**Сетевые операционные системы**

**Практическое занятие №4**

Командная оболочка ОС UNIX Shell

***Цель:*** Изучить основное назначение возможности командной оболочки shell ОС UNIX.

**Теоретические сведения:**

**1. Командная оболочка ОС UNIX**

Интерфейсом для ОС UNIX служит shell (*оболочка),* являющаяся программным уровнем, который обеспечивает среду для ввода команд и параметров для получения желаемого результата. Для удовлетворения различных потребностей в UNIX существуют различные оболочки. Например, оболочки Bourne (sh), Bourne Again (bash), Korn (ksh) и С (csh). У каждой из них — свои свойства и способы взаимодействия с UNIX. В настоящей практической работе рассматриваются следующие темы:

Способ взаимодействия пользователя с командной оболочкой в ОС UNIX, свойства командного процессора shell, управление средой окружения shell

**Взаимодействие ядра и оболочки**

Когда система UNIX переводится в интерактивный режим, ядро системы загружается в основную память компьютера, где остается до выключения компьютера. Во время процесса загрузки программа **init** выполняется в качестве фоновой задачи и продолжает выполняться до выключения. Эта программа сканирует файл **/etc/inittab,** в котором перечисляются порты, имеющие терминалы, и их характеристики. Когда эта программа активна и открытый терминал найден, **init** вызывает программу **getty,** которая вызывает на монитор терминала приглашение **login:.** Когда этот процесс вызван и выполняется, пользователь может начинать взаимодействовать с системой.

**Вызовы оболочки UNIX при входе в систему**

На рис. 1 показано протекание процесса от загрузки ядра до входа в оболочку. С этого момента пользователь находится в активной оболочке и готов к вводу команд в систему.

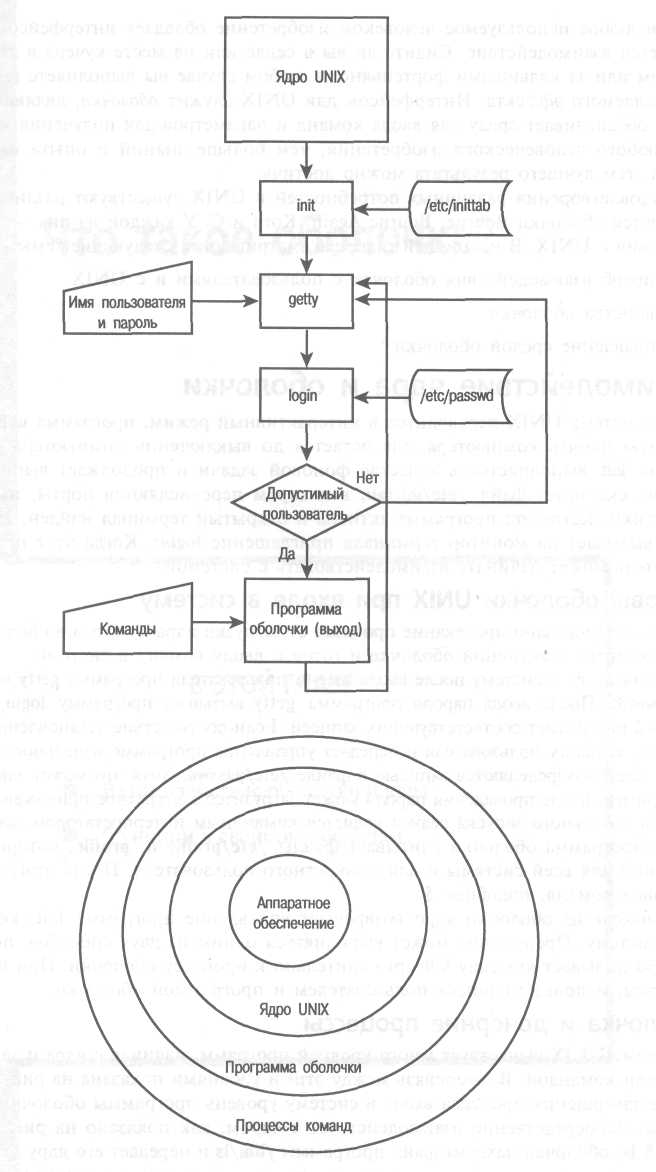


Рис. 1. Запуск shell при входе пользователя в систему.

Во время входа в систему после ввода имени пользователя программа **getty** выводит на монитор приглашение **password:.** После ввода пароля программа getty вызывает программу **login,** которая просматривает файл **/etc/passwd** на предмет соответствующих записей. Если соответствие установлено, **login** осуществляет переход к начальному каталогу пользователя и передает управление программе начального запуска сеанса; и имя пользователя, и пароль определяются записью в файле /etc/passwd. Хотя это может зависеть от программы приложения, например, после введения пароля может загрузится конкретное приложение, управляемое меню, обычно программа начального запуска сеанса является командным интерпретатором, например таким как оболочка Bourne **/bin/sh**.

Затем программа оболочки считывает файлы **/etc/profile** и **.profile** (из рабочего каталога пользователя)**,** которые устанавливают параметры, характерные для всей системы и для конкретного пользователя. После этого оболочка выводит приглашение на ввод команд, подобное $.

При выходе из оболочки ядро возвращает управление программе **init,** которая перезапускает процесс входа в систему. Прерывание может выполняться одним из двух способов:

- посредством команды **exit;**

**-** или когда ядро вызывает команду **kill** применительно к процессу оболочки.

При прерывании ядро освобождает все ресурсы, использовавшиеся пользователем и связанной с ним программой оболочки.

**Оболочка и дочерние процессы**

В системе UNIX существует много уровней программ, начиная с ядра и заканчивая конкретным приложением или командой. Взаимосвязь между этими уровнями показана на рис. 1.

После завершения процесса входа в систему уровень программы оболочки (уровень командного интерпретатора) непосредственно взаимодействует с ядром, как показано на рис. 1. При вводе команды, например, $**ls,** оболочка находит файл программы **/bin/ls** и передает его ядру для исполнения. Ядро выделяет память для нового дочернего процесса, загружает программу и выполняет инструкции, содержащиеся в **/bin/ls.** По завершении программы ядро освобождает выделенную процессу память и возвращает управление родительской программе оболочки. Для ознакомления с примером такого процесса введите следующую команду:

$ps

В результате выводится перечень выполняющихся в данный момент команд: программа оболочки и программа **ps.** Теперь введите следующее:

$sleep 10 & $ps

Первая команда создает дочерний процесс **sleep,** подлежащий выполнению в фоновом режиме, который отображается посредством команды ps. При каждом вводе команды создается дочерний процесс, который выполняется независимо от родительского процесса или оболочки. Это оставляет родительский процесс неизменным при выполнении другой задачи.

**Автоматическое выполнение оболочки**

Некоторые ресурсы UNIX, например **сгоn** (демон выполнения заданий по расписанию)**,** могут запускать программу оболочки без вмешательства пользователя. При использовании этого свойства пользователю нужно в первой строке файла с кодом программы указать оболочку, которую необходимо запустить, например, так:

#!/bin/sh

Эта строка определяет оболочку Bourne.

Необходимо также переназначить любой вывод, поскольку ни один терминал не связан с автоматическим выполнением.

**Функции и свойства оболочки**

Какую бы оболочку не выбрал пользователь, все они имеют одно и то же назначение — предоставить пользователю интерфейс к UNIX. С этой целью все оболочки обладают некоторыми общими характеристиками:

* Интерпретация командной строки.
* Зарезервированные ключевые слова (имена команд и переменных окружения).
* Метасимволы оболочки (шаблоны).
* Доступ к программам и их исполнение.
* Обработка файлов: переназначение ввода/вывода и программные каналы.
* Поддержка переменных среды окружения командной оболочки.
* Управлением средой окружения командной оболочки.
* Программирование на основе команд оболочки.

**2. Интерпретация командной строки**

При входе в систему запускается специальная версия shell, называемая *интерактивной оболочкой,* отображается приглашение оболочки, обычно в форме знака доллара ($), процента (%) или фунта (#). При вводе строки в ответ на это приглашение оболочка пытается интерпретировать ее. Ввод в ответ на приглашение оболочки иногда называют *командной строкой.* Основной формат командной строки выглядит следующим образом:

*команда аргументы*

**Команда** — это исполняемая команда UNIX, программа, утилита или программа оболочки.

**Аргументы**передаются исполняемому модулю. Большинство программ утилит UNIX предполагают, что **аргументы**будут иметь следующую форму:

*опции имена\_файлов*

Например, в командной строке

$ls -l filel file2

- имеется три аргумента команды ls; первый является опцией, а последние два — именами файлов.

Одно из действий, выполняемых shell для ядра — исключение необязательной информации. Так, применительно к компьютеру одним из типов необязательной информации являются пробелы; следовательно, важно знать, что делает оболочка, встречая пробел. Пробелами являются символы пробелов, горизонтальной табуляции и символы новой строки. Рассмотрим следующий пример:

$echo part A part В part С part A part В part С

В данном случае оболочка интерпретирует командную строку как команду **echo,** имеющую шесть аргументов, и удаляет пробелы между ними. Например, при печати заголовков отчета и при необходимости сохранить пробелы, нужно было бы заключить данные в символы кавычек, как в следующем примере:

$echo 'part A part В part С part A part B part С'

Символ одинарных кавычек мешает оболочке “заглядывать” внутрь кавычек. Теперь оболочка интерпретирует эту строку как команду **echo** с единственным аргументом, который в данном случае является строкой символов, содержащей пробелы.

**Зарезервированные слова**

Все версии оболочек имеют слова, обладающие специальным значением. При программировании средствами оболочек такие слова, как **do, done, for** и **while** обеспечивают управление циклом, a **if, then, else if -** условиями. Каждая версия оболочки обладает различными зарезервированными словами, присущими именно ей.

**Метасимволы оболочки (шаблоны)**

Шаблоны (таблица 1) используются для указания в команде нескольких имен обрабатываемых файлов.

|  |  |
| --- | --- |
| Все версии оболочек поддерживают метасимволы, которые дают возможность указывать имена файлов. Шаблоны приводятся ниже:*Шаблоны* | *Описание* |
| **?** | Соответствие любому символу |
| **\*** | Соответствие любой части символьной строки |
| **[a-z]** | Соответствие диапазону приведенных символов |
| **[0-9]** | Соответствие диапазону приведенных цифр |

Шаблоны могут быть удобны при обработке ряда специфичных файлов. Ниже приведен ряд примеров:

$ls t\*

Эта команда выводит перечень всех файлов, чьи имена начинаются с t.

$ls test?5.dat

Эта команда выводит перечень всех файлов, имена которых начинаются с test, далее следует любой одиночный символ, а затем — 5.dat.

$ls [а-с]\*

Эта команда выводит перечень всех файлов, имена которых начинаются с символов от а до с.

$ls [e,m,t]\*

В результате выполнения этой команды отображается список всех файлов, имена которых начинаются с е, m или t.

**Команды запуска программ**

При вводе команды оболочка считывает переменную среды $PATH содержащую список каталогов с файлами программ. Оболочка просматривает этот набор каталогов, отыскивая файл программы, соответствующий команде. Затем оболочка передает ядру действительное имя файла.

**Обработка файлов - переправления ввода/вывода и прграммные каналы**

Большинство команд UNIX принимают ввод с клавиатуры терминала и отправляют вывод на его дисплей, если иное не указано аргументами. Для переназначения вывода в файл используется символ '>'. Например, команда

$ls > myfiles

выводит список файлов текущего каталога и помещает его в файл, названный myfiles. Аналогично можно переназначить ввод с помощью символа <. Например, команда

$wc -l < myfiles

осуществляет ввод в команду wc из файла myfiles. Хотя такой же результат можно было бы получить, использовав имя файла в качестве аргумента, необходимость в переназначении ввода становится более очевидной при программировании на языке оболочки shell.

Для направления вывода одной команды во ввод следующей можно использовать символ | (канал). Например, команда:

$ls -s | -nr | pg

выдает список файлов текущего каталога и их размер в блоках, направляет вывод в команду sort, которая упорядочивает файлы в порядке уменьшения номеров, а затем направляет вывод в команду формирования страниц pg для окончательного отображения на мониторе терминала. Команда канала — одно из наиболее используемых инструментальных средств при создании конструкций команд.

**Подстановка команд**

Подстановка команд аналогична переназначению, за исключением того, что она используется для получения аргументов команды из вывода другой команды. Например, команда

$grep 'wc -l myfiles` \*'

берет число строк в файле myfiles из команды wc и помещает это число в качестве аргумента в команду grep для отыскания в текущем каталоге всех файлов, соответствующих этому критерию.

**Поддержка переменных окружения оболочки**

Оболочка может поддерживать переменные. *Переменные* — это области памяти, в которых можно хранить данные для последующего использования. Значение переменной присваивается с помощью знака равенства (=):

$LOOKUP=/usr/mydir

В этой строке оболочка устанавливает LOOKUP в качестве переменной и присваивает ей значение /usr/mydir. В дальнейшем можно использовать значение, сохраненное в LOOKUP, в командной строке, предваряя имя переменной знаком доллара ($). Рассмотрите следующие примеры:

$ echo $LOOKUP /usr/mydir

$ echo $LOOKUP LOOKUP

Чтобы сделать переменную доступной дочерним процессам, можно использовать команду export, например:

$LOOKUP=/usr/mydir

$export LOOKUP

Примечание. Присвоение значений переменным в оболочке С отличается от выполнения этого в оболочках Bourn и Корн, Для присвоения переменной в оболочке С используется команда set:

% set LOOKUP = /usr/mydir

Замечание. Обратите внимание, что знак равенства с обеих сторон выделяется пробелами.

Подобно подстановке имени файла подстановка имени переменной происходит до вызова программы. Во втором примере знак доллара ($) опущен. Следовательно, оболочка просто передает строку команде **echo** в качестве аргумента. При подстановке имени переменной значение переменной замещает имя переменной.

Например, в строке

$ ls $LOOKUP/filename

программа ls вызывается с единственным аргументом **/usr/mydir/filename**.

**Запуск оболочки — управление средой**

Когда пользователь начинает сеанс работы с UNIX то запускается оболочка (если иное не определено в учетной записи пользователя), оболочка создает для пользователя специфичную рабочую среду, параметры которой сохраняются в файле профиля учетной записи пользователя (**~имя\_пользователя/.profile**).

**Переменные среды оболочки**

Когда программа login вызывает оболочку, она устанавливает переменные среды, которые считываются из файлов инициализации оболочки **/etc/profile** и **.profile.** Обычно эти файлы устанавливают тип терминала в переменной **$TERM** и путь по умолчанию для исполняемых файлов — в переменной **$РАТН.** Посмотрите следующие примеры:

$echo $term

$echo $PATH

Переменные легко можно изменить так же, как присваиваются значения любой переменной оболочки.

Примечание. Оболочка С присваивает значения переменным среды посредством команды setenv:

%setenv TERM = vt100

**Файлы начального запуска оболочки**

Файл **.profile** — локальный файл начального запуска для оболочки Bourne. Оболочка Коrn использует файл **.kshrc**, а оболочка С — файл **.cshrc.** Для манипулирования средой начального запуска эти файлы можно редактировать. При необходимости можно добавлять дополнительные переменные. Можно также добавлять программирование оболочки для получения установок среды, зависящих от различных условий.

**Опции начального запуска оболочки**

При вызове оболочки из **/etc/passwd** или из командной строки, несколько опций можно устанавливать в качестве аргументов программы оболочки. Например, оболочка Bourne имеет опцию -х, которая перед исполнением отображает команды и их аргументы. Это удобно при отладке программы оболочки.

**Заключение.**

Оболочка shell обеспечивает интерфейс между пользователем и основой ОС UNIX — ядром. Shell интерпретирует командные строки в качестве ввода, осуществляет подстановку имен файлов и переменных, переназначает ввод и вывод, отыскивает исполняемые файлы, инициирует и взаимодействует с программами. Shell создает дочерние процессы и может управлять их выполнением. Shell поддерживает переменные среды каждого пользователя. Кроме того, оболочка является мощным языком программирования.

**Задание для выполнения.**

1. Изучите раздел теоретических сведений, выполните все примеры, приведенные в этом разделе. В случае затруднений, используйте справочную систему **man**.

2. Выполнение примеров и упражнений производиться в ВМ Linux Ubuntu и ВМ CentOS.

2.1 Получение доступа к консоли Linux Ubuntu.

Запустить ВМ Linux Ubuntu.

Вызвать окно терминала.

В окне терминала вызовите Midnight Commander с помощью следующей команды:

$mc

Примечание. Если программа mc отсутствует, то ее необходимо установить.

Для Ubuntu:

$ sudo aptget mc

Для CentOS:

$ sudo yum install mc

3. Изучить 50 наиболее часто используемых команд UNIX, приведенных в Приложении 1 к данной лабораторной работе, на примере ОС Linux Ubuntu и FreeBSD.

3.1. Читайте Приложение 1, к настоящей практической работе и выполняйте команды описанные в нем команды. В случае затруднений, используйте справочную систему man.

3.3. Протокол выполнения команд сохраните в качестве отчета.

4. Выполните упражнения, приведенные ниже (заменяет ответы на контрольные вопросы), результаты выполнения упражнений включите в отчет.

**Упражнения:**

1. Объясните, что произойдет в случае ввода команды:

$ date \@

2. Большинство интерпретаторов shell интерпретируют символ # как символ начала комментария и игнорируют весь текст от символа # до конца строки. Учитывая это, объясните приведенный ниже диалог. Предполагается, что для стирания также используется символ #.

$date

Mon Sep 26 12:39:56 EET 2006

$# date

Mon Sep 26 12:39:56 EET 2006

$\#date

$\\#date

#date : not found

$

3. Попробуйте выполнить команду:

$ls /usr/games

а, затем что-либо из предложенного ею. Вы довольны?

Остановите выполнение запущенной программы, одним из известных Вам способов.

Объясните, что выполнила система в ответ на ваши команды.

4. В чем состоит разница между следующими командами:

$ls junk

$ls /

$ls

$ls \*

$ls’\*’

$echo junk

$echo /

$echo

$echo \*

$echo ‘\*’

$‘\*’

5. Объясните, почему команда

$ls > ls.out

Включает ls.out в список имен []?

6. Объясните, результат выполнения команды:

$wc temp > temp

7. Что произойдет, если вы ошибетесь в имени команды, задав

$who > temp

8. Объясните разницу между командами

$who | sort

и

$who > sort

5. Оформите отчет по работе.