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?**

Фреймворк – набор библиотек и правил работы с ними.

Набор функций для взаимодействия с ОС.

1. **Что такое POSIX?**

Portable Operating System Interface – переносимый интерфейс ОС – набор стандартов взаимодействия между ОС и прикладной программой.

Используется для совместимости UNIX-подобных систем, но может быть использован и на не-UNIX системах.

Благодаря этим стандартам можно запускать код на разных UNIX-подобных системах.

1. **Что такое аппаратное прерывание?**

Вообще, прерывание – это запрос на остановку текущего выполняемого кода, чтобы выполнить определенное действие, описанное в обработчике прерывания

Аппаратные – реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, часы, клавиатура и т.д.).

Являются асинхронными, т.е. происходят в случайные моменты времени.

1. **Что такое программное прерывание?**

Программные – вызываются искусственно с помощью команды из соответствующей программы (например, int)

Предназначены для выполнения некоторых действий операционной системы.

Являются синхронными.

1. **Что такое системный вызов?**

обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

1. **Что такое процесс OS?**

Единица работы ОС. Объект ядра ОС + адресное пространство.

1. **Что такое контекст процесса OS?**

Это вся информация, необходимая для продолжения процесса после его приостановки.

1. **Что такое адресное пространство процесса?**

Адресное пространство процесса – диапазон адресов, которые выделены процессу. В этом диапазоне выделяются адреса, которые процесс может так или иначе использовать.

Каждому процессу выделяется "плоское" (состоит из одного подряд идущего диапазона адресов) 32- или 64-битовое адресное пространство.

Некоторые ОС предоставляют сегментированное адресное пространство —состоящее больше чем из одного диапазона адресов, т.е. из сегментов.

Современные ОС обычно предоставляют плоское адресное пространство.

1. **Перечислите области памяти адресного пространства 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. **Что такое стандартные потоки процесса?**

Это потоки процесса, имеющие зарезервированный номер (дескриптор) для выполнения некоторых стандартных функций:

0 – поток ввода stdin

1 – поток вывода stdout

2 – поток вывода ошибок stderr

[в DOS были потоки с номерами 4 – stdprn и 5 – stdaux]

1. **Перечислите системные вызовы Windows для создания процесса?**

CreateProcess()

CreateProcessAsUser()

CreateProcessWithTokenW()

CreateProcessWithLogonW()

1. **Перечислите системные вызовы Linux для создания процесса?**

fork() – создает клон текущего процесса (дочерний процесс)

exec() – заменяет текущий процесс новым процессом, запускающим другую программу

1. **С помощью каких утилит можно увидеть перечень процессов в Windows?**

tasklist – отображение списка процессов

taskkill – управление процессами и их завершение

1. **С помощью каких утилит можно увидеть перечень процессов в Linux?**

ps – запущенные процессы

top – процессы в режиме реального времени

lsof – список открытых файлов – также выводит процессы

1. **Перечислите свойства процесса OS.**

* Создается ядром по системному вызову
* Может создавать дочерние процессы (с помощью системного вызова)
* При создании выделяется адресное пространство
* Соответствует исполняемому файлу (exe)
* Процессы изолированы друг от друга
* При приостановке или переключении процесса вся необходимая для продолжения информация сохраняется в контексте
* Имеет PID и PPID