Белорусский государственный университет

информатики и радиоэлектроники

Факультет компьютерных технологий

Контрольная работа

по дисциплине «Операционные системы»

Вариант 1

Выполнил ст. гр. 980161: Алейчик И.Д.

Принял(а): Герман Ю.О.

Минск 2020

1. Виды памяти. Регистровая память (описать виды памяти, например, на магнитном диске, на лазерном диске, полупроводниковую память и пр.)

**Регистровая память** – еще называют буферезированной, это тип памяти, который содержит между микросхемами памяти и контроллером некий регистр, куда буферизируется сигналы управления.

Виды памяти подразделяються на **внешние** и **внутренние**.

Внутренняя память:

* ОЗУ (Оперативное запоминающее устройство) или же оперативаня память;
* ПЗУ (Постоянное запоминающее устройство) ;
* Кеш-память.

Внешняя память :

* Магнитная;

1. Жесткие магнитные диски (Жеский диск HDD);
2. Гибкие магнитные диски (Дискеты);
3. Магнитные ленты.

* Оптическая;

1. Компакт диски (CD-ROM);
2. DVD.

* Флешь пямять.

1. USB-Flash
2. SSD

Внутренная память это память установленная непосредсвтенно в устройство например в компьютер, может быть представлена в виде интегрированого чипа(ов) и/или платы с микросхемами памяти которая встраиваеться в специальные разьемы на материнской плате.

Внешняя память это устройства хранения и чтения данных или просто накопители которые могут быть использованы непосредственно в ПК или как внешнее ЗУ.

**ОЗУ** – энергозависимая память компьютера, используется для хранения временных данных программ в виде машинного кода.

**ПЗУ** – энергонезависимая память компьютера, используется для хранения постоянно используемых данных, кроме ПЗУ есть и **ППЗУ**( Прогаммируемое постоянное запоминающее устройство) это точно такая память как и ПЗУ за исключением что она индивидуально программируется и устанавливается на какое либо оборудование или устройство.

**Кеш-память** – высокоскоростная память предназаначенная для хранения и произвольного доступа к наиболее часто используемым данным, наиболее часто можно встретить в процессорах, могут иметь некий фиксированный размер в зависимости от уровня.

**Жесткий диск (HDD)** – энергонезависимый накопитель, предназаченный для хранения и произвольного доступа к данным, работает на принципах магнитной записи.

**Гибкий диск** или же **дискета –** предназначена для длительного хранения программ и данных, представляет собой гибкую платиковую пластину в защитной оболочке, работает на принципах магнитной записи.

**Магниная лента** – так же предназначена для длительного хранения программ и данных, представляет собой гибкую платиковую ленту с дорожками в защитной оболочке, работает на принципах магнитной записи, на сегодня магнтные ленты можно лишь встретить в виде картриджей форматов LTO (Linear Tape-Open) применяется в некоторых дата центрах.

**Компакт диски CD-DVD –** представляю собой пластиковый диск, с поверхностью на кторой распологаются незаметные глазу дорожки, запись и чтение осуществляется путем прожига и сканированием оптическим лазером CD-DVD приводом.

**Флешь память –** потсоянно запоминающее устройства прдензначенное для хранения, переноса, чтения и записи каки либо данных, примерами флешь памяти являются USB флешки и SSD.

**Полупроводниковая память** – это память в которой микросехы памяти построены на полупроводниках, если коротко на микрочипах и контроллерах, такой тип памяти энергозависим по этому содержимое при отключении питния теряется. Отличительной способностью полупроводниковой памяти считается высокое быстродействие и компактность.

По энергозависимости память делят на:

* Энергозависимую;
* Энергонезависимую;
* Статическую ;
* Динамическую.

**Статическая память** – это энэргозависимая полупроводниковая память с произвольным доступом, способная сохранять каждый разряд в триггере, позваляющей поддерживать состоянии сигнала без постоянной перезаписи, но так как это тип пямяти энэргозависим он может сохрнять данные только при налии питания.

**Динамическая память** – тоже в какой-то степени считается энергозависимой памятью, при отсуствии обращения к ячейке за счет утечки токов кондексаторы разрежаются и информация теряется (деградирует), по этому такой тип памяти требует переодической подзорядки.

Некторые виды памяти не требуют наличия питания, такая память называется энергонезавсисимой, при использовании такой пямяти за сохранность данных при отключении питания можно не волноваться, чего нельзя сказать про энергозависимую, ведь при ее использвании все временные и промежуточные данные при отключении питания обнуляются. К энэргонезависимой памяти относят все виды внешней пямяти и ПЗУ, а к энэргозависимой ОЗУ и кеш-память.

1. Создать приложение – например, с двумя потоками-вкладчиками и двумя потоками-потребителями. Каждый поток-вкладчик в случайный момент времени пишет в ячейку (случайное целое число от 1 до 100), представляющую сумму, помещаемую на счет. Поток-потребитель считывает число (снимает случайную сумму с вклада). Синхронизацию сделать на основе механизма семафора или критической секции.

**Листинг кода:**

#define \_CRT\_SECURE\_NO\_WARNINGS // off secure mode

#define \_USE\_MATH\_DEFINES // general math defines for example number PI and etc.

#include <iostream> // I/O

#include <cstdio> // I/O

#include <iomanip> // I/O

#include <conio.h> // I/O

#include <cstdlib> // function general purpose

#include <string> // work is string

#include <vector> // vector array

#include <regex> // regular expressions

#include <fstream> // work is fiel

#include <windows.h> // windows api

#include <tchar.h> // data type tchar

#include <wchar.h> // data type wchar

#include <tlhelp32.h> // work process helper

#include <math.h> // math functions

#include <ctime> // work is time

#include <time.h> // work is

#include <math.h> // work is math function

#include <stack> // work is stack

CRITICAL\_SECTION cs; // объявляем объект критической секции

int bank = 0; // счет банка (общий счет)

int random(int a, int b) //генерация случайного числа для пополнения или снятия счета

{

srand(time(0));

if (a > 0) return a + rand() % (b - a);

else return a + rand() % (a + b);

}

DWORD WINAPI myThreadConsumer1(void\* lpParameter) // Потребитель 1

{

srand(time(0));

int take1 = random(1, 100); // сколько потребитель хочет снять

EnterCriticalSection(&cs); // входим в критическую секцию

printf("-------------------------------------------------------------------Потребитель №1 \n");

printf("Потребитель считывает сумму вклада, сейчас на счете находится = %d \n", bank);

printf("Снимает случайную сумма вклада в размере = %d\n", take1);

if (bank > take1) // если на счете денег больше - снимаем

{

bank = bank - take1;

printf("Снято: %d\n", take1);

printf("Осталось: %d\n\n", bank);

}

else { // если нет – оставляем все как есть

printf("На основном счете нет либо недостаточно денег: %d\n\n", bank);

}

LeaveCriticalSection(&cs); // выходим из критической секции

return 0;

}

DWORD WINAPI myThreadConsumer2(void\* lpParameter) // Потребитель 2

{

srand(time(0));

int take2 = random(1, 100 / 5); // сколько потребитель хочет снять

EnterCriticalSection(&cs); // входим в критическую секцию

printf("-------------------------------------------------------------------Потребитель №2\n");

printf("Потребитель считывает сумму вклада, сейчас на счете находится = %d \n", bank);

printf("Снимает случайную сумму вклада в размере: %d\n", take2);

if (bank > take2) // если на счете денег больше - снимаем

{

bank = bank - take2; // снимаем со счета желаемую сумму

printf("Снято: %d\n", take2);

printf("Осталось: %d\n\n", bank);

}

else {// если нет – оставляем все как есть

printf("На основном счете нет либо недостаточно денег: %d\n\n", bank);

}

LeaveCriticalSection(&cs); // выходим из критической секции

return 0;

}

DWORD WINAPI myThreadDepositor1(void\* lpParameter) // Вкладчик 1

{

int i = 0;

int pay = random(1, 100 / 10); // сколько вкладчик хочет вложить

EnterCriticalSection(&cs); // входим в критическую секцию

printf("-------------------------------------------------------------------Вкладчик №1 \n");

printf("Вкладчик помещает на счет сумму в размере: %d\n", pay);

bank = bank + pay; помещаем на счет желаемую сумму

printf("Транзакция успешно проведена\nНа счету: %d\n\n", bank);

LeaveCriticalSection(&cs); // выходим из критической секции

return 0;

}

DWORD WINAPI myThreadDepositor2(void\* lpParameter) // Вкладчик 2

{

int i = 0;

int pay = random(1, 100); // сколько вкладчик хочет вложить

EnterCriticalSection(&cs); // входим в критическую секцию

printf("-------------------------------------------------------------------Вкладчик №2 \n");

printf("Вкладчик помещает на счет сумму в размере: %d\n", pay);

bank = bank + pay; // помещаем на счет желаемую сумму

printf("Транзакция успешно проведена\nНа счету: %d\n\n",bank);

LeaveCriticalSection(&cs); // выходим из критической секции

return 0;

}

void mainThread()

{

DWORD myThreadID1,myThreadID2, myThreadID3, myThreadID4; // идентификаторы потоков

HANDLE myHandleConsumer1, myHandleConsumer2, myHandleDepositor1, myHandleDepositor2; // дескрипторы потоков

InitializeCriticalSection(&cs); // инизиалищзруем критическую секицю

//инициализирем потоки

myHandleConsumer1 = CreateThread(0, 0, myThreadConsumer1, 0, CREATE\_SUSPENDED, &myThreadID1); // Потребитель 1

myHandleConsumer2 = CreateThread(0, 0, myThreadConsumer2, 0, CREATE\_SUSPENDED, &myThreadID2);// Потребитель 2

myHandleDepositor1 = CreateThread(0, 0, myThreadDepositor1, (void\*)myHandleConsumer1, NULL, &myThreadID3); // Вкладчик 1

myHandleDepositor2 = CreateThread(0, 0, myThreadDepositor2, (void\*)myHandleConsumer2, NULL, &myThreadID4);// Вкладчик 2

//определяем способность системы временно повышать приоритет потока

BOOL b = SetThreadPriorityBoost(myHandleDepositor1, false);

if (b) // если значение 1 - true

{

SetThreadPriority(myHandleDepositor1, THREAD\_PRIORITY\_HIGHEST); // задаем потоку myHandleDepositor1 наивысший приоритет

}

//запускаем потоки

ResumeThread(myHandleConsumer1);

ResumeThread(myHandleConsumer2);

ResumeThread(myHandleDepositor1);

ResumeThread(myHandleDepositor2);

//каждый поток переходит к устанволенному состоянию либо по истечении 10 сек завершается

WaitForSingleObject(myHandleConsumer1, 10000);

WaitForSingleObject(myHandleConsumer2, 10000);

WaitForSingleObject(myHandleDepositor1, 10000);

WaitForSingleObject(myHandleDepositor2, 10000);

//завершаем работу потоков

TerminateThread(myHandleConsumer1, 0);

TerminateThread(myHandleConsumer2, 0);

TerminateThread(myHandleDepositor1, 0);

TerminateThread(myHandleDepositor2, 0);

DeleteCriticalSection(&cs); // понимаем что критическая секция нам уже не нужна, удаляем ее

system("pause");

}

void \_tmain\_list() {

printf("Operating System WorkLabs and KR\n\n1.Example\_1\n2.Example\_2\n3.Example\_3\n4.Example\_4\n5.Example\_5\n6.Example\_6\n7.Task\n8.KR\_OS\n\n");

}

int \_tmain\_menu(int set) {

while (set != 0) {

if (set == 1)

{

mainThread();

return 0;

}

}

}

int \_tmain(int argc, \_TCHAR\* argv[]) {

int input;

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

scanf("%d",&input)

\_tmain\_menu(input);

system("pause");

}

**Примеры работы программы:**



