



МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный технологический институт
(технический университет)»
СПбГТИ(ТУ)

УГНС	09.00.00	Информатика и вычислительная техника
Направление подготовки	09.03.01	Информатика и вычислительная техника
Направленность (профиль)		Системы автоматизированного проектирования
Факультет		Информационных технологий и управления
Кафедра		Систем автоматизированного проектирования и управления
Учебная дисциплина		ОПЕРАЦИОННЫЕ СИСТЕМЫ
Курс 2		Группа 494

Отчет по лабораторной работе №4

Тема: Управление процессами

Студент _____

А.А. Гусев

Преподаватель _____

Р.В. Макарук

Отметка о зачете _____

(подпись преподавателя)

Санкт-Петербург
2021

Цель работы

Знакомство с алгоритмами управления процессами, средствами, обеспечивающими возможность организации многопроцесности и многопоточности, а также средствами синхронизации процессов. Получение практических навыков использования средств ОС, как инструментальных средств при проведении различных работ.

Ход работы

В качестве тестовой системы для выполнения практического задания использовалась: операционная система — MS Windows XP. Также для сравнения была взята ОС — MS Windows 10. В качестве исследуемого приложения был выбран редактор Блокнот. В процессе выполнения работы в нем открывался файл racket64.exe - установочный файл компилятора языка LISP. В диспетчере задач MS Windows виден созданный процесс.

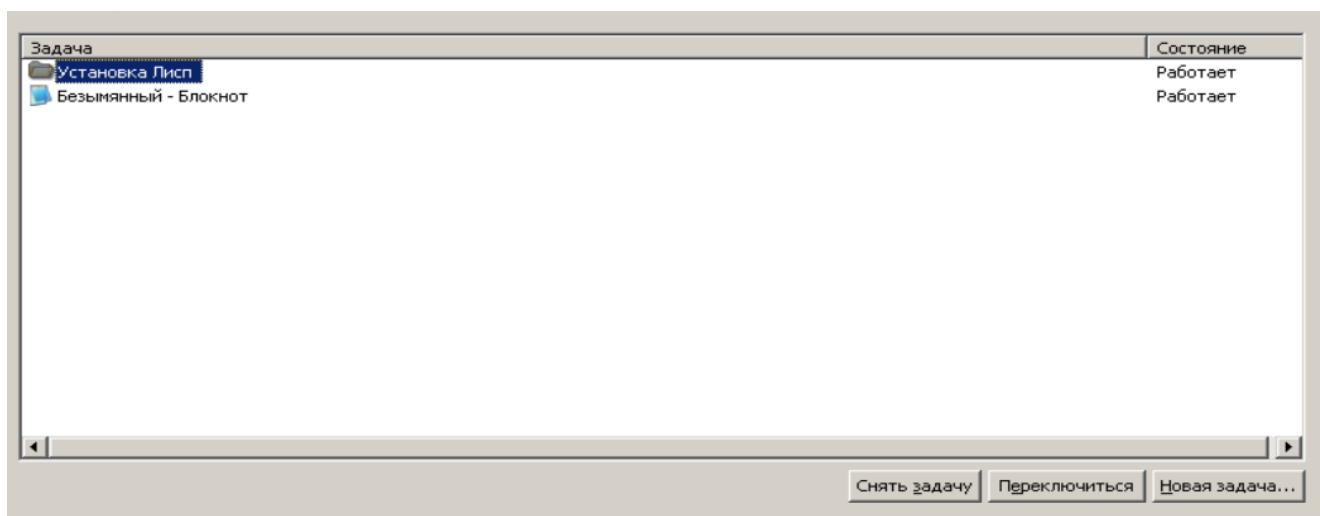


Рисунок 1 - Диспетчер задач

Диспетчер задач позволяет получить обобщенную информацию об использовании основных ресурсов компьютера: общее количество процессов и потоков, участвующих в системе и т.д.

В диспетчере задач отображаются сведения о программах и процессах, выполняемых на компьютере.

Имя образа	PID	Имя пользователя	ЦП	Время ЦП	Память	Вирт.п.	Баз.пр.	Потоков
explorer.exe	1460	konarparti	00	0:00:54	24 448 КБ	29 000 КБ	Средний	12
svchost.exe	996	SYSTEM	00	0:00:00	18 272 КБ	11 600 КБ	Средний	67
wuauclt.exe	1996	konarparti	00	0:00:00	4 936 КБ	5 496 КБ	Средний	3
svchost.exe	824	SYSTEM	00	0:00:00	4 444 КБ	2 904 КБ	Средний	18
spoolsv.exe	1552	SYSTEM	00	0:00:00	4 244 КБ	2 904 КБ	Средний	11
winlogon.exe	612	SYSTEM	00	0:00:18	4 152 КБ	5 804 КБ	Высокий	18
svchost.exe	1200	LOCAL SERVICE	00	0:00:00	4 140 КБ	1 560 КБ	Средний	14
svchost.exe	904	NETWORK SERVICE	00	0:00:00	3 900 КБ	1 524 КБ	Средний	7
alg.exe	484	LOCAL SERVICE	00	0:00:00	3 340 КБ	1 048 КБ	Средний	6
svchost.exe	1044	NETWORK SERVICE	00	0:00:00	3 276 КБ	1 108 КБ	Средний	4
csrss.exe	588	SYSTEM	00	0:00:01	3 168 КБ	1 368 КБ	Высокий	9
services.exe	656	SYSTEM	00	0:00:00	2 868 КБ	1 316 КБ	Средний	14
ctfmon.exe	572	konarparti	00	0:00:00	2 744 КБ	768 КБ	Средний	1
notepad.exe	544	konarparti	00	0:00:00	2 156 КБ	592 КБ	Средний	1
wsntfy.exe	1360	konarparti	00	0:00:00	1 772 КБ	448 КБ	Средний	1
taskmgr.exe	1632	konarparti	02	0:00:11	1 708 КБ	1 052 КБ	Высокий	4
lsass.exe	668	SYSTEM	00	0:00:00	992 КБ	3 456 КБ	Средний	18
smss.exe	364	SYSTEM	00	0:00:00	372 КБ	164 КБ	Средний	3
System	4	SYSTEM	00	0:00:14	212 КБ	28 КБ	Средний	53
Бездействие сис...	0	SYSTEM	98	0:29:58	16 КБ	0 КБ	Н/Д	1

Рисунок 4 - Характеристика запущенного процесса

Системный монитор служит для сбора и просмотра в реальном времени данных памяти, диска, процессора, сети и других параметров в виде графика, гистограммы или отчета.

Анализ данных наблюдения позволяет обнаружить такие явления, как избыточный спрос на определенные ресурсы, приводящий к возникновению узкого места в работе системы. Здесь также есть возможность выбрать параметры, характеризующие запущенный процесс. Перед выполнением исследуемой задачи устанавливаем сначала минимальный приоритет процесса.

Имя образа	PID	Имя пользователя	ЦП	Время ЦП	Память	Вирт.п.	Баз.пр.	Потоков
explorer.exe	1460	konarparti	02	0:00:54	24 372 КБ	28 968 КБ	Средний	12
svchost.exe	996						Средний	67
wuauclt.exe	1996						Средний	3
svchost.exe	824						Средний	18
spoolsv.exe	1552						Средний	11
winlogon.exe	612						Высокий	18
svchost.exe	1200						Средний	14
svchost.exe	904						Средний	7
alg.exe	484						Средний	6
svchost.exe	1044						Средний	4
csrss.exe	588						Высокий	9
services.exe	656	SYSTEM	00	0:00:00	2 868 КБ	1 316 КБ	Средний	14
ctfmon.exe	572	konarparti	00	0:00:00	2 744 КБ	768 КБ	Средний	1
taskmgr.exe	1632	konarparti	02	0:00:12	2 156 КБ	1 052 КБ	Высокий	4
notepad.exe	544	konarparti	00	0:00:00	2 156 КБ	592 КБ	Средний	1
wsntfy.exe	1360	konarparti	00	0:00:00	1 772 КБ	448 КБ	Средний	1
lsass.exe	668	SYSTEM	00	0:00:00	880 КБ	3 456 КБ	Средний	18
smss.exe	364	SYSTEM	00	0:00:00	372 КБ	164 КБ	Средний	3
System	4	SYSTEM	00	0:00:14	212 КБ	28 КБ	Средний	53
Бездействие сис...	0	SYSTEM	96	0:31:14	16 КБ	0 КБ	Н/Д	1

Рисунок 5 — Понижение базового приоритета

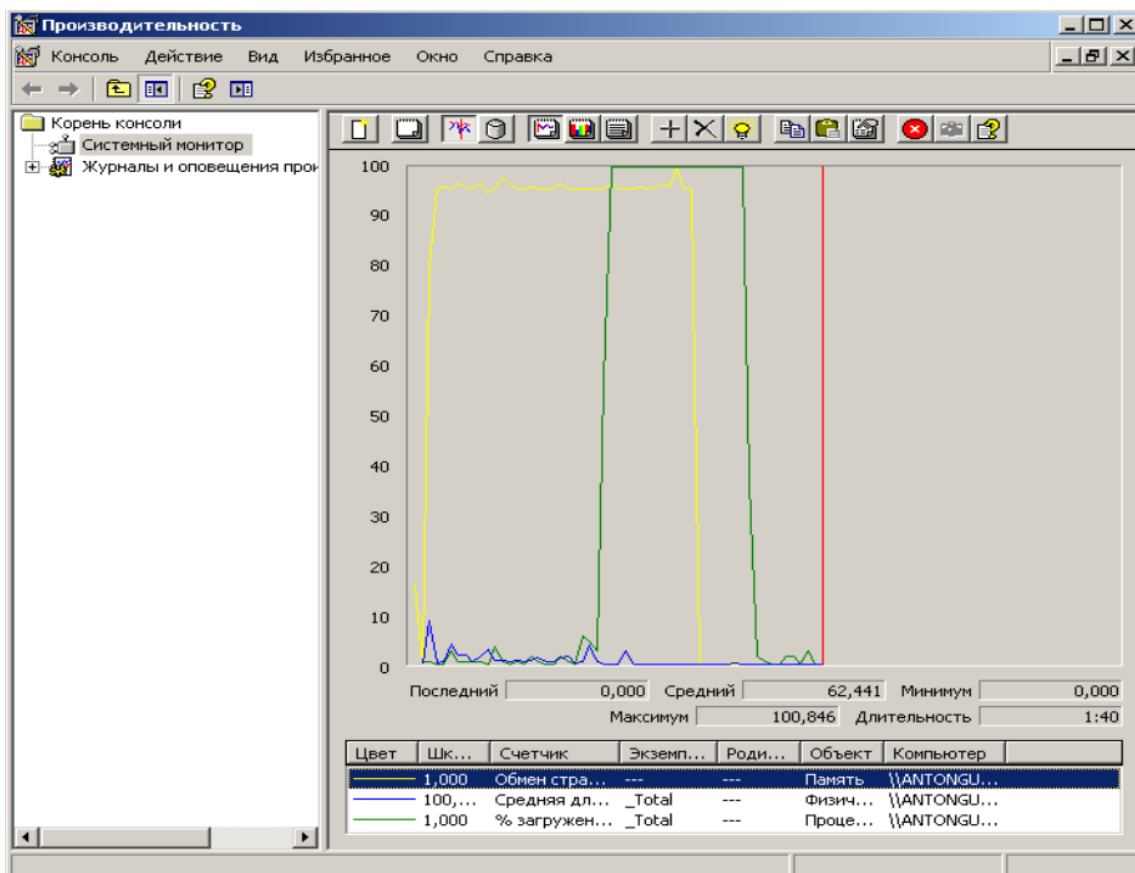


Рисунок 6 — Выполнение операции при пониженном приоритете
Теперь изменим приоритет на «Реального времени»

csrss.exe	588	SYSTEM	00	0:00:01	3 200 КБ	1 368 КБ	Высокий	9
services.exe	656	SYSTEM	00	0:00:00	2 908 КБ	1 340 КБ	Средний	15
ctfmon.exe	572	konarparti	00	0:00:00	2 748 КБ	768 КБ	Средний	1
notepad.exe	1444	konarparti	00	0:00:00	2 164 КБ	592 КБ	Реального времени	1
taskmgr.exe	1632	konarparti	99	0:00:13	1 964 КБ	1 052 КБ	Высокий	4
wscntfv.exe	1360	konarparti	00	0:00:00	1 772 КБ	448 КБ	Средний	1

Рисунок 7 - Повышение приоритета задачи

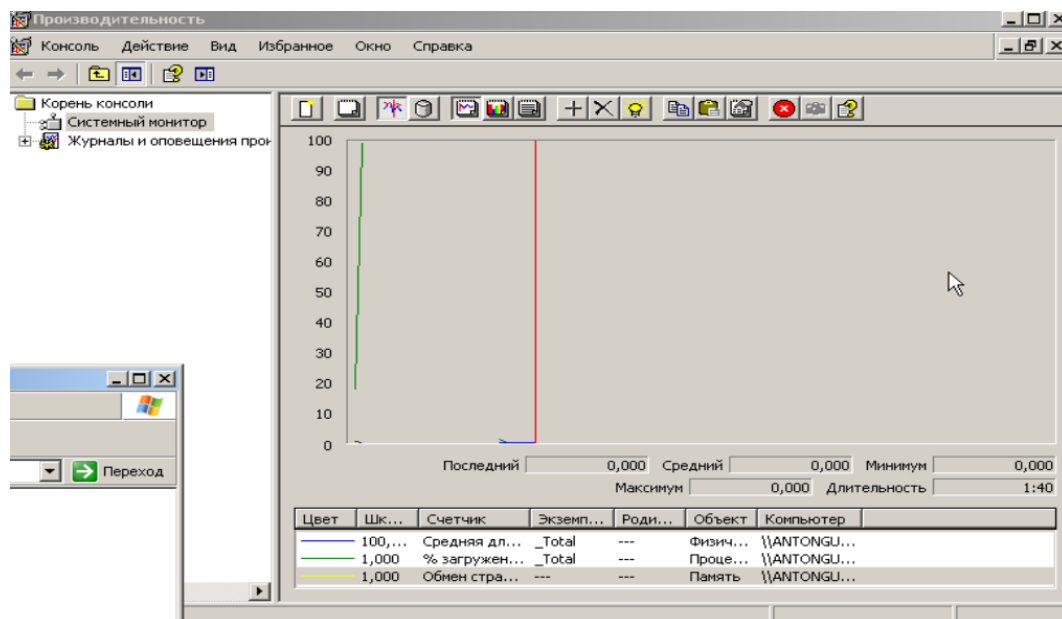


Рисунок 8 — Процесс выполняется с приоритетом реального времени

Из представленных рисунков очевидно, что при изменении приоритета процесса резко изменяется характер работы процесса в пользовательском режиме в много процессной системе - задача выполняется быстрее.

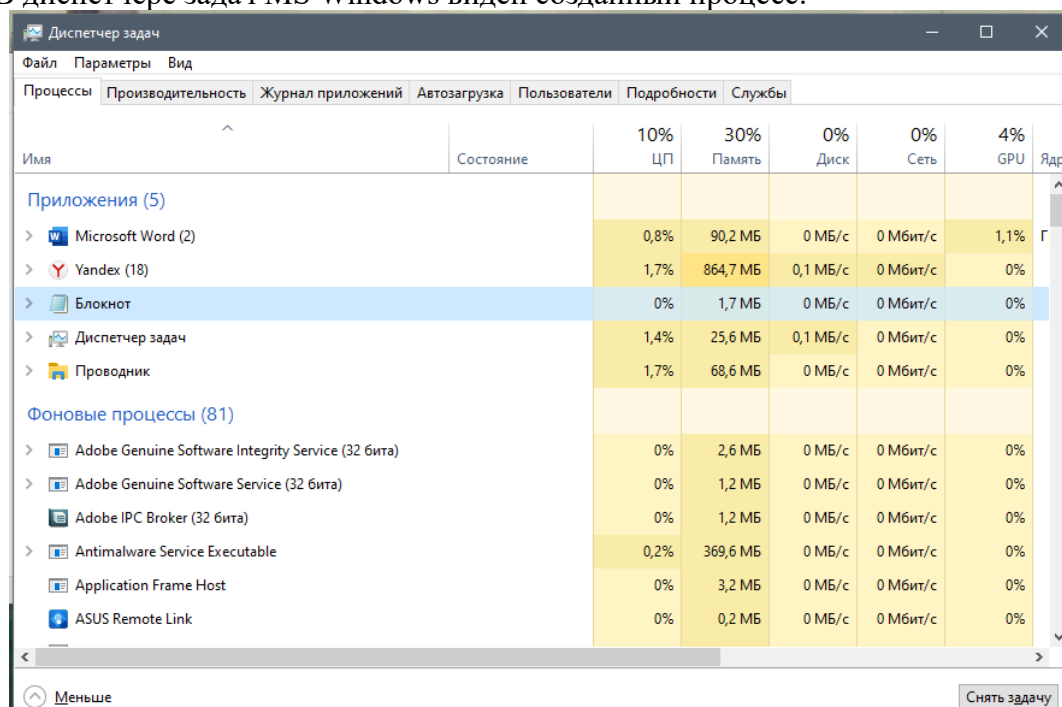
Диспетчер задач может показать количество потоков, созданных в конкретном процессе. В рассматриваемом процессе был создан 1 поток.

Однако управлять потоками штатными средствами системы нельзя. Сегодня разработчики программного обеспечения предлагают разнообразные программы для получения более детальной информации о компонентах вычислительного процесса.

Теперь в качестве тестовой системы для выполнения практического задания возьмем ОС MS Windows 10.

В качестве исследуемого приложения был выбран редактор Блокнот. Выполнимая операция – открытие того же .exe файла в блокноте.

В диспетчере задач MS Windows виден созданный процесс.



Имя	Состояние	10% ЦП	30% Память	0% Диск	0% Сеть	4% GPU	Ядро
Приложения (5)							
> Microsoft Word (2)		0,8%	90,2 МБ	0 МБ/с	0 Мбит/с	1,1%	Г
> Yandex (18)		1,7%	864,7 МБ	0,1 МБ/с	0 Мбит/с	0%	
> Блокнот		0%	1,7 МБ	0 МБ/с	0 Мбит/с	0%	
> Диспетчер задач		1,4%	25,6 МБ	0,1 МБ/с	0 Мбит/с	0%	
> Проводник		1,7%	68,6 МБ	0 МБ/с	0 Мбит/с	0%	
Фоновые процессы (81)							
> Adobe Genuine Software Integrity Service (32 бита)		0%	2,6 МБ	0 МБ/с	0 Мбит/с	0%	
> Adobe Genuine Software Service (32 бита)		0%	1,2 МБ	0 МБ/с	0 Мбит/с	0%	
> Adobe IPC Broker (32 бита)		0%	1,2 МБ	0 МБ/с	0 Мбит/с	0%	
> Antimalware Service Executable		0,2%	369,6 МБ	0 МБ/с	0 Мбит/с	0%	
> Application Frame Host		0%	3,2 МБ	0 МБ/с	0 Мбит/с	0%	
> ASUS Remote Link		0%	0,2 МБ	0 МБ/с	0 Мбит/с	0%	

Рисунок 9 - Диспетчер задач Windows 10

Диспетчер задач позволяет получить обобщенную информацию об использовании основных ресурсов компьютера: общее количество процессов и потоков, участвующих в системе и др.

В диспетчере задач отображаются сведения о программах и процессах, выполняемых на компьютере. Кроме того, там можно просмотреть наиболее часто используемые показатели быстродействия процессов.

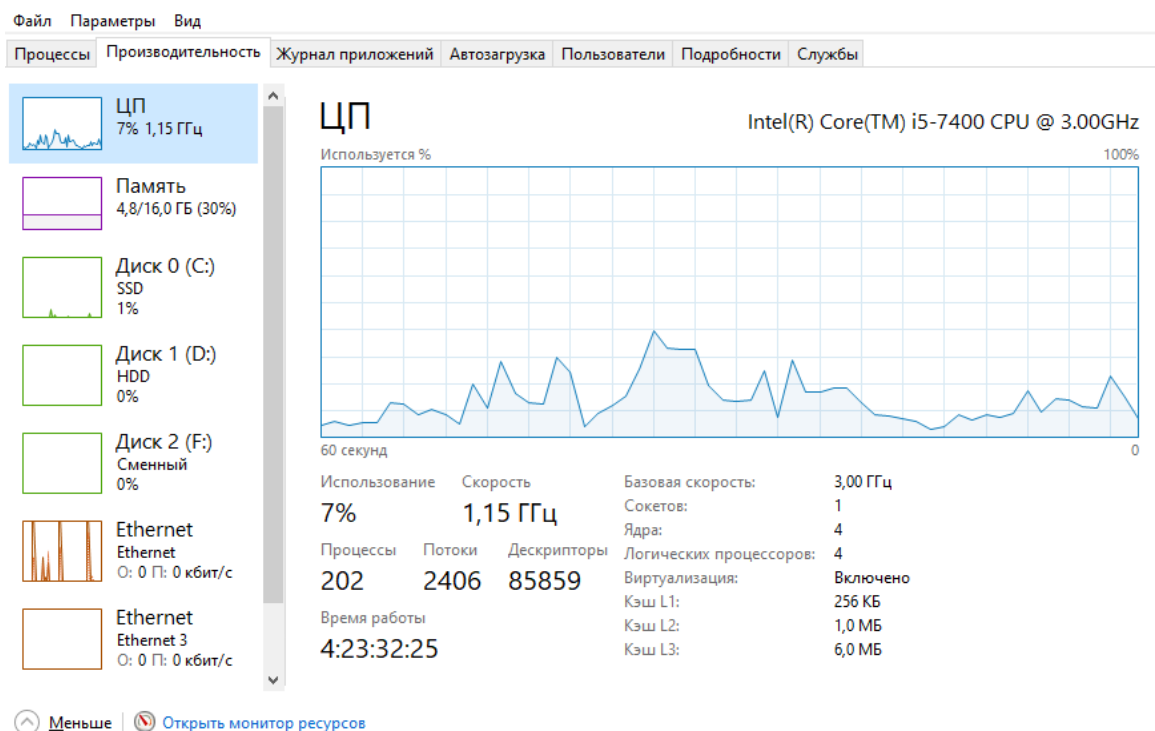


Рисунок 10 - Использование основных ресурсов компьютера

Просмотр (мониторинг) процессов осуществляется переходом на вкладку процессы. Выбрать просматриваемые характеристики можно с помощью нажатия ПКМ по столбцу.

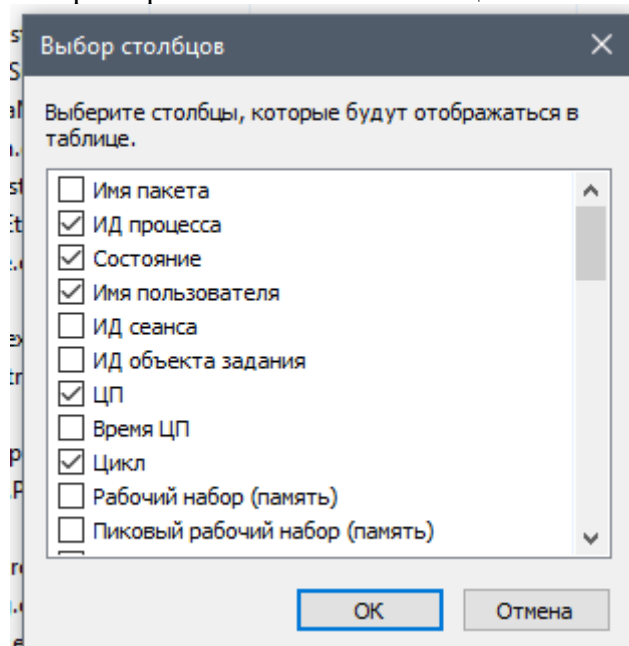


Рисунок 11 - Выбор отображаемых характеристик

Файл Параметры Вид									
Процессы Производительность Журнал приложений Автозагрузка Пользователи Подробности Службы									
Имя	ИД п...	Состояние	Имя польз...	ЦП	Цикл	Память (а...	Базовый п...	П...	Виртуализаци...
fontdrvhost.exe	680	Выполняется	UMFD-0	00	0	284 K	Обычный	5	Отключено
HuaweiHiSuiteServic...	3760	Выполняется	СИСТЕМА	00	0	816 K	Обычный	5	Не разрешено
IAStorDataMgrSvc.exe	3348	Выполняется	СИСТЕМА	00	0	10 656 K	Обычный	8	Не разрешено
IAStorIcon.exe	10796	Выполняется	anton	00	0	7 656 K	Обычный	7	Отключено
InstallAssistService.exe	4188	Выполняется	СИСТЕМА	00	0	288 K	Обычный	2	Не разрешено
JetBrains.Etw.Collect...	3816	Выполняется	СИСТЕМА	00	0	252 K	Обычный	2	Не разрешено
jhi_service.exe	5060	Выполняется	СИСТЕМА	00	0	340 K	Обычный	2	Не разрешено
LMS.exe	7180	Выполняется	СИСТЕМА	00	0	1 164 K	Обычный	4	Не разрешено
LockApp.exe	11432	Приостановлено	anton	00	0	0 K	Обычный	16	Отключено
LogiRegistryService...	3936	Выполняется	СИСТЕМА	00	0	560 K	Обычный	3	Не разрешено
lsass.exe	824	Выполняется	СИСТЕМА	00	0	7 568 K	Обычный	12	Не разрешено
mDNSResponder.exe	3668	Выполняется	СИСТЕМА	00	0	992 K	Обычный	2	Не разрешено
Microsoft.Photos.exe	16748	Приостановлено	anton	00	0	0 K	Обычный	24	Отключено
mmc.exe	14332	Выполняется	anton	00	0	11 696 K	Обычный	30	Не разрешено
MoUsCoreWorker.e...	7480	Выполняется	СИСТЕМА	00	0	58 168 K	Обычный	8	Не разрешено
MsMpEng.exe	13572	Выполняется	СИСТЕМА	00	0	430 584 K	Обычный	30	Не разрешено
mysqld.exe	4384	Выполняется	NETWORK...	00	0	472 K	Обычный	3	Не разрешено
mysqld.exe	5128	Выполняется	NETWORK...	00	0	1 652 K	Обычный	49	Не разрешено
NisSrv.exe	10532	Выполняется	LOCAL SE...	00	0	2 308 K	Обычный	8	Не разрешено
node.exe	10100	Выполняется	anton	00	0	25 632 K	Обычный	20	Отключено
notepad.exe	12988	Не отвечает	anton	25	25	167 076 K	Обычный	1	Отключено
nvcontainer.exe	4100	Выполняется	СИСТЕМА	00	0	6 604 K	Обычный	39	Не разрешено
nvcontainer.exe	1120	Выполняется	anton	00	0	3 596 K	Обычный	17	Отключено

Меньше Снять задачу

Рисунок 12 - Выполнение процесса

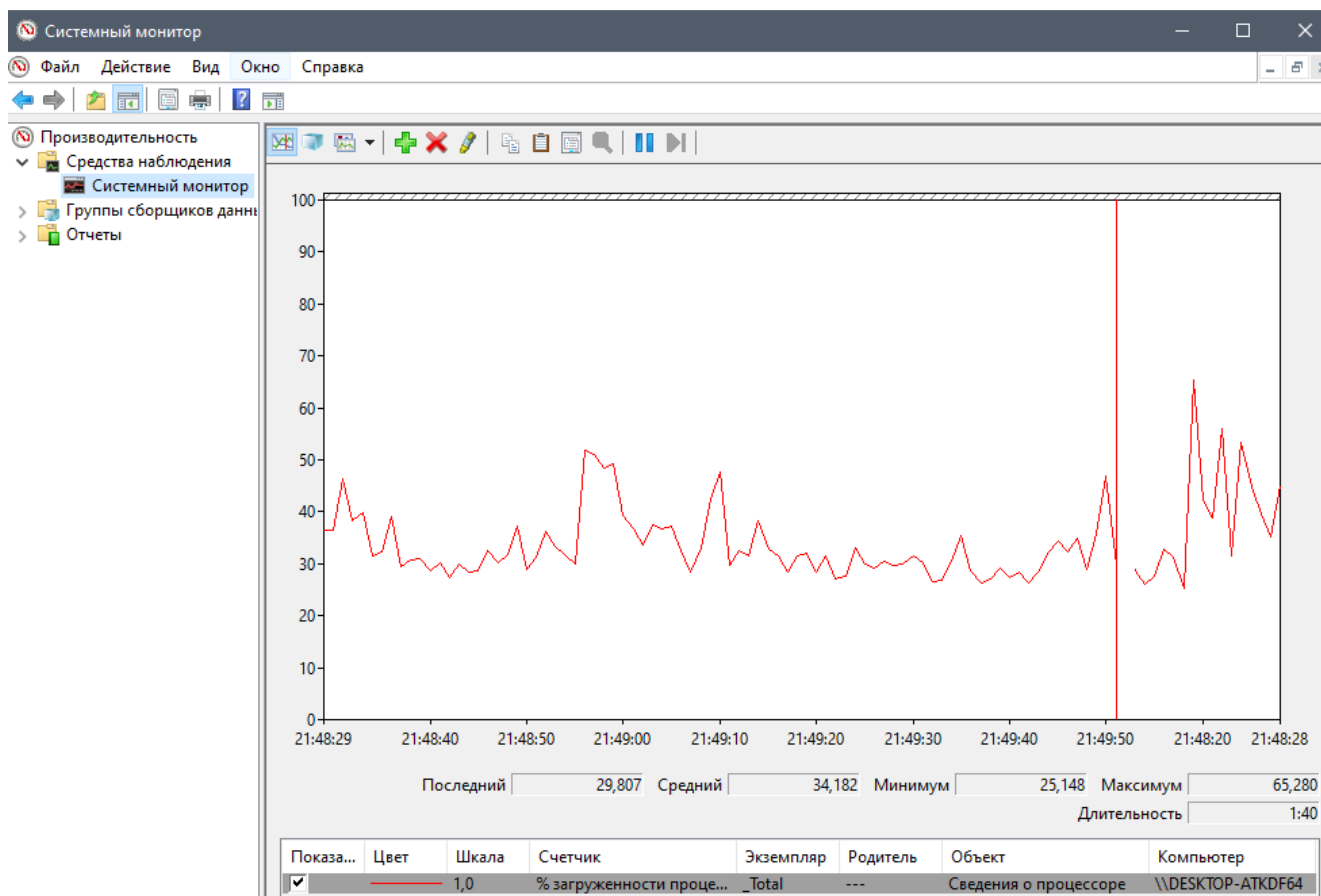


Рисунок 13 - Системный монитор в процессе выполнения задачи

3. Средства синхронизации процессов в MS Windows

Процессам часто нужно взаимодействовать друг с другом с целью совместного использования аппаратных и информационных ресурсов вычислительной системы, например, один процесс может передавать данные другому процессу, или несколько процессов могут обрабатывать данные из общего файла.

Процессы в общем случае протекают независимо, асинхронно друг другу. Одной из функций ОС является обеспечение санкционированного взаимодействия процессов.

Взаимодействие процессов в вычислительной системе напоминает жизнь в коммунальной квартире. Постоянное ожидание в очереди к местам общего пользования (процессору) и ежедневная борьба за ресурсы (кто опять занял все конфорки на плите?)

Для нормального функционирования процессов ОС старается максимально обособить их друг от друга. Каждый процесс имеет собственное адресное пространство (каждая семья должна жить в отдельной комнате), нарушение которого, как правило, приводит к аварийной остановке процесса (вызов милиции). Каждому процессу по возможности предоставляются свои дополнительные ресурсы (каждая семья предпочитает иметь собственный холодильник). Тем не менее для решения некоторых задач процессы могут объединять свои усилия.

Существуют в основном две проблемы взаимодействия процессов, это:

-«конкуренция» процессов в борьбе за ресурсы;

-«сотрудничество» с использованием разделяемых переменных, совместно используемых данных или баз данных.

Для исключения проблем при обмене данными между процессами, разделении данных, при доступе к процессору и устройствам ввода-вывода необходима синхронизация.

Синхронизация (Synchronization) - установка соответствия момента запуска процессов с определенными событиями в системе, т.е. обеспечение временной упорядоченности действий параллельных или асинхронных процессов или их выравнивание.

Приемы синхронизации:

На нижнем уровне:

- блокировка памяти (Storage Interlock);
- семафоры (Semaphore);

На верхнем уровне:

- почтовый ящик;
- буфера сообщений;
- программируемые каналы;
- монитор;

На нижнем уровне применяются следующие приемы синхронизации:

Блокировка памяти (блокировка техническими средствами) (Lockout Memory) - заключается в том, что если два процесса пытаются обратиться в одну ячейку памяти, то одному из них разрешается выполнить операцию, а другому приходится ждать. Машинная операция проверки установки решает проблему критического участка средствами блокировки памяти.

Семафор (Semaphore) (ввел Дейкстра) - не отрицательная целая величина, для которой введены две операции:

- V(S)-увеличение S на единицу (операция показывает, что ресурс возвращен);
- P(S)-уменьшение S на единицу (операция показывает, что ресурс выделяется).

Эти операции реализуются в ядре ОС.

Если $S=0$, то уменьшение невозможно и процесс, вызывающий P, вынужден ждать. Операции P и V неделимы, т.е. в каждый момент времени только один процесс может выполнить операции P и V над данным семафором.

С помощью семафора организуется взаимное исключение процессов с заключением критических участков в скобки, роль которых выполняют P и V.

С каждым семафором связан список процессов, отдающих разрешения пройти семафор.

Если два процесса используют семафор $S=1$, то ни один из них не находится в критическом участке, а при $S=0$ находится один.

Семафоры — это всегда средства задержки процессов, которые тормозятся или засыпают на семафорах и пробуждаются и приходят в движение при изменении состояния семафоров.

Семафор — это также средство регистрации некоего события в системе. О наступлении такого события сообщают процессы, используя системные вызовы, что приводит к переустановке текущих значений семафора (в системе создается дескриптор семафора - информационно управляющая структура).

Реализация взаимoisключения семафорами имеют ряд слабостей: эти операции слишком элементарны, для того чтобы учесть серьезные проблемы параллельных вычислений.

Поэтому используются механизмы более высокого уровня: например, различные буфера сообщений или монитор.

На верхнем уровне применяются следующие приемы синхронизации:

Почтовый ящик - если процесс P_1 хочет общаться с процессом P_2 , он запрашивает у системы почтовый ящик. Почтовый ящик — это структура, состоящая из заголовка и гнезд для передачи сообщений. P_1 посылает сообщение, пока не заполнены гнезда. P_2 - получает сообщений, пока гнезда не пусты. Может быть и двухсторонняя связь. Гнездо не будет занято, пока на запрос не придет ответ. Образуется многовходовые почтовые ящики, если обращаются многие процессы.

Программные каналы (Pipes) (или буфера сообщений) — средства обмена потоками информации между процессами, и средства синхронизации процессов, т.к. параллельные процессы могут корректно взаимодействовать между собой. Попытка записи в канал, который никто не читает, приведет к приостановке процесса производителя, и наоборот, попытка чтения из канала, в который никто не пишет, приведет к приостановке процесса потребителя до появления процесса производителя.

Кольцевой буфер - структура данных, широко применяемая в ОС для буферизации обменов между процессом-производителем и процессом-потребителем.

Монитор (предложил Дейкстра) — это конструкция параллелизма, которая содержит как данные, так и процедуры, необходимые для управления распределением общего ресурса или группы общих ресурсов (не путать с монитором работ). Т.о. монитор — это набор процедур и информационных структур, которыми процессы пользуются в режиме разделения, причем, в каждый момент времени только один процесс (пример монитора - планировщик ресурсов).

Чтобы обеспечить выделение нужного ресурса, процесс должен обратиться к конкретной процедуре монитора. Необходимость входа в монитор в различные моменты времени может возникать у разных процессов. Однако вход в монитор жестко контролируется, и именно здесь осуществляется взаимoisключение процессов, таким образом, что в каждый момент времени один процесс может войти в монитор, остальным приходится ждать, и режимом ожидания также управляет монитор. Информация спрятана внутри монитора, процессы, обращаясь к монитору, не знают какие данные находятся внутри монитора и не получают к ним доступа.

Если процесс, находящийся внутри монитора, не может продолжить свое выполнение, пока не выполнятся какие-либо условия, он ждет вне монитора, пока ресурс освободится, при этом другой процесс обращается к монитору (команда ждать (WAIT) и команда сигнал (SIGNAL) с именем переменной, т.е. условия). До его освобождения это выполняется с помощью операции ждать и сигнал. При блокировке монитор указывает условие, при котором процесс продолжается.

При этом процесс, ожидающий некоторого ресурса, имеет более высокий приоритет по сравнению с новым процессом, т.о. есть гарантия, что он его получит. Монитор может задержать процесс, если последний будет запрашивать занятый ресурс.

В мониторе может находиться одна или несколько процедур, допускающих параллельную работу.

Мониторы имеют преимущество перед семафорами: они могут реализовывать как семафорные, так и почтовые операции. Кроме этого, дают возможность процессам совместно использовать программу, представляющую собой критический участок.

4. Пример дисплейного фрагмента, характеризующего возможности разработанной программы

Ввод данных осуществляется программно. После запуска программы, пользователь видит на экране работу алгоритма банкира. В результате работы программа определяет надежность состояния

```
Ресурсы 4 R1, 4 R2, 4 R3 и 4 R4.
Предоставлено Максимум
R1 R2 R3 R4   R1 R2 R3 R4
A  2  0  0  0   2  0  2  2
B  2  2  0  0   2  2  2  2
C  0  2  2  0   2  4  2  4
D  0  0  2  2   0  0  2  4
Оставшиеся свободные ресурсы:
  0  0  0  2
D процесс выполнен
Свободные ресурсы после завершения 0 0 2 4
A процесс выполнен
Свободные ресурсы после завершения 2 0 2 4
B процесс выполнен
Свободные ресурсы после завершения 4 2 2 4
C процесс выполнен
Свободные ресурсы после завершения 4 4 4 4
Состояние является надёжным
```

Рисунок 14 — Пример работы приложения

5. Текст программы, реализующей алгоритм банкира

Начало программы ---]

[Начало Main.java ---]

```
package konarparti;

import java.util.ArrayList;

public class Main {

    public static void showArrays(ArrayList<Integer> list){
        for(int i = 0; i< list.size(); i++){
            {
                System.out.print(list.get(i) + "  ");
            }
            System.out.print("  ");
        }
    }

    public static ArrayList<Integer> SearchFreeRes(Resources res){
        System.out.println("Оставшиеся свободные ресурсы: ");
        System.out.print("  ");
        for (int i = 0; i < 4; i++){
            res.availResources.add(res.amountofResources.get(i)
(res.AprocessProvided.get(i) + res.BprocessProvided.get(i) + res.CprocessProvided.get(i) +
res.DprocessProvided.get(i)));
            System.out.print(res.availResources.get(i) + "  ");
        }
        System.out.println();
        return res.availResources;
    }
}
```

```

    public static boolean CheckLaunchProc(Resources res, ArrayList<Integer> processProvided,
ArrayList<Integer> processRequire){
        ArrayList<Integer> temp = new ArrayList<Integer>();

        for(int i = 0; i < processRequire.size(); i++){
            temp.add(processProvided.get(i) + res.availResources.get(i));
        }
        for(int i = 0; i < temp.size(); i++){
            if(temp.get(i) < processRequire.get(i)){
                return false;
            }
        }
        res.availResources.clear();
        for (var i: temp) {
            res.availResources.add(i);
        }
        return true;
    }

    public static void main(String[] args) {
        Resources res = new Resources();

        System.out.println("Песчурсы 4 R1, " +
            "4 R2, " +
            "4 R3 и 4 R4.");
        System.out.println("    Предоставлено Максимум");
        System.out.println("    R1 R2 R3 R4    R1 R2 R3 R4");

        System.out.print("A  ");
        showArrays(res.AprocessProvided);
        showArrays(res.AprocessRequire);
        System.out.println();

        System.out.print("B  ");
        showArrays(res.BprocessProvided);
        showArrays(res.BprocessRequire);
        System.out.println();

        System.out.print("C  ");
        showArrays(res.CprocessProvided);
        showArrays(res.CprocessRrequire);
        System.out.println();

        System.out.print("D  ");
        showArrays(res.DprocessProvided);
        showArrays(res.DprocessRrequire);
        System.out.println();

        res.availResources = SearchFreeRes(res);

        ArrayList<ArrayList<Integer>> processProvided = new ArrayList<ArrayList<Integer>>();
        processProvided.add(res.AprocessProvided);
        processProvided.add(res.BprocessProvided);          processProvided.add(res.CprocessProvided);
        processProvided.add(res.DprocessProvided);

        ArrayList<ArrayList<Integer>> processRequire = new ArrayList<ArrayList<Integer>>();
        processRequire.add(res.AprocessRequire);    processRequire.add(res.BprocessRequire);
        processRequire.add(res.CprocessRrequire); processRequire.add(res.DprocessRrequire);

        for(int i = 0; i < processProvided.size(); i++){
            if(CheckLaunchProc(res, processProvided.get(i), processRequire.get(i))){
                System.out.println(res.nameOfProcess.get(i) + " процесс выполнен");
                System.out.print("Свободные ресурсы после завершения ");
                for(int j = 0; j < res.availResources.size(); j++)
                {
                    System.out.print(res.availResources.get(j) + " ");
                }
                System.out.println();
                processRequire.remove(i);
                processProvided.remove(i);
                res.nameOfProcess.remove(i);
            }
        }
    }

```

```

        i = -1;
    }
}
if(res.nameOfProcess.isEmpty()){
    System.out.println("Состояние является надёжным");
}
else{
    System.out.println("Состояние не является надёжным");
}
}
}
}

```

[Конец Main.java ---]

[Начало Resources.java ---]

```

package konarparti;

import java.util.ArrayList;
import java.util.Arrays;

public class Resources {
    public ArrayList<Integer> amountofResources = new ArrayList<> (Arrays.asList(4, 4, 4, 4)); //предоставлено процессу A
    public ArrayList<String> nameOfProcess = new ArrayList<> (Arrays.asList ("A", "B", "C", "D" ));

    public ArrayList<Integer> AprocessProvided = new ArrayList<> (Arrays.asList (2, 0, 0, 0)); //предоставлено процессу A
    public ArrayList<Integer> BprocessProvided = new ArrayList<> (Arrays.asList (2, 2, 0, 0 )); //предоставлено процессу B
    public ArrayList<Integer> CprocessProvided = new ArrayList<> (Arrays.asList (0, 2, 2, 0 )); //предоставлено процессу C
    public ArrayList<Integer> DprocessProvided = new ArrayList<> (Arrays.asList (0, 0, 2, 2 )); //предоставлено процессу D

    public ArrayList<Integer> AprocessRequire = new ArrayList<> (Arrays.asList (2, 0, 2, 2) ); //предоставлено процессу A
    public ArrayList<Integer> BprocessRequire = new ArrayList<> (Arrays.asList (2, 2, 2, 2 )); //предоставлено процессу B
    public ArrayList<Integer> CprocessRequire = new ArrayList<> (Arrays.asList (2, 4, 2, 4 )); //предоставлено процессу C
    public ArrayList<Integer> DprocessRequire = new ArrayList<> (Arrays.asList (0, 0, 2, 4 )); //предоставлено процессу D

    public ArrayList<Integer> availResources = new ArrayList<>(amountofResources.size());
    // доступные ресурсы
}

```

[Конец Resources.java ---]

Вывод

В ходе работы были изучены базовые алгоритмы управления процессами средств, обеспечивающих возможность организации многопроцесности и многопоточности, а также средств синхронизации процессов. Получены практические навыки использования средств ОС и средств, поставляемых другими разработчиками, как инструментальных средств при проведении различных работ.

Сравнение Windows XP и Windows 10

В обеих операционных системах присутствуют возможности просмотра объёма установленной физической памяти, объём виртуальной памяти, величину файла подкачки и его размещение. В обеих ОС есть оснастка Монитор производительности и различные счётчики для отслеживания производительности компьютера.

Можно сделать вывод, что по возможностям работы с памятью операционные системы не имеют больших различий.

Вывод

В ходе выполнения лабораторной работы был изучен принцип работы памяти компьютера. Рассмотрены основные встроенные инструменты ОС для работы с памятью.

Написана программа реализующая алгоритмы замещения страниц FIFO и LRU.