Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №7
по дисциплине: **ОСИСП**
Тема: Семафоры

Выполнил
студент 2 курса
Корнасевич И. Д.

Проверил
Давидюк Ю. И.

Брест 2021

Задание Первый процесс в цикле ожидает ввода символа в stdin, после чего пишет его в файл, каждый раз открывая и закрывая его. Второй процесс забирает символ и этого файла и выводит его на экран несколько раз.

producer.c

```c
#include <semaphore.h>
#include <unistd.h>
#include <stdio.h>
#include <uv.h>
#include "consts.h"


int main() {
    printf("Producer started\n");
    sem_unlink(PRODUCER_SEM_NAME);
    sem_unlink(CONSUMER_SEM_NAME);
    sem_unlink(TRANSFER_SEM_NAME);
    sem_t *producerSem = sem_open(PRODUCER_SEM_NAME, O_CREAT | O_EXCL, 0666, 1);
    sem_t *consumerSem = sem_open(CONSUMER_SEM_NAME, O_CREAT | O_EXCL, 0666, 0);
    sem_t *transferSem = sem_open(TRANSFER_SEM_NAME, O_CREAT | O_EXCL, 0666, 0);

    u_int iteration = 0;
    char buf[1];
    while (1){
        sem_wait(producerSem);
        printf("Producer waits input\n");
        read(0, buf, 10);
        printf("Iteration %d\n", iteration++);
        sem_init(transferSem, 1, buf[0]);
        printf("Producer posts\n");
        sem_post(consumerSem);
    }
}
```

consumer.c

```c
#include <semaphore.h>
#include <unistd.h>
#include <stdio.h>
#include <uv.h>
#include "consts.h"


int main() {
    printf("Consumer started\n");
    sem_t *producerSem = sem_open(PRODUCER_SEM_NAME, 0);
    sem_t *consumerSem = sem_open(CONSUMER_SEM_NAME, 0);
    sem_t *transferSem = sem_open(TRANSFER_SEM_NAME, 0);

    u_int iteration = 0;
    char buf[2] = ".\n";
    int intBuf[1];
    while (1) {
```

```
18          printf("Consumer waits\n");
19          sem_wait(consumerSem);
20          printf("Iteration %d\n", iteration++);
21          sem_getvalue(transferSem, intBuf);
22          buf[0] = (char)intBuf[0];
23          write(0, buf, 2);
24          sem_post(producerSem);
25      }
26  }
```

producer.txt

```
1  Consumer started
2  Consumer waits
3  Iteration 0
4  1
5  Consumer waits
6  Iteration 1
7  2
8  Consumer waits
9  Iteration 2
10 3
11 Consumer waits
12 Iteration 3
13 4
14 Consumer waits
15 Iteration 4
16 5
17 Consumer waits
18 Iteration 5
19 6
20 Consumer waits
21 Iteration 6
22 .
23 Consumer waits
24 Iteration 7
25 ,
26 Consumer waits
27 Iteration 8
28 /
29 Consumer waits
30 Iteration 9
31 a
32 Consumer waits
33 Iteration 10
34 z
35 Consumer waits
36 Iteration 11
37 x
38 Consumer waits
39 Iteration 12
40 c
41 Consumer waits
42 Iteration 13
43 v
44 Consumer waits
```

```
Iteration 14
b
Consumer waits
Iteration 15
n
Consumer waits
Iteration 16
m
Consumer waits
Iteration 17
t
Consumer waits
Iteration 18
i
Consumer waits
Iteration 19
a
Consumer waits
^C
```

consumer.txt

```
Producer started
Producer waits input
1
Iteration 0
Producer posts
Producer waits input
2
Iteration 1
Producer posts
Producer waits input
3
Iteration 2
Producer posts
Producer waits input
4
Iteration 3
Producer posts
Producer waits input
5
Iteration 4
Producer posts
Producer waits input
6
Iteration 5
Producer posts
Producer waits input
.
Iteration 6
Producer posts
Producer waits input
,
Iteration 7
Producer posts
Producer waits input
```

```
/
Iteration 8
Producer posts
Producer waits input
a
Iteration 9
Producer posts
Producer waits input
z
Iteration 10
Producer posts
Producer waits input
x
Iteration 11
Producer posts
Producer waits input
c
Iteration 12
Producer posts
Producer waits input
v
Iteration 13
Producer posts
Producer waits input
b
Iteration 14
Producer posts
Producer waits input
n
Iteration 15
Producer posts
Producer waits input
m
Iteration 16
Producer posts
Producer waits input
t
Iteration 17
Producer posts
Producer waits input
i
Iteration 18
Producer posts
Producer waits input
a
Iteration 19
Producer posts
Producer waits input
^C
```

Вывод: Семафор представляет собой atomic unsigned int. По сути семафор — не бинарный мьютекс. При помощи семафора можно синхронизировать работу нескольких процессов. В Linux существуют именованные и неименованные семафоры. Для выполнения задания я выбрал именно неименованную вариацию, так

как это позволяет получить к такому семафору доступ из любого процесса. Операции с семафорами:

- `sem_open()` — создание семафора или получение доступа на уже существующий. Также в эту функцию может входить инициализация семафора.
- `sem_unlink()` — удаление именованного семафора.
- `sem_wait()` — уменьшение значения семафора. Если оно уже равно нулю, то процесс останавливается до тех пор, пока значение не увеличится.
- `sem_post()` — увеличение значения семафора. Если существует остановленный этим семафором процесс, то он возобновляется.