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(SEM_NAME);
    sem_t *sem = sem_open(SEM_NAME, O_CREAT | O_EXCL, 0666, 0);
    if (!sem) {
        printf("Cannot open semaphore\n");
        perror("error: ");
        return 1;
    }
    u_int iteration = 0;
    char buf[1];
    while (1){
        printf("Producer waits input\n");
        read(0, buf, 10);
        printf("Iteration %d\n", iteration++);
        int fd = open("../main.txt", O_WRONLY | O_CREAT, 0666);
        if (fd < 0){
            printf("Cannot open the file");
            return -1;
        }
        write(fd, buf, 1);
        close(fd);
        printf("Producer posts\n");
        sem_post(sem);
    }
}
```

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 *sem = sem_open(SEM_NAME, 0);
    if (!sem){
        printf("Semaphore not exist\n");
        perror("error: ");
        return -1;
    }
```

```
16      char buf[1];
17      u_int iteration = 0;
18      while (1) {
19          printf("Consumer waits\n");
20          sem_wait(sem);
21          printf("Iteration %d\n", iteration++);
22          int fd = open("../main.txt", O_RDONLY, 0666);
23          if (fd < 0){
24              printf("Cannot open the file.");
25              return -1;
26          }
27          read(fd, buf, 1);
28          remove("../main.txt");
29          write(1, buf, 1);
30          write(1, buf, 1);
31          write(1, buf, 1);
32          write(1, "\n\n", 2);
33      }
34  }
```

producer.txt

```
1  Producer started
2  Producer waits input
3  1
4  Iteration 0
5  Producer posts
6
7  Producer waits input
8  t
9  Iteration 1
10 Producer posts
11
12 Producer waits input
13 k
14 Iteration 2
15 Producer posts
16
17 Producer waits input
18 a
19 Iteration 3
20 Producer posts
21
22 Producer waits input
23 -
24 Iteration 4
25 Producer posts
26
27 Producer waits input
28 \
29 Iteration 5
30 Producer posts
31
32 Producer waits input
33 *
34 Iteration 6
```

```
Producer posts

Producer waits input
n
Iteration 7
Producer posts

Producer waits input
^C
```

consumer.txt

```
Consumer started
Consumer waits
Iteration 0
111

Consumer waits
Iteration 1
ttt

Consumer waits
Iteration 2
kkk

Consumer waits
Iteration 3
aaa

Consumer waits
Iteration 4
---

Consumer waits
Iteration 5
\\\

Consumer waits
Iteration 6
***

Consumer waits
Iteration 7
nnn

Consumer waits
^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.