceptos previos

Mutex o pausa es aquella que le dice a los demás threads que no pueden acceder al procesador mientras se realiza una operación.

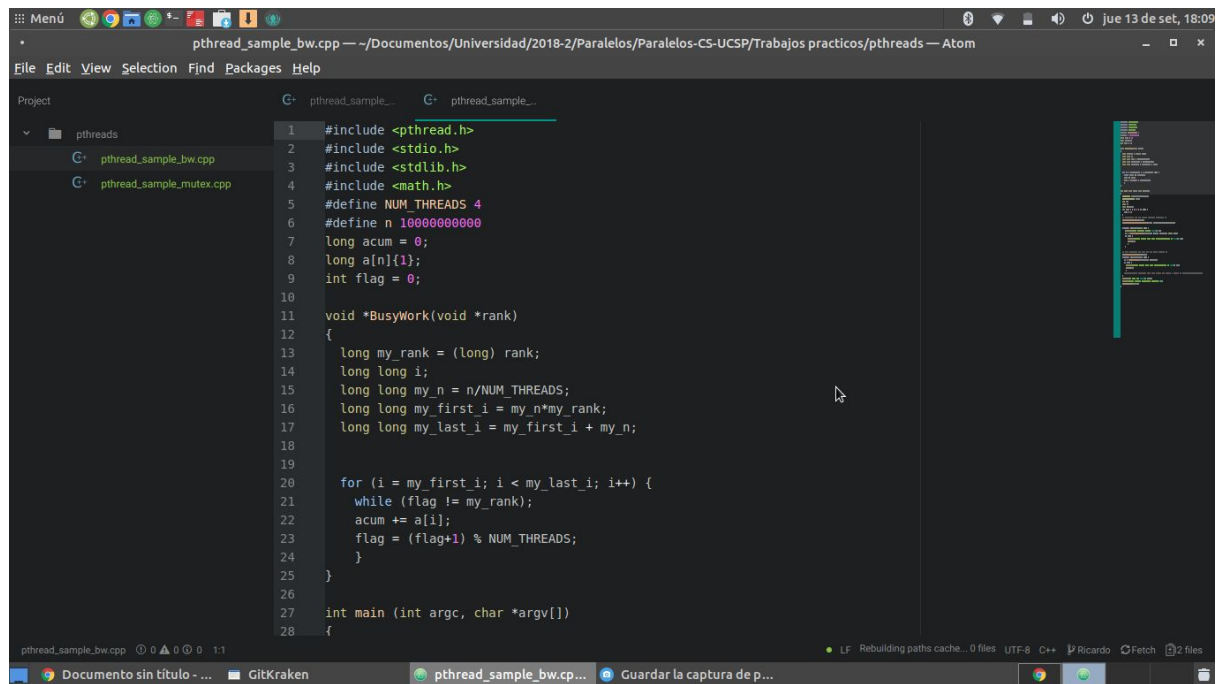
Busy Waiting es aquella que dependiendo de si el proceso tiene permitido el acceso a los recursos, por ende espera indefinidamente en el procesador simplemente haciendo nada. Para esta evaluación se debe contar hasta 10^{**7} .

Código de Mutex



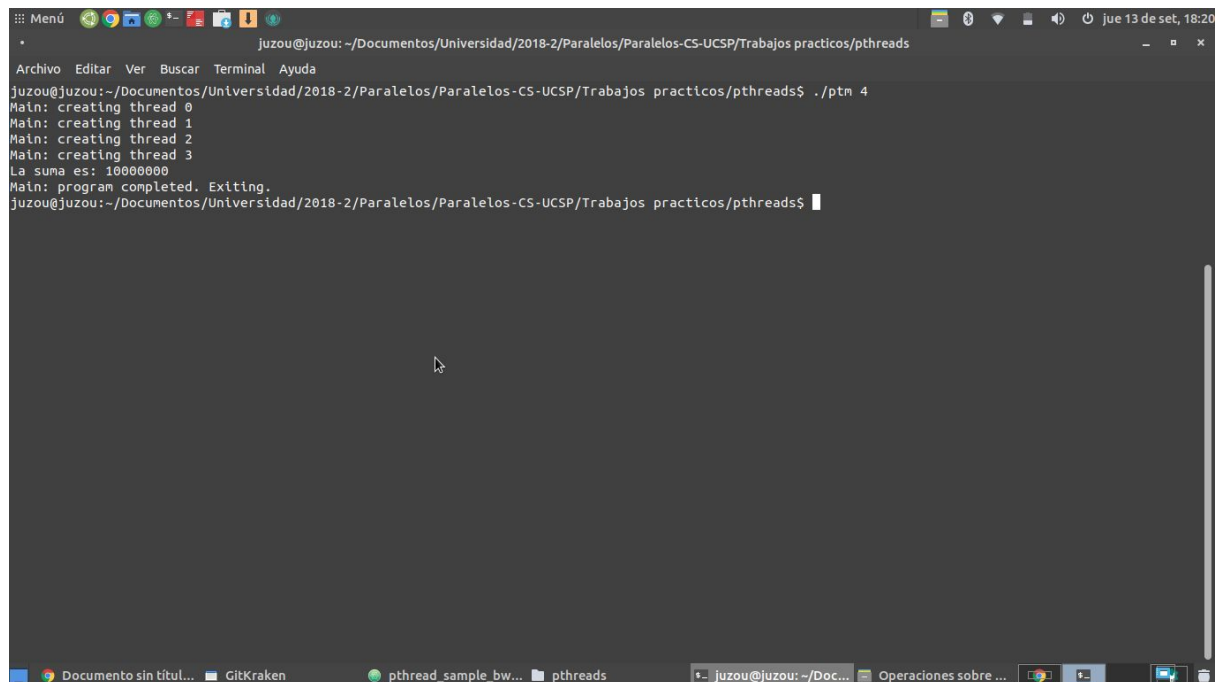
```
13
14
15 void *BusyWork(void *rank) {
16     long my_rank = (long) rank;
17     long long i;
18     long long my_n = n/NUM_THREADS;
19     long long my_first_i = my_n*my_rank;
20     long long my_last_i = my_first_i + my_n;
21
22     double my_sum = 0.0;
23     for (i = my_first_i; i < my_last_i; i++) {
24         my_sum += a[i];
25     }
26     pthread_mutex_lock(&mtx);
27     acum += my_sum;
28     pthread_mutex_unlock(&mtx);
29     return NULL;
30 }
31
32 int main (int argc, char *argv[])
33 {
34     pthread_t thread[NUM_THREADS];
35     pthread_attr_t attr;
36     int rc;
37     long t;
38     void *status;
39     for (int i = 0; i < n; i++) {
40         // ...
41     }
42 }
```

Código de Busy Waiting



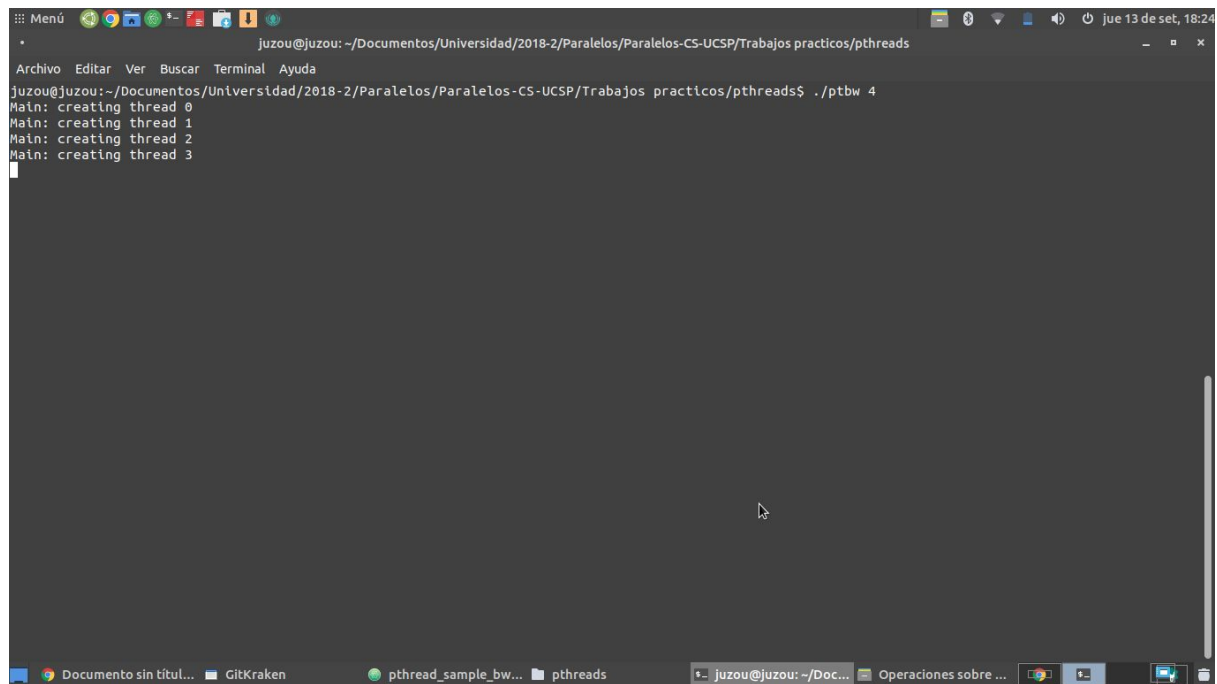
```
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5 #define NUM_THREADS 4
6 #define n 1000000000
7 long acum = 0;
8 long a[n]{1};
9 int flag = 0;
10
11 void *BusyWork(void *rank)
12 {
13     long my_rank = (long) rank;
14     long long i;
15     long long my_n = n/NUM_THREADS;
16     long long my_first_i = my_n*my_rank;
17     long long my_last_i = my_first_i + my_n;
18
19     for (i = my_first_i; i < my_last_i; i++) {
20         while (flag != my_rank);
21         acum += a[i];
22         flag = (flag+1) % NUM_THREADS;
23     }
24 }
25
26 int main (int argc, char *argv[])
27 {
28     pthread_t t[NUM_THREADS];
29     for (int i = 0; i < NUM_THREADS; i++) {
30         pthread_create(&t[i], NULL, BusyWork, (void *) i);
31     }
32     for (int i = 0; i < NUM_THREADS; i++) {
33         pthread_join(t[i], NULL);
34     }
35     printf("La suma es: %ld\n", acum);
36     return 0;
37 }
```

Salida de Mutex



```
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos/pthreads
Archivo Editar Ver Buscar Terminal Ayuda
juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos/pthreads$ ./ptm 4
Main: creating thread 0
Main: creating thread 1
Main: creating thread 2
Main: creating thread 3
La suma es: 1000000000
Main: program completed. Exiting.
juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos/pthreads$
```

Salida de Busy Waiting



```
juzou@juzou: ~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos/threads
juzou@juzou:~/Documentos/Universidad/2018-2/Paralelos/Paralelos-CS-UCSP/Trabajos practicos/threads$ ./ptbw 4
Main: creating thread 0
Main: creating thread 1
Main: creating thread 2
Main: creating thread 3
```

Salida en tiempos

threads	bw	mutex
4	29.88233	0.039635
8	300+	0.034696
16	300+	0.061290
32	300+	0.058947

Comparación

En tiempo de ejecución, el código de Busy Waiting es más lento, esto por el detalle de que malgasta los recursos del procesador no haciendo nada, en cambio el código de Mutex demora mucho menos, esto por que en realidad no se queda haciendo nada, sino que ese thread sale del procesador para darle apertura a otro; este solo entra cuando el mutex no esta activado, por ende es más efectivo.

Se ve además que el tiempo del busy waiting aumenta de manera notoria cuando se aumenta el número de threads.