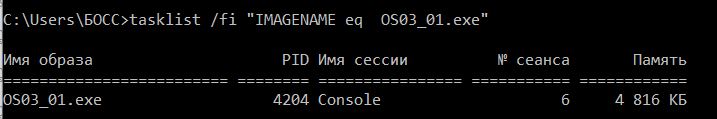
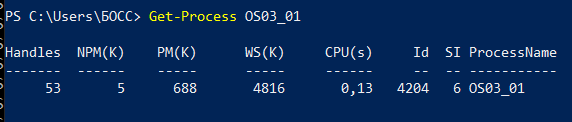
Лабораторная работа 03

OC, ПОИТ-3

**Задание 01**

1. Разработайте консольное Windows-приложение **OS03\_01** на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
2. Продемонстрируйте информацию о процессе **OS03\_01** в с помощью утилит **Task Manager**, **tasklist,**  **PowerShell ISE** и **Performance Monitor**.



****

**Задание 02**

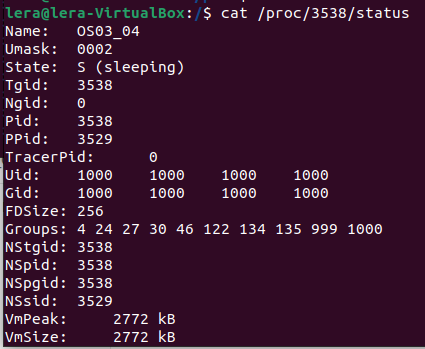
1. Разработайте консольное Windows-приложение **OS03\_02** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Приложение **OS03\_02** должно создавать два дочерних процесса **OS03\_02\_1** и **OS03\_02\_2.**
3. Процесс **OS03\_02\_1** - консольное Windows-приложение выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
4. Процесс **OS03\_02\_2** - консольное Windows-приложение выполняющее цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
5. Продемонстрируйте информацию о процессах **OS03\_02, OS03\_02\_1** и **OS03\_02\_2** в с помощью утилит **Task Manager**, **tasklist,**  **PowerShell ISE** и **Performance Monitor**.

**Задание 03.**

1. Разработайте консольное Windows-приложение **OS03\_03** на языке С++, выводящее на консоль перечень выполняющихся процессов в данный момент в OS.
2. Запустите приложение **OS03\_02** и продемонстрируйте с помощью приложения **OS03\_03** в перечне процессов **OS03\_02, OS03\_02\_1, OS03\_02\_2** и **OS03\_03**.

**Задание 04**

1. Разработайте консольное Linux-приложение **OS03\_04** на языке С, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.
2. Продемонстрируйте информацию о процессе **OS03\_04** с помощью файловой системы **/proc**.



1. Продемонстрируйте информацию о процессе **OS03\_04** с помощью утилиты **ps**.



**Задание 05**

1. Разработайте консольное Linux-приложение **OS03\_05** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Приложение **OS03\_05** должно создавать один дочерний процесс **OS03\_05\_1** с помощью системного вызова **fork.** Процесс **OS03\_05\_1** в этом случае неявляется отдельным модулем, а встроен (fork) в программный модуль **OS03\_05**.
3. Процесс **OS03\_05\_1** - консольное Linux-приложение выполняющее цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
4. Продемонстрируйте информацию о процессах **OS03\_05** и **OS03\_05\_1** с помощью файловой системы **/proc**.
5. Продемонстрируйте информацию о процессах **OS03\_05** и **OS03\_05\_1** с помощью утилиты **ps**.

**Задание 06**

1. Разработайте консольное Linux-приложение **OS03\_06** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.
2. Приложение **OS03\_06** должно создавать один дочерний процесс **OS03\_05\_1** (отдельный модуль)с помощью системного вызова **system.**
3. Продемонстрируйте информацию о процессах **OS03\_06** и **OS03\_05\_1** с помощью файловой системы **/proc**.
4. Продемонстрируйте информацию о процессах **OS03\_06** и **OS03\_05-1** с помощью утилиты **ps**.

**Задание 07**

1. Разработайте консольное Linux-приложение **OS03\_07** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса
2. Приложение **OS03\_07** должно создавать один дочерний процесс **OS03\_05\_1** (отдельный модуль)с помощью системного вызова **exec.**
3. Продемонстрируйте информацию о процессах **OS03\_07** и **OS03\_05\_1** с помощью файловой системы **/proc**.
4. Продемонстрируйте информацию о процессах **OS03\_07** и **OS03\_05-1** с помощью утилиты **ps**.
5. Продемонстрируйте разницу системных вызовов **system** и **exec**.

**Задание 08.ответьте на следующие вопросы**

1. 1 Что такое POSIX?
2. 2 Что такое системный вызов?
3. 3 Что такое аппаратное прерывание, программное прерывание?
4. 4 Что такое процесс?
5. 5 Что такое контекст процесса?
6. 6 Что такое родительский и дочерний процесс?
7. 7 Что такое процесс инициализации OS?
8. 8 Перечислите области памяти процесса и поясните их назначение.
9. 9 Чем отличаются системные процессы от пользовательских?
10. 10 Что такое Windows-сервисы, Linux-демоны?
11. 11 С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.
12. 12 С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.
13. 13 Какие потоки данных доступны любому процессу автоматически?
14. 14 Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.
15. 15 Поясните назначение системного вызова wait в Linux-приложении.
16. 16 Дайте развернутое определение процесса OS.

**+1. Что такое POSIX?**

Набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой, библиотеку языка С и набор приложений и их интерфейсов

**+2. Что такое системный вызов?**

Механизм вызова прикладной программой функции ядка OS

**+3. Что такое аппаратное прерывание, программное прерывание?**

**Аппаратное** - возникает как реакция микропроцессоров на физический сигнал от некоторого устройства (клавиатура, системные часы, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени

**Программное** - вызываются искусственно с помощью соответствующей команды из программы, предназначены для выполнения некоторых действий операционной системы, являются синхронными;

**+4. Что такое процесс?**

Выполняемая в данный момент программа. Единица работы OS - объект ядра

**+5. Что такое контекст процесса?**

данные, которые сохраняются при переключении процессов и предназначенные для продолжения работы

**6. Что такое родительский и дочерний процесс?**

Системный вызов fork() создает точную копию исходного процесса, называемого родительским процессом(parent process) Новый процесс называется дочерним процессом(child process)

**7. Что такое процесс инициализации OS?**

Процесс инициализации запускает все другие процессы, которые должны быть запущены, это родительский процесс для всего, что выполняется в системе. Другие процессы могут тоже создавать дочерние процессы, но если родительский процесс завершается, для его дочерних процессов родительским становится процесс инициализации.

**+8. Перечислите области памяти процесса и поясните их назначение.**

code, static, heap, data, stack

-----Возрастание адресов----->

**Code:** отображение программного кода в памяти  
**Static:** для константных объектов  
**Data:** отображение инициализированных переменных  
**Stack:** стек работает по схеме LIFO (последним вошел, первым вышел). Всякий раз, когда вызывается новый метод, содержащий примитивные значения или ссылки на объекты, то на вершине стека под них выделяется блок памяти.  
Когда метод завершает выполнение, блок памяти, отведенный для его нужд, очищается, и пространство становится доступным для следующего метода. Переменные в стеке существуют до тех пор, пока выполняется метод в котором они были созданы.

**Heap:** используется для выделения памяти во время выполнения программы. Но в отличие от стека, память, выделенная в куче, сохраняется и после того, как функция, вызвавшая выделение этой памяти, завершит работу. Язык С предоставляет программисту целый ряд средств управления памятью в куче.

**9. Чем отличаются системные процессы от пользовательских?**

Системные процессы являются частью ядра и всегда находятся в оперативной памяти. Такие процессы не имеют соответствующих им программ в виде исполняемых файлов и запускаются особым образом при инициализации ядра системы

**10. Что такое Windows-сервисы, Linux-демоны?**

**Windows-сервисы** - приложения, автоматически выполняемое системой при запуске операционной системы Windows и выполняющиеся вне зависимости от статуса пользователя

**Linux-демоны** - это программа, у которой есть определенная уникальная цель. Обычно, это служебные программы, которые незаметно работают в фоновом режиме для того чтобы отслеживать состояние и обслуживать определенные подсистемы и гарантировать правильную работу всей операционной системы в целом

**+11. С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.**

CreateProcess - данный вызов имеет множество параметров, многие из которых «по умолчанию» Функция CreateProcess создает новый процесс и его первичный (главный) поток

**+12. С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.**

fork, system

**fork()** создает точную копию вызывающего процесса. После выполнения системного вызова fork два процесса, родительский и дочерний, имеют единый образ памяти, единые строки описания конфигурации и одни и те же открытые файлы

**exec()** не создает новый процесс, а заменяет данные текущего процесса новыми данными. Вызов exec происходит таким образом, что переданная в качестве аргумента программа загружается в память вместо старой, которая вызвала exec.

**system()** неявно запускает новый процесс для обслуживания запроса и возвращает значение, полученное через дочерний процесс, который он изначально разветвил

**+13. Какие потоки данных доступны любому процессу автоматически?**

ввода, вывода, вывод ошибок

**14. Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.**

Приостанавливает выполнение потока до тех пор, пока не произойдет одно из двух событий:

* истечет таймаут ожидания;
* ожидаемый объект перейдет в сигнальное (signaled) состояние.

**15. Поясните назначение системного вызова wait в Linux-приложении.**

Блокирует вызывающий процесс до тех пор, пока один из его дочерних процессов не завершится (или не произойдет ошибка).

**16. Дайте развернутое определение процесса OS.**

процесс OS – единица работы OS - объект ядра OS+адресное пространство:

* создается ядром OS по системному вызову;
* адресное пространство (данные, программа, стек, куча);
* ресурсы: регистры, открытые файлы, родительский процесс, перечень связанных (дочерних) процессов, реальные страницы памяти, виртуальное адресное пространство, маркеры доступа (безопасность);
* процесс может создавать (с помощью системного вызова) дочерние процессы, в общем случае может образовываться дерево процессов;чя
* обычно соответствует работающей программе (например, exe-файлу в Windows);
* OS хранит список (или таблицу) объектов работающих процессов;
* при приостановке процесса в объекте процесса сохраняется вся информация (регистры, уведомления OS,… - контекст процесса), позволяющая возобновить работу процесса;
* процессы изолированы друг от друга;
* для обмена данными между процессами, применяется специальный механизм – механизм межпроцессного взаимодействия (IPC);

--------