**BC/NW 2017 № 2 (31):3.1**

**РАБОТА В КОМАНДНОМ ИНТЕРФЕЙСЕ ПОЛЬЗОВАТЕЛЯ UNIX-СИСТЕМ**

Орлов Д.А., Филатов А.В.

*Изложенный в работе материал имеет учебно-практическое назначение и оформлен в двух частях.(3.1 и 3.2)*

**Часть I.**

Первой целью данной работы (в первой её части) является ознакомление с основными понятиями командного интерфейса пользователя ОС UNIX. Практическая цель - выполнение индивидуальных заданий в рамках сеанса работы в среде ОС UNIX, суть которых состоит в выполнении процедур начальной установки, входа в систему, проверки программного и системного окружения, освоении приемов переназначения ввода/вывода и конвейеризации, использовании сервисных средств ОС UNIX для самообучения.

Второй целью данной работы (также в первой её части) является ознакомление с возможностями, предоставляемыми командным интерфейсом пользователя ОС UNIX, для работы с файловой системой. Практическая цель ― выполнение индивидуальных заданий в рамках сеанса работы в среде ОС UNIX, суть которых состоит в выполнении процедур копирования, перемещения, переименования и удаления файлов, создания, переименования и удаления директорий, использование процедур связи пользователей, а также использовании периферийных устройств.

**1. ОСНОВНЫЕ ПОНЯТИ ОПЕРАЦИОННОЙ СИСТЕМЫ UNIX**

*Командный интерфейс пользователя. Основные сведения*

Для всех пользователей, вот уже скоро сорок пять лет сталкивающихся с ОС UNIX, основной проблемой является отсутствие удобного и естественного, т.н. "дружественного" интерфейса пользователя. В семействах ОС типа UNIX в связи с чрезвычайно широким разнообразием видеотерминального оборудования и по историческим причинам господствует идеология построчного диалога с пользователем. При этом основной принцип построения пользовательского интерфейса - "в ОС типа UNIX: если что-то можно, то это можно везде и в любом количестве". Подобный подход означает достаточность возможностей даже простейших алфавитно-цифровых видеотерминалов для ввода любых UNIX-команд.

Итак, для ввода любой команды ОС UNIX правила командного интерфейса пользователя предоставляют следующий строчный формат:

$ <команда> [<ключи\_вызова (модификаторы)>] [<параметры\_вызова>] <ВВ>

, где

* $ - системное приглашение или «подсказка» (prompt), выводимая в качестве признака готовности ОС UNIX воспринимать ввод команды пользователем;
* <команда> - любая из команд UNIX, либо утилита ОС;
* <ключи-вызова (модификаторы)> - последовательность однобуквенных символов — кодов режимов исполнения команды, префиксируемая символом "-" либо "--", либо "+" (ключи еще иногда называют флагами);
* <параметры\_вызова> - аргументы команды;
* [...] - необязательное присутствие значения, заключенного между [и];
* <...> - нажатие клавиши терминала, обозначаемой "...";
* <ВВ> - нажатие клавиши "ENTER" ("RETURN", "CR/LF", "ввод" или в зависимости от вида терминала.

Признаком завершения реакции системы на введенную команду является появление на экране видеотерминала пользователя системного приглашения.

Для входа в режим взаимодействия с ОС типа UNIX и получения доступа к командному интерфейсу пользователь один раз за сеанс работы должен пройти процедуру регистрации, т.е. сообщения системе своих индивидуальных, присвоенных системным администратором, параметров входа: login – имени и, может быть, пароля (при вводе на экране видеотерминала не отображается). Процедура регистрации может быть проиллюстрирована следующим фрагментом:

login: user1 <ВВ>

[password: <....><ВВ>]

$

где:

* login, password и $ – реакция системы;
* user1 – пример login-имени пользователя;
* <....> – некоторая последовательность нажимаемых клавиш консоли, составляющая пароль.

В конце сеанса пользователю для правомочного завершения работы с системой необходимо отсоединиться от ОС с помощью команд logout, exit, halt либо reboot (если установлены права использования).

После регистрации в системе и получения системного промптера пользователь получает в распоряжение на время сеанса часть файловой системы ОС типа UNIX с собственной "домашней" директорией и копией командной оболочки. Командный интерфейс пользователя подразумевает для каждой команды наличие трех стандартных файлов:

* stdin - файла ввода информации,
* stdout - файла вывода информации,
* stderr - файла вывода сообщений об ошибках (файл диагностики).

По умолчанию в ОС типа UNIX для процесса исполнения UNIX-команды пользователя stdin по умолчанию назначен на клавиатуру, а stdout и stderr — по умолчанию на экран видеотерминала пользователя.

Для переназначения стандартных файлов ввода либо вывода существуют в ОС типа UNIX процедуры переадресации:

* > - переадресация вывода (запись не в stdout по умолчанию)
* < - переадресация ввода (чтение не из stdin по умолчанию)
* >> - переадресация ввода с добавлением.

Кроме переадресации связь между процессами исполнения двух команд (UNIX-процессами в общем смысле) может быть осуществлена в виде т.н. конвейера – подачи данных с выхода одного UNIX-процесса на вход другого с помощью процедуры программного канала. Символом для обозначения программного канала является вертикальная черта │. Так например, в командной строке, вида

$ <команда-1>│<команда-2>

здесь выход процесса (а именно stdout), определяемый командой <команда-1>, является входом (stdin) процесса, определяемого командой <команда-2>.

Командный интерфейс пользователя кроме того предоставляет возможность UNIX-пользователю с помощью команд самого командного интерфейса изучать свойства наиболее употребительных команд этого интерфейса и настраивать, изменять, модифицировать комплекс программно-технических средств реализации этого интерфейса. Наиболее простым средством информационного обеспечения командного интерфейса является команда man, которой в качестве параметра вызова передается имя команды, а для настройки теримнала служит команда stty. В стандартном, наиболее часто встречающемся варианте, пользователь для реализации интерфейса имеет в распоряжении виртуальный терминал, эмулирующий протокол общения с терминалом VT52/VT100, где для редактирования командной строки используются следующие последовательности:

* <ctrl>+<h> - для удаления отдельного символа в командной строке;
* <@> - удаление (смысловое) набранной, но не введенной строки командного интерфеса ("сброс команды");
* <DEL> - завершение зависшего процесса и восстановление системного приглашения.

Для адаптации скорости выводимой процессами на экран видеотерминала информации служат последовательности:

* <ctrl>+<S> - приостановка вывода;
* <ctrl>+<Q> - возобновление вывода.

**2. Основные сведения о процедурах работы с файловой структурой**

В ОС UNIX единицей хранения является файл, причем доктрина, принятая отцами-основателями (Томпсон, Ритчи и примкнувший к ним Керниган), подразумевает, что объектом хранения в файле системы UNIX могут быть тексты, команды, а также каталоги и содержимое спецфайлов. Массив памяти, отведенный системой UNIX под файловое пространство, носит структурированный иерархический характер, где все файлы сгруппированы в директории (каталоги), а директории рекурсивно входят в корневую директорию, являющуюся точкой входа в файловую систему суперпользователя. Корневая директория не является при этом ничьим подкаталогом. Обычный пользователь имеет свою собственную точку входа в файловую систему – «домашнюю директорию». Именование файлов (помните, что с точки зрения ОС UNIX директория – это тоже некоторый файл) определяется именованием пути от какого-либо однозначно определенного в текущий момент каталога, т.е. перечислением в порядке спуска по иерархии файловой системы имен каталогов, где в качестве разделителя используется символ slash – «/», причем корневая директория ОС UNIX – уникальна, существует всегда и однозначно определена на одно и то же место файловой системы для всех пользователей – определяется единственным символом slash, а текущая или рабочая (вспомните, что такое pwd для обозначения команды) директория пользователя – однозначно определены каждая для своего пользователя и может меняться пользователем во время сеанса – для краткости именуется символом «точка» – независимо от реального имени, а директория, для которой текущая директория пользователя является подкаталогом (если таковая существует), именуется удвоенным символом «точка» – «..», такая директория еще может называться «родительским каталогом».

Командный интерфейс ОС UNIX позволяет пользователю выполнять следующие действия в файловой структуре, помимо тех тех, которые уже изучены ранее (см. ls, file, cat, pwd и, конечно же cd):

* переименование или перемещение файла mv;
* копирование простого файла cp;
* удаление простого файла rm;
* создание нового каталога mkdir;
* удаление каталога rmdir;
* изменение атрибутов защиты файла chmod;
* сортировку или слияние файлов sort;
* поиск в файле по шаблону grep;
* построчное сравнение файлов-I diff и comm.

Отличительной особенностью ОС UNIX является то, что пользователь, если он имеет соответствующие права доступа, может получить доступ через рабочую директорию к файлу из любой директории не путем копирования его в рабочий каталог, а путем занесения только ссылки на этот файл в свою рабочую директорию путем исполнения команды связывания («линкования») ln.

Идеология единого структурированного иерархического и допускающего множественность ссылок файлового пространства как файловой структуры позволяет в среде ОС UNIX относительно легко решать как проблему использования периферийных устройств, так и проблему информационного обмена между пользователями. Для выполнения, например, вывода файла (записи) с использованием периферийного устройства, определяемого спецфайлом /dev/dev1 необходимо переназначить стандартный файл вывода на этот спецфайл. Для связи с пользователями можно воспользоваться дополнительно возможностями командного интерфейса ОС UNIX, который позволяет:

* копировать строки («посылать сообщение») с терминала одного пользователя на терминал другого пользователя непосредственно внутри одной файловой системы – write;
* управлять использованием с других терминалов своего терминала в качестве устройства - назначения («мишени») команды write – mesg.

Для построения более сложных коммуникационных процессов, например, для связи пользователей в разных файловых системах ОС UNIX или вообще в разных ОС служит система электронной почты ("E-Mail"). Для работы с реальными сетями электронной почты (например, BITNET, INTERNET) служит стандартизованная в ОС UNIX на уровне применения утилиты mail или elm, работа с которыми строится по принципу экранного меню и потому достаточно проста и не требует высокой квалификации. Для использования утилит электронной почты пользователь должен лишь иметь доступ к какому-либо серверу E-Mail и быть зарегистрированным в нем в качестве абонента.

**Задание к лабораторной работе 1.**

*Домашняя подготовка*

1. Изучить по прилагаемому списку литературы следующие вопросы и темы:

- системные ресурсы пользователя;

- вход в систему, выход из системы;

- рабочая директория, "домашняя" директория;

- Текущая директория:

- терминал пользователя и его параметры.

2. Изучить по прилагаемой литературе функциональные особенности следующих команд командного интерфейса:

- login,

- logout (exit),

- ls,

- pwd,

- who,

- date,

- (time),

- ps,

- cd,

- file,

- cat,

- more,

- man,

- stty.

3. Подготовить ответы по вопросам коллоквиума.

Вопросы коллоквиума

1). Что такое многопользовательская операционная система?

2). Что такое мультизадачная операционная система?

3). Что такое мобильная операционная система?

4). Что такое ядро ОС типа UNIX?

5). Каким образом ОС типа UNIX "различает" пользователей системы?

6). Когда заканчивается регистрация пользователя в системе?

7). Кто устанавливает идентификатор и пароль пользователя в системе?

8). Есть ли различие между понятиями "идентификатор пользователя" и "имя терминала пользователя"?

9). Есть ли различие между понятиями "рабочая директория" и "домашняя директория" для пользователя?

10). Что такое "процесс" в ОС типа UNIX?

11). Какие команды могут информировать пользователя о процессах этого пользователя?

12). Информация, записываемая некоторым процессом в stdout

25 строк текста. Какими средствами, Вам уже известными, можно прочитать эту информацию пользователю?

13). Каким образом (уже известным Вам) можно создать файл нулевой длины?

14). Каким образом можно сохранить информацию о времени входа в систему в виде файла?

15). Каким образом можно сохранить информацию о имени терминала пользователя в виде файла?

16). Каким образом (возможно, несколькими способами) можно в одном файле сохранить информацию о дате входа в систему, текущем пути к директории пользователя и дате, непосредственно предшествующей выходу из системы?

17). Каким образом узнать соответствие идентификаторов активных пользователей и имен терминалов?

18). Что такое блокориентированный специальный файл?

19). Что такое байториентированный специальный файл?

20). Как определить наличие в директории обычных файлов?

21). Как определить наличие в директории блокориентированных специальных файлов?

22). Как определить наличие в директории байториентированных специальных файлов?

23). Как определить наличие в директории поддиректории?

24). Что такое группа пользователей в системе UNIX?

25). Для чего служат специальные файлы в ОС типа UNIX?

26). Как определить, является ли какой либо файл в директории пользователя ОС типа UNIX исполняемым непосредственно?

27). С помощью какой из команд ОС типа UNIX можно определить наличие среди файлов пользователя stdin, stdout и stderr?

28). Какие категории пользователей в ОС типа UNIX Вы можете назвать?

29). Могут ли быть в ОС типа UNIX два и более пользователя с одинаковыми именами?

30). Могут ли существовать одновременно в ОС типа UNIX два и более процесса с одинаковыми именами пользователя и разными идентификаторами процесса?

31). Могут ли существовать одновременно в ОС типа UNIX два и более процесса с одинаковыми идентификаторами процесса и разными именами пользователя?

32). Из скольких полей должно состоять имя файла в ОС типа UNIX?

33). Из скольких полей может состоять имя файла в ОС типа UNIX?

34). К чему может привести применение команды cat к исполняемому файлу?

4. Просмотрите лабораторное задание, соответствующее Вашему варианту и обдумайте его выполнение.

*Лабораторное задание*

1. Перед выполнением лабораторного задания получите у администратора системы (преподавателя) регистрационные имя пользователя и пароль.

2. Зарегистрируйтесь в системе с терминала.

3. Определите следующие параметры среды:

- текущее время;

- имя Вашего терминала;

- место Вашей рабочей ("домашней") директории;

- имена и соответствующие им терминалы всех активных пользователей;

- номера всех доступных для Вашего наблюдения активных процессов;

- параметры Вашего терминала.

4). Создайте пустой файл (файл, не содержащий никакой информации) в Вашей директории по имени first.snap;

5). В соответствии с номером в журнале учебной группы выполните пункт задания (см. таблицу 1).

**Таблица 1**

|  |  |
| --- | --- |
| № варианта | Содержание пункта задания |
|  | Создайте в Вашей "домашней" директории файл contens.of.partent и запишите в него имя и листинг каталога, подкаталогом которого является Ваша "домашняя" директория |
|  | Создайте в Вашей "домашней" директории файл contens.of.par.sort содержащий отсортированный по времени создания и кроме того еще отсортированный по времени последнего доступа листинг каталога, подкаталогом которого является Ваша "домашняя" директория |
|  | Создайте в Вашей директории файл time.my, содержащий текущее значение даты и времени, причем во второй строке файла зафиксируйте десятичный номер текущей недели в году, в третьей строке - десятичный номер года вместе с номером столетия, в четвертой - информация о часовом поясе |
|  | Создайте в Вашей директории файл time.our, содержащий кроме текущего значения даты и времени национальное представление полного названия дня недели, его десятичного номера и полного названия месяца. |
|  | Создайте в Вашей директории файл users.of.session имена и номера терминалов всех активных пользователей, текущее значение системного времени и имя Вашего терминала |
|  | Создайте в Вашей директории файл process.of.session содержащий список всех выполняющихся процессов, текущее значение системного времени и имя Вашего терминала |
|  | Создайте в Вашей директории файл my.own, содержащий ее имя, содержимое и имя Вашего терминала вместе с регистрационным именем |
|  | Создайте в Вашей директории файл partent.of.my, содержащий листинг каталога, подкаталогом которого является Ваша директория, а также информацию о текущем значении системного времени. |
|  | Создайте в Вашей директории файл all.of.my.par, содержащий рекурсивный вывод содержимого подкаталогов каталога, подкаталогом которого является Ваша директория |
|  | Создайте в Вашей директории файл users.and.time, создающий имена и номера терминалов всех активных пользователей, имя Вашей директории и имя каталога, подкаталогом которой является Ваша директория |
|  | Создайте в Вашей директории файл cleven.example, содержащий список всех выполняющихся процессов, имя Вашей директории и имя каталога, подкаталогом которой является Ваша директория |
|  | Создайте в Вашей директории файл zwoelf.exam, содержащий ее листинг, информацию о собственно директории а также ее листинг с указанием размера файлов директории в блоках |
|  | Создайте в Вашей директории файл learn.no1, содержащий время начала получения листинга текущей директории, сам листинг и время окончания выдачи листинга текущей директории (времена – с точностью до ввода кода командной строки). |
|  | Создайте в Вашей директории файл example.sample, содержащий время начала получения информации обо всех активных пользователях в системе, саму эту информацию и время окончания выдачи этой информации (время - приблизительное) |
|  | Создайте в Вашей директории файл process.es, содержащий (приблизительно) время начала получения информации о всех активных процессах в системе, саму эту информацию и (приблизительно) время окончания выдачи этой информации |
|  | Создайте в Вашей директории файл column.partent.ls, содержащий листинг каталога, подкаталогом которого является Ваша директория, сформированный в несколько колонок, а также имя этого каталога. |

6). Убедителсь в том, что Ваша директория непуста. Создайте файл second.type, содержащий информацию о типах всех файлов Вашей директории, включая и файл second.type.

7). Вызовите команду man для просмотра информации о командах:

login,logout (exit, halt), ls, pwd, who, date (time), ps, cd, file, cat, more, man, stty. Отметьте в создаваемом Вами файле third.exeption имена команд, по которым отсутствует информация при обращении к команде man. Не смущайтесь тем, что сообщения команды man не на русском языке - это язык, хотя и напоминающий английский, однако максимально адаптированный к международному жаргону программистов и потому должен быть понятен Вам.

8). Повторите пункт 3 лабораторного задания, сохраняя всю необходимую информацию в файле first.snap и используя в качестве разделителя данных о действующих параметрах последовательность символов типа "-------------...----" (или иных, по Вашему желанию). Отметьте совпадения и несовпадения при выполнении команд пункта 3 и содержимого файла first.snap.

9). Попробуйте добиться того, чтобы вывод на Ваш терминал осуществлялся темными символами на светлом фоне.

10). Создайте в Вашей домашней директории поддиректорию Mysecondir;

11). Скопируйте Ваш файл second.type в директорию Mysecondir;

12). Переместите Ваш файл first.snap в директорию Mysecondir с переименованием его в second.snap;

13). Установите для файла second.snap права доступа с абсолютным значением атрибута 777;

14). Установите для директории права доступа внешнего пользователя такие, чтобы он мог копировать из Вашей директории;

15). Создайте из Mysecondir ссылку на файл first.snap пользователя, определяемого по списку Вашей группы по формуле (n+1)mod(k), где n - Ваш номер в списке, k - число студентов в Вашей группе. Что Вам мешает?

16). Создайте директорию Mysecondir в Вашей домашней директории и создайте в ней ссылку на директорию Mysecondir пользователя с номером (n+2)mod(k). Что Вам мешает?

17). Выполните пункт задания в соответствии с Вашим номером в журнале учебной группы (см. таблицу 2)

**Таблица 2.**

|  |  |
| --- | --- |
|  | Содержание пункта задания |
|  | Выполните пункт 5).3. задания к лабораторной работе. Выполните сравнение на несовпадение с выводом строк, отсутствующих в Вашем time.my по сравнению с time.my пользователя с номером 3. Результат поместить в файл differ в Mysecondir |
|  | Выполните пункт 5).4. задания к лабораторной работе 1. Выполните сравнение на несовпаднеие с выводом строк, отсутствующих в чужом файле time.our и файла time.our пользователя с номером 4. Результат поместить в файл differ в Mysecondir |
|  | Выполните пункт 5).5. задания к лабораторной работе 1. Выполните сравнение на совпадение с файлом user.of.session пользователя с номером 5. с выводом общих строк в файл differ в Mythirdir. |
|  | Выполните пункт 5).6. задания к лабораторной работе 1. выполните построчное сравнение с файлом process.of.session пользователя с номером 6 по полному формату в файл compare в Mythirdir. |
|  | Выполните два раза (бесконфликтно, например, в разных директориях) Ваше задание пункта 5) лабораторной работы 1 файлов users.of.session. Подсчитайте количество строк среди строк совпадения, содержащих информацию о Вашем терминале, занося эту информацию в файл compare в Mythirdir. (Попробуйте все это сделать в одной командной строке). |
|  | Выполните два раза (бесконфликтно, например, в разных директориях) Ваше задание пункта 5) лабораторной работы 1. Подсчитайте количество слов среди строк несовпадения файлов process.of.session, содержащих информацию о Вашем терминале, занося эту информацию в файл compare в Mythirdir. (Попробуйте все это выполнить в одной командной строке). |
|  | Выполните два раза (бесконфликтно) Ваше задание пункта 5) лабораторной работы 1. Подсчитайте количество символов среди файлов my.own, содержащих имя Вашего терминала, занося эту информацию в файл compare в Mythirdir. (Попробуйте все это выполнить в одной командной строке). |
|  | Выполните пункт 5).1. лабораторной работы 1. Выполните сравнение сначала Вашей домашней директории с директорией пользователя с номером 1, потом - наоборот, занося получаемую информацию в файл compare в Mythird. |
|  | Выполните пункт 5).2. лабораторной работы 1. Выполните сравнение сначала Вашей домашней директории с директорией пользователя с номером 2, потом - наоборот, занося получаемую информацию в файл compare в Mythird. |
|  | Создайте в Mythird файл Sorted, содержащий в первой своей части построчную информацию о единицах хранения в корневой директории, отсортированную по именам файлов, во второй части - ту же информацию, но отсортированную по размерам файлов и в третьей части - отсортированную по правилам доступа, подсчитайте число строк в этом файле. |
|  | Создайте в Mythird файл Sorted, содержащий в первой своей части построчную информацию о всех активных пользователях, отсортированную по именам пользователей, во второй части - ту же информацию, но отсортированную по именам терминалов, в третьей- отсортированную по строкам входа в систему, подсчитайте число строк в этом файле. |
|  | Создайте в Mythird файл Sorted, содержащий в первой своей части построчную информацию о всех активных процессах, отсортированную по именам пользователей, во второй части - ту же информацию, отсортированную в соответствии с номером процесса - по именам терминалов, в третьей - отсортированную по времени инициализации процесса, подсчитайте число строк в файле Sorted. |
|  | Создайте в Мythird ссылки на все Ваши существующие файлы, а также файл Sorted, содержащий в каждой своей строке полную информацию(т.е. права доступа, размер, срок создания, владелец, имя и т.д.) о всех Ваших файлах, включая информацию о возможном содержимом файла, отсортированный по последнему. |
|  | Выполните дважды (бесконфликтно, например, в разных директориях) Ваше задание пункта 5). лабораторной работы 1. Из первого из полученных файлов получите строки, содержащие текст ttyAnn, где А- некоторая (любая) буква латиницы, nn - десятичный номер, а из второго строки, не содержащие этот текст. Подсчитайте количество строк в первом и во втором случае, записав эту информацию в файл example.2 в Мythirdir. |
|  | Выполните дважды (бесконфликтно, например, в разных директориях) Ваше задание пункта 5). лабораторной работы 1. Из первого из полученных файлов получите строки, содержащие информацию о процессах с номерами от 100 до 999, из второго - строки, не соответствующие терминалам пользователей. Полученную информацию запишите в файл process.2 с подсчетом числа строк в нем самом. |
|  | Выполните пункт 5).9. лабораторной работы 1. Выполните построчное сравнение Вашего файла all.of.my.par с одноименным файлом пользователя с номером 9 и совпадающие пронумерованные строки запишите в файл differ в Мythirdir. |

18). Вызовите команду man для просмотра информации о командах mv, cp, rm, mkdir, chmod, sort, grep (Grep, egrep, fgrep), diff, comm (cmp). Добавьте в Ваш файл therd.exeption имена команд, по которым отсутствует информация при обращении к команде man. Не правда ли, "английский", на котором написаны разделы, читаемые man, ненамного сложнее диалекта Э. Щукиной и потому должен быть понятен Вам.

19). Включите Ваш терминал для приема сообщений от других пользователей, использующих команду write.

20). Перешлите пользователю с номером (n+3) mod (k) сообщение, типа "Hello, I am a user from ttyxx".

21). Выберите любой из разделов команды man пользователю с номером (n+4) mod (k) с припиской в конце о том, кто Вы с точки зрения ОС UNIX. Продемонстрируйте этот пункт работы преподавателю.

22). Завершите сеанс работы с ОС.

*Контрольные вопросы:*

1). Могут ли одновременно в ОС типа UNIX существовать два пользователя с одним и тем же именем, но работающие на разных терминалах?

2). Могут ли одновременно в ОС типа UNIX существовать два обычных разноименных пользователя, запускающие свои задания с одного и того же терминала?

3). Могут ли в одной и той же директории работать два различных пользователя одновременно?

4). Может ли один и тот же пользователь одновременно выполнять более одного процесса и при этом каждый процесс в своей директории?

5). Как можно выяснить связь между процессом с известным идентификатором и командой или программой, соответствующей этому процессу?

6). Какие файлы позволяют бесконфликтно применять к себе команду cat?

7). Какие команды из изученных в данной лабораторной работе возможно объединить через конвейер?

8). Какие команды из изученных в данной лабораторной работе допускают переназначение ввода и каким образом?

9). Какие команды из изученных в данной лабораторной работе допускают переназначение вывода и каким образом?

10). Какие команды из изученных в данной лабораторной работе допускают переназначение диагностики и каким образом?

11). Какие команды из изученных в данной лабораторной работе в результате верного вызова формируют ненулевой stderr и нулевой stdout?

12). Каким образом можно установить последовательность регистраций всех активных пользователей?

13). Может ли в ОС типа UNIX одновременно исполняться одинаковые процессы с одинаковыми именами у разных пользователей?

14). Может ли в ОС типа UNIX пользователь, перемещаясь по иерархической файловой структуре, попадать в директории, принадлежащие другим пользователям?

15). Какой последовательностью может считаться конец файла в ОС типа UNIX?

16). Поддиректорией какой директории является Ваша рабочая директория и поддиректорией какой директории является директория, поддиректорией которой является Ваша рабочая директория.

17). К чему может привести выполнение команды, удаляющей файл, на который другими пользователями установлена ссылка?

18). Есть ли различие в используемом объеме памяти при копировании файла в директорию и "линкования" его туда же?

19). Может ли домашняя директория содержать в качестве поддиректории корневую директорию?

20). Попробуйте написать команду перехода в директорию, одной из поддитекторий которой является директория, одной из поддиректорий которой является директория, являющаяся рабочей для этой команды?

21). К чему могло бы привести выполнение команды rmdir ..?

22). К чему могло бы привести выполнение команды mv . ..?

23). К чему могло бы привести выполнение команды chmod u-rwx . ?

24). К чему приведет выполнение команды:

- mkdir dire?tory ?

- mkdir dire\*tory ?

- mkdir dire[ckCK]tory ?

25). При каких необходимых и достаточных условиях пользователь в рабочей директории может исполнить команду mkdir и команду rkdir?

26). Что делает (и делает ли) команда

grep Los Angeles>file

и что делает (и делает ли) команда

grep 'Los Angeles'<file?

27). Может ли директория иметь то же имя, что и директория, поддиректорией которой она является? То же имя, что и директория, в поддиректории которой она находится?

28). Может ли директория иметь несколько родительских каталогов? Несколько установленных ссылок на каталоги более высоких уровней?

29). Объясните значение полей строки команды ls -la для файла /dev/dev1.

30). Как получить распечатку файла /usr/mydir/myfile.text?

31). Что происходит при выполнении команды write user1 tty5 < file.text и что происходит при выполнении команды cat file.text > /dev/tty5 ?

32). Как узнать, разрешено ли в данный момент поступление на Ваш терминал сообщений от других пользователей?

**Часть II.**

Целью данной части является ознакомление с основными возможностями, предоставляемыми языком программирования командных последовательностей shell. Основная практическая цель – выполнение индивидуальных заданий в рамках сеанса работы с ОС UNIX, смысл которых состоит в выполнении процессов пользователя в фоновом режиме, в перенастроенной среде, в выполнении принудительного завершения процессов пользователя, а также в создании .profile- файла и файлов командных последовательностей.

**Командный интерпретатор shell.**

Командный интерпретатор shell не входит в ядро операционной системы UNIX, что позволяет пользователю во-первых, выбирать такой командный интерпретатор, который в наибольшей степени удовлетворял бы запросам пользователя, а во-вторых, создавать свои собственные программы на языке shell. С точки зрения операционной системы командный интерпретатор shell обеспечивает удобный интерфейс между пользователем и ядром с помощью выполнения вводимых с терминала команд. Кроме того, командный интерпретатор shell (чаще всего используется bourne shell) – предоставляет пользователю полноценный язык программирования с возможностью введения переменных, алгоритмических структур и подпрограмм (содержащих командные файлы в том числе), обеспечением возможности передачи параметров и обработки прерываний.

Программы (или скрипты), написанные на языке shell, представляют из себя последовательность командных строк, предназначенных для обработки интерпретатором языка shell. Сам интерпретатор обычно расположен в директории /bin под именем sh.

Командная строка в языке shell – это последовательность слов, разделенная пробелами, причем всегда первое слово определяет имя команды, утилиты или shell-программы, которые будут выполнены, а оставшиеся слова, как правило, передаются команде в качестве аргументов. Файл, содержащий программу, определяемую первым словом строки, может храниться как в системном каталоге, так и в личных каталогах и быть продуктом деятельности пользователя. Таким образом, набор команд языка shell является расширяемым – его пополняет каждая программа, написанная пользователем, и даже ссылка на уже существующий файл. Для процедур shell остаются таким образом в силе понятия «стандартный ввод», «стандартный вывод», «стандартный протокол» и «конвейер». Кроме того, мощь языку shell придаёт возможность использования переменных и аргументов. Переменные в языке shell обозначаются идентификаторами, а определяются строками типа:

var=value;

export var;

где var – соответствующий текстовому значению value идентификатор. Доступ к переменной (к её значению) осуществляется по выполнению по отношению к идентификатору префиксной операции $. Строка, являющаяся значением (value) некоторой переменной языка shell, не должна содержать символы переопределения ввода/вывода, фонового процесса и канала. Предопределенные (служебные) переменные shell состоят из двух подмножеств: аргументов процедур и внутренних переменных идентификатора, некоторые из которых можно только читать, а другие – ещё и модифицировать.

К внутренним shell-переменным относятся такие, например, как:

**HOME** - содержит имя "домашнего" каталога пользователя.

**PATH** - множество каталогов, в которых будет проходить поиск команды. Разделитель – «:».

**PS1** и / или **PS2** - первичная и вторичная подсказки системы ("промптеры").

**TERM** - содержит мнемоническое имя типа терминала пользователя.

**MAIL** - содержит полное имя файла почтового ящика

пользователя.

Аргументы – это особый тип переменных, именами которых служат цифры. Аргументы – это по сути своей позиционные параметры командной строки, причем имя команды – это нулевой аргумент. Значения аргументов обозначают как **$d**,

где **d** - любая десятичная цифра. Поскольку в некоторых случаях десяти переменных может оказаться недостаточно, то в языке shell введена команда shift, сдвигающая аргументы на одну позицию влево за исключением **$0**, который остается без изменений.

Значения некоторых внутренних переменных (так называемых "зарезервированных") устанавливаются самим shell'ом, а не пользователем:

**#** - количество фактических аргументов – позиционных параметров, кроме **$0**,

**?** - код возврата,

**$** - значение PID (см. лабораторную работу 1) текущего процесса,

**!** - значение PID фонового процесса, вызванного последним когда-либо,

**-** - состояние флагов интерпретатора shell,

**\*** - совокупность всех позиционных параметров, начиная с **$1**.

Для выполнения арифметических действий используется команда expr, помещающая результат в стандартный вывод.

Знаки операций команды expr:

**+** - для сложения,

**-** - для вычитания,

**'\*'** или **"\*"** - для умножения,

**/** - для деления,

**%** - для взятия остатка от деления.

Язык shell кроме того, как всякий универсальный язык программирования предоставляет ряд конструкций для управления последовательностью действий:

- оператор цикла for;

- условный оператор if;

- команда test для вычисления условных выражений;

- оператор цикла while;

- оператор цикла until;

- оператор выбора case;

- операторы выдачи сообщений echo и ewal.

Наиболее часто используются операторы цикла и условный оператор, при этом формат оператора цикла for можно представить в виде:

**for** <имя**>** [**in** <слово>...]

**do**

<командная последовательность>

**done**

где переменная <имя> последовательно принимает значения из набора <слово>, или, в случае отсутствия конструкции in <слово>, соответствует значению формального аргумента; do... done - тело цикла; <командная последовательность> - любая последовательность команд shell'a.

Построение циклов возможно и с помощью следующих форматов:

**while** <условие>

**do**

<командная последовательность>

**done**

где <условие> - некоторая переменная, константа или выражение, вырабатывающее код возврата. При значении кода возврата <условие> равного нулю будет выполняться тело цикла.

**until** <условие>

**do**

<командная последовательность>

**done**

В этом формате тело цикла исполняется при ненулевых значениях кода возврата <условие>.

Условный оператор имеет несколько форматов, при этом одним из наиболее употребительных является следующий:

**if** <условие>

**then**

<командная последовательность 1>

[**else**

<командная последовательность 2>]

**fi**

где <командная последовательность 1> выполняется в случае выработки в <условие> кода возврата, равного нулю, после чего выполняется, если она есть, команда, непосредственно следующая за ограничителем fi; <командная последовательность 2> выполняется сразу после выработки в <условие> ненулевого кода возврата.

Язык командного интерпретатора shell дает большие возможности пользователю по созданию гибкого программного окружения. Другими распространенными версиями shell являются т.н. C-shell (автор - У.Джой) и Korn-shell (автор - Д. Корн), каждый из которых при идейной близости к bourne-shell обладает отличительными особенностями только в пла не улучшения сервисных служб (введение буфера диалогового режима, массива переменных и т.д.), в связи с чем изучение языков командного интерпретатора в ОС UNIX происходит, как правило, на примере Bourne-shell'a.

**Задания к лабораторной работе**

*Домашняя подготовка*

1. Выбрать для изучения вариант языка командного интерпретатора Bourne-shell.

2. Изучить по прилагаемому списку литературы следующие вопросы и темы: файлы инициализации; переменные и аргументы; операторы и процедуры; построение условных операторов, циклов; выполнение проверок, сдвигов, ветвления, вычислений; замены, подготовки и переназначения.

3. Для изучивших Bourne-shell ответить на следующие вопросы коллоквиума (изучающие С-shell отвечают на те же вопросы, но заменяют в тексте вопроса "Bourne-shell" на "С-shell"):

1) Как определен в Bourne-shell файл начала сеанса?

2) Каким образом строится процедура в Bourne-shell?

3) Каков порядок поиска по каталогам ОС типа UNIX при использовании bourne-shell?

4) Как называются командные файлы в bourne-shell?

5) Каким образом устанавливается в bourne-shell первичная подсказка-запрос системы?

6) Каким образом устанавливаются каталоги обязательного поиска команд в bourne-shell?

7) Каким образом устанавливается средствами bourne-shell тип и характеристики терминала пользователя?

8) Каким образом устанавливается в bourne-shell вторичная (собственно bourne-shell'овская) подсказка-промптер?

9) Какие типы переменных и аргументов доступны в bourne-shell?

10) Какие возможности группирования команд Вам известны в Bourne-shell, каким образом они определены для интерпретатора?

11) Что такое shell-файл в Bourne-shell?

12) Каким образом выполнить Shell-файл в Bourne-shell?

13) Каким образом осуществляется вывод любого готового текста на экран пользователя Bourne-shell'a?

14) Каким образом может быть осуществлена проверка файлов в Bourne-shell?

15) Каким образом может быть произведено построение условных операторов в Bourne-shell?

16) Каким образом получить в Bourne-shell аргумент с номером i?

17) Каким образом получить в Bourne-shell идентификационные номера процессов и каких процессов?

18) Каким образом присваивается переменная в Bourne-shell?

19) Каким образом отображается (возвращается) в Bourne-shell значение переменной?

20) Можно ли присваивать переменным Bourne-shell'a текстовые значения, содержащие разделители (например, пробелы)?

21) Каким образом происходит вычисление арифметических выражений в Bourne-shell?

22) Каким образом Bourne-shell'у доступно число аргументов команды?

23) Каким образом может быть произведено построение операторов цикла в Bourne-shell?

24) Возможно ли и каким образом построение вложенных циклов в Bourne-shell?

25) Возможно ли принудительное завершение цикла в Bourne-shell?

26) Возможно ли возобновление выполнения цикла в Bourne-shell?

27) Возможно ли принудительное завершение процедуры Bourne-shell'a?

28) Каким образом можно построить многоальтернативный выбор в Bourne-shell?

29) Каким образом получить в Bourne-shell доступ к аргументам некоторой команды, если число таких аргументов более 10?

30) Каким образом в Bourne-shell производится обработка ситуаций, вызывающих выработку сигналов прерываний?

4. Просмотрите лабораторное задание, соответствующее вашему варианту, продумайте его выполнение.

*Лабораторное задание*

1) Перед выполнением лабораторного задания :

- ответьте на вопросы коллоквиума, указанные преподавателем;

- получите у администратора системы или преподавателя право на использование терминала;

- зарегистрируйтесь в системе.

2) Создайте в вашем домашнем каталоге подкаталог lab4, перейдите в него.

3) Сохранить в виде ascii- файла F1.var значение внутренних переменных shell и номера процессов в системе.

4) Переопределить значения переменных PS1, PS2, сохранить в ascii- файле F2.var значения внутренних переменных.

5) Вернуться к исходному значению внутренних переменныхshell.

6) В каталоге lab4 получить копию файла

/ete/skel/local.profile

с именем .profile, определить в нем значения переменных DIGIT\_FOR\_LAB, MYOWNNAME, TIMEZONE, MYGROUP, DAY\_OF\_WEEK\_TODAY, а также измените на действующие значения внутренних переменных интерпретатора PATH и HOME.

Стартуйте модифицированную таким образом командную оболочку. Убедитесь в результате.

7) Напишите программу на языке shell - "скрипт", которая по вводимой вами произвольной последовательности десятичных чисел осуществляла бы вычисление промежуточных сумм и выводила конечный результат на экран и в файл sh.result. Используйте в программе значение переменной интерпретатора S#. Если номер Вашей бригады четный - постарайтесь обойтись оператором for, если нечетный - конструкцией if.

8) Напишите скрипт, который выводил бы через каждые десять секунд на экран некоторое сообщение только в том случае, если наступит определенное время в определенный день. Режим запуска - фоновый. Уточнение задания - см. табл.4.1

9) Напишите скрипт, который проверял бы наличие процесса, связанного со скриптом по пункту номер 8) и останавливал бы его, если количество сообщений, выведенных на экран скриптом по пункту номер 8) превысит некоторое число. Уточнение задания - см. табл.4.2.

10) Напишите скрипт без использования диалога с оператором на языке shell, который через несколько минут (5-10) закроет все процессы, связанные с пунктами 7 - 9. Если номер Вашей бригады четный - постарайтесь обойтись конструкцией if, иначе - конструкцией for. Примечание: перед запуском скрипта покажите его текст преподавателю и только после получения разрешения - запустите на исполнение. Подсказка к выполнению: кое-о-чем Вы позаботились в начале лабораторного задания; это "кое-что" позволит Вам выполнить задание.

11) После запуска скрипта п.10 вернитесь в оболочку shell которая была в Вашем распоряжении до выполнения п.6).

12) Завершите сеанс работы с ОС через минуту запуска некоторого скрипта.

**Таблица 4.1**

|  |  |
| --- | --- |
| № бригады | Уточнение задания к п.8) |
|  | Время срабатывания скрипта "запомнить" в переменных DAY\_OF\_WEEK\_TODAY, TIMEZONE |
|  | Аргумент при запуске скрипта – значения для переустановки PS1 |
|  | После каждого вывода на экран сообщения значение переменной MYOWNNAME «удлинялось» бы на один символ из последовательности: A,B,C,D,E,F,.... |
|  | При выводе сообщения скрипт создает новую переменную MESSAGEn (n=1,...,N), значение которой состоит из номера сообщения и его текста |
|  | Сообщение представляет собой следующий текст:  - строка заголовка,  - порядковый номер текущего сообщения,  - номер PID скрипта и его PPID,  - время,  - имя процесса из активных в этот момент с его PID. |
|  | Сообщение представляет собой следующий текст:  - строка заголовка  - \*\*\*\*  - " "x" , где х - значение MYOWNNAME  - ' 'y' , где у - значение DAY\_OF\_WEEK\_TODAY  - $$$z$# , где z - значение MYGROUP |
|  | Сообщение представляет собой следующий текст:  - строка заголовка  - текущее системное время  - "звонок"  - список всех пользователей, у которых активны про-  цессы с PID>=PID скрипта по п.8) |
|  | Запуск скрипта выполните в последовательности с  командой date, строка сообщения должна включать в  себя количество секунд, прошедших с начала запуска  скрипта. |

**Таблица 4.2**

|  |  |
| --- | --- |
| № | Уточнение задания к п.9) |
| 1 - 8 | Необходимое число повторов - в переменной DIGIT\_FOR\_LAB |
| 1,3,5,7 | Значение DIGIT\_FOR\_LAB установить в диалоговом режиме после старта скрипта п.9). |
| 2,4,6,8 | Значение DIGIT\_FOR\_LAB установить на N больше, чем количество повторов по скрипту п.8), установка N - в диалоговом режиме. |
| 1,2,3,4 | Значение DIGIT\_FOR\_LAB не может быть отрицательным, нулевым и больше 5. |
| 5,6,7,8 | Значение DIGIT\_FOR\_LAB может быть только одним из следующих: 2,3,5,7,10,100,120. |
| 1 - 8 | Первое сообщение после старта скрипта по п.9) - вывод значения текущего числа повторов сообщений, выведенных скриптом по п.8). |