Лабораторная работа 03

OC

**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**.
3. 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. Что такое POSIX?

**POSIX** (Portable Operating System Interface) - это набор стандартов, разработанных для обеспечения совместимости между различными операционными системами UNIX. Он определяет интерфейсы программирования приложений (API), библиотеки и другие инструменты, которые позволяют разработчикам создавать переносимое программное обеспечение.

1. Что такое системный вызов?

**Системный вызов** - это механизм, который позволяет программе взаимодействовать с ядром операционной системы. Он предоставляет доступ к функциям ядра, таким как управление памятью, вводом-выводом и создание процессов.

1. Что такое аппаратное прерывание, программное прерывание?

**Аппаратное прерывание** - это сигнал, отправляемый устройством в процессор для прерывания текущей работы и обработки важного события. **Программное прерывание** - это сигнал, отправляемый программой для прерывания текущей работы и выполнения определенного кода.

1. Что такое процесс?

**Процесс** - это экземпляр выполняющейся программы. Он содержит код программы, данные и ресурсы, необходимые для выполнения программы.

1. Что такое контекст процесса?

**Контекст процесса** - это состояние процесса в определенный момент времени, включая значения регистров процессора, указатели на память и другие данные.

1. Что такое родительский и дочерний процесс?

**Родительский процесс** - это процесс, который создает другой процесс (дочерний процесс). **Дочерний процесс** - это процесс, созданный другим процессом (родительским процессом).

1. Что такое процесс инициализации OS?

**Процесс инициализации OS** - это первый процесс, который запускается при загрузке операционной системы. Он отвечает за запуск других процессов и инициализацию системных ресурсов.

1. Перечислите области памяти процесса и поясните их назначение.

Области памяти процесса включают:

* 1. **Код**: Область памяти, содержащая исполняемый код программы.
  2. **Данные**: Область памяти, содержащая глобальные переменные и статические данные.
  3. **Стек**: Область памяти, используемая для хранения локальных переменных и вызовов функций.
  4. **Куча**: Область памяти, используемая для динамического выделения памяти во время выполнения программы.

1. Чем отличаются системные процессы от пользовательских?

**Системные процессы** выполняются от имени операционной системы и имеют привилегии доступа к системным ресурсам. **Пользовательские процессы** выполняются от имени пользователя и имеют ограниченные привилегии доступа к системным ресурсам.

1. Что такое Windows-сервисы, Linux-демоны?

**Windows-сервисы** - это приложения, которые выполняются в фоновом режиме без присутствия пользователя и предоставляют функциональность операционной системы или других приложений. **Linux-демоны** (также известные как службы) - это фоновые процессы, которые выполняются без присутствия пользователя и предоставляют определенные функции или службы.

1. С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.

[В Windows дочерний процесс можно создать с помощью системных вызовов CreateProcess или CreateThread](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output)[1](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output). [Разница между ними заключается в том, что CreateProcess создает новый процесс с отдельным адресным пространством памяти, в то время как CreateThread создает новый поток внутри существующего процесса](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output)[1](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output).

При использовании CreateProcess, новый процесс получает свое собственное адресное пространство памяти, что означает, что он не разделяет память с родительским процессом. [Это позволяет каждому процессу иметь свои собственные переменные, стек и другие ресурсы](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output)[1](https://learn.microsoft.com/ru-ru/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output).

1. С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.

В Linux дочерний процесс можно создать с помощью системного вызова fork(). Он создает точную копию родительского процесса, включая его память, файловые дескрипторы и другие атрибуты. [Разница между родительским и дочерним процессами заключается в том, что fork() возвращает 0 в дочернем процессе и идентификатор дочернего процесса в родительском процессе1](https://www.opennet.ru/docs/RUS/linux_parallel/node7.html).

1. Какие потоки данных доступны любому процессу автоматически?

Любому процессу автоматически доступны три потока данных: стандартный ввод (stdin), стандартный вывод (stdout) и стандартный поток ошибок (stderr). [Они представляют собой потоки данных, которые используются для ввода и вывода информации](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)[2](https://bing.com/search?q=%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%B8+%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85+%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BD%D1%8B%D0%B5+%D0%BB%D1%8E%D0%B1%D0%BE%D0%BC%D1%83+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D1%83+%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8)[3](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

1. Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.

[В Windows-приложении системный вызов WaitForSingleObject используется для ожидания завершения выполнения одного объекта4](https://question-it.com/questions/6494191/v-windows-mne-nuzhno-vyzyvat-waitforsingleobject-posle-vyzova-createprocess).

1. Поясните назначение системного вызова wait в Linux-приложении.

[В Linux-приложении системный вызов wait используется для приостановки выполнения текущего процесса до тех пор, пока дочерний процесс не завершится или не появится сигнал5](https://www.opennet.ru/man.shtml?topic=wait&category=2&russian=0).

1. Дайте развернутое определение процесса OS.
2. [Процесс операционной системы (OS) — это программное обеспечение, управляющее компьютерами и позволяющее запускать на них прикладные программы6](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).