

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №3 по дисциплине «Технологии промышленного программирования»

Обучающийся группы 6232-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Портнов К.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Баландин А.В.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147860736)

[2 Результаты работы 4](#_Toc147860737)

[ПРИЛОЖЕНИЕ А Листинг модуля M1 5](#_Toc147860738)

1. Постановка задачи

Разработать приложение, состоящее из одного процесса с тремя запущенными нитями:

1. М(main).
2. Т1(F1).
3. Т2(F2).

В качестве нити М(main) выступает функция main(). Нити Т1(F1) и Т2(F2) запускаются нитью М(main) на базе соответственно функций F1() и F2(). Все три нити, работая параллельно, должны совместно динамически сформировать текст вида:

"Text0, Text1, Text2.\n"

Порядок работы приложения

Вначале нить М(main) запускает нить Т1(F1), передавая ей в качестве параметра указатель совместно формируемого текста и функцию F2.

Далее нить М(main), записывая в текст букву за буквой, формирует свою часть текста:

"Text0, "

После формирования нитями всего текста нить М(main) выдаёт его на печать и завершает свою работу.

Запущенная нить Т1(F1) запускает нить Т2(F2), после чего тем же способом добавляет в формируемый текст свою часть: "Text1, "

После завершения записи соей части текста нить Т1(F1) ожидает завершения выполнения нити Т2(F2), после чего завершает свою работу.

Запущенная нить Т2(F2) должна так же добавить в формируемый текст свою часть:

"Text2.\n "

после чего завершает свою работу. Замечание. Для имитации времени записи в текст одной буквы использовать пустой цикл в 1000 итераций.

Необходимо использовать: Блокировки чтения/записи. Ждущие блокировки. Барьеры.

1. Результаты работы

Результаты работы представлены в виде вывода на консоль сообщений во время выполнения программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Результат выполнения программы в консоли

ПРИЛОЖЕНИЕ А  
Листинг модуля M1

**#include** <cstdlib>

**#include** <iostream>

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <string.h>

**#include** <pthread.h>

**#include** <pthread.h>

**#include** <time.h>

**#include** <unistd.h>

**#define** GetCurrentDir getcwd

**using** std::cout;

**using** std::endl;

// Блокировки чтения/записи. Ждущие блокировки. Барьеры.

// Отображение текущей директории

**void** **viewWorkDirectory**();

// Функция потока T1

**void**\* **funcT1**(**void**\* args);

// Функция потока T2

**void**\* **funcT2**(**void**\* args);

// Запись одного символа с имитацией времени записи

**void** **appendChar**(**char** \*text, **char** c);

// Распечатывание буфера в консоли

**void** **printBuffer**(**char** \*text);

// Структура параметров для T1

**struct** threadT1dArg {

**void**\* func;

**char**\* textBuf;

};

// Объект "блокировка чтение/запись"

pthread\_rwlock\_t rwlock;

// Объект "барьер"

pthread\_barrier\_t barrier;

// Флаг завершения записи нитью T1

**bool** isProcessT1 = **false**;

**char**\* mainText = (**char**\*) "Text0, ";

**char**\* t1Text = (**char**\*) "Text1, ";

**char**\* t2Text = (**char**\*) "Text2.\n ";

**int** **main**(**int** argc, **char** \*argv[]) {

cout << "Main: Старт" << endl;

viewWorkDirectory();

// Буфер формируемого текста

**char** textBuf[200] = {};

// Инициализация блокировки чтения/записи

**pthread\_rwlock\_init**(&rwlock, NULL);

**int** rc;

// Инициализация барьера

**pthread\_barrier\_init**(&barrier, NULL, 3);

cout << "Main: Получение блокировки для записи" << endl;

rc = **pthread\_rwlock\_wrlock**(&rwlock);

**if**(rc != 0){

cout << "Main: Ошибка получения блокировки для записи " << strerror(rc) << endl;

exit(EXIT\_FAILURE);

}

**struct** threadT1dArg t1Arg = {(**void**\*) funcT2, textBuf};

// дескриптор нити, атрибутная запись, функция нити, аргументы - буфер текста

pthread\_t threadT1;

**int** threadT1Res = **pthread\_create**(&threadT1, NULL, funcT1, &t1Arg);

**if**(threadT1Res != 0){

cout << "Main: Ошибка старта T1 " << strerror(threadT1Res) << endl;

**return** EXIT\_FAILURE;

}

cout << "Main: Поток T1 создан - " << threadT1 << endl;

// Посимвольная запись в буфер текста нитью main

**for** (**unsigned** **int** i = 0; i < strlen(mainText); i++) {

appendChar(textBuf, mainText[i]);

}

cout << "Main: Освобождение блокировки для записи" << endl;

rc = **pthread\_rwlock\_unlock**(&rwlock);

**if**(rc != 0){

cout << "Main: Ошибка освобождения блокировки для записи " << strerror(rc) << endl;

exit(EXIT\_FAILURE);

}

cout << "Main: Закончил запись текста" << endl;

cout << "Main: У барьера" << endl;

**pthread\_barrier\_wait**(&barrier);

cout << "Main: Прошёл барьер" << endl;

printBuffer(textBuf);

**return** EXIT\_SUCCESS;

}

// Функция потока T1

**void**\* **funcT1**(**void**\* args) {

cout << "T1: Старт" << endl;

**struct** threadT1dArg arg = \*((**struct** threadT1dArg\*)args);

// дескриптор нити, атрибутная запись, функция нити, аргументы - буфер текста

pthread\_t threadT2;

**int** threadT2Res = **pthread\_create**(&threadT2, NULL, (**void** \*(\*)(**void** \*)) arg.func, (**void**\*) arg.textBuf);

**if**(threadT2Res != 0){

cout << "T1: Ошибка старта T2 " << strerror(threadT2Res) << endl;

exit(EXIT\_FAILURE);

}

cout << "T1: Поток T2 создан - " << threadT2 << endl;

**int** rc;

**int** count = 0;

**bool** isWait = **true**;

**while**(isWait){

rc = **pthread\_rwlock\_tryrdlock**(&rwlock);

// EBUSY - 16

**if** (rc == 16) {

**if** (count >= 15) {

cout << "T1: Достигнуто ограничение на число попыток" << endl;

exit(EXIT\_FAILURE);

}

++count;

cout << "T1: Не удалось получить блокировку - ожидание" << endl;

**sleep**(1);

} **else**{

**if**(rc != 0){

cout << "T1: Ошибка получения блокировки для записи " << strerror(rc) << endl;

exit(EXIT\_FAILURE);

} **else** {

isWait = **false**;

}

}

}

cout << "T1: Приступаю к работе" << endl;

cout << "T1: Захват буфера для записи текста" << endl;

**for** (**unsigned** **int** i = 0; i < strlen(t1Text); i++) {

//pthread\_sleepon\_lock();

appendChar(arg.textBuf, t1Text[i]);

//isProcessT1 = false;

//pthread\_sleepon\_signal(&isProcessT1);

//pthread\_sleepon\_unlock();

}

cout << "T1: Закончил запись текста" << endl;

isProcessT1 = **true**;

**pthread\_sleepon\_lock**();

**pthread\_sleepon\_signal**(&isProcessT1);

**pthread\_sleepon\_unlock**();

cout << "T1: У барьера" << endl;

**pthread\_barrier\_wait**(&barrier);

cout << "T1: Прошёл барьер" << endl;

**return** EXIT\_SUCCESS;

}

// Функция потока T2

**void**\* **funcT2**(**void**\* args) {

cout << "T2: Старт" << endl;

**char**\* textBuf = (**char**\*) args;

**pthread\_sleepon\_lock**();

//Проверка надо ли ждать и ожидание смены флага

**while**(isProcessT1 == **false**){

cout << "T2: Ожидание T1" << endl;

**pthread\_sleepon\_wait**(&isProcessT1);

}

**pthread\_sleepon\_unlock**();

cout << "T2: Приступаю к работе" << endl;

cout << "T2: Захват буфера для записи текста" << endl;

**const** **char**\* t2Text = "Text2.\n ";

**for** (**unsigned** **int** i = 0; i < strlen(t2Text); i++) {

appendChar(textBuf, t2Text[i]);

}

**pthread\_sleepon\_unlock**();

cout << "T2: Закончил запись текста" << endl;

cout << "T2: У барьера" << endl;

**pthread\_barrier\_wait**(&barrier);

cout << "T2: Прошёл барьер" << endl;

**return** EXIT\_SUCCESS;

}

// Запись одного символа с имитацией времени записи

**void** **appendChar**(**char** \*text, **char** c) {

size\_t p = strlen(text);

text[p] = c;

text[p + 1] = '\0';

**for** (**int** i = 0; i < 100000000; i++);

**sleep**(1);

}

// Отображение текущей директории

**void** **viewWorkDirectory**(){

**char** cCurrentPath[FILENAME\_MAX];

**if** (!GetCurrentDir(cCurrentPath, **sizeof**(cCurrentPath)))

exit(EXIT\_FAILURE);

cCurrentPath[**sizeof**(cCurrentPath) - 1] = '\0';

cout << "Log: Текущая рабочая директория - " << cCurrentPath << endl;

time\_t rawtime;

time(&rawtime);

**struct** tm\* ptm = gmtime(&rawtime);

cout << "Log: Текущая дата/время - " << ptm->tm\_mday << "." << 1 + ptm->tm\_mon << "." << 1900 + ptm->tm\_year << " " << ptm->tm\_hour << ":" << 1 + ptm->tm\_min << ":" << 1 + ptm->tm\_sec << endl;

}

// Распечатывание буфера в консоли

**void** **printBuffer**(**char** \*text){

size\_t p = strlen(text);

cout << "Log: Буфер сейчас: ";

**for**(**int** i = 0; i < p; i++){

cout << text[i];

}

cout << endl;

}