Создать виртуальную машину и установить на неё FreeBSD

Изучить особенности работы с FreeBSD в текстовом режиме, используя виртуальные терминалы, научиться переключению между терминалами и выполнению команд от имени другого пользователя. Познакомиться и научиться использовать простые команды. Получить представление о процессах, как о способе управления ресурсами. Научиться получать и анализировать информацию о процессах и управлять состоянием выполняющихся процессов.

**Теоретические сведения**

Изначально, терминал - это интерфейс ввода/вывода, состоящий из физических устройства ввода (клавиатура) и вывода (дисплей). Терминал предназначен исключительно для ввода информации и ее отображения на экране. Терминалы бывают физическими (реальными), виртуальными и псевдотерминалами (т.е. программами, которые «притворяются» терминалами). При выполнении лабораторных работ Вы будете использовать виртуальные терминалы.

**Вход в FreeBSD.**

FreeBSD это многопользовательская, многопроцессорная система. Любой многопользовательской системе нужен способ отличать каждого «пользователя» от остальных. В FreeBSD, эта задача решается путем авторизации («входа») пользователя в систему с использованием «персональных» данных. У каждого пользователя есть уникальное имя (''имя пользователя'') и пароль. Перед тем, как разрешить пользователю выполнять какие-либо действия, FreeBSD просит пользователя представиться. В случае правильности (с т.з. системы) введенных данных пользователю предоставляется возможность дальнейшей работы.

**Завершение работы**

FreeBSD требует корректного завершения работы, позволяющего сохранить необходимую системе служебную информацию, завершить в обычном порядке работающие программы, закрыть сеансы и соединения, уведомив об этом всех пользователей.

Для завершения работы во многих UNIX-системах используется утилита **shutdown**. Формат команды: **shutdown** ***action time***.

***action*** - указывает на действие, которое следует совершить,

***-r***(от restart) - перезагрузить компьютер;

***-p***(от power) - выключить питание компьютера;

***time*** - указывает на время совершения события ***action***, например ***now***заставит выполнить утилиту ***shutdown***немедленно.

**Множественные консоли**

При работе в FreeBSD можно использовать несколько виртуальных консолей (терминалов). Вы можете переключаться с одной виртуальной консоли на другую, нажимая соответствующие сочетания клавиш на клавиатуре. У каждой консоли есть свой канал вывода и FreeBSD заботится о том, чтобы правильно перенаправить ввод с клавиатуры и вывод на монитор, как только вы переключитесь с одной консоли на другую.

В конфигурации по умолчанию FreeBSD запускает восемь виртуальных консолей. Тем не менее, это не ограничение оборудования, и вы можете легко настроить систему для загрузки большего или меньшего числа виртуальных консолей. Число и параметры виртуальных консолей задаются в файле ***/etc/ttys***.

Вы можете использовать это файл для настройки виртуальных консолей FreeBSD. Любая не закомментированная строка в этом файле (строка, не начинающаяся с символа #), содержит настройки для одного терминала или виртуальной консоли.

Для переключения между консолями зарезервированы специальные комбинации клавиш. Вы можете использовать сочетания **Alt**-**F1**, **Alt**-**F2**, до **Alt**-**F8**, чтобы переключаться между различными виртуальными консолями в FreeBSD.

При переключении от одной консоли к другой, FreeBSD заботится о сохранении и восстановлении вывода на экран. Программы, которые Вы запускаете на одной виртуальной консоли, не прекращают выполнение, когда консоль становится невидимой, они продолжают выполняться, когда вы переключаетесь на другую виртуальную консоль.

**Процессы. Управление процессами.**

**Процесс** (process) — блок адресного пространства, в котором выполняются одна или более нитей, экземпляр выполняемой программы. Любой процесс может запускать другие процессы. Таким образом, процессы в среде UNIX образуют иерархическую структуру. На вершине этой структуры находится процесс **init**, являющийся предком всех остальных процессов.

**Атрибуты процессов**

С каждым процессом связан набор атрибутов, которые помогают системе контролировать выполнение процессов и распределять между ними ресурсы системы.

**Идентификатор процесса** (process ID) – это целое число однозначно идентифицирующее процесс. Процесс с идентификатором 1 это процесс init. Процесс идентификатором 0 – процесс свопинга (swapping).

**Идентификатор родительского процесса** (parent process ID) указывает на родительский процесс.

**Идентификатор группы процессов** (process group ID). Процессы могут объединяться в группы. Каждая группа обозначается идентификатором группы. Процесс, идентификатор которого совпадает с идентификатором группы, называется лидером группы.

**Идентификатор сеанса** (session ID). Каждая группа процессов принадлежит к сеансу. Сеанс связывает процессы с управляющим терминалом. Когда пользователь входит в систему, все создаваемые им процессы будут принадлежать сеансу, связанному с его текущим терминалом.

Процесс может быть:

* **выполняемым** в текущий момент (R, Runned);
* **находящимся в режиме ожидания** (S, Suspended);
* **прерванным** (T, Terminated), например, при использовании клавиш Ctrl+Z;
* **«зомби»** (Z, Zombied) - завершившимся, но от которого родительский процесс еще не принял сигнала завершения. Спустя некоторое время «зомби» завершаются окончательно и освобождают ресурсы;
* **зависшим**, или в состоянии непрерывного ожидания (uninterruptible sleep). Такой процесс не реагирует на какие-либо сигналы и может быть снят только перезагрузкой системы.

**Идентификаторы пользователя и группы.** С каждым процессом связаны действительные идентификаторы пользователя (real user ID) и группы (real group ID), совпадающие с соответствующими идентификаторами пользователя, запустившего процесс. Кроме того, с процессом связаны эффективные идентификаторы пользователя (effective user ID) и группы, определяющие права процесса в системе. Обычно, действительные и эффективные идентификаторы совпадают.

**Приоритет** (nice) Значение nice показывает готовность процесса уступить свое процессорное время другим процессам. Чем больше значение nice, тем ниже приоритет процесса.

**Даемоны**

Если вы запускаете редактор, им можно легко управлять, открывать в нем файлы и т.д. Вы можете делать это, поскольку редактор предоставляет такие возможности и потому, что редактор присоединен к терминалу. Некоторые программы разработаны без поддержки интерфейса пользователя, поэтому они отсоединяются от терминала при первой возможности. Н-пример, веб-сервер целый день отвечает на запросы из сети, и ему, как правило, не требуется ваше вмешательство. Программы, передающие почту от сервера к серверу – другой пример приложений этого класса.

Мы называем эти программы даемонами (от англ.: daemon). Даемоны (демоны) это персонажи греческой мифологии; хорошие или плохие, они были спутниками человека и, вообще говоря, выполняли полезную работу для людей. Почти как веб- и почтовые серверы выполняют полезную работу сегодня.

Есть соглашение, по которому имя программы, которая обычно запускается как даемон, заканчивается на «d». BIND это Berkeley Internet Name Daemon (выполняемая программа называется named), программа веб-сервера Apache называется httpd, даемон очереди печати это lpd и так далее. Например, главный почтовый даемон для Sendmail называется sendmail.

**Механизмы взаимодействия**

UNIX имеет большое число механизмов межпроцессного взаимодействия. Наиболее популярными средствами являются сигналы, программные каналы (pipes) и именованные каналы (FIFO).

**Сигналы**

Сигналы обеспечивают простой метод прерывания работы процессов. Сигналы используются в основном для обработки исключительных ситуаций. Процесс может определять действия, выполняемые при поступлении сигнала, блокировать сигналы, посылать сигналы другим процессам.

**Основные сигналы:**

SIGCHLD – сигнал о завершении дочернего процесса.

SIGHUP – сигнал освобождения линии. Посылается всем процессам, подключенным к управляющему терминалу при отключении терминала. Многие демоны при получении данного сигнала заново просматривают файлы конфигурации и перезапускаются.

SIGINT – сигнал посылается всем процессам сеанса, связанного с терминалом, при нажатии пользователем клавиши прерывания (CTRL-C).

SIGTERM – сигнал приводит к немедленному прекращению работы получившего сигнал процесса.

SIGKILL – сигнал приводит к немедленному прекращению работы получившего сигнал процесса. В отличие от SIGTERM процесс не может блокировать и перехватывать данный сигнал.

SIGSEGV – сигнал посылается процессу, если тот пытается обратиться к неверному адресу памяти.

SIGSTOP – сигнал приводящий к остановке процесса. Для отправки сигнала SIGSTOP активному процессу текущего терминала можно воспользоваться комбинацией клавиш (CTRL-Z).

SIGCONT – сигнал, возобновляющий работу остановленного процесса.

Для того, чтобы отправить процессу сигнал можно использовать команду **kill**. Для того чтобы процесс мог отправить сигнал другому процессу, необходимо, чтобы эффективные идентификаторы пользователя у посылающего процесса и у процесса получателя совпадали. Процессы с эффективным идентификатором пользователя равным нулю могут посылать сигналы любым процессам.

**Каналы**

Часто возникает ситуация когда два процесса последовательно обрабатывают одни и те же данные. Для обеспечения передачи данных от одного процесса к другому в подобных ситуациях используются программные каналы. Программный канал (pipe) служит для установления связи, соединяющей один процесс с другим. Запись данных в канал и чтение из него осуществляются при помощи системных вызовов write и read, т.е. работа с каналами аналогична работе с файлами. Для создания программного канала используется системный вызов pipe. Вызов возвращает два дескриптора файлов, первый из которых открыт для чтения из канала, а второй для записи в канал. Каналы используются, например, при организации конвейера.

При выполнении команды:

**find /usr/bin -name a\* | sort**

создается канал, команда find выводит в него результаты своей работы, а команда sort считывает из этого канала данные для сортировки.

Главным недостатком программных каналов является то, что они могут использоваться только для связи процессов имеющих общее происхождение (напр., родительский процесс и его потомок). Другой недостаток ограниченное время существования канала (программные каналы уничтожаются после завершения обращающегося к ним процесса). Именованные каналы идентичны программным в отношении записи и чтения данных, но они являются объектами файловой системы. Именованный канал имеет имя, владельца и права доступа. Открытие и закрытие именованного канала осуществляется как открытие и закрытие любого файла, но при чтении и записи он ведет себя аналогично каналу.

**Команды для работы с процессами**

**Top** – консольная команда, которая выводит список работающих в системе процессов и информации о них. По умолчанию она в реальном времени сортирует их по нагрузке на процессор. Чтобы выйти из программы top, необходимо нажать клавишу [q].

Полезные интерактивные команды, которые можно использовать в top:

* **[Пробел]** – Немедленно обновить содержимое экрана.
* **[h]** – Вывести справку о программе.
* **[k]** – Уничтожить процесс. Программа запрашивает у вас код процесса и сигнал, который будет ему послан.
* **[n]** – Изменить число отображаемых процессов. Вам предлагается ввести число.
* **[u]** – Сортировать по имени пользователя.
* **[M]** – Сортировать по объёму используемой памяти.
* **[P]** – Сортировать по загрузке процессора.

**ps [-axewjlu] [-o формат] [-U пользователь] [-p pid]**

Выводит список и статус процессов работающих в системе. Без аргументов выводит список процессов текущего пользователя, подключенных к терминалу, в текущей сессии. Значения параметров следующие:

* **-a** – вывести информацию о процессах всех пользователей.
* **-x** – вывести информацию о процессах, не подключенных к терминалу.
* **-e** – вывести значения переменных окружения процесса.
* **-w** – использовать строки длиной 132 символа. Если указан несколько раз, то строки не обрезаются совсем.
* **-c, -j, -l, -u, -v** – меняют формат вывода информации.
* **-o** – формат вывести информацию в указанном формате.
* **-U** ***пользователь*** – вывести информацию о процессах указанного пользователя.
* **-p** ***pid*** – вывести информацию о процессе с указанным идентификатором.

Значение формата для параметра **-o** является списком из следующих ключевых слов, разделенных запятыми (без пробелов):

* **сommand** командная строка и аргументы.
* **nice** уровень nice (приоритет).
* **pgid** идентификатор группы процессов.
* **pid** идентификатор процесса.
* **ppid** идентификатор родительского процесса.
* **rgid, ruid** реальные идентификаторы группы и пользователя.
* **uid** реальный идентификатор пользователя.
* **tty** управляющий терминал

**kill [-s signal| -signal] pid**

Посылает сигнал указанному процессу. Если значение сигнала опущено, предполагается SIGTERM. signal – символическое имя сигнала без префикса SIG, либо номер сигнала.

Пример: **kill -HUP 172** – послать сигнал SIGHUP процессу с идентификатором 172.

Выполнение **kill -s KILL 1** – быстрый способ перегрузить систему, т.к. главный процесс (**init**) имеет идентификатор **1**.

**nice [-nice] команда [аргументы]**

Выполняет команду с меньшим приоритетом. Если nice не задан, то предполагается значение nice может быть от -20 (наивысший приоритет) до 20 (наименьший приоритет). Отрицательные числа задаются как –nice. Увеличение приоритета может выполнить только суперпользователь.

Пример: **nice -10 john users** – запустить программу john с пониженным приоритетом.

**Перенаправление стандартного вывода**

Для обозначения перенаправления используются символы **">"**, **"<"** и **">>"**. Чаще всего используется перенаправление вывода команды в файл. Вот соответствующий пример:

**ls -l > /home/Student/dir.txt**

По этой команде в файле **/home/ Student/dir.txt** будет сохранен перечень файлов и подкаталогов того каталога, который был текущим на момент выполнения команды **ls**, при этом если указанного файла не существовало, то он будет создан; если он существовал, то будет перезаписан; чтобы вывод команды был дописан в конец существующего файла, то надо вместо символа **>** использовать **>>**.

Новый текстовый файл можно создать, например, с помощью перенаправления стандартного ввода с клавиатуры в файл:

**cat > f1.txt**

**Hello World!**  // для завершения ввода необходимо нажать Ctrl+C

или перенаправить вывод команды в файл

**echo “Privet Student!” > f2.txt**

Можно и посмотреть созданный файл с помощью команды: **cat f1.txt**

Команды **more** и **less**

**more** – это основная команда постраничной или построчной прокрутки большого текста на экране и поиска в этом тексте.

После нажатия клавиши Enter на экране появится первая страница текста, в последней строке которой будет одно слово -More- (еще). Нажатие клавиши пробела прокручивает текст на одну страницу, нажатие Enter - на одну строку. В конце текста на экране появится подсказка командной строки.

Команде **more** можно передать результаты выполнения другой команды, воспользовавшись методом конвейеризации.

**ls -l /usr/ports | more**

**less** – усовершенствованный вариант команды **more**. В дополнение к функциям, описанным выше (постраничная или построчная прокрутка текста от начала до конца и поиск), команда less позволяет выполнять следующие операции:

* переход на указанную строку;
* переход в начало или конец текста;
* прокрутка текста от конца к началу;
* поиск в обратном направлении.

Чтобы перейти на определенную строку текста, введите ее номер с буквой **g** в конце. Если опустить номер строки, будет выполнен переход к первой строке. Вместо строчной буквы **g** можно указать заглавную (**G**), но тогда при отсутствии номера строки будет выполняться переход не к первой, а к последней строке.

Для построчной прокрутки текста вверх или вниз достаточно нажать соответствующую клавишу со стрелкой. Постраничная прокрутка вверх выполняется нажатием клавиш **Ctrl+B**. Для выхода нажмите **q**.

**Выполнение**

Сразу после загрузки FreeBSD и завершения работы стартовых скриптов, система предлагает ввести имя пользователя:

**login:**

В этом примере, будем использовать предопределенное имя «главного» пользователя - root. Вводим *root* в приглашении и нажимаем **Enter**. Далее должно появиться приглашение ввести «пароль»:

**login: root**

**Password:**

Введите соответствующий имени root пароль и нажмите **Enter**. Пароль *не виден*! Не беспокойтесь об этом. Это сделано по соображениям безопасности. Если вы ввели пароль правильно, то сразу же войдете в FreeBSD и можете начать выполнять команды. Вы увидите сообщение дня (MOTD, или message of the day, которое хранится в текстовом файле /etc/motd) за которым последует командная строка (с символом *#*, *$*, или *%*). Это означает, что вы успешно вошли в FreeBSD.

Информацию о командах вы можете получить через встроенную справочную систему формата **man** (от manuals) или **info**. Для получения справки достаточно ввести **man** (или **info**) с именем нужной команды в качестве параметра. Получите справку по команде **pwd**.

Изучите справку по простым командам:

* **pwd** – возвращает имя текущей директории
* **tty** – возвращает имя текущего терминала
* **who** – возвращает список пользователей вошедших в систему
* **w** – немного более подробная информация о работающих пользователях
* **uname** – возвращает информацию о системе
* **uptime** – возвращает время работы системы
* **date** – сообщает или устанавливает дату и время
* **exit** – завершение сеанса
* **ls [-F|-l|-a]** – возвращает содержание директории; **-F** выдает список файлов в текущей директории со значком «\*» после исполняемых файлов, «/» – после директорий и «@»– после символических ссылок. Ключ **-a** – все файлы, в том числе скрытые; **-l** – в длинном формате, с указанием даты и прочего.
* **cd [..|~]** ***имя\_директории*** – позволяет изменить текущую директорию
* **cat *filename*** – выдаёт содержимое ***filename*** на экран, если он слишком длинный и вы можете увидеть только его конец, нажмите ScrollLock и используйте клавишу стрелка вверх для движения назад; вы можете также использовать ScrollLock и со страницами справки. Нажмите ScrollLock снова для прекращения прокрутки

**Практика**

Используя простые команды, выполните следующее:

1. выясните, какая директория является текущей
2. выясните, в каком терминале выполняется текущий сеанс
3. откройте новый сеанс в **ttyv3** для пользователя **Student**
4. повторите пункты 1-2
5. перейдите в **ttyv0**, выясните, какие пользователи и в каких терминалах работают
6. при помощи команд **tty**, **w**, **uname**, **uptime** выведите в файл **info.txt** имя текущего терминала, информацию о пользователях, работающих в системе, название и версию операционной системы, время работы системы
7. завершите сеанс в **ttyv3**

Используя теоретические сведения и справочную систему FreeBSD, выполните следующее:

1. войдите в систему с учетной записью **Student** (например, в терминале **ttyv2**)
2. получить справку о команде **ps**
3. командой **ps** выведите краткую информацию о выполняющихся процессах в текущем терминале и определите PID текущей оболочки
4. получите подробную информацию о загруженных процессах, какой из них использует максимальный объем памяти, а какой - максимально загружает процессор
5. по информации, полученной в пункте 4, выясните, какой PID имеет процесс **init** и от чьего имени он запущен
6. изучите ключи **-c, -v, -j, -u, -l**, которые изменяют формат вывода команды; выполните команду **ps** с каждым из этих ключей, результаты сохраните в соответствующих файлах
7. определите, какие процессы порождены процессом **init**
8. работая под учетной записью **root**, выполните команду **top**
9. переключитесь в терминал, где работаете от имени **Student**
10. найдите в списке процессов идентификатор процесса **root** для команды **top**
11. используя команду **kill**, попытайтесь «убить» этот процесс, объясните результат
12. запустите команду **top** от имени **Student** и попробуйте «убить» это процесс, работая от имени **root**
13. работая под учетной записью **Student**, выведите в файл отчета (**ps.txt**) следующую информацию о запущенных **Вами** процессах: идентификатор процесса, терминал, идентификатор пользователя, приоритет, команда
14. работая под учетной записью **Student**, выведите в файл отчета (**ps1.txt**) информацию о процессах, запущенных пользователем **root**

Выключите виртуальную машину, работая от имени Student. Если не удалось, попробуйте от имени **root**. Для этого используйте команду:

**shutdown –p now**

В качестве выполненного ДЗ отправьте скриншот виртуальной машины. Так же можно для каждого пункта из практического руководства указать набор команд для выполнения задания.  
  
Пример:  
 1. pwd

2. tty

3. …