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»

Отчет по лабораторной работе №5 по курсу «Операционные системы»

Тема Взаимодействие параллельных процессов

Студент Богаченко А.Е.

Группа ИУ7-56Б

Оценка (баллы) ______________________________

Преподаватели Рязанова Н. Ю.

Москва — 2020 г.

Задача «Производство-потребление»

Листинги кода

```c
#include "buffer.h"

int buff_init(cycle_buff_t *const buffer) {
    if (!buffer) {
        return -1;
    }
    memset(buffer, 0, sizeof(cycle_buff_t));

    return 0;
}

int buff_write(cycle_buff_t *const buffer, const char c) {
    if (!buffer) {
        return -1;
    }
    buffer->data[buffer->wpos++] = c;
    buffer->wpos %= N;

    return 0;
}

int buff_read(cycle_buff_t *const buffer, char *const dst) {
    if (!buffer) {
        return -1;
    }
    *dst = buffer->data[buffer->rpos++];
    buffer->rpos %= N;

    return 0;
}
```

Листинг 1 – Очередь на основе циклического массива (буфера). Код.

```c
#ifndef __BUFFER_H__
#define __BUFFER_H__

#include <string.h>
#include <unistd.h>

#define N 24
```

```c

typedef char data_t[N];

typedef struct {
    size_t rpos;
    size_t wpos;
    data_t data;
} cycle_buff_t;

int buff_init(cycle_buff_t *const buffer);
int buff_write(cycle_buff_t *const buffer, const char c);
int buff_read(cycle_buff_t *const buffer, char* const dst);

#endif  // __BUFFER_H__
```

Листинг 2 – Очередь на основе циклического массива (буфера). Заголовочник.

```c
#include "runners.h"

struct sembuf PROD_LOCK[2] = {{BUF_EMPTY, -1, 0}, {BIN_SEM, -1, 0}};
struct sembuf PROD_RELEASE[2] = {{BUF_FULL, 1, 0}, {BIN_SEM, 1, 0}};

struct sembuf CONS_LOCK[2] = {{BUF_FULL, -1, 0}, {BIN_SEM, -1, 0}};
struct sembuf CONS_RELEASE[2] = {{BUF_EMPTY, 1, 0}, {BIN_SEM, 1, 0}};

int run_producer(cycle_buff_t *const buffer,
                 const int sid,
                 const int prod_id) {
    if (!buffer) {
        return -1;
    }

    srand(time(NULL) + prod_id);

    int sleep_time;
    char ch;
    for (short i = 0; i < ITERATIONS_AMOUNT; ++i) {
        sleep_time = rand() % MAX_RANDOM_PROD + 1;
        sleep(sleep_time);

        if (semop(sid, PROD_LOCK, 2) == -1) {
            perror("prod lock error!");
            exit(EXIT_FAILURE);
        }

        // !!! --- CRITICAL --- !!!
        ch = 'a' + (char)(buffer->wpos % 26);
        if (buff_write(buffer, ch) == -1) {
            perror("buffer write error!");
            return EXIT_FAILURE;
        }
        printf(" Producer #%d write: %c — idle %ds\n", prod_id,
               ch, sleep_time);
        // !!! —— CRITICAL —— !!!

        if (semop(sid, PROD_RELEASE, 2) == -1) {
            perror("prod release error!");
            exit(EXIT_FAILURE);
        }
    }
    return EXIT_SUCCESS;
}

int run_consumer(cycle_buff_t *const buffer,
                 const int sid,
                 const int cons_id) {
    if (!buffer) {
        return -1;
    }

    srand(time(NULL) + cons_id + PROD_COUNT);

    int sleep_time;
    char ch;
    for (short i = 0; i < ITERATIONS_AMOUNT; ++i) {
        sleep_time = rand() % MAX_RANDOM_CONS + 1;
        sleep(sleep_time);

        if (semop(sid, CONS_LOCK, 2) == -1) {
            perror("consumer lock error!");
            exit(EXIT_FAILURE);
        }

        // !!! —— CRITICAL —— !!!
        if (buff_read(buffer, &ch) == -1) {
            perror("buffer read error!");
            return EXIT_FAILURE;
        }
        printf(" Consumer #%d read:  %c — idle %ds\n", cons_id,
               ch, sleep_time);
        // !!! —— CRITICAL —— !!!

        if (semop(sid, CONS_RELEASE, 2) == -1) {
            perror("consumer release error!");
            exit(EXIT_FAILURE);
        }
```

```c
    }
    return EXIT_SUCCESS;
}
```

Листинг 3 – Реализация задачи. Код.

```c
#ifndef __RUNNERS_H__
#define __RUNNERS_H__

#include <stdio.h>
#include <stdlib.h>
#include <sys/sem.h>
#include <time.h>

#include "buffer.h"

#define ITERATIONS_AMOUNT 8

#define CONS_COUNT 3
#define PROD_COUNT 3

#define BIN_SEM 0
#define BUF_FULL 1
#define BUF_EMPTY 2

#define MAX_RANDOM_PROD 2
#define MAX_RANDOM_CONS 5

int run_producer(cycle_buff_t *const buffer, const int sid, const int prod_id);
int run_consumer(cycle_buff_t *const buffer, const int sid, const int cons_id);

#endif  // __RUNNERS_H__
```

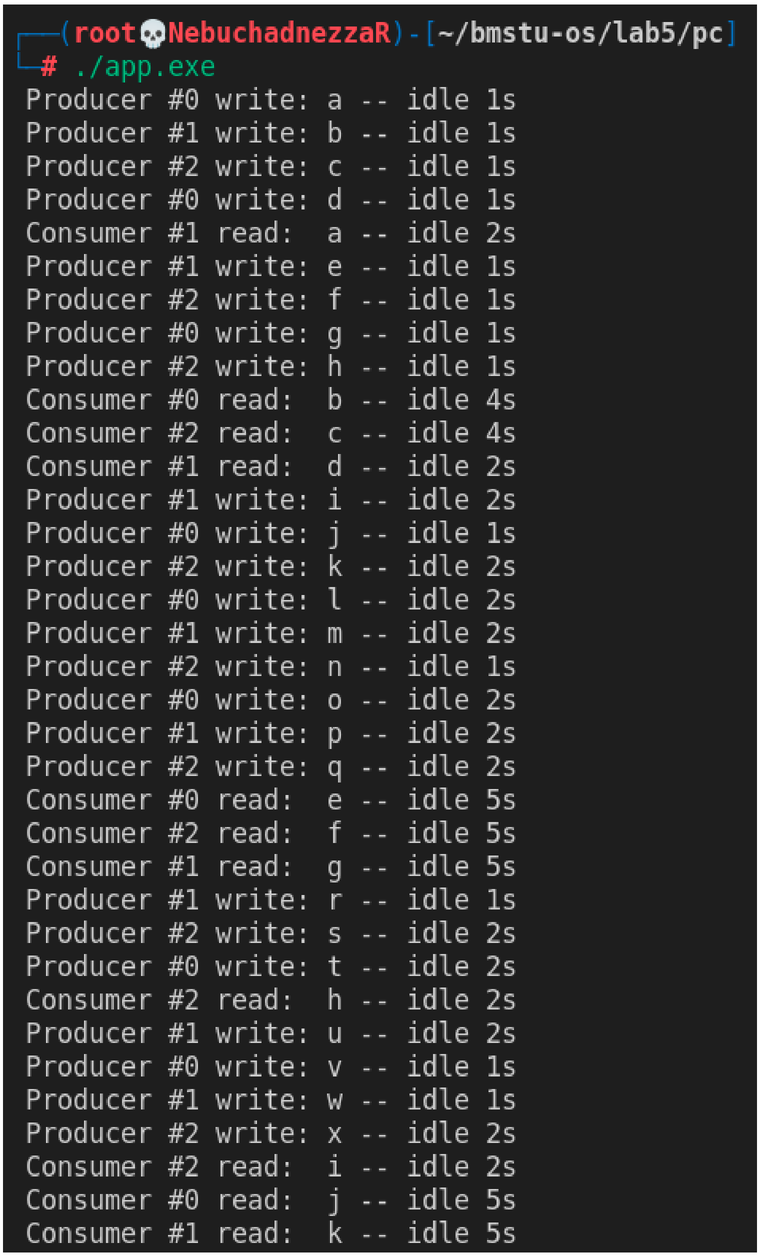
Листинг 4 – Реализация задачи. Заголовочник.

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

```

```c
#include "buffer.h"
#include "runners.h"

#define MAX_SEMS 3

int main(void) {
    setbuf(stdout, NULL);
    int fd = shmget(IPC_PRIVATE,
                    sizeof(cycle_buff_t),
                    IPC_CREAT | S_IRWXU | S_IRWXG | S_IRWXO);
    if (fd == -1) {
        perror("shmget failed!");
        return EXIT_FAILURE;
    }

    cycle_buff_t *buffer;
    if ((buffer = shmat(fd, 0, 0)) == (void *)-1) {
        perror("shmat failed!");
        return EXIT_FAILURE;
    }

    if (buff_init(buffer) == -1) {
        perror("init failed!");
        return EXIT_FAILURE;
    }

    int sid = semget(IPC_PRIVATE,
                     MAX_SEMS,
                     IPC_CREAT | S_IRWXU | S_IRWXG | S_IRWXO);
    if (sid == -1) {
        perror("semget failed!");
        return EXIT_FAILURE;
    }

    semctl(sid, BIN_SEM, SETVAL, 1);
    semctl(sid, BUF_EMPTY, SETVAL, N);
    semctl(sid, BUF_FULL, SETVAL, 0);

    int child_pid;
    for (short i = 0; i < PROD_COUNT; ++i) {
        switch ((child_pid = fork())) {
            case -1:
                perror("producer fork failed!");
                exit(EXIT_FAILURE);
                break;
            case 0:
                run_producer(buffer, sid, i);
                return EXIT_SUCCESS;
        }
    }

    for (short i = 0; i < CONS_COUNT; ++i) {
        switch ((child_pid = fork())) {
            case -1:
                perror("consumer fork failed!");
                exit(EXIT_FAILURE);
                break;
            case 0:
                run_consumer(buffer, sid, i);
                return EXIT_SUCCESS;
        }
    }

    for (short i = 0; i < CONS_COUNT + PROD_COUNT; ++i) {
        int status;
        if (wait(&status) == -1) {
            perror("children error!");
            exit(EXIT_FAILURE);
        }
        if (!WIFEXITED(status))
            puts("unexpected termination");
    }

    if (shmdt((void *)buffer) == -1 ||
        shmctl(fd, IPC_RMID, NULL) == -1 ||
        semctl(sid, IPC_RMID, 0) == -1) {
        perror("exit error!");

        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

Листинг 5 – Точка входа в программу

Работа программы



```
┌──(root💀NebuchadnezzaR)-[~/bmstu-os/lab5/pc]
└─# ./app.exe
Producer #0 write: a -- idle 1s
Producer #1 write: b -- idle 1s
Producer #2 write: c -- idle 1s
Producer #0 write: d -- idle 1s
Consumer #1 read:  a -- idle 2s
Producer #1 write: e -- idle 1s
Producer #2 write: f -- idle 1s
Producer #0 write: g -- idle 1s
Producer #2 write: h -- idle 1s
Consumer #0 read:  b -- idle 4s
Consumer #2 read:  c -- idle 4s
Consumer #1 read:  d -- idle 2s
Producer #1 write: i -- idle 2s
Producer #0 write: j -- idle 1s
Producer #2 write: k -- idle 2s
Producer #0 write: l -- idle 2s
Producer #1 write: m -- idle 2s
Producer #2 write: n -- idle 1s
Producer #0 write: o -- idle 2s
Producer #1 write: p -- idle 2s
Producer #2 write: q -- idle 2s
Consumer #0 read:  e -- idle 5s
Consumer #2 read:  f -- idle 5s
Consumer #1 read:  g -- idle 5s
Producer #1 write: r -- idle 1s
Producer #2 write: s -- idle 2s
Producer #0 write: t -- idle 2s
Consumer #2 read:  h -- idle 2s
Producer #1 write: u -- idle 2s
Producer #0 write: v -- idle 1s
Producer #1 write: w -- idle 1s
Producer #2 write: x -- idle 2s
Consumer #2 read:  i -- idle 2s
Consumer #0 read:  j -- idle 5s
Consumer #1 read:  k -- idle 5s
```

Рисунок 1 – «Производство-Потребление». Максимальная задержка потребителя – 5с, производителя – 2с.

Задача «Читатели-Писатели»

Листинги кода

```c
#include "io.h"

struct sembuf READER_QUEUE[] = {
    {READ_QUEUE, 1, 0},
    {WRITER, 0, 0},
    {WRITE_QUEUE, 0, 0},
};

struct sembuf READER_LOCK[] = {
    {READER, 1, 0},
    {READ_QUEUE, -1, 0},
};

struct sembuf READER_RELEASE[] = {
    {READER, -1, 0},
};

struct sembuf WRITER_QUEUE[] = {
    {WRITE_QUEUE, 1, 0},
    {READER, 0, 0},
    {WRITER, 0, 0},
};

struct sembuf WRITER_LOCK[] = {
    {WRITER, 1, 0},
    {WRITE_QUEUE, -1, 0},
};

struct sembuf WRITER_RELEASE[] = {
    {WRITER, -1, 0},
};

static inline int start_read(int sid) {
    return semop(sid, READER_QUEUE, 3) != -1 &&
           semop(sid, READER_LOCK, 2) != -1;
}
static inline int stop_read(int sid) {
    return semop(sid, READER_RELEASE, 1) != -1;
}

int reader_run(int *const shared_counter,
               const int sid,
               const int reader_id) {
    if (!shared_counter) {
        return -1;
    }

    srand(time(NULL) + reader_id);

    int sleep_time;
    for (short i = 0; i < ITERATIONS; ++i) {
        sleep_time = rand() % MAX_RANDOM + 1;
        sleep(sleep_time);

        if (!start_read(sid)) {
            perror("Something went wrong with start_read!");
            exit(EXIT_FAILURE);
        }

        // !!! --- CRITICAL --- !!!
        int val = *shared_counter;
        printf(" Reader #%d read:  %3d -- idle %ds\n", reader_id,
               val, sleep_time);
        // !!! --- CRITICAL --- !!!

        if (!stop_read(sid)) {
            perror("Something went wrong with stop_read!");
            exit(EXIT_FAILURE);
        }
    }
    return EXIT_SUCCESS;
}

static inline int write_start(int sid) {
    return semop(sid, WRITER_QUEUE, 3) != -1 &&
           semop(sid, WRITER_LOCK, 2) != -1;
}

static inline int write_stop(int sid) {
    return semop(sid, WRITER_RELEASE, 1) != -1;
}

int writer_run(int *const shared_counter,
               const int sid,
               const int writer_id) {
    if (!shared_counter) {
        return -1;
    }

    srand(time(NULL) + writer_id + READERS_COUNT);

    int sleep_time;
    for (short i = 0; i < ITERATIONS; ++i) {
        sleep_time = rand() % MAX_RANDOM + 1;
        sleep(sleep_time);

        if (!write_start(sid)) {
            perror("Something went wrong with write_start!");
            exit(EXIT_FAILURE);
        }

        // !!! --- CRITICAL --- !!!
        int val = ++(*shared_counter);
        printf(" Writer #%d write: %3d -- idle %ds\n", writer_id,
               val, sleep_time);
        // !!! --- CRITICAL --- !!!

        if (!write_stop(sid)) {
            perror("Something went wrong with write_stop!");
            exit(EXIT_FAILURE);
        }
    }

    return EXIT_SUCCESS;
}
```

Листинг 6 – Реализация задачи. Код.

```c
#ifndef __IO_H__
#define __IO_H__

#include <stdio.h>
#include <stdlib.h>
#include <sys/sem.h>
#include <time.h>
#include <unistd.h>

#define ITERATIONS 20
#define WRITERS_COUNT 3
#define READERS_COUNT 5

#define MAX_SEMS 4
#define READER 0
#define WRITER 1

#define READ_QUEUE 2
#define WRITE_QUEUE 3

#define MAX_RANDOM 3
```

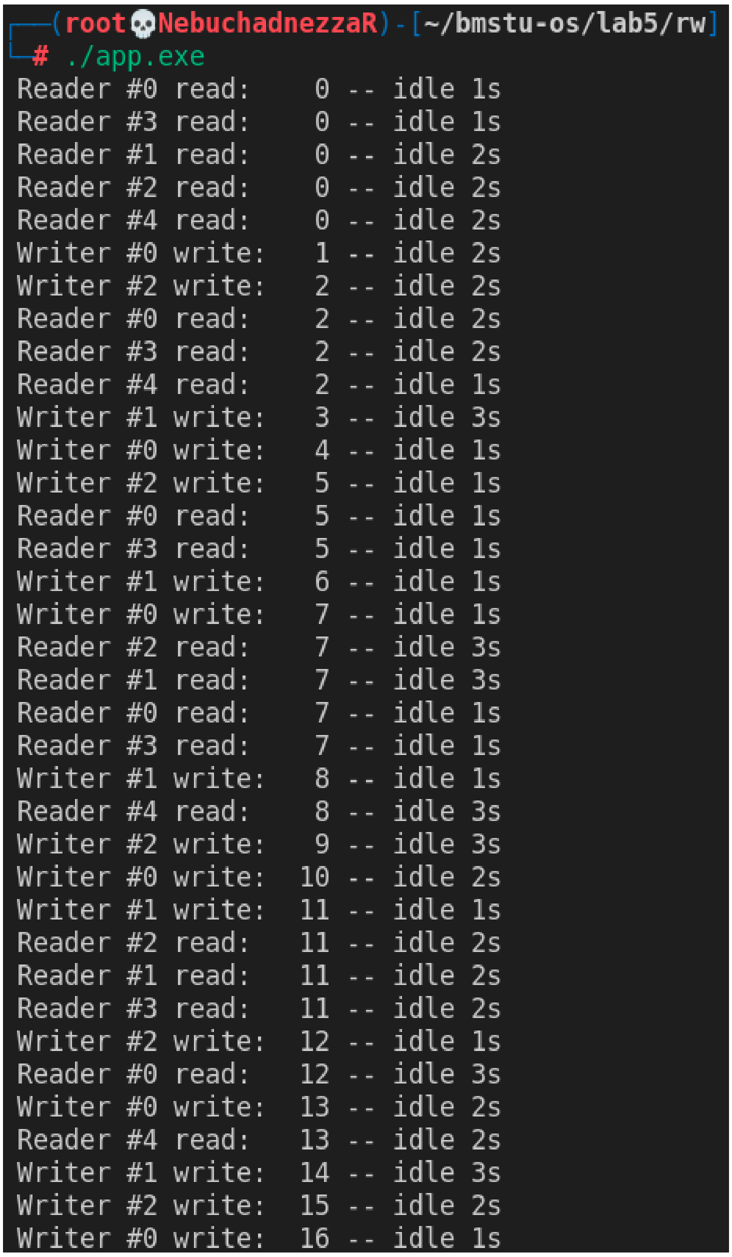
```c

int reader_run(int *const shared_counter, const int sid, const int reader_id);
int writer_run(int *const shared_counter, const int sid, const int writer_id);

#endif // __IO_H__
```

Листинг 7 – Реализация задачи. Заголовочник.

```c
#include <sys/shm.h>
#include <sys/stat.h>
#include <wait.h>

#include "io.h"

int main(void) {
    setbuf(stdout, NULL);
    int fd = shmget(IPC_PRIVATE,
                    sizeof(int),
                    IPC_CREAT | S_IRWXU | S_IRWXG | S_IRWXO);
    if (fd == -1) {
        perror("shmget failed!");
        return EXIT_FAILURE;
    }

    int *shared_counter;
    if ((shared_counter = shmat(fd, 0, 0)) == (void *)-1) {
        perror("shmat failed!");
        return EXIT_FAILURE;
    }

    int sid = semget(IPC_PRIVATE,
                     MAX_SEMS,
                     IPC_CREAT | S_IRWXU | S_IRWXG | S_IRWXO);
    if (sid == -1) {
        perror("semget failed!");
        return EXIT_FAILURE;
    }

    semctl(sid, READER, SETVAL, 0);
    semctl(sid, WRITER, SETVAL, 0);
    semctl(sid, WRITE_QUEUE, SETVAL, 0);
    semctl(sid, READ_QUEUE, SETVAL, 0);

    int child_pid;
    for (short i = 0; i < READERS_COUNT; ++i) {
        switch ((child_pid = fork())) {
            case -1:
                perror("reader fork failed!");
                exit(EXIT_FAILURE);
                break;
            case 0:
                reader_run(shared_counter, sid, i);
                return EXIT_SUCCESS;
        }
    }

    for (short i = 0; i < WRITERS_COUNT; ++i) {
        switch ((child_pid = fork())) {
            case -1:
                perror("writer fork failed!");
                exit(EXIT_FAILURE);
                break;
            case 0:
                writer_run(shared_counter, sid, i);
                return EXIT_SUCCESS;
        }
    }

    for (short i = 0; i < WRITERS_COUNT + READERS_COUNT; ++i) {
        int status;
        if (wait(&status) == -1) {
            perror("children error!");
            exit(EXIT_FAILURE);
        }

        if (!WIFEXITED(status)) {
            puts("unexpected termination");
        }
    }

    if (shmdt((void *)shared_counter) == -1 ||
        shmctl(fd, IPC_RMID, NULL) == -1 ||
        semctl(sid, IPC_RMID, 0) == -1) {

        perror("exit error!");

        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

Листинг 8 – Точка входа в программу

Работа программы



```
┌──(root💀NebuchadnezzaR)-[~/bmstu-os/lab5/rw]
└─# ./app.exe
Reader #0 read:    0 -- idle 1s
Reader #3 read:    0 -- idle 1s
Reader #1 read:    0 -- idle 2s
Reader #2 read:    0 -- idle 2s
Reader #4 read:    0 -- idle 2s
Writer #0 write:   1 -- idle 2s
Writer #2 write:   2 -- idle 2s
Reader #0 read:    2 -- idle 2s
Reader #3 read:    2 -- idle 2s
Reader #4 read:    2 -- idle 1s
Writer #1 write:   3 -- idle 3s
Writer #0 write:   4 -- idle 1s
Writer #2 write:   5 -- idle 1s
Reader #0 read:    5 -- idle 1s
Reader #3 read:    5 -- idle 1s
Writer #1 write:   6 -- idle 1s
Writer #0 write:   7 -- idle 1s
Reader #2 read:    7 -- idle 3s
Reader #1 read:    7 -- idle 3s
Reader #0 read:    7 -- idle 1s
Reader #3 read:    7 -- idle 1s
Writer #1 write:   8 -- idle 1s
Reader #4 read:    8 -- idle 3s
Writer #2 write:   9 -- idle 3s
Writer #0 write:  10 -- idle 2s
Writer #1 write:  11 -- idle 1s
Reader #2 read:   11 -- idle 2s
Reader #1 read:   11 -- idle 2s
Reader #3 read:   11 -- idle 2s
Writer #2 write:  12 -- idle 1s
Reader #0 read:   12 -- idle 3s
Writer #0 write:  13 -- idle 2s
Reader #4 read:   13 -- idle 2s
Writer #1 write:  14 -- idle 3s
Writer #2 write:  15 -- idle 2s
Writer #0 write:  16 -- idle 1s
```

Рисунок 2 – «Читатели-Писатели». Максимальная задержка – 3с.