Лабораторная работа № 5 Асылбек уулу Бакыта из группы ИСП-308

«Потоки»

1. Реализовать задачу «производитель-потребитель» в виде консольной утилиты. На вход подаются в качестве аргументов количество потоков-потребителей и количество потоков-производителей. В функции, где «что-то производится», пускай в консоли «что-то выводится». В функции, где «что-то потребляется» - аналогично.

#include <windows.h>

#include <stdio.h>

#include <time.h>

SECURITY\_ATTRIBUTES SA;//Это структура

HANDLE object;

HANDLE\* manu;//Производители

HANDLE\* cons;//Потребители

int\* A, n = 0;

//Поток потребитель

DWORD consumers(LPVOID F)

{

bool k = true;//Переменная для работы цикла

while (k)//Пока k истина цикл продолжает работать

{

WaitForSingleObject(object, INFINITE);//Ждем объект

printf("\n %d consumer %d", (int)F, A[(int)F]);

ReleaseMutex(object);//Освобождаем объект мьютекса

k = false;

}

return 0;

}

//Поток производитель

DWORD manufacturers(LPVOID F)

{

srand(time(NULL));

bool k = true;//Переменная для работы цикла

while (k)//Пока k истина цикл продолжает работать

{

WaitForSingleObject(object, INFINITE);//Ждем объект

A[(int)F] = (int)F \* rand() % 9 + 1;//Заполняем массив

printf("\n %d manufacturer %d", (int)F, A[(int)F]);

ReleaseMutex(object);//Освобождаем объект мьютекса

k = false;

}

return 0;

}

int main()

{

printf(" Enter the number of threads: ");

scanf\_s("%d", &n);//Вводим количество потоков

//Выделяем динамическую память

cons = (HANDLE\*)malloc(sizeof(HANDLE) \* n);

manu = (HANDLE\*)malloc(sizeof(HANDLE) \* n);

A = (int\*)malloc(sizeof(int) \* n);

if (manu == NULL || cons == NULL || A == NULL)//Проверяем

goto stop;

//Задаем начальные значения структуры

SA.nLength = sizeof(SA);//Размер структуры в байтах

SA.lpSecurityDescriptor = NULL;//Объекту назначен дескриптор безопасности

SA.bInheritHandle = TRUE;//Процесс наследует дескриптор.

object = CreateMutex(&SA, false, NULL);//Создаем объекты мьютекса

//Создаем потоки

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

{

manu[i] = CreateThread(NULL, 0, manufacturers, (void\*)i, 0, NULL);

cons[i] = CreateThread(NULL, 0, consumers, (void\*)i, 0, NULL);

}

WaitForMultipleObjects(n, manu, TRUE, INFINITE);//Ждем несколько объектов

WaitForMultipleObjects(n, cons, TRUE, INFINITE);//Ждем несколько объектов

//Проверяем и освобождаем память. Закрываем открытые дескрипторы

stop:

{

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

{

if (manu[i] != NULL) CloseHandle(manu[i]);

if (cons[i] != NULL) CloseHandle(cons[i]);

}

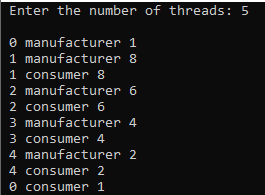
if (A != NULL) free(A);

}

if (object != NULL) CloseHandle(object);

return 0;

}



1. Реализовать задачу «читатели-писатели» в виде консольной утилиты. На вход так же вводится количество читателей и количество писателей. Функции для «чтения», «записи» также реализовать с выводом какой-то информации в консоли.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <stdio.h>

HANDLE\* writers, \* readers, f, object, mapf;

bool readonly;

int n = 0;

LPVOID address;

DWORD writer(LPVOID F)

{

readonly = false;//режим только для чтения отключено

bool k = true;//Переменная для работы цикла

while (k)//Пока k истина цикл продолжает работать

{

WaitForSingleObject(object, INFINITE);//Ждем объект

printf("\n\n Writer %d wrote: %s", (int)F, (char\*)address);

ReleaseMutex(object);//Освобождем мьютекс

k = false;

}

readonly = true;//режим только для чтения включена

return 0;

}

DWORD reader(LPVOID F)

{

bool k = true;//Переменная для работы цикла

while (k)//Пока k истина цикл продолжает работать

{

if (readonly) continue;//Проверка того, что установлен режим только для чтения

printf("\n\n Reader %d read: %s", (int)F, (char\*)address);

k = false;

}

return 0;

}

int main()

{

f = CreateFileA("C://text.txt", GENERIC\_WRITE | GENERIC\_READ, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

if (f == INVALID\_HANDLE\_VALUE)

{

printf("CreateFile failed\n");

}

mapf = CreateFileMappingA(f, NULL, PAGE\_READWRITE, 0, 0, NULL);

if (mapf == NULL)

{

printf("CreateFileMapping failed\n");

CloseHandle(f);

}

address = MapViewOfFile(mapf, FILE\_MAP\_ALL\_ACCESS, 0, 0, GetFileSize(f, NULL));

if (address == NULL)

{

printf("MapViewOfFile failed\n");

CloseHandle(mapf);

CloseHandle(f);

}

printf(" Enter the number of threads:");

scanf\_s("%d", &n);

//Выделяем динамическую память

writers = (HANDLE\*)malloc(sizeof(HANDLE) \* n);

readers = (HANDLE\*)malloc(sizeof(HANDLE) \* n);

//Проверяем выполнилось ли выделение динамической памяти

if (writers == NULL || readers == NULL)

goto stop;

object = CreateMutex(NULL, false, NULL);//Создаем мьютекс

//Создаем потоки

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

{

writers[i] = CreateThread(NULL, 0, writer, (void\*)i, 0, NULL);

readers[i] = CreateThread(NULL, 0, reader, (void\*)i, 0, NULL);

}

//Ждем объекты

WaitForMultipleObjects(n, writers, TRUE, INFINITE);

WaitForMultipleObjects(n, readers, TRUE, INFINITE);

//Закрываем открытые дескрипторы

stop:

{

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

{

if (writers[i] != NULL) CloseHandle(writers[i]);

if (readers[i] != NULL) CloseHandle(readers[i]);

}

}

if (address != NULL) UnmapViewOfFile(address);

if (mapf != NULL) CloseHandle(mapf);

if (f != NULL) CloseHandle(f);

if (object != NULL) CloseHandle(object);

return 0;

}

