МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №5

З предмету «Операційні системи»

Виконав

Студент групи КН-36а

Рубан Ю.Д.

Перевірив:

Кондратов О.М.

Харків 2017

ПРИМИНЕНИЕ ПОТОКОВ

**Цель:** Ознакомиться с теоретическим материалом.

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:**

Рассмотрим взаимодействие двух потоков, один из которых пишет данные в буферный пул, а другой считывает их из пула. Буферный пул состоит из N буферов, каждый содержит одну запись. В общем случае поток-писатель и поток-читатель имеют разные скорости и обращаются к пулу с переменной интенсивностью. Для правильной работы поток-писатель приостанавливается, когда все буферы заняты, и переходит в активное состояние при наличии хотя бы одного свободного буфера. Поток-читатель приостанавливается, когда все буферы пусты, и активизируется, когда появляется, по крайней мере, одна запись. Описанный процесс происходит бесконечно. Воспользоваться объектами синхронизации типа «мьютекс».

Код программы:

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

#include<stdio.h>

#include<semaphore.h>

#include<iostream>

using namespace std;

const int SIZE = 1024;

char\*\* buffer\_pool;

pthread\_mutex\_t mutex;

bool is\_empty(char\*\* pool,int n)

{

if(pool[n][0]!='\0'){return false;}

return true;

}

bool is\_full(char\*\*pool,int n)

{

if(pool[n][0]=='\0')return false;

return true;

}

void \*thread\_read(void \*arg)

{

int n = \*((int\*)(arg));

for(;;)

{

for(int i=0;i<n;i++)

{

if(!is\_empty(buffer\_pool,i)){

pthread\_mutex\_lock(&mutex);

printf("reading from buffer pool\n");

printf("%s\n",buffer\_pool[i]);

buffer\_pool[i][0]='\0';

pthread\_mutex\_unlock(&mutex);}

sleep(1);

}

//sleep(1);

}

return NULL;

}

void\* thread\_write(void\* arg)

{

int n = \*(int\*)(arg);

for(;;)

{

for(int i=0;i<n;i++)

{

if(!is\_full(buffer\_pool,i))

{

pthread\_mutex\_lock(&mutex);

printf("writing in pools\n");

cin>>buffer\_pool[i];

pthread\_mutex\_unlock(&mutex);

}

sleep(1);

}

}

}

int main(){

int n;

pthread\_t thread\_r,thread\_w;

printf("enter n\n");

cin>>n;

buffer\_pool = (char\*\*)calloc(n,sizeof(char\*));

for(int i=0;i<n;i++)

{

buffer\_pool[i] = (char\*)calloc(SIZE,sizeof(char));

buffer\_pool[i][0] = '\0';

}

if(pthread\_mutex\_init(&mutex,NULL)!=0)return -1;

if(pthread\_create(&thread\_w,NULL,thread\_write,&n)!=0)return -1;

if(pthread\_create(&thread\_r,NULL,thread\_read,&n)!=0)return -1;

pthread\_join(thread\_w,NULL);

pthread\_join(thread\_r,NULL);

return 0;

}

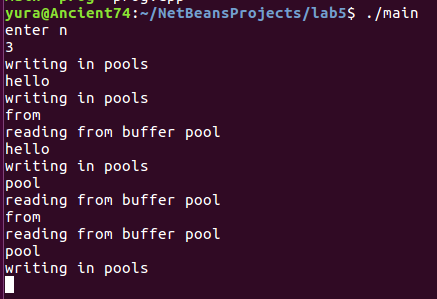


Рисунок 1.1 – результат выполнения индивидуального задания

Код основного задания с использованием семафоров:

#include<pthread.h>

#include<iostream>

#include<semaphore.h>

using namespace std;

int variable = 0;

sem\_t sem;

void\* inc(void\*)

{

sem\_wait(&sem);

for(int i =0;i<=10;i++)variable++;

sem\_post(&sem);

return NULL;

}

void\* decr(void\*)

{

sem\_wait(&sem);

for(int i =0;i<=10;i++)variable--;

sem\_post(&sem);

return NULL;

}

int main()

{

sem\_init(&sem,0,1);

pthread\_t inct,dect;

pthread\_create(&inct,NULL,inc,NULL);

pthread\_create(&dect,NULL,decr,NULL);

pthread\_join(inct,NULL);

pthread\_join(dect,NULL);

cout<<variable<<endl;

return 0;

}

Код основного задания с использованием мьютексов:

#include<pthread.h>

#include<iostream>

#include<semaphore.h>

using namespace std;

int variable = 0;

pthread\_mutex\_t mutex;

void\* inc(void\*)

{

pthread\_mutex\_lock(&mutex);

for(int i =0;i<=10;i++)variable++;

pthread\_mutex\_unlock(&mutex);

return NULL;

}

void\* decr(void\*)

{

pthread\_mutex\_lock(&mutex);

for(int i =0;i<=10;i++)variable--;

pthread\_mutex\_unlock(&mutex);

return NULL;

}

int main()

{

pthread\_mutex\_init(&mutex,NULL);

pthread\_t inct,dect;

pthread\_create(&inct,NULL,inc,NULL);

pthread\_create(&dect,NULL,decr,NULL);

pthread\_join(inct,NULL);

pthread\_join(dect,NULL);

cout<<variable<<endl;

return 0;

}

Код основного задания с использованием общей переменной:

#include<pthread.h>

#include<iostream>

#include<semaphore.h>

using namespace std;

int variable = 0;

int protected\_variable=1;

void\* inc(void\*)

{

if(protected\_variable==1)

{

for(int i =0;i<=10;i++)variable++;

protected\_variable=0;

}

return NULL;

}

void\* decr(void\*)

{

if(protected\_variable==0)

{

for(int i =0;i<=10;i++)variable--;

protected\_variable=1;

}

return NULL;

}

int main()

{

pthread\_t inct,dect;

pthread\_create(&inct,NULL,inc,NULL);

pthread\_create(&dect,NULL,decr,NULL);

pthread\_join(inct,NULL);

pthread\_join(dect,NULL);

cout<<variable<<endl;

return 0;

}

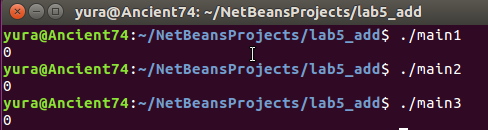


Рисунок 1.2 – Результат выполнения программ основного задания

**Выводы:**

В данной лабораторной работе был изучен теоретический материал по использованию потоков и средств синхронизации, таких как семафоры, мьютексы и общие переменные. Так же были получены практические навыки по использованию потоков.