Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2

Синхронизация потоков в ОС GNU/Linux
Тема

Преподаватель _____ А. С. Кузнецов
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цель

Изучение программных средств синхронизации потоков в ОС.

2 Задачи

1. Ознакомиться с краткими теоретическими сведениями по управлению протоками в ОС GNU/Linux.

2. Разработать программу в виде Linux-приложения, для выполнения различных частей которой создаются и запускается потоки управления, а для синхронизации доступа к требуемым ресурсам используются соответствующие объекты ОС

3. Вывести результат программы на консоль.

Описание варианта:

Имеется группа из N студентов, в начале ленты они договариваются между собой в каком порядке будут защищать лабораторные работы преподавателю. Не сдавший студент идет в конец очереди. Каждый студент защищает лабораторную какое-то время, а также с какой-то вероятностью (равномерный закон распределения) положительного или отрицательного результата.

Описанный процесс происходит бесконечно. Значение N также задается пользователем при старте процесса. В конце вывести список студентов, с пометкой ≪защитил≫/≪не защитил≫, сколько было попыток.

3 Исходные тексты программ

На листинге 1 представлен код программы с реализацией основного алгоритма.

Листинг 1 – Реализация программы по заданию

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
#include <unistd.h>
#include <stdbool.h>
#include <string.h>

#define DEFENSE_OPTIONS_NUMBER 2
#define MAX_PASS_TIME 20
#define MIN_PASS_TIME 5
#define TIME_ACCELERATION_COEFFICIENT 8 // Чем выше это значение, тем быстрее будет работать программа

typedef struct _student
{
    int studID;
    int retakeNumber;
    int isDefended;
} student_t;

typedef struct _threadNode
{
    student_t info;
    pthread_t thread;
    struct _threadNode* next;
} threadNode;

typedef struct _list
{
    threadNode* head;
    threadNode* tail;
} threadList;

/*! \brief Input validation for natural number
 *
```

```
 * \param number Int pointer where the appropriate number will be loaded
 *
 *  \return Nothing
 */
void inputNat(int* number)
{
    while (!scanf("%d", number))
    {
        fflush(stdin);
        printf("Please, input correct information!\n");
    }
    if (*number <= 0)
    {
        printf("Please, enter natural number!\n");
        inputNat(number);
    }
}

// Объявление мьютекса
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;

/*! \brief Generates the defence result using uniform distribution.
 *
 *  \return Has the student defend the lab ("has defended" or "hasn't defended").
 */
char* getRandomMark()
{
    int remainder = rand() % DEFENSE_OPTIONS_NUMBER;
    if (remainder == 1)
        return "has defended";
    else
        return "hasn't defended";
}

/*! \brief Generates a defence time using uniform distribution.
 *
 *  \return Lab defence time (from 5 to 20 minutes).
 */
int getRandomPassTime()
{
    return MIN_PASS_TIME + rand() % (MAX_PASS_TIME - MIN_PASS_TIME + 1);
}
```

```
/*! \brief Lab defence of one student (thread). Time and mark are random. This function is used while creating thread.
 *
 *  \param vars Variables that is passed during thread creation.
 *
 *  \return Nothing.
 */
void* defendLab(void* vars)
{
    student_t* info = (student_t*) vars;

    pthread_mutex_lock(&mtx);

    // Определение результатов экзамена
    srand(time(NULL));
    int passTime = getRandomPassTime();
    char* mark = getRandomMark();

    // Задержка потока
    int passSeconds = passTime / TIME_ACCELERATION_COEFFICIENT; // Количество секунд
    int passNanoseconds = (int) (((double) passTime / TIME_ACCELERATION_COEFFICIENT -
                                  passTime / TIME_ACCELERATION_COEFFICIENT) *
                                 1000000000); // Дробная часть секунд в наносекундах
    struct timespec delay = {passSeconds, passNanoseconds};
    nanosleep(&delay, NULL);

    printf("The defence has taken %d minutes. The student %d %s lab.\n", passTime, info->studID, mark);

    if (strcmp(mark, "has defended") == 0)
        info->isDefended = true;

    pthread_mutex_unlock(&mtx);

    return NULL;
}
```

```
/*! \brief Pushes the node to the end of the linked list. Can transform the pushed
node into a new thread.
 *
 *  \param plist Linked list.
 *  \param info Information that will be loaded into the new node.
 *  \param makeThread Whether it necessary to make a new node into a thread or
not.
 *
 *  \return Nothing.
 */
void push(threadList* plist, student_t info, bool makeThread)
{
    int threadStatus;

    // "Списковая" часть
    threadNode* node = (threadNode*) malloc(sizeof(threadNode));
    node->next = NULL;

    if (plist->head == NULL)
    {
        plist->head = node;
        plist->tail = node;
    }
    else
    {

        plist->tail->next = node;
        plist->tail = node;
    }

    if (!makeThread)
    {
        node->info = info;
        return;
    }

    // "Потоковая" часть
    threadStatus = pthread_create(&node->thread, NULL, defendLab, &info);
    if (threadStatus != 0)
    {
        printf("Error in creating thread. Thread number error: %d\n",
threadStatus);
```

```
        exit(threadStatus);
    }

    threadStatus = pthread_join(node->thread, NULL);
    if (threadStatus != 0)
    {
        printf("Error in joining thread. Thread number error: %d\n",
threadStatus);
        exit(threadStatus);
    }

    node->info = info;
}

/*! \brief Removes the first node from the linked list.
 *
 *  \param plist Linked list.
 *
 *  \return Nothing.
 */
void poll(threadList* plist)
{
    if (plist->head == NULL)
        return;

    if (plist->head == plist->tail)
    {
        plist->head = NULL;
        plist->tail = NULL;
        return;
    }

    plist->head = plist->head->next;
}

/*! \brief Displays the linked list content.
 *
 *  \param plist Linked list.
 *
 *  \return Nothing.
 */
void showInfo(threadList plist)
```

```
{
    puts("------------------------------------------------------------------------
-------------------------");
    for (threadNode* i = plist.head; i != NULL; i = i->next)
    {
        printf("Student  %d  has  passed  on  %d  try.\n",  i->info.studID,  i-
>info.retakeNumber + 1);
    }
    puts("------------------------------------------------------------------------
-------------------------");
}

/*! \brief Frees the memory of every node in linked list.
 *
 *  \param plist Linked list.
 *
 *  \return Nothing.
 */
void freeList(threadList* plist)
{
    threadNode* tmp = NULL;
    while (plist->head != NULL)
    {
        tmp = plist->head;
        plist->head = plist->head->next;
        free(tmp);
    }
}

int main()
{
    while (true)
    {
        srand(time(NULL));
        int studentsNumber;
        puts("Input the number of students in group:");
        inputNat(&studentsNumber);

        threadList group = {NULL, NULL};

        printf("Group of %d students is ready to defend labs.\n", studentsNumber);
```

```
        // Инициализация потоков в списке. Первая попытка сдачи
        for (int i = 0; i < studentsNumber; i++)
        {
            student_t studInfo = {i + 1, 0, false};
            push(&group, studInfo, true);
        }


        puts("");

        // Пересдачи
        threadList defendedGroup = {NULL, NULL};
        do
        {
            if (!group.head->info.isDefended)
            {
                group.head->info.retakeNumber++;
                push(&group, group.head->info, true);
            }
            else
                push(&defendedGroup, group.head->info, false);
            poll(&group);
        } while (group.head != NULL);

        showInfo(defendedGroup);
        freeList(&defendedGroup);
    }
    return 0;
}
```

4 Тестовые примеры работы программ

```
root@brain:/mnt/lab2# ./main
Group of 5 students is ready to defend labs.
The defence has taken 7 minutes. The student 1 hasn't defended lab.
The defence has taken 13 minutes. The student 2 hasn't defended lab.
The defence has taken 14 minutes. The student 3 has defended lab.
The defence has taken 7 minutes. The student 4 hasn't defended lab.
The defence has taken 20 minutes. The student 5 has defended lab.

The defence has taken 20 minutes. The student 1 has defended lab.
The defence has taken 13 minutes. The student 2 has defended lab.
The defence has taken 14 minutes. The student 4 has defended lab.
-----------------------------------------------------------------------------------------
Student 3 has passed on 1 try.
Student 5 has passed on 1 try.
Student 1 has passed on 2 try.
Student 2 has passed on 2 try.
Student 4 has passed on 2 try.
-----------------------------------------------------------------------------------------
Group of 6 students is ready to defend labs.
The defence has taken 20 minutes. The student 1 has defended lab.
The defence has taken 19 minutes. The student 2 hasn't defended lab.
The defence has taken 10 minutes. The student 3 hasn't defended lab.
The defence has taken 20 minutes. The student 4 hasn't defended lab.
The defence has taken 12 minutes. The student 5 has defended lab.
The defence has taken 9 minutes. The student 6 has defended lab.
```

Рисунок 1 – Консольный вывод программы