Министерство образования и науки Российской Федерации

Югорский государственный университет

Институт цифровой экономики

Отчет о лабораторной работе №4

По дисциплине «Разработка системных приложений»

Выполнил: студент группы 1161б

Стародубов Юрий

Проверил: Шицелов А.В.

Ханты-Мансийск

2019

Цель:

Освоение средств параллельного программирования, изучение механизмов синхронизации процессов при помощи семафоров, а также обмена данными при помощи сегментов разделяемой памяти.

Ход работы:

Задание

Процесс 1 порождает потомка 2, они присоединяют к себе разделяемую память объемом (N\*sizeof(int)). Процесс 1 пишет в нее k1 чисел сразу, процесс 2 переписывает k2 чисел из памяти в файл. Процесс 1 может производить запись, только если в памяти достаточно места, а процесс 2 переписывать, только если имеется не меньше, чем k2 чисел. После каждой записи (чтения) процессы засыпают на t секунд.

После каждой записи процесс 1 увеличивает значение записываемых чисел на 1. Используя семафоры, обеспечить синхронизацию работы процессов в соответствии с заданными условиями. Параметры N, k1, k2, t задаются в виде аргументов командной строки.

Листинг 1. Код задания

#include <sys/mman.h>

#include <semaphore.h>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <fstream>

#include <sys/stat.h>

using namespace std;

int input (int par) //Функция проверки ввода номера слова

{

while (!(cin>>par)||(cin.peek()!='\n')||par<1)

{

cin.clear();

while(cin.get()!='\n');

cout<<"Неверный тип данных. Введите заново\n";

}

return par;

}

int main()

{

cout << "Введите N" <<endl;

int N = input(N);

cout << "Введите k1" <<endl;

int k1 = input(k1);

cout << "Введите k2" <<endl;

int k2 = input(k2);

cout << "Введите t" <<endl;

int t = input(t);

ofstream fout;

fout.open("file.txt");

sem\_t\* sem1 = (sem\_t\*)mmap(0, sizeof(sem\_t), PROT\_READ|PROT\_WRITE, MAP\_ANONYMOUS|MAP\_SHARED, 0, 0 );

sem\_t\* sem2 = (sem\_t\*)mmap(0, sizeof(sem\_t), PROT\_READ|PROT\_WRITE, MAP\_ANONYMOUS|MAP\_SHARED, 0, 0 );

int\* shmem = (int\*)mmap(NULL, N\*sizeof(int), PROT\_READ | PROT\_WRITE, MAP\_ANONYMOUS | MAP\_SHARED, -1, 0);

sem\_init(sem1, 1, 0);

sem\_init(sem2, 1, 1);

int z = N;

pid\_t PID\_\_ = fork();

if (PID\_\_ != 0) {

for (int i = 0; i < z; i++) {

sem\_wait(sem2);

if (k1\*sizeof(int)>N\*sizeof(int)) exit(1);

for(int a= 0; a<k1; a++){

shmem[a] = a;

}

k1++;

sleep(t);

sem\_post(sem1);

}

wait(NULL);

}

if (PID\_\_ == 0) {

for (int i = 0; i < z; i++) {

sem\_wait(sem1);

if(k2<k1) {

for (int b = 0; b < k2; b++) {

fout << shmem[b] << " ";

}

fout << endl;

sleep(t);

}

else cout<<"Can`t read"<<endl;

cout<<k1<<endl;

sem\_post(sem2);

}

}

fout.close();

sem\_close(sem1);

sem\_destroy(sem1);

sem\_close(sem2);

sem\_destroy(sem2);

sleep(1);

}

Вывод: в ходе данной лабораторной работы были получены навыки работы со средствами параллельного программирования, изучены механизмы синхронизации процессов при помощи семафоров, а также обмена данными при помощи сегментов разделяемой памяти.