Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **«ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 3

Название: Взаимодействие параллельных процессов

Дисциплина: Операционные системы

Студент ИУ7-52Б _____ В.А. Иванов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Ю. Рязанова
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Задание 1
«Производство-потребление»

Листинг программы:

main_header.h

```
#ifndef MAIN_H
#define MAIN_H

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/stat.h>
#include <sys/shm.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define FULL_SEMN   0
#define EMPT_SEMN   1
#define BIN_SEMN    2

#define QUEUE_SIZE  5
#define BUF_SIZE    2 + QUEUE_SIZE
#define PROC_N      3

#endif // MAIN_H
```

main.c

```
#include "main_header.h"

int rand_dt()
{
   return 1 + rand() % 3;
}

void prod_func(size_t b_size, int isem_descry, char* buf, int my_n)
{
   srand(time(NULL) + my_n*100);
   struct sembuf pre_sem[2] = { {EMPT_SEMN, -1, SEM_UNDO},
                                {BIN_SEMN, -1, SEM_UNDO} };
   struct sembuf post_sem[2] = { {FULL_SEMN, 1, SEM_UNDO},
                                 {BIN_SEMN, 1, SEM_UNDO} };
   char cur_letter = 'A';
   while (1)
   {
      if (semop(isem_descry, pre_sem, 2) == -1)
      {
         perror("semop");
```

```
            exit(1);
        }

        buf[buf[1]] = cur_letter;
        if (++buf[1] >= b_size)
            buf[1] = 2;
        printf("Producer №%d wrote:\t %c\n", my_n+1, cur_letter);

        if (semop(isem_descry, post_sem, 2) == -1)
        {
            perror("semop");
            exit(1);
        }

        if (++cur_letter > 'Z')
            cur_letter = 'A';
        sleep(rand_dt());
    }

}

void cons_func(size_t b_size, int isem_descry, char* buf, int my_n)
{
    srand(time(NULL) + my_n*10);
    struct sembuf pre_sem[2] = { {FULL_SEMN, -1, SEM_UNDO},
                                 {BIN_SEMN, -1, SEM_UNDO} };
    struct sembuf post_sem[2] = { {EMPT_SEMN, 1, SEM_UNDO},
                                  {BIN_SEMN, 1, SEM_UNDO} };
    char cur_letter;
    while (1)
    {
        if (semop(isem_descry, pre_sem, 2) == -1)
        {
            perror("semop");
            exit(1);
        }

        cur_letter = buf[buf[0]];
        printf("Consumer №%d read:\t\t %c\n", my_n+1, cur_letter);
        if (++buf[0] >= b_size)
            buf[0] = 2;

        if (semop(isem_descry, post_sem, 2) == -1)
        {
            perror("semop");
            exit(1);
        }

        sleep(rand_dt());
    }

}


int main(void)
{
```

```
int perms = S_IRWXU | S_IRWXO | S_IRWXG;

int isem_descry = semget(IPC_PRIVATE, 3, IPC_CREAT | perms );
if (isem_descry == -1)
{
   perror("semget");
   return 1;
}
int full_ctl = semctl(isem_descry, FULL_SEMN, SETVAL, 0);
int empt_ctl = semctl(isem_descry, EMPT_SEMN, SETVAL, QUEUE_SIZE);
int bin_ctl = semctl(isem_descry, BIN_SEMN, SETVAL, 1);
if (full_ctl == -1 || empt_ctl == -1 || bin_ctl == -1)
{
   perror("semctl");
   return 1;
}

int mem_id = shmget(IPC_PRIVATE, (BUF_SIZE)*sizeof(char), IPC_CREAT | perms);
if (mem_id == -1)
{
   perror("shmget");
   return 1;
}
char* addr = shmat(mem_id, 0, 0);
if (addr == (char*)(-1))
{
   perror("shmat");
   return 1;
}

addr[0] = (char)2;
addr[1] = (char)2;

printf("> Start of simulation\n");
for (size_t i=0; i<PROC_N; i++)
{
   pid_t prod_pid = fork();
   switch (prod_pid)
   {
   case -1:
      perror("fork");
      return 1;
   case 0:
      prod_func(BUF_SIZE, isem_descry, addr, i);
      return 0;
   default:
      printf("> Producer created\n");
      break;
   }

   pid_t cons_pid = fork();
   switch (cons_pid)
   {
   case -1:
      perror("fork");
      return 1;
   case 0:
```

```
            cons_func(BUF_SIZE, isem_descry, addr, i);
            return 0;
        default:
            printf("> Consumer created\n");
            break;
        }
    }

    int status, pid;
    for (size_t i=0; i<PROC_N*2; i++)
    {
        pid = wait(&status);
        if (pid == -1)
        {
            perror("wait");
            return 1;
        }
    }


    if (semctl(isem_descry, 0, IPC_RMID, 0) == -1)
    {
        perror("semctl");
        return 1;
    }
    if (shmctl(mem_id, IPC_RMID, NULL) == -1)
    {
        perror("shmctl");
        return 1;
    }
    if (shmdt(addr) == -1)
    {
        perror("shmdt");
        return 1;
    }

    return 0;
}
```

Пример работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab3_1$ gcc -o main.o main.c
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab3_1$ ./main.o
> Start of simulation
> Producer created
> Consumer created
> Producer created
Producer №1 wrote:      A
Consumer №1 read:               A
> Consumer created
Producer №2 wrote:      A
> Producer created
> Consumer created
Producer №3 wrote:      A
Consumer №2 read:               A
Consumer №3 read:               A
Producer №1 wrote:      B
Consumer №1 read:               B
Producer №3 wrote:      B
Consumer №3 read:               B
Producer №1 wrote:      C
Producer №2 wrote:      B
Consumer №1 read:               C
Consumer №2 read:               B
Producer №3 wrote:      C
Producer №2 wrote:      C
Consumer №3 read:               C
Producer №3 wrote:      D
Producer №1 wrote:      D
Consumer №1 read:               C
Consumer №2 read:               D
Producer №3 wrote:      E
Producer №2 wrote:      D
Producer №1 wrote:      E
Consumer №1 read:               D
Consumer №3 read:               E
Producer №3 wrote:      F
Producer №2 wrote:      E
Consumer №2 read:               D
Producer №1 wrote:      F
Consumer №1 read:               E
Consumer №2 read:               F
Consumer №3 read:               E
```

Задание 2
«Читатели-писатели»

Листинг программы:

main_header.h

```
#ifndef _MAIN_H
#define _MAIN_H

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/stat.h>
#include <sys/shm.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ACT_W_SEMN   0
#define ACT_R_SEMN   1
#define WAI_W_SEMN   2
#define WAI_R_SEMN   3


#define WRITER_N    2
#define READER_N    5

#endif // _MAIN_H
```

main.c

```
#include "main_header.h"

int rand_dt()
{
   return 1 + rand() % 3;
}

void start_read(int isem_descry)
{
   static struct sembuf wait_sem[1] = {
      {WAI_R_SEMN, 1, SEM_UNDO} };
   static struct sembuf act_sem[4] = {
      {ACT_W_SEMN, 0, SEM_UNDO},
      {WAI_W_SEMN, 0, SEM_UNDO},
      {ACT_R_SEMN, 1, SEM_UNDO},
      {WAI_R_SEMN,-1, SEM_UNDO}};

   if (semop(isem_descry, wait_sem, 1) == -1)
   {
```

```
        perror("semop");
        exit(1);
    }

    if (semop(isem_descry, act_sem, 4) == -1)
    {
        perror("semop");
        exit(1);
    }
}
void stop_read(int isem_descry)
{
    static struct sembuf act_sem[1] = {
        {ACT_R_SEMN,-1, SEM_UNDO}};

    if (semop(isem_descry, act_sem, 1) == -1)
    {
        perror("semop");
        exit(1);
    }
}

void start_write(int isem_descry)
{
    static struct sembuf wait_sem[1] = {
        {WAI_W_SEMN, 1, SEM_UNDO} };
    static struct sembuf act_sem[4] = {
        {ACT_W_SEMN, 0, SEM_UNDO},
        {ACT_R_SEMN, 0, SEM_UNDO},
        {ACT_W_SEMN, 1, SEM_UNDO},
        {WAI_W_SEMN,-1, SEM_UNDO}};

    if (semop(isem_descry, wait_sem, 1) == -1)
    {
        perror("semop");
        exit(1);
    }

    if (semop(isem_descry, act_sem, 4) == -1)
    {
        perror("semop");
        exit(1);
    }
}
void stop_write(int isem_descry)
{
    static struct sembuf act_sem[1] = {
        {ACT_W_SEMN,-1, SEM_UNDO}};

    if (semop(isem_descry, act_sem, 1) == -1)
    {
        perror("semop");
        exit(1);
    }

}
```

```
void read_func(int isem_descry, char* buf, int my_n)
{
   srand(time(NULL) + my_n*100);
   while (1)
   {
      start_read(isem_descry);

      printf("Reader №%d get:\t\t %c\n", my_n, *buf);

      stop_read(isem_descry);
      sleep(rand_dt());
   }
}

void write_func(int isem_descry, char* buf, int my_n)
{
   srand(time(NULL) + my_n*10);
   while (1)
   {
      start_write(isem_descry);

      *buf = (char)(rand() % ('z' - 'a') + 'a');
      printf("Writer №%d send:\t %c\n", my_n, *buf);

      stop_write(isem_descry);
      sleep(rand_dt());
   }
}

int main(void)
{
   int perms = S_IRWXU | S_IRWXO | S_IRWXG;

   int isem_descry = semget(IPC_PRIVATE, 4, IPC_CREAT | perms );
   if (isem_descry == -1)
   {
      perror("semget");
      return 1;
   }

   int ctl[4] = { 0, 0, 0, 0 };
   ctl[0] = semctl(isem_descry, ACT_W_SEMN, SETVAL, 0);
   ctl[1] = semctl(isem_descry, ACT_R_SEMN, SETVAL, 0);
   ctl[2] = semctl(isem_descry, WAI_W_SEMN, SETVAL, 0);
   ctl[3] = semctl(isem_descry, WAI_R_SEMN, SETVAL, 0);
   for (size_t i=0; i<4; i++)
      if (ctl[i] == -1)
      {
         perror("semctl");
         return 1;
      }

   int mem_id = shmget(IPC_PRIVATE, sizeof(char), IPC_CREAT | perms);
   if (mem_id == -1)
   {
      perror("shmget");
      return 1;
```

```
}
char* addr = shmat(mem_id, 0, 0);
if (addr == (char*)(-1))
{
    perror("shmat");
    return 1;
}
*addr = '!';

printf("> Start of simulation\n");
for (size_t i=0; i<WRITER_N; i++)
{
    pid_t prod_pid = fork();
    switch (prod_pid)
    {
    case -1:
        perror("fork");
        return 1;
    case 0:
        write_func(isem_descry, addr, i);
        return 0;
    default:
        printf("> Writer created\n");
        break;
    }
}
for (size_t i=0; i<READER_N; i++)
{
    pid_t prod_pid = fork();
    switch (prod_pid)
    {
    case -1:
        perror("fork");
        return 1;
    case 0:
        read_func(isem_descry, addr, i);
        return 0;
    default:
        printf("> Reader created\n");
        break;
    }
}


int status, pid;
for (size_t i=0; i<WRITER_N+READER_N; i++)
{
    pid = wait(&status);
    if (pid == -1)
    {
        perror("wait");
        return 1;
    }
}


if (semctl(isem_descry, 0, IPC_RMID, 0) == -1)
```

```
    {
        perror("semctl");
        return 1;
    }
    if (shmctl(mem_id, IPC_RMID, NULL) == -1)
    {
        perror("shmctl");
        return 1;
    }
    {
        perror("shmdt");
        return 1;
    }

    return 0;
}
    if (shmdt(addr) == -1)
```

Пример работы:

```
> Start of simulation
> Writer created
> Writer created
Writer №0 send:  e
> Reader created
Writer №1 send:  a
> Reader created
Reader №0 get:           a
> Reader created
Reader №1 get:           a
Reader №2 get:           a
> Reader created
> Reader created
Reader №3 get:           a
Reader №4 get:           a
Writer №0 send:  o
Writer №1 send:  o
Reader №2 get:           o
Reader №0 get:           o
Writer №1 send:  w
Reader №3 get:           w
Reader №4 get:           w
Reader №1 get:           w
Writer №0 send:  t
Reader №0 get:           t
Reader №2 get:           t
Reader №3 get:           t
Reader №4 get:           t
Reader №1 get:           t
Writer №0 send:  f
Writer №1 send:  a
Reader №2 get:           a
Reader №4 get:           a
Reader №0 get:           a
Writer №1 send:  y
Reader №3 get:           y
Reader №4 get:           y
Reader №1 get:           y
Writer №0 send:  j
Reader №2 get:           j
Reader №3 get:           j
Reader №4 get:           j
Reader №1 get:           j
```