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 (англ. Portable Operating System Interface — переносимый интерфейс операционных систем) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем

1. Что такое системный вызов?

системный вызов - механизм вызова прикладной программой функции ядра OS

1. Что такое аппаратное прерывание, программное прерывание?

Прерывание – передача управления специальной процедуре для выполнения функции ядра ОС. Аппаратное вызывается аппаратным обеспечением, например, периферийным устройством, программное – пользовательской программой

1. Что такое процесс?

Экземпляр выполняемой программы и контекст выполнения (значения регистров, адресное пространство и т.д.)

1. Что такое контекст процесса?

- программный счетчик процесса, т.е. адрес команды для исполнения процесса;

- содержимое регистров процессора;

- данные, необходимые для планирования использования процессора и управления памятью (приоритет процесса, размер и расположение адресного пространства и т. д.);

- учетные данные (идентификационный номер процесса, какой пользователь инициировал его работу, общее время использования процессора данным процессом и т. д.);

- сведения об устройствах ввода-вывода, связанных с процессом (например, какие устройства закреплены за процессом, таблица открытых файлов).

1. Что такое родительский и дочерний процесс?
2. Что такое процесс инициализации OS?

родитель всех прочих процессов

1. Перечислите области памяти процесса и поясните их назначение.

стек (локальные переменные и адреса возврата), данные (переменные), куча (динамическое выделение памяти), код

1. Чем отличаются системные процессы от пользовательских?

системные стартуют автоматически при запуске системы и управляют системными ресурсами, пользовательские запускаются пользователем

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

фоновые процессы, предназначенные для обработки событий, например, для работы с периферией

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

функция CreateProcess и ей подобные

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

есть один – fork. Остальные либо используют его (system), либо не создают дочерний процесс, а замещают родительский (exec-подобные)

1. Какие потоки данных доступны любому процессу автоматически?

stdin, stdout, stderr (Чаще всего цель этого потока совпадает с stdout, однако, в отличие от него, цель потока stderr не меняется при «>» и создании конвейеров («|»). То есть, отладочные сообщения процесса, вывод которого перенаправлен, всё равно попадут пользователю)

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

ожидает переход указанного объекта в сигнальное состояние или истечение времени. До тех пор блокирует поток выполнения

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

ожидание изменения состояния дочернего процесса и получение информации о нём. Изменением состояния считается: завершение работы дочернего процесса; остановка дочернего процесса по сигналу; возобновление дочернего процесса по сигналу. В случае, если дочерний процесс был завершён, выполнение ожидания позволяет системе освободить ресурсы, связанные с этим дочерним процессом

1. Дайте развернутое определение процесса OS.
2. **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;
* 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;