Лабораторная работа 02

OC, ПОИТ-3

**Задание 01. Команды (утилиты) Linux**

1. Исследуйте назначение следующих стандартных утилит Linux:

echo,

ls, pwd, cd,

mkdir, rmdir,

touch,

cp, mv, rm

su, whoami,

man, whereis, whatis, apropos,

cat, less, head, tail,

lpstat, lpr, lpq, lpstat, lprm,

chgrp, chown, chmod,

zip, gzip, gunzip, bzip2, bunzip2, tar,

locate, grep, find,

history,

alias, unalias,

ps, top, lsof,

free, df, du,

yum,

ifconfig, ping, traceroute, host, iwconfig, dhclient, ifup, ifdown, route,

ssh, sftp, scp, rsync, wget, curl,

1. Составьте таблицу поясняющую назначение команд (п.1).

|  |  |
| --- | --- |
| Команда Linux | Краткое описание команды |
| **ls** | вывод списка файлов и каталогов… |
| … | … |
| **curl** | Копирование файлов web-узла… |

**Задание 02.Переменные окружения среды Linux**

1. Исследуйте назначение следующих стандартных переменных окружения

$HOME

$PATH

$PS1

$PS2

$IFS

1. Составьте таблицу поясняющую назначение переменных окружения (п.3).

|  |  |
| --- | --- |
| Переменная окружения | Краткое описание |
| $HOME | исходный каталог … |
| … | … |
| $IFS | разделитель … |

**Задание 04.Разработка С-приложения и bash-скрипт**

1. Разработайте приложение на языке С, выводящее 1000 сообщений на консоль. Сообщения должны выводиться каждые 2 секунды на консоль и содержать PID процесса и номер сообщения.



1. Разработайте bash-скрипт **os\_0007.sh**, который принимает два параметра: Pid процесса, строку fd (необязательный параметр). В результате работы bash-скрипт выводит в консоль: наименование исполняемого файла, Pid-процесса, Pid-родительского процесса (PPid), перечень дескрипторов (номеров) дескрипторов (fd) открытых потоков. Приведенные ниже скриншоты демонстрируют работу скрипта os\_0007.sh.









**Задание 05.Построение цепочки родительских процессов**

1. Применив, разработанный скрипт **os\_0007.sh**  постройте цепочку родительских процессов до процесса инициализации system.Ниже приводится пример, первых трех шагов построения цепочки.



**Задание 06.ответьте на следующие вопросы**

1. Что такое фреймворк OS?
2. Что такое POSIX?
3. Что такое аппаратное прерывание?
4. Что такое программное прерывание?
5. Что такое системный вызов?
6. Что такое процесс OS?
7. Что такое контекст процесса OS?
8. Что такое адресное пространство процесса?
9. Перечислите области памяти адресного пространства процесса и поясните их назначение.
10. Что такое стандартные потоки процесса?
11. Перечислите системные вызовы Windows для создания процесса?
12. Перечислите системные вызовы Linux для создания процесса?
13. С помощью каких утилит можно увидеть перечень процессов в Windows?
14. С помощью каких утилит можно увидеть перечень процессов в Linux?
15. Перечислите свойства процесса OS.
16. Фреймворк операционной системы (OS framework) - это набор программных компонентов, библиотек и инструментов, предназначенных для упрощения разработки операционных систем или приложений, работающих внутри операционных систем. Фреймворк OS предоставляет разработчикам средства для управления ресурсами компьютера, взаимодействия с аппаратным обеспечением и выполнения задач, связанных с обработкой процессов, потоков, файловой системой и сетевой коммуникацией.
17. POSIX (Portable Operating System Interface for Unix) - это набор стандартов, разработанных для обеспечения переносимости программного обеспечения между различными операционными системами, основанными на UNIX. POSIX определяет интерфейсы для работы с файлами, потоками, процессами, сетью и другими аспектами операционных систем, что делает его основой для разработки кросс-платформенных приложений.
18. Аппаратное прерывание (hardware interrupt) - это сигнал, который генерируется аппаратным устройством, таким как процессор, контроллер устройства или периферийное устройство, для того чтобы привлечь внимание центрального процессора (CPU). Прерывание используется для уведомления операционной системы или программы о наступлении события, требующего немедленной обработки, например, обработки ввода от клавиатуры или сетевых данных.
19. Программное прерывание (software interrupt) - это сигнал, который генерируется программой или операционной системой, чтобы вызвать выполнение определенного обработчика или функции. Программные прерывания часто используются для выполнения системных вызовов или переходов между режимами выполнения, такими как переключение из пользовательского режима в режим ядра.
20. Системный вызов (system call) - это специальная функция, предоставляемая операционной системой для обеспечения доступа к ее службам и ресурсам. Программы могут использовать системные вызовы для выполнения задач, которые требуют привилегий или доступа к аппаратному обеспечению, таким как создание процессов, чтение/запись файлов, управление памятью и другие операции.
21. Процесс операционной системы (OS process) - это выполнимая программа в рамках операционной системы, которая имеет свой собственный набор ресурсов, включая память, файловые дескрипторы и стек выполнения. Процесс представляет собой изолированную среду выполнения, и операционная система управляет их выполнением.
22. Контекст процесса операционной системы (OS process context) - это состояние процесса, включая значения регистров процессора, содержимое памяти и другие ресурсы, которые необходимы для продолжения выполнения процесса после переключения контекста. Операционная система сохраняет и восстанавливает контексты процессов при их переключении для обеспечения многозадачности.
23. Адресное пространство процесса (process address space) - это область памяти, выделенная для конкретного процесса в операционной системе. В адресном пространстве процесса хранятся исполняемый код, данные, стек вызовов и другие ресурсы, необходимые для работы процесса.
24. Области памяти адресного пространства процесса могут включать:

* Код (Text Segment): Область, где хранится исполняемый код программы.
* Данные (Data Segment): Область, где хранятся глобальные и статические переменные.
* Стек (Stack): Область, используемая для хранения локальных переменных и данных о вызовах функций.
* Куча (Heap): Область, в которой выделяются динамические данные, такие как объекты и массивы.
* Системные области: Различные системные ресурсы и информация о процессе, такие как таблицы файловых дескрипторов и информация о процессе.

1. Стандартные потоки процесса (standard process streams) - это три основных потока ввода-вывода, связанных с каждым процессом в операционной системе:

* stdin (стандартный ввод): Поток для ввода данных, который по умолчанию связан с клавиатурой или другим источником ввода.
* stdout (стандартный вывод): Поток для вывода данных, который по умолчанию связан с экраном или другим устройством вывода.
* stderr (стандартная ошибка): Поток для вывода сообщений об ошибках и диагностической информации.

1. Некоторые системные вызовы Windows для создания процесса включают:

* CreateProcess(): Создание нового процесса.
* CreateThread(): Создание нового потока внутри процесса.
* LoadLibrary(): Загрузка динамических библиотек (DLL) в процесс.

1. Некоторые системные вызовы Linux для создания процесса включают:

* fork(): Создание нового процесса, который является копией текущего процесса.
* exec(): Замещение текущего процесса новым исполняемым файлом.
* clone(): Создание нового процесса или потока с определенными параметрами.

1. В Windows можно увидеть перечень процессов с помощью следующих утилит:

* Task Manager: Графическая утилита для мониторинга и управления процессами.
* Command Prompt (команда **tasklist**): Командная строка для вывода списка процессов.

1. В Linux можно увидеть перечень процессов с помощью следующих утилит:

* ps: Команда для вывода списка процессов.
* top: Интерактивная утилита для мониторинга ресурсов процессов в реальном времени.
* htop: Улучшенная альтернатива утилите top.

1. Свойства процесса операционной системы могут включать:

* Идентификатор процесса (PID): Уникальный числовой идентификатор процесса.
* Состояние процесса: Текущее состояние процесса (например, работает, ожидает, завершен).
* Родительский процесс: Идентификатор родительского процесса, который создал данный процесс.
* Пользователь и группа: Информация о пользователе и группе, в рамках которых выполняется процесс.
* Ресурсы процесса: Используемые ресурсы, такие как память и файловые дескрипторы.
* Приоритет и политики планирования: Информация о приоритете и способе планирования выполнения процесса.
* Время выполнения: Время, проведенное процессом в системе и в пользовательском режиме.
* Список открытых файловых дескрипторов: Список файлов, с которыми процесс взаимодействует.
* Стек вызовов: Содержимое стека выполнения процесса.

Эти свойства позволяют операционной системе управлять и мониторить выполнение процессов в системе.