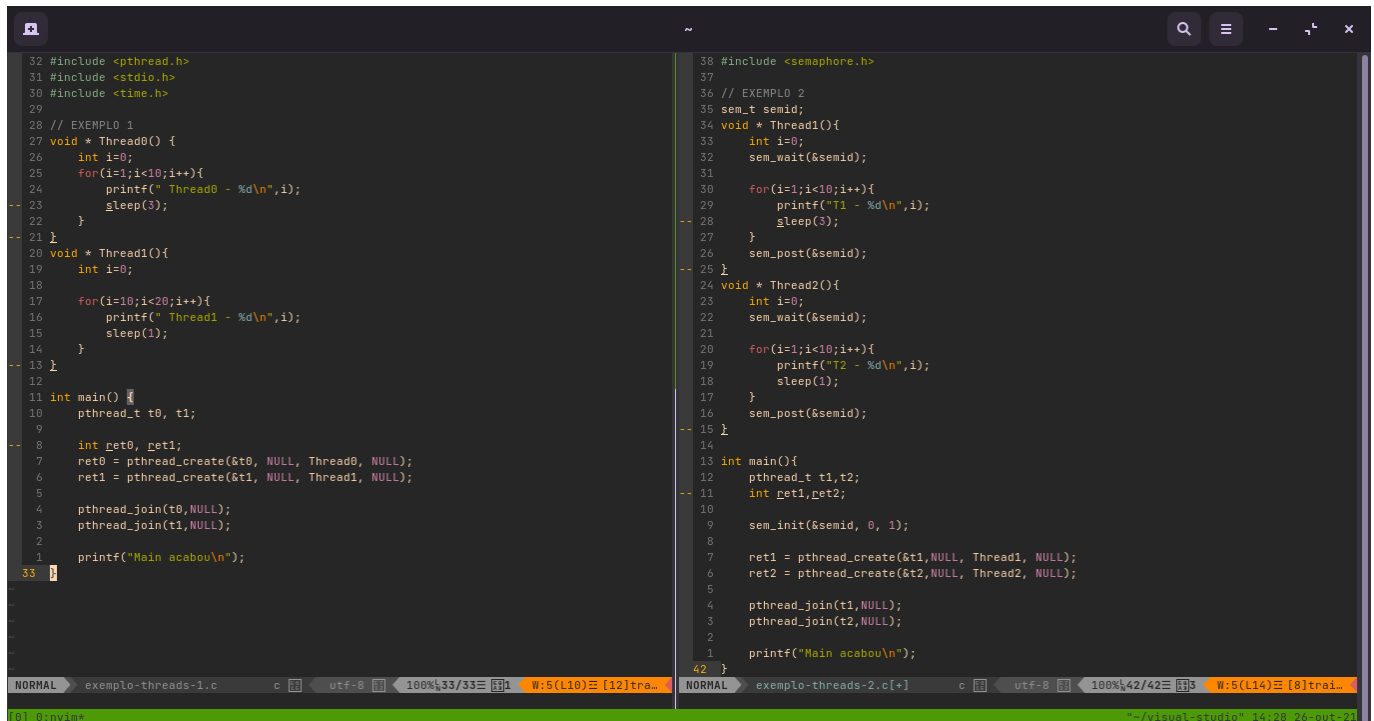


Exercício de Threads e Semáforos:

Implementando exemplo 1/2:

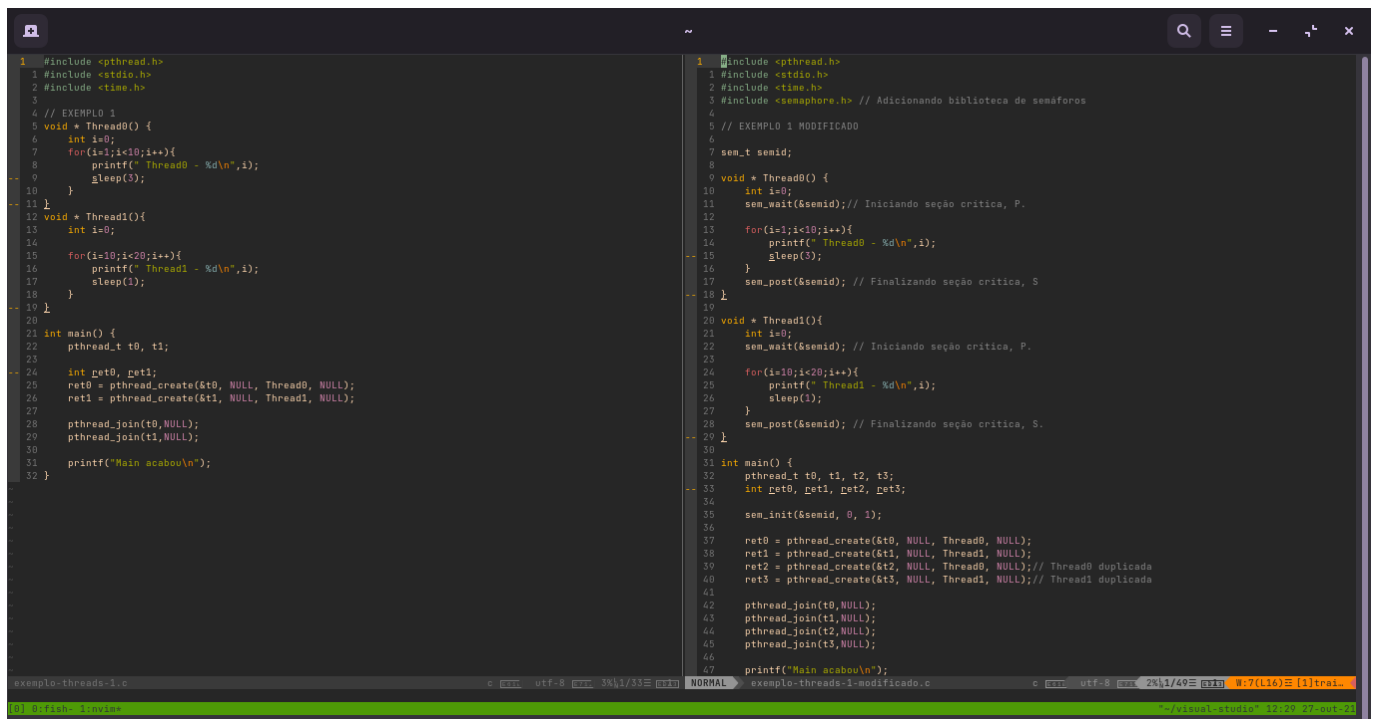


```
32 #include <pthread.h>
31 #include <stdio.h>
30 #include <time.h>
29
28 // EXEMPLO 1
27 void * Thread0() {
26     int i=0;
25     for(i=1;i<10;i++){
24         printf(" Thread0 - %d\n",i);
23         sleep(3);
22     }
21 }
20 void * Thread1(){
19     int i=0;
18     for(i=10;i<20;i++){
17         printf(" Thread1 - %d\n",i);
16         sleep(1);
15     }
14 }
13 }
12
11 int main() {
10     pthread_t t0, t1;
9
8     int ret0, ret1;
7     ret0 = pthread_create(&t0, NULL, Thread0, NULL);
6     ret1 = pthread_create(&t1, NULL, Thread1, NULL);
5
4     pthread_join(t0,NULL);
3     pthread_join(t1,NULL);
2
1     printf("Main acabou\n");
33 }
```

```
38 #include <semaphore.h>
37
36 // EXEMPLO 2
35 sem_t semid;
34 void * Thread1(){
33     int i=0;
32     sem_wait(&semid);
31
30     for(i=1;i<10;i++){
29         printf("T1 - %d\n",i);
28         sleep(3);
27     }
26     sem_post(&semid);
25 }
24 void * Thread2(){
23     int i=0;
22     sem_wait(&semid);
21
20     for(i=1;i<10;i++){
19         printf("T2 - %d\n",i);
18         sleep(1);
17     }
16     sem_post(&semid);
15 }
14
13 int main(){
12     pthread_t t1,t2;
11     int ret1,ret2;
10
9     sem_init(&semid, 0, 1);
8
7     ret1 = pthread_create(&t1,NULL, Thread1, NULL);
6     ret2 = pthread_create(&t2,NULL, Thread2, NULL);
5
4     pthread_join(t1,NULL);
3     pthread_join(t2,NULL);
2
1     printf("Main acabou\n");
42 }
```

De acordo com as aulas do dia 18/10 e 20/10

Modificando exemplo 1:



```
1 #include <pthread.h>
1 #include <stdio.h>
2 #include <time.h>
3
4 // EXEMPLO 1
5 void * Thread0() {
6     int i=0;
7     for(i=1;i<10;i++){
8         printf(" Thread0 - %d\n",i);
9         sleep(3);
10     }
11 }
12 void * Thread1(){
13     int i=0;
14     for(i=10;i<20;i++){
15         printf(" Thread1 - %d\n",i);
16         sleep(1);
17     }
18 }
19 }
20
21 int main() {
22     pthread_t t0, t1;
23
24     int ret0, ret1;
25     ret0 = pthread_create(&t0, NULL, Thread0, NULL);
26     ret1 = pthread_create(&t1, NULL, Thread1, NULL);
27
28     pthread_join(t0,NULL);
29     pthread_join(t1,NULL);
30
31     printf("Main acabou\n");
32 }
```

```
1 #include <pthread.h>
1 #include <stdio.h>
2 #include <time.h>
3 #include <semaphore.h> // Adicionando biblioteca de semáforos
4
5 // EXEMPLO 1 MODIFICADO
6
7 sem_t semid;
8
9 void * Thread0() {
10     int i=0;
11     sem_wait(&semid); // Iniciando seção crítica, P.
12
13     for(i=1;i<10;i++){
14         printf(" Thread0 - %d\n",i);
15         sleep(3);
16     }
17     sem_post(&semid); // Finalizando seção crítica, S
18 }
19 void * Thread1(){
20     int i=0;
21     sem_wait(&semid); // Iniciando seção crítica, P.
22
23     for(i=10;i<20;i++){
24         printf(" Thread1 - %d\n",i);
25         sleep(1);
26     }
27     sem_post(&semid); // Finalizando seção crítica, S.
28 }
29
30 int main() {
31     pthread_t t0, t1, t2, t3;
32     int ret0, ret1, ret2, ret3;
33
34     sem_init(&semid, 0, 1);
35
36     ret0 = pthread_create(&t0, NULL, Thread0, NULL);
37     ret1 = pthread_create(&t1, NULL, Thread1, NULL);
38     ret2 = pthread_create(&t2, NULL, Thread0, NULL); // Thread0 duplicada
39     ret3 = pthread_create(&t3, NULL, Thread1, NULL); // Thread1 duplicada
40
41     pthread_join(t0,NULL);
42     pthread_join(t1,NULL);
43     pthread_join(t2,NULL);
44     pthread_join(t3,NULL);
45
46     printf("Main acabou\n");
47 }
```

Adicionando semáforos e duplicando as threads.

Compilando exemplo 1:

```
~/visual-studio ➤ echo "COMPILANDO EXEMPLO 1" && echo " " && gcc -o exemplo-threads-1.o exemplo-threads-1.c -lpthread
COMPILANDO EXEMPLO 1

exemplo-threads-1.c: In function 'Thread0':
exemplo-threads-1.c:10:3: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  10 |     sleep(3);
      |     ^~~~~~
~/visual-studio
```

```
~/visual-studio ➤ echo "COMPILANDO EXEMPLO 1 MODIFICADO" && echo " " && gcc -o exemplo-threads-1-modificado.o exemplo-threads-1-modificado.c -lpthread
COMPILANDO EXEMPLO 1 MODIFICADO

exemplo-threads-1-modificado.c: In function 'Thread0':
exemplo-threads-1-modificado.c:16:3: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  16 |     sleep(3);
      |     ^~~~~~
~/visual-studio
```

Executando exemplo 1:

```
~/visual-studio ➤ echo "EXECUTANDO EXEMPLO 1" && echo " " && ./exemplo-threads-1.o
EXECUTANDO EXEMPLO 1

Thread0 - 1
Thread1 - 10
Thread1 - 11
Thread1 - 12
Thread0 - 2
Thread1 - 13
Thread1 - 14
Thread1 - 15
Thread0 - 3
Thread1 - 16
Thread1 - 17
Thread1 - 18
Thread0 - 4
Thread1 - 19
Thread0 - 5
Thread0 - 6
Thread0 - 7
Thread0 - 8
Thread0 - 9
Main acabou
~/visual-studio
```

```
~/visual-studio ➤ echo "EXECUTANDO EXEMPLO 1 MODIFICADO" && echo " " && ./exemplo-threads-1-modificado.o
EXECUTANDO EXEMPLO 1 MODIFICADO

Thread0 - 1
Thread0 - 2
Thread0 - 3
Thread0 - 4
Thread0 - 5
Thread0 - 6
Thread0 - 7
Thread0 - 8
Thread0 - 9
Thread0 - 10
Thread1 - 11
Thread1 - 12
Thread1 - 13
Thread1 - 14
Thread1 - 15
Thread1 - 16
Thread1 - 17
Thread1 - 18
Thread1 - 19
Thread1 - 10
Thread1 - 11
Thread1 - 12
Thread1 - 13
Thread1 - 14
Thread1 - 15
Thread1 - 16
Thread1 - 17
Thread1 - 18
Thread1 - 19
Main acabou
~/visual-studio
```

Resultados:

Ao implementar duas seções críticas nas funções Thread0() e Thread1() , a execução manteve os outputs das threads em ordem:

```

sem_t semid;

void * Thread0() {
    int i=0;
    sem_wait(&semid); // Iniciando seção crítica, P.

    for(i=1;i<10;i++){
        printf(" Thread0 - %d\n",i);
        sleep(3);
    }
    sem_post(&semid); // Finalizando seção crítica, S
}

void * Thread1(){
    int i=0;
    sem_wait(&semid); // Iniciando seção crítica, P.

    for(i=10;i<20;i++){
        printf(" Thread1 - %d\n",i);
        sleep(1);
    }
    sem_post(&semid); // Finalizando seção crítica, S.
}

```

Dentro da função `main()` , mais duas threads são criadas, `ret2` e `ret3`, executando as funções `Thread0()` e `Thread1()` respectivamente:

```

int main(){
    pthread_t t0, t1, t2, t3;
    int ret0, ret1, ret2, ret3;

    sem_init(&semid, 0, 1);

    ret0 = pthread_create(&t0, NULL, Thread0, NULL);
    ret1 = pthread_create(&t1, NULL, Thread1, NULL);
    ret2 = pthread_create(&t2, NULL, Thread0, NULL); // Thread0 duplicada
    ret3 = pthread_create(&t3, NULL, Thread1, NULL); // Thread1 duplicada

    pthread_join(t0, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);

    printf("Main acabou\n");
}

```

Modificando exemplo 2:

The image shows a side-by-side comparison of two C source files in Visual Studio Code. The left file, 'exemplo-threads-2.c', contains the original code with two threads (Thread1 and Thread2) and a main function that creates them and joins them. The right file, 'exemplo-threads-2-modificado.c', shows the modified version where the threads are duplicated (Thread1 and Thread2 are each created twice) and the main function is updated to join all four threads. The status bar at the bottom indicates the current file is 'exemplo-threads-2-modificado.c' and shows various editor settings like encoding (utf-8) and font size (14pt).

```

1 #include <pthread.h>
2 #include <stdio.h>
3 #include <time.h>
4 #include <semaphore.h>
5
6 // EXEMPLO 2
7 sem_t semid;
8 void * Thread1(){
9     int i=0;
10    sem_wait(&semid);
11    for(i=1;i<10;i++){
12        printf("T1 - %d\n",i);
13        sleep(3);
14    }
15    sem_post(&semid);
16 }
17 void * Thread2(){
18     int i=0;
19    sem_wait(&semid);
20    for(i=1;i<10;i++){
21        printf("T2 - %d\n",i);
22        sleep(1);
23    }
24    sem_post(&semid);
25 }
26
27 int main(){
28     pthread_t t1,t2;
29     int ret1,ret2;
30
31     sem_init(&semid, 0, 1);
32
33     ret1 = pthread_create(&t1,NULL, Thread1, NULL);
34     ret2 = pthread_create(&t2,NULL, Thread2, NULL);
35
36     pthread_join(t1,NULL);
37     pthread_join(t2,NULL);
38
39     printf("Main acabou\n");
40 }

```

```

45 #include <pthread.h>
46 #include <stdio.h>
47 #include <time.h>
48 #include <semaphore.h>
49
50 // EXEMPLO 2 MODIFICADO
51 sem_t semid;
52 void * Thread1(){
53     int i=0;
54    sem_wait(&semid);
55    for(i=1;i<10;i++){
56        printf("T1 - %d\n",i);
57        sleep(1);
58    }
59    sem_post(&semid);
60 }
61 void * Thread2(){
62     int i=0;
63    sem_wait(&semid);
64    for(i=1;i<10;i++){
65        printf("T2 - %d\n",i);
66        sleep(1);
67    }
68    sem_post(&semid);
69 }
70
71 int main(){
72     pthread_t t1,t2, t3, t4;
73     int ret1,ret2, ret3, ret4;
74
75     sem_init(&semid, 0, 1);
76
77     ret1 = pthread_create(&t1,NULL, Thread1, NULL);
78     ret2 = pthread_create(&t2,NULL, Thread2, NULL);
79     ret3 = pthread_create(&t3,NULL, Thread1, NULL); // Thread1 duplicada
80     ret4 = pthread_create(&t4,NULL, Thread2, NULL); // Thread2 duplicada
81
82     pthread_join(t1,NULL);
83     pthread_join(t2,NULL);
84     pthread_join(t3,NULL);
85     pthread_join(t4,NULL);
86
87     printf("Main acabou\n");
88 }

```

Compilando exemplo 2:

```
~/visual-studio echo "COMPILANDO EXEMPLO 2" && echo " " && gcc -o exemplo-threads-2.o exemplo-threads-2.c -lpthread
COMPILANDO EXEMPLO 2
exemplo-threads-2.c: In function 'Thread1':
exemplo-threads-2.c:14:3: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
   14 |     sleep(3);
      |     ^~~~~~
~/visual-studio

~/visual-studio echo "COMPILANDO EXEMPLO 2 MODIFICADO" && echo " " && gcc -o exemplo-threads-2-modificado.o exemplo-threads-2-modificado.c -lpthread
COMPILANDO EXEMPLO 2 MODIFICADO
exemplo-threads-2-modificado.c: In function 'Thread1':
exemplo-threads-2-modificado.c:14:3: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
   14 |     sleep(3);
      |     ^~~~~~
~/visual-studio
```

Executando exemplo 2:

```
~/visual-studio echo "EXECUTANDO EXEMPLO 2" && echo " " && ./exemplo-threads-2.o
EXECUTANDO EXEMPLO 2
T1 - 1
T1 - 2
T1 - 3
T1 - 4
T1 - 5
T1 - 6
T1 - 7
T1 - 8
T1 - 9
T2 - 1
T2 - 2
T2 - 3
T2 - 4
T2 - 5
T2 - 6
T2 - 7
T2 - 8
T2 - 9
Main acabou
~/visual-studio

~/visual-studio echo "EXECUTANDO EXEMPLO 2 MODIFICADO" && echo " " && ./exemplo-threads-2-modificado.o
EXECUTANDO EXEMPLO 2 MODIFICADO
T1 - 1
T1 - 2
T1 - 3
T1 - 4
T1 - 5
T1 - 6
T1 - 7
T1 - 8
T1 - 9
T2 - 1
T2 - 2
T2 - 3
T2 - 4
T2 - 5
T2 - 6
T2 - 7
T2 - 8
T2 - 9
T1 - 1
T1 - 2
T1 - 3
T1 - 4
T1 - 5
T1 - 6
T1 - 7
T1 - 8
T1 - 9
T2 - 1
T2 - 2
T2 - 3
T2 - 4
T2 - 5
T2 - 6
T2 - 7
T2 - 8
T2 - 9
Main acabou
~/visual-studio
```

Resultados:

Os outputs de cada uma das funções foram intercalados e duplicados. Primeiro todos os outputs da `Thread1()` e depois todos os outputs da `Thread2()`. Após isso o processo se repetiu pois duas threads a mais, executando as funções `Thread1()` e `Thread2()`, foram adicionadas.

```

int main(){
    pthread_t t1,t2, t3, t4;
    int ret1,ret2, ret3, ret4;

    sem_init(&semid, 0, 1);

    ret1 = pthread_create(&t1,NULL, Thread1, NULL);
    ret2 = pthread_create(&t2,NULL, Thread2, NULL);
    ret3 = pthread_create(&t3,NULL, Thread1, NULL); // Thread1 duplicada
    ret4 = pthread_create(&t4,NULL, Thread2, NULL); // Thread2 duplicada

    pthread_join(t1,NULL);
    pthread_join(t2,NULL);
    pthread_join(t3,NULL);
    pthread_join(t4,NULL);

    printf("Main acabou\n");
}

```

Conclusão:

Devido o tempo de espera de cada função, o sistema operacional as executou com prioridade diferente sobre as outras.

No exemplo 1, o tempo de `sleep()` da `Thread0()` é maior que o da `Thread1()`, o que resultou em outputs sequenciais, primeiro todas as threads que executavam `Thread0()` e depois todas que executavam `Thread1()`.

Já no exemplo 2, como o tempo de `sleep()` de ambas funções eram o mesmo, o resultado foi um input intercalado.

Manaus/AM - 26/10/2021