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?**

Стандарты для переносимости программ между UNIX-подобными системами.

1. **Что такое системный вызов?**

Механизм для приложений для взаимодействия с операционной системой.

1. **Что такое аппаратное прерывание, программное прерывание?**

Аппаратное прерывание - сигнал от устройства(аппаратуры)(асинх), программное прерывание - событие созданное программой(синх).

1. **Что такое процесс?**

Выполняющаяся программа со своим контекстом выполнения.

1. **Что такое контекст процесса?**

Данные для восстановления состояния процесса.

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

Процессы, связанные иерархически, где родитель порождает дочерний процесс.

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

Это первый процесс, который запускается при загрузке операционной системы. Его основная задача - инициализировать ядро операционной системы и запустить другие процессы.

1. **Перечислите области памяти процесса и поясните их назначение.**

Адреса возрастают от CODE до STACK.

CODE – инструкции ЦП скомпилированной программы, создаются автоматически. Read-only память

STATIC – сохраняется на протяжении всей жизни программы, используется, например, для глобальных переменных. Часто имеет размер 4 байта. Хранится в отделах .bss и .data

HEAP – тут хранятся основные данные. Динамическая обширная память

DATA – содержит глобальные и статические переменные

STACK – контекст функции: аргументы, возвращаемое значение, адрес возврата. Тут хранятся локальные переменные. Эта память автоматически выделяется ЦП. Часто имеет размер 8 MB.

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

Системные процессы выполняются с привилегированными правами и управляют внутренними аспектами операционной системы. Пользовательские процессы работают с пользовательскими приложениями и выполняются с ограниченными привилегиями. Разница в уровне доступа и целях выполнения.

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

**Windows-сервисы** - это программы, которые выполняются как фоновые службы в операционной системе Windows и могут работать без входа пользователя. **Linux-демоны** (или демоны) - это аналогичные фоновые процессы в Linux, часто используемые для задач, таких как управление системными службами и серверами.

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

CreateProcess() – (было бы неплохо знать, что это просто макрос, который скрывает две функции: CreateProcessA() и CreateProcessW(), которые отличаются только кодировкой – ANSI и Unicode соответственно).

CreateProcessAsUser() – дочерний процесс запускается в контексте системы безопасности пользователя, обозначенного параметром hToken.

CreateProcessWithTokenW() – дочерний процесс с привилегией NT Authority/SYSTEM. Может загружать профили пользователей и работать с WindowsStation и рабочим столом Winsta0 вместо вас (че бля? но я больше ниче не нашел мне впадлу)

CreateProcessWithLogonW() – дочернему процессу не нужно вызывать функцию LogonUser, чтобы подтвердить подлинность пользователя и получить маркер безопасности.

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

**Linux**: Для создания дочернего процесса в Linux можно использовать системный вызов **fork**. Он создает точную копию текущего процесса, включая код и данные. После **fork**, дочерний процесс может использовать системный вызов **exec** для замены своего адресного пространства новой программой. Это позволяет создавать новые процессы, которые могут выполнять разные программы.

1. **Какие потоки данных доступны любому процессу автоматически?**
2. Любой процесс в операционной системе имеет три стандартных потока данных:

* **stdin**: Поток стандартного ввода, через который процесс может принимать ввод от пользователя или других программ.
* **stdout**: Поток стандартного вывода, через который процесс может отправлять вывод или результаты выполнения программы.
* **stderr**: Поток стандартного вывода ошибок, используется для отправки сообщений об ошибках.

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

Этот системный вызов используется в Windows-приложениях для ожидания завершения выполнения другого процесса или объекта. Он блокирует текущий процесс до тех пор, пока объект (например, процесс) не завершит свою работу или не изменит свое состояние. Это позволяет синхронизировать выполнение разных частей программы.

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

Этот системный вызов используется в Linux-приложениях для ожидания завершения выполнения дочернего процесса. Он блокирует родительский процесс до завершения дочернего процесса и возвращает информацию о его завершении, такую как код завершения. **wait** позволяет родительскому процессу контролировать выполнение и обработку дочерних процессов.

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

Процесс операционной системы (OS) - это специальный процесс, который управляет и контролирует различными аспектами операционной системы, такими как планирование задач, ввод/вывод данных, управление памятью и обработка прерываний. Он является основой для функционирования операционной системы и обеспечивает выполнение других процессов.