МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра Стратегічного управління

ЗВІТ

з лабораторної роботи № 5

з дисципліни «Операційні системи»

на тему «Застосування потоків»

Перевірила: старший викладач

Вільхівська О. В.

Виконав: ст. гр. КН-27

Харків, 2018

Л**абораторная работа №5**

**Тема:** Применением потоков

**Задания**

**Задание 1**

1. Разработать три многопоточные программы с использованием минимум двух потоков и различных средств синхронизации. Например: два потока записывают и читают информацию из одного файла; два потока увеличивают значение общей переменной; два потока с различной частотой считывают и записывают данные в общий буфер памяти.

2. Необходимо обеспечить синхронизированную работу потоков в критической секции с использованием: мьютексов; семафоров; условных переменных.

3. Убедиться в результативности применения средств синхронизации потоков, сравнив результаты работы программ с использованием и без использования средств синхронизации.

**Задание 2. Индивидуальное задание (Вариант №2)**

В пансионе отдыхают и предаются размышлениям 5 философов, пронумерованные от 1 до 5. В столовой расположен круглый стол, вокруг которого расставлены 5 стульев, также пронумерованные от 1 до 5. На столе находится одна большая тарелка со спагетти, которая пополняется бесконечно, также там расставлены 5 тарелок, в которые накладывается спагетти, и 5 вилок, назначение которых очевидно.

Для того чтобы пообедать, философ входит в столовую и садится на стул со своим номером. При этом есть философ сможет только в том случае, если свободны две вилки – справа и слева от его тарелки. При выполнении этого условия философ поднимает одновременно обе вилки и может поглощать пищу в течение какого-то заданного времени. В противном случае, философу приходится ждать освобождения обеих вилок. Пообедав, философ кладет обе вилки на стол одновременно и уходит.

Описанный процесс происходит бесконечно.

Воспользоваться объектами синхронизации типа «мьютекс».

**Ход Работы**

**Задание 1**

Код программы (чтение и запись в файл с использованием двух потоков и двух двоичных семафоров)

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <string.h>  #include <pthread.h>  #include <semaphore.h>  #include <iostream>  #include <fstream>  #include <sstream>  std::string firstLine(){  std::ifstream fin;  fin.open("example.txt");  std::string sLine;  if (fin.good())  {  getline(fin, sLine);  }  fin.close();  return sLine.c\_str();  }  int writeFile (char\* str)  {  std::ofstream myfile;  myfile.open ("example.txt");  myfile << str;  myfile.close();  return 0;  }  void \*thread\_function(void \*arg);  sem\_t bin\_sem1;  sem\_t bin\_sem2;  #define WORK\_SIZE 1024  char work\_area[WORK\_SIZE];  void \*thread\_function (void \*arg) { // Thread 2 (used to count characters)  while(strncmp("end", work\_area, 3) != 0) {  sem\_wait(&bin\_sem2);  writeFile(work\_area);  printf("Thread 2. You wrote to file: %s", work\_area);  sleep(1);  sem\_post(&bin\_sem1);  }  pthread\_exit(NULL);  }  int main() {  int res,res2;  pthread\_t a\_thread;  void \*thread\_result;  res = sem\_init(&bin\_sem1, 0, 1);  res2 = sem\_init(&bin\_sem2, 0, 0);  if ((res != 0)||(res2!=0)) {  perror("Semaphore initialization failed");  exit(EXIT\_FAILURE);  }  res = pthread\_create(&a\_thread, NULL, thread\_function, NULL);  if (res != 0) {  perror("Thread creation failed");  exit(EXIT\_FAILURE);  }  // Thread 1 (Used to enter characters)  while (strncmp("end", work\_area, 3) != 0) {  sem\_wait(&bin\_sem1);  printf("Input some text. Enter 'end' to finish\n");  fgets(work\_area, WORK\_SIZE, stdin);  std::ifstream infile("example.txt");    printf("Thread 1. Fetched from file: %s\n", firstLine().c\_str());  sem\_post(&bin\_sem2);  }  printf("\nWaiting for thread to finish...\n");  sem\_destroy(&bin\_sem1);  sem\_destroy(&bin\_sem2);  exit(EXIT\_SUCCESS);  } |

Пример работы

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/lab5$ g++ -pthread task\_1.cpp  reacher@reacher-X550CC:~/NetBeansProjects/lab5$ ./a.out  Input some text. Enter 'end' to finish lol Thread 1. Fetched from file: end Thread 2. You wrote to file: lol Input some text. Enter 'end' to finish end Thread 1. Fetched from file: lol  Waiting for thread to finish... Thread joined |

Код программы. Два потока, изменяющие общую переменную и мьютекс

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <string.h>  #include <pthread.h>  #include <semaphore.h>  void \*thread\_function(void \*arg);  pthread\_mutex\_t work\_mutex;/\* защищает work\_area и time\_to\_exit \*/  #define WORK\_SIZE 1024  char work\_area[WORK\_SIZE];  int time\_to\_exit = 0;  int commonVar=0,temp=0;  void \*thread\_function(void \*arg) {  while(commonVar<10) {  pthread\_mutex\_lock(&work\_mutex);  commonVar++;  temp=commonVar;  printf("Thread 2: commonVar = %d\n", commonVar);  pthread\_mutex\_unlock(&work\_mutex);  sleep(1);  }  time\_to\_exit = 1;  pthread\_mutex\_unlock(&work\_mutex);  pthread\_exit(0);  }  int main() {  int res;  pthread\_t a\_thread;  void \*thread\_result;  res = pthread\_mutex\_init(&work\_mutex, NULL);  if (res !=0) {  perror("Mutex initialization failed");  exit(EXIT\_FAILURE);  }  res = pthread\_create (&a\_thread, NULL, thread\_function, NULL);  if (res !=0) {  perror("Thread creation failed");  exit(EXIT\_FAILURE);  }  while (!time\_to\_exit) {  pthread\_mutex\_lock(&work\_mutex);  commonVar++;  temp=commonVar;  printf("Thread 1: commonVar = %d\n", commonVar);  pthread\_mutex\_unlock(&work\_mutex);  sleep(1);  }    pthread\_mutex\_unlock(&work\_mutex);  printf("\nWaiting for thread to finish...\n");  res = pthread\_join(a\_thread, &thread\_result);  if (res !=0) {  perror("Thread join failed");  exit(EXIT\_FAILURE);  }  printf("Thread joined\n");  pthread\_mutex\_destroy(&work\_mutex);  exit(EXIT\_SUCCESS);  } |

Пример работы

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/lab5$ g++ -pthread mutex.cpp reacher@reacher-X550CC:~/NetBeansProjects/lab5$ ./a.out Thread 1: commonVar = 1 Thread 2: commonVar = 2 Thread 1: commonVar = 3 Thread 2: commonVar = 4 Thread 1: commonVar = 5 Thread 2: commonVar = 6 Thread 1: commonVar = 7 Thread 2: commonVar = 8 Thread 1: commonVar = 9 Thread 2: commonVar = 10 Thread 1: commonVar = 11  Waiting for thread to finish... Thread joined |

Код программы

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <string.h>  #include <signal.h>  #include <pthread.h>  #include <unistd.h>  /\* compile with gcc -pthread lockwait.c \*/  pthread\_cond\_t cv;  pthread\_mutex\_t lock;  char work\_area[1024];  void storeData(char\* text){  //functions that copies text to clipboard  }  void \*threadOne(void \*v) {    // pthread\_mutex\_lock(&lock);  while(strncmp("end", work\_area, 3) != 0) {  pthread\_mutex\_lock(&lock);  printf("Thread 1: Enter some text\n");  fgets(work\_area, 1024, stdin);  storeData(work\_area);  printf("Thread 1: Copied to buffer\n");  pthread\_cond\_wait(&cv, &lock);  pthread\_mutex\_unlock(&lock);  }  /\* do something that requires holding the mutex and condition is true \*/  pthread\_mutex\_unlock(&lock);    pthread\_exit(NULL);  }  void \*threadTwo(void \*v){    // pthread\_mutex\_lock(&lock);  /\* do something that might make condition true \*/  while(strncmp("end", work\_area, 3) != 0) {  pthread\_mutex\_lock(&lock);  printf("Thread 2: Enter some text\n");  fgets(work\_area, 1024, stdin);  storeData(work\_area);  printf("Thread 2: Copied to buffer\n");  pthread\_cond\_signal(&cv);  pthread\_mutex\_unlock(&lock);  sleep(1);  }  pthread\_exit(NULL);  }  main() {  int res;  pthread\_t thread\_one;  pthread\_t thread\_two;  void \*thread\_result;  res = pthread\_create (&thread\_one, NULL, threadOne, NULL);  if (res !=0) {  perror("Thread creation failed");  exit(EXIT\_FAILURE);  }  res = pthread\_create (&thread\_two, NULL, threadTwo, NULL);  if (res !=0) {  perror("Thread creation failed");  exit(EXIT\_FAILURE);  }  pthread\_join(thread\_one, &thread\_result);  pthread\_join(thread\_two, &thread\_result);  exit(EXIT\_SUCCESS);  } |

Пример работы

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/Threads$ g++ -pthread main.cpp  reacher@reacher-X550CC:~/NetBeansProjects/Threads$ ./a.out  Thread 1: Enter some text  1  Thread 1: Copied to buffer  Thread 2: Enter some text  2  Thread 2: Copied to buffer  Thread 1: Enter some text  3  Thread 1: Copied to buffer  Thread 2: Enter some text  4  Thread 2: Copied to buffer  Thread 1: Enter some text |

**2. Индивидуальное задание (Вариант №2)**

Код программы

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <string.h>  #include <pthread.h>  #define NUM\_THREADS 5  pthread\_mutex\_t work\_mutex;  bool forks[] = {true,true,true,true,true};  void \*thread\_function(void \*arg) {  int my\_number = \*(int\*)arg;  int rand\_num = 1 + (int)(9.0\*rand() / (RAND\_MAX+1.0));  while(true){  pthread\_mutex\_lock(&work\_mutex);  if ((forks[(my\_number+NUM\_THREADS-1)%NUM\_THREADS])&&(forks[(my\_number+NUM\_THREADS)%NUM\_THREADS])){  printf("Philosopher #%d is having a meal\n", my\_number+1);  // printf("Argument was %d %d %d\n", my\_number+1,forks[(my\_number+NUM\_THREADS-1)%NUM\_THREADS],forks[(my\_number+NUM\_THREADS)%NUM\_THREADS]);  forks[(my\_number+NUM\_THREADS-1)%NUM\_THREADS]=false;  forks[(my\_number+NUM\_THREADS)%NUM\_THREADS]=false;  pthread\_mutex\_unlock(&work\_mutex);  sleep(rand\_num);  forks[(my\_number+NUM\_THREADS-1)%NUM\_THREADS]=true;  forks[(my\_number+NUM\_THREADS)%NUM\_THREADS]=true;  printf("Philosopher #%d left\n", my\_number+1);  }  pthread\_mutex\_unlock(&work\_mutex);  sleep(1); // break before next meal  }  pthread\_exit(NULL);  }  int main() {  int res;  void \*thread\_result;  pthread\_t a\_thread[NUM\_THREADS];  int lots\_of\_threads;  for(lots\_of\_threads=0;lots\_of\_threads<NUM\_THREADS;lots\_of\_threads++) {  pthread\_create(&(a\_thread[lots\_of\_threads]),NULL,thread\_function, (void\*)&lots\_of\_threads);  sleep(1);  }  for(lots\_of\_threads = NUM\_THREADS - 1; lots\_of\_threads >= 0;lots\_of\_threads--) {  pthread\_join(a\_thread[lots\_of\_threads], &thread\_result);  }  exit(EXIT\_SUCCESS);  } |

Пример работы

|  |
| --- |
| reacher@reacher-X550CC:~/NetBeansProjects/lab5$ g++ -pthread philosophers.cpp reacher@reacher-X550CC:~/NetBeansProjects/lab5$ ./a.out Philosopher #1 is having a meal Philosopher #3 is having a meal Philosopher #1 left Philosopher #5 is having a meal Philosopher #3 left Philosopher #2 is having a meal Philosopher #2 left Philosopher #3 is having a meal Philosopher #5 left Philosopher #1 is having a meal Philosopher #3 left Philosopher #4 is having a meal Philosopher #1 left Philosopher #2 is having a meal Philosopher #2 left |

**Выводы**

Во время выполнения данной лабораторной было продолжено изучение потоков в C++. Создание потока отличается от функции fork тем, что при его создании указывается ссылка на функцию, которая будет выполняться.

При работе с потоками используют функцию pthread\_join – эквивалент функции wait при работе с процессами.

Для синхронизации потоков могут использоваться семафоры, мьютексы, условные переменные. Все эти способы синхронизации были рассмотрены в текущей лабораторной работе.