Оглавление

[1. Структура операционной системы. 3](#_Toc61715517)

[2. Ядро операционной системы, его основные компоненты. 3](#_Toc61715518)

[3. Режим ядра и пользовательский режим выполнения кода. Системные вызовы. 5](#_Toc61715519)

[4. Процесс. Ресурсы процесса. 7](#_Toc61715520)

[5. Выполнение потоков в среде с вытесняющей многозадачностью. 7](#_Toc61715521)

[6. Таблица процессов. Планировщик процессов. 7](#_Toc61715522)

[7. Особенности создания процесса с помощью системного вызова UNIX/LINUX fork(). 8](#_Toc61715523)

[8. Особенности создания процесса с помощью функций стандартного интерфейса POSIX – расширения языка C process.h – exec\*, spawn\*. 8](#_Toc61715524)

[9. Интерфейс системных вызовов Win32 API. 8](#_Toc61715525)

[10. Создание процессов в Windows с помощью функции CreateProcess. 9](#_Toc61715526)

[11. Получение информации о процессах с помощью интерфейса PSAPI. 9](#_Toc61715527)

[12. Механизм виртуальной памяти. Технология страничной организации памяти. 9](#_Toc61715528)

[13. Таблица страниц. Менеджер виртуальной памяти. 11](#_Toc61715529)

[14. Структура адресного пространства в MS Windows. 11](#_Toc61715530)

[15. Получение информации об организации памяти с помощью функций Win32 API. 11](#_Toc61715531)

[16. Библиотеки динамической компоновки. 11](#_Toc61715532)

[17. Структура PE-файла. Получение информации о PE-файлах с помощью интерфейса IMAGEHLP. 12](#_Toc61715533)

[18. Обмен данными между процессами в Windows с помощью библиотек динамической компоновки. 12](#_Toc61715534)

[19. Обмен данными между процессами в Windows с помощью отображения файлов в память. 13](#_Toc61715535)

[20. Неименованные каналы. 13](#_Toc61715536)

[21. Именованные каналы. 13](#_Toc61715537)

[22. Технология сокетов. Сокеты Беркли. 14](#_Toc61715538)

[23. Особенности реализации технологии сокетов winsock2. 14](#_Toc61715539)

[24. Архитектура приложения MS Windows. 15](#_Toc61715540)

[25. Передача данных между процессами с помощью сообщений Windows. 15](#_Toc61715541)

[26. Ловушки Windows. 15](#_Toc61715542)

[27. Потоки и их синхронизация. Критические секции. 16](#_Toc61715543)

[28. Алгоритм Петерсона. 16](#_Toc61715544)

[29. Управление потоками в Windows с помощью функций интерфейса PSAPI. 17](#_Toc61715545)

[30. Объекты ядра MS Windows и управление ими. 17](#_Toc61715546)

[31. Мьютексы и семафоры. 17](#_Toc61715547)

[32. Реализация мьютексов в Windows. 17](#_Toc61715548)

[33. Реализация семафоров в Windows. 18](#_Toc61715549)

[34. Синхронизация потоков с помощью объектов Event. 18](#_Toc61715550)

[35. Синхронизация потоков с помощью механизма критических секций в Windows. 18](#_Toc61715551)

[36. Реализация службы MS Windows. 19](#_Toc61715552)

[37. Реализация службы GNU/Linux 19](#_Toc61715553)

# Структура операционной системы.

Большинство ОС поддерживают двухуровневую систему привилегий: привилегированный режим (режим ядра, пространство ядра) и пользовательский режим (пространство пользователя).

В режиме ядра (kernel mode) выполняются все разрешенные инструкции, в ходе выполнения доступна вся оперативная память и любые регистры. На время выполнения кода ОС микропроцессор переключается в режим ядра.

В пользовательском режиме (user mode) доступ к регистрам и памяти ограничен. Приложению не будет позволено работать с памятью за пределами набора адресов, установленного ОС, или обращаться напрямую к регистрам устройств.

<https://calcsbox.com/post/naznacenie-i-struktura-operacionnyh-sistem.html>

<https://intuit.ru/studies/courses/10471/1078/lecture/16565>

<https://mf.grsu.by/UchProc/livak/b_lecture/lec34_OS.htm>

<https://portal.tpu.ru/f_ic/files/school/materials/8.pdf>

[(К оглавлению)](#_top)

# Ядро операционной системы, его основные компоненты.

Ядро́ (kernel) — центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера.

Как основополагающий элемент ОС, ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам системы, необходимым для их работы. Оно действует как интерфейс (мост) между пользовательским приложением (программным обеспечением) и аппаратным обеспечением. Поэтому связь между программным обеспечением и аппаратным обеспечением осуществляется через ядро.

Ядро выполняет все функции по управлению ресурсами системы - как физическими, так и логическими - и разделяет доступ пользователей (программ пользователей) к этим ресурсам. При помощи системного программного обеспечения пользователь управляет средствами, предоставляемыми ядром.

Интерфейс ядра ОС с прикладными программами осуществляется при помощи программного интерфейса системных вызовов, интерфейс с аппаратным обеспечением - при помощи драйверов.

В ядро типичной ОС входят следующие компоненты:

* УПРАВЛЕНИЕ ПРОЦЕССАМИ

Создание, выполнение и завершение процессов выполняются внутри системы всякий раз, когда система находится во включенном состоянии (режиме ON). Процесс содержит всю информацию о задаче, которую необходимо выполнить. Таким образом, для выполнения любой задачи внутри системы создается процесс. В то же время существует множество процессов, которые находятся в активном состоянии внутри системы. Управление всеми этими процессами очень важно для предупреждения тупиковых ситуаций и для правильного функционирования системы, и оно осуществляется ядром.

* УПРАВЛЕНИЕ ПАМЯТЬЮ

Всякий раз, когда процесс создается и выполняется, он занимает память, и когда он завершается, память должна быть освобождена и может быть использована снова. Но память должна быть обработана кем-то, чтобы освобожденная память могла быть снова назначена новым процессам. Эта задача также выполняется ядром. Ядро отслеживает, какая часть памяти в данный момент выделена и какая часть доступна для выделения другим процессам.

* УПРАВЛЕНИЕ УСТРОЙСТВАМИ

Ядро также управляет всеми различными устройствами, подключенными к системе, такими как устройства ввода и вывода и т. д.

* ОБРАБОТКА ПРЕРЫВАНИЙ

При выполнении процессов возникают условия, при которых сначала необходимо решить задачи с большим приоритетом. В этих случаях ядро должно прерывать выполнение текущего процесса и обрабатывать задачи с большим приоритетом, которые были получены в промежутке.

* ОПЕРАЦИИ ВВОДА/ВЫВОДА

Поскольку ядро управляет всеми подключенными к нему устройствами, оно также отвечает за обработку всех видов входных и выходных данных, которыми обмениваются эти устройства. Таким образом, вся информация, которую система получает от пользователя, и все выходные данные, которые пользователь получает через различные приложения, обрабатываются ядром.

<https://portal.tpu.ru/f_ic/files/school/materials/8.pdf>

<https://wiki.merionet.ru/servernye-resheniya/42/chto-takoe-yadro-operacionnoj-sistemy/>

[(К оглавлению)](#_top)

# Режим ядра и пользовательский режим выполнения кода. Системные вызовы.

Для предотвращения доступа приложений к критически важным данным ОС и устранения риска их модификации Windows использует два режима доступа к CPU: пользовательский (user mode), ядра (kernel mode).

Код приложений работает в пользовательском режиме, тогда как код ОС (например, системные сервисы и драйверы устройств) -- в режиме ядра. В режиме ядра предоставляется доступ ко всей системной памяти и разрешается выполнять любые машинные команды CPU.

Хотя каждый Windows-процесс имеет свою (закрытую) память, код ОС и драйверы устройств, работающие в режиме ядра, делят единое виртуальное адресное пространство. Каждая страница в виртуальной памяти помечается тэгом, определяющим, в каком режиме должен работать CPU для чтения и/или записи данной страницы. Страницы в системном пространстве доступны лишь в режиме ядра, а все страницы в пользовательском адресном пространстве -- в пользовательском режиме. Страницы только для чтения (например, содержащие лишь исполняемый код) ни в каком режиме для записи недоступны.

Windows не предусматривает никакой защиты системной памяти от компонентов, работающих в режиме ядра. Иначе говоря, код ОС и драйверов устройств в режиме ядра получает полный доступ к системной памяти и может обходить средства защиты Windows для обращения к любым объектам. Поскольку основная часть кода Windows выполняется в режиме ядра, крайне важно, чтобы компоненты, работающие в этом режиме, были тщательно продуманы и протестированы.

Прикладные программы могут переключаться из пользовательского режима в режим ядра, обращаясь к системному сервису. Переключение из пользовательского режима в режим ядра осуществляется специальной командой CPU. ОС перехватывает эту команду, обнаруживает запрос системного сервиса, проверяет аргументы, которые поток передал системной функции, и выполняет внутреннюю подпрограмму. Перед возвратом управления пользовательскому потоку CPU переключается обратно в пользовательский режим. Благодаря этому ОС защищает себя и свои данные от возможной модификации пользовательскими процессами.

**Про вызовы:**

<https://coderlessons.com/tutorials/akademicheskii/osnovy-operatsionnykh-sistem/10-sistemnyi-vyzov-v-os>

[(К оглавлению)](#_top)

# Процесс. Ресурсы процесса.

<https://life-prog.ru/2_82038_protsessi-i-resursi.html>

<https://ru.bmstu.wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)>

[(К оглавлению)](#_top)

# Выполнение потоков в среде с вытесняющей многозадачностью.

Preemptive multitasking - вытесняющая многозадачность - это такой способ, при котором решение о переключении процессора с выполнения одного процесса на выполнение другого процесса принимается планировщиком операционной системы, а не самой активной задачей

В системе определен квант времени (порядка десятков миллисекунд) – процессорное время, выделяемое одному потоку (каждому - своё). Длительность выполнения одного потока не может превышать одного кванта. Когда это время заканчивается, диспетчер процессов переключает процессор на выполнение другого потока. При этом состояние регистров, стека и областей памяти – контекст потока, сохраняется в стеке потока. Очередность потоков определяется их состоянием и приоритетом.

<file:///C:/Users/kchip/Desktop/lectures/Lecture3.pdf>

[(К оглавлению)](#_top)

# Таблица процессов. Планировщик процессов.

<file:///C:/Users/kchip/Desktop/lectures/Lecture3.pdf>

<https://studopedia.ru/2_43384_planirovanie-protsessov.html>

<http://fkn.ktu10.com/?q=node/683>

[(К оглавлению)](#_top)

# Особенности создания процесса с помощью системного вызова UNIX/LINUX fork().

<https://www.opennet.ru/docs/RUS/linux_parallel/node7.html>

<https://www.ibm.com/developerworks/ru/library/au-unixprocess/#listing3>

<file:///C:/Users/kchip/Desktop/lectures/Lecture3.pdf>

[(К оглавлению)](#_top)

# Особенности создания процесса с помощью функций стандартного интерфейса POSIX – расширения языка C process.h – exec\*, spawn\*.

<https://www.ibm.com/developerworks/ru/library/au-unixprocess/>

<https://ru.qaz.wiki/wiki/Exec_(system_call)>

<https://ru.qaz.wiki/wiki/Spawn_(computing)>

[(К оглавлению)](#_top)

# Интерфейс системных вызовов Win32 API.

<https://gigabaza.ru/doc/30014-p2.html>

<https://tstu.ru/book/elib3/mm/2016/evdokimov/site/page3.3.html>

[(К оглавлению)](#_top)

# Создание процессов в Windows с помощью функции CreateProcess.

<http://www.vsokovikov.narod.ru/New_MSDN_API/Process_thread/fn_createprocess.htm>

<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa>

d

[(К оглавлению)](#_top)

# Получение информации о процессах с помощью интерфейса PSAPI.

<https://www.rsdn.org/article/qna/baseserv/enumproc.xml>

<https://docs.microsoft.com/en-us/windows/win32/api/_psapi/>

[(К оглавлению)](#_top)

# Механизм виртуальной памяти. Технология страничной организации памяти.

Виртуальная память — схема адресации памяти компьютера, при которой память представляется программному обеспечению непрерывной и однородной, в то время как в реальности для фактического хранения данных используются отдельные (разрывные) области различных видов памяти, включая кратковременную (оперативную) и долговременную (жёсткие диски, твёрдотельные накопители).

Также существует термин swap (англ. swap) также означающий виртуальную память (точнее способ её представления), или же означает подкачку данных с диска. Применение механизма виртуальной памяти позволяет:

* упростить адресацию памяти клиентским программным обеспечением;
* рационально управлять оперативной памятью компьютера (хранить в ней только активно используемые области памяти);
* изолировать процессы друг от друга (процесс полагает, что монопольно владеет всей памятью);

В большинстве современных операционных систем виртуальная память организуется с помощью страничной адресации. Оперативная память делится на страницы: области памяти фиксированной длины (например, 4096 байт), которые являются минимальной единицей выделяемой памяти (то есть даже запрос на 1 байт от приложения приведёт к выделению ему страницы памяти). Процесс обращается к памяти с помощью адреса виртуальной памяти, который содержит в себе номер страницы и смещение внутри страницы. Операционная система преобразует виртуальный адрес в физический, при необходимости подгружая страницу с жёсткого диска в оперативную память. При запросе на выделение памяти операционная система может «сбросить» на жёсткий диск страницы, к которым давно не было обращений. Критические данные (например, код запущенных и работающих программ, код и память ядра системы) обычно находятся в оперативной памяти (исключения существуют, однако они не касаются той части, которая отвечает за использование файла подкачки).

Важна: <https://vseloved.github.io/pdf/mem-ru.pdf>

[https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F\_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82% https://ru.bmstu.wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F\_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C\_(%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5\_%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)D1%8C](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C)

[(К оглавлению)](#_top)

# Таблица страниц. Менеджер виртуальной памяти.

<https://www.nrjetix.com/fileadmin/doc/publications/Lectures/Lecture5-2.pdf>

<https://ru.qaz.wiki/wiki/Page_table>

<http://komputercnulja.ru/operacionnye-sistemy/stranichnaya-organizaciya-pamyati>

<file:///C:/Users/kchip/Desktop/lectures/Lecture6.pdf>

[(К оглавлению)](#_top)

# Структура адресного пространства в MS Windows.

<https://intuit.ru/studies/courses/14491/1078/lecture/16578>

<https://infopedia.su/17xa8e4.html>

<https://www.nrjetix.com/fileadmin/doc/publications/Lectures/Lecture11-2.pdf>

[(К оглавлению)](#_top)

# Получение информации об организации памяти с помощью функций Win32 API.

FFF

[(К оглавлению)](#_top)

# Библиотеки динамической компоновки.

<https://ru.qaz.wiki/wiki/Dynamic-link_library>

<https://frolov-lib.ru/books/bsp.old/v13/ch3.htm#ch3_3>

<https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8_%D0%BF%D0%BE%D0%B4%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0>

<https://acm.bsu.by/wiki/C2018/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8#.D0.9E.D0.B1.D1.89.D0.B8.D0.B5_.D0.BA.D0.BE.D0.BD.D1.86.D0.B5.D0.BF.D1.86.D0.B8.D0.B8>

[(К оглавлению)](#_top)

# Структура PE-файла. Получение информации о PE-файлах с помощью интерфейса IMAGEHLP.

<https://ru.wikipedia.org/wiki/Portable_Executable#%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0>

<http://cs.usu.edu.ru/docs/pe/>

<https://ru.bmstu.wiki/PE_(Portable_Executable)>

<https://codeby.net/threads/0x01-issleduem-portable-executable-exe-fajl-format-pe-fajla.65415/>

<file:///C:/Users/kchip/Desktop/lectures/Lecture7.pdf>

<https://habr.com/ru/post/266831/>

[(К оглавлению)](#_top)

# Обмен данными между процессами в Windows с помощью библиотек динамической компоновки.

<https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2008/h90dkhs0(v=vs.90)?redirectedfrom=MSDN>

<file:///C:/Users/kchip/Desktop/lectures/Lecture11.pdf>

[(К оглавлению)](#_top)

# Обмен данными между процессами в Windows с помощью отображения файлов в память.

<https://lektsii.org/12-77543.html>

<https://frolov-lib.ru/books/bsp.old/v27/ch2.htm#ch2_1>

<file:///C:/Users/kchip/Desktop/lectures/Lecture12.pdf>

[(К оглавлению)](#_top)

# Неименованные каналы.

<https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B8%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB>

<https://parallel.uran.ru/book/export/html/464>

<http://www.iakovlev.org/index.html?p=1392>

<https://www.nrjetix.com/fileadmin/doc/publications/Lectures/Lecture9.pdf>

<file:///C:/Users/kchip/Desktop/lectures/Lecture13.pdf>

[(К оглавлению)](#_top)

# Именованные каналы.

<http://www.uhlib.ru/kompyutery_i_internet/sistemnoe_programmirovanie_v_srede_windows/p12.php>

<https://coderlessons.com/tutorials/kompiuternoe-programmirovanie/izuchite-mezhprotsessnoe-vzaimodeistvie/mezhprotsessnoe-vzaimodeistvie-imenovannye-kanaly>

<https://frolov-lib.ru/books/bsp/v27/ch2_3.htm>

<https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB>

<file:///C:/Users/kchip/Desktop/lectures/Lecture13.pdf>

[(К оглавлению)](#_top)

# Технология сокетов. Сокеты Беркли.

<https://lecturesnet.readthedocs.io/net/low-level/ipc/socket/intro.html>

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)>

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82%D1%8B_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8#%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D1%81%D0%BE%D0%BA%D0%B5%D1%82%D0%B0_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8>

<file:///C:/Users/kchip/Desktop/lectures/Lecture14.pdf>

[(К оглавлению)](#_top)

# Особенности реализации технологии сокетов winsock2.

<https://ru.wikipedia.org/wiki/Winsock>

<https://infourok.ru/interfejs-soketov-windows-4841151.html>

<https://club.shelek.ru/viewart.php?id=35>

<https://docs.microsoft.com/en-us/windows/win32/api/winsock2/>

<file:///C:/Users/kchip/Desktop/lectures/Lecture14.pdf>

<file:///C:/Users/kchip/Desktop/lectures/Lecture14m.pdf>

[(К оглавлению)](#_top)

# Архитектура приложения MS Windows.

<http://nauchebe.net/2010/04/arxitektura-windows%E2%80%91prilozheniya/>

<file:///C:/Users/kchip/Desktop/lectures/Lecture5.pdf>

[(К оглавлению)](#_top)

# Передача данных между процессами с помощью сообщений Windows.

<http://www.interface.ru/home.asp?artId=8317>

<https://frolov-lib.ru/books/bsp.old/v27/ch2.htm#ch2_1>

<https://books.ifmo.ru/file/pdf/909.pdf>

[(К оглавлению)](#_top)

# Ловушки Windows.

<file:///C:/Users/kchip/Desktop/lectures/Lecture17.pdf>

<https://swblog.ru/articles/programming/lovushki-v-microsoft-windows.html>

<http://www.delphikingdom.com/asp/viewitem.asp?catalogid=548>

[(К оглавлению)](#_top)

# Потоки и их синхронизация. Критические секции.

Поток — это по сути последовательность инструкций, которые выполняются параллельно с другими потоками. Каждая программа создает по меньшей мере один поток: основной, который запускает функцию main(). Программа, использующая только главный поток, является однопоточной; если добавить один или более потоков, она станет многопоточной.

<https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B5%D0%BA%D1%86%D0%B8%D1%8F>

<https://ru.qaz.wiki/wiki/Synchronization_(computer_science)>

<https://www.rsdn.org/article/baseserv/mt.xml>

<file:///C:/Users/kchip/Desktop/lectures/Lecture8.pdf>

<file:///C:/Users/kchip/Desktop/lectures/Lecture9.pdf>

[(К оглавлению)](#_top)

# Алгоритм Петерсона.

<http://espressocode.top/petersons-algorithm-for-mutual-exclusion-set-1/>

<https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9F%D0%B5%D1%82%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0>

<https://intuit.ru/studies/courses/2192/31/lecture/976?page=4>

[(К оглавлению)](#_top)

# Управление потоками в Windows с помощью функций интерфейса PSAPI.

FFF

[(К оглавлению)](#_top)

# Объекты ядра MS Windows и управление ими.

FFF

[(К оглавлению)](#_top)

# Мьютексы и семафоры.

<https://coderlessons.com/tutorials/akademicheskii/osnovy-operatsionnykh-sistem/3-miuteks-protiv-semafora>

<https://ru.gadget-info.com/difference-between-semaphore>

<https://ru.wikipedia.org/wiki/%D0%9C%D1%8C%D1%8E%D1%82%D0%B5%D0%BA%D1%81>

<https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)>

[(К оглавлению)](#_top)

# Реализация мьютексов в Windows.

<https://ru.wikipedia.org/wiki/%D0%9C%D1%8C%D1%8E%D1%82%D0%B5%D0%BA%D1%81#%D0%9C%D1%8C%D1%8E%D1%82%D0%B5%D0%BA%D1%81%D1%8B_%D0%B2_Win32_API>

<https://it.wikireading.ru/1419>

[(К оглавлению)](#_top)

# Реализация семафоров в Windows.

<https://it.wikireading.ru/1424>

[(К оглавлению)](#_top)

# Синхронизация потоков с помощью объектов Event.

<https://devpractice.ru/python-lesson-23-concurrency-part-2/#p15>

<https://compress.ru/article.aspx?id=11782#06>

<file:///C:/Users/kchip/Desktop/lectures/Lecture10.pdf>

[(К оглавлению)](#_top)

# Синхронизация потоков с помощью механизма критических секций в Windows.

<https://duckduckgo.com/?q=%D0%9A%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5+%D1%81%D0%B5%D0%BA%D1%86%D0%B8%D0%B8+Windows&atb=v179-4__&ia=web>

[(К оглавлению)](#_top)

# Реализация службы MS Windows.

<https://microsoft.fandom.com/ru/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows>

<file:///C:/Users/kchip/Desktop/lectures/Lecture15/Lecture15.pdf>

[(К оглавлению)](#_top)

# Реализация службы GNU/Linux

<http://dnny.ru/os_and_po/67-demony-v-linux-kto-oni-ili-chto-oni.html>

<https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BC%D0%BE%D0%BD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)>

<file:///C:/Users/kchip/Desktop/lectures/Lecture16.pdf>

[(К оглавлению)](#_top)