Лабораторная работа
Изучение основных команд для работы с файлами.

1 Общие сведения

UNIX — многопользовательская, многозадачная операционная система с разделением времени. В любой момент в системе выполняется множество процессов, каждый процесс принадлежит некоторому пользователю. Пользователь это объект обладающий определенными правами в системе. Каждый пользователь идентифицируется уникальным идентификатором пользователя (UID — user identifier). Пользователю присваиваются имя и пароль. Пользователь с UID 0 (root) обладает неограниченными правами. Кроме того каждый пользователь входит в одну или несколько групп. Принадлежность к группе добавляет пользователю определенные права в системе. Каждая группа идентифицируется уникальным идентификатором группы (GID — group identifier). Информация о пользователях хранится в файле /etc/passwd. Каждая строка файла содержит информацию об одном пользователе: регистрационное имя, зашифрованный пароль[1] , UID, GID, полное имя, домашний каталог, командную оболочку. Командная оболочка (командный интерпретатор, shell) — средство интерактивного взаимодействия с системой. Домашний каталог — каталог в котором хранятся файлы пользователя. При входе пользователя в систему этот каталог становится текущим для оболочки.

2 Файловая система

Файловая система — это структура, с помощью которой ядро операционной системы организует и представляет пользователям ресурсы памяти системы. Сюда относится память на различного рода носителях информации. Емкость и количество носителей различно в разных системах. Ядро объединяет эти ресурсы в единую иерархическую структуру, которая начинается в каталоге / и разветвляется, охватывая произвольное число подкаталогов.

Цепочка имен каталогов, через которые необходимо пройти для доступа к заданному файлу, вместе с именем этого файла называется *путевым именем файла (pathname)*. Путевые имена могут быть полными или относительными. В любой момент каждый процесс привязан к некоторому *текущему каталогу*. Относительные имена интерпретируются с текущего каталога.

Файловое дерево может быть произвольного размера. Однако существуют определенные ограничения зависящие от конкретной операционной системы. Как правило имя каталога не должно содержать более 256 символов, а в определении одного пути не должно быть более 1023 символов.

В ОС UNIX существует восемь типов файлов:

Обычный файл
: — это просто последовательность байтов. Обычный файл может содержать выполняемую программу, главу книги, графическое изображение и т.п.

Каталоги
: могут содержать файлы любых типов в любых сочетаниях. Специальные имена `.` и `..` обозначают соответственно сам каталог и его родительский каталог.

Файлы устройств
: позволяют программам взаимодействовать с аппаратными средствами и периферийными устройствами системы. При конфигурировании ядра к нему добавляются те модули, которые знают, как взаимодействовать с каждым из устройств системы. За всю работу по управлению конкретным устройством отвечает специальная программа, называемая *драйвером устройства*.

  Драйверы устройств образуют стандартный коммуникационный интерфейс, который выглядит как обычный файл. Когда ядро получает запрос к байт-ориентированному или блок-ориентированному файлу устройства, оно просто передает этот запрос соответствующему драйверу устройства.

[1] В настоящее время это поле не используется, пароль хранится в другом месте

Каждому типу устройств системы может соответствовать несколько файлов устройств. Поэтому файлы устройств характеризуются двумя номерами: старшим и младшим.*Старший определяет драйвер, а младший конкретное устройство.*

Доменные гнезда (sokets) UNIX

— это соединения между процессами, которые позволяют им взаимодействовать, не подвергаясь влиянию других процессов. Доменные гнезда UNIX локальны для конкретного хост-компьютера. Обращение к ним осуществляется через объект файловой системы, а не через сетевой порт.

Именованные каналы

, также как и доменные гнезда обеспечивают взаимодействие двух несвязанных процессов, выполняемых на одной машине.

Жесткие ссылки

— это скорее не тип файла, а его дополнительное имя. У каждого файла имеется как минимум одна ссылка. Как правило, это имя, под которым он был создан. Добавлением ссылки создается псевдоним файла.

Ссылку невозможно отличить от имени файла, к которому она присоединена: в ОС UNIX они идентичны. UNIX подсчитывает количество ссылок, указывающих на каждый файл, и не освобождает блоки данных файла до тех пор, пока не удалит его последнюю ссылку.

Символические ссылки

— обеспечивают возможность указывать вместо путевого имени файла имя ссылки. Символическая ссылка содержит путевое имя файла, на который она ссылается.

Имена файлов могут состоять из любых символов, за исключением слэша и символа с кодом ноль. Максимальная длина имени файла определяется конкретной системой. Для каждого файла определен владелец этого файла и группа владелец данного файла. Для каждого файла определяются права доступа владельца файла, группы, всех остальных. Есть три типа прав доступа: чтение, запись, выполнение/поиск. Изменить права доступа к файлу может только владелец и суперпользователь (root).

3 Перенаправление ввода и вывода

Если некоторый процесс намерен производить ввод или вывод информации в файл, то он должен сначала открыть этот файл. При открытии файла процесс получает дескриптор файла — некоторое число, которое используется, в дальнейшем для обращения к файлу. При запуске процесса ему передаются дескрипторы трех открытых файлов: 0 – стандартный ввод, 1 – стандартный вывод, 2 – стандартный вывод ошибок. Как правило все эти дескрипторы указывают на терминал – tty. Оболочка позволяет назначать другие файлы для ввода и вывода при помощи команд перенаправления:

команда < *файл*

При запуске *команды* дескриптор 0 будет связан с *файлом*, т.е. программа будет считывать данные не с клавиатуры, а из *файла*. Файл будет открыт для чтения.

команда > *файл*

При запуске *команды* дескриптор 1 будет связан с *файлом*, т.е. программа будет выводить результаты работы не на экран, а в заданный *файл*. Файл будет открыт для записи, если файл существовал, он будет очищен, если нет, то он будет создан.

команда >> *файл*

При запуске *команды* дескриптор 1 будет связан с указанным *файлом*, как и в предыдущем случае. Однако в данном случае, если файл существовал, то он не будет перезаписан, данные будут добавляться в конец файла.

команда `n>` *файл*

При запуске *команды* дескриптор с номером n будет связан с указанным *файлом*. Например, если указать 2>err.log, то вывод сообщений об ошибках будет производиться в файл *err.log*. Аналогично, можно указывать дескриптор перед операторами перенаправления > и >>.

команда `n<>` *файл*

При запуске *команды* дескриптор с номером n будет связан с указанным *файлом*. Файл будет открыт для чтения и записи.

При перенаправлении можно вместо имени файла указывать дескриптор, для этого следует поставить перед дескриптором знак &. Например: `2>&1` скопирует содержимое дескриптора 1 в дескриптор 2. Копируемый дескриптор должен быть открыт для чтения или записи в зависимости от операции.

Операции перенаправления выполняются слева направо. В случаях, когда используется копирование дескрипторов, порядок выполнения операций может влиять на результат.

4 Основы работы с командным интерпретатором

слово — последовательность символов, воспринимаемая интерпретатором как одна единица.

имя — слово состоящее только из алфавитно-цифровых символов и знаков подчеркивания, начинающееся с буквы или знака подчеркивания. Также называется идентификатором.

метасимвол

— символ, который, не будучи заключен в кавычки, разделяет слова. Один из следующих:

```
| & ; ( ) < > space tab
```

управляющий оператор

— слово выполняющее функции управления. Один из следующих:

```
|| & && ; ;; ( ) | <newline>
```

Зарезервированные слова — это слова имеющие специальное значение для интерпретатора. Следующие слова являются зарезервированными:

```
! case do done elif else esac fi for function if
in select then until while { } time [[ ]]
```

Простая команда это последовательность из необязательного присвоения значения переменной с последующими словами и перенаправлениями, прерываемая управляющим оператором. Первое слово определяет выполняемую команду. Последующие слова передаются команде в качестве аргументов.

```
[VAR=val] command argument ...
```

Возвращаемое значение простой команды — код завершения или 128+n если команда была прервана по сигналу n.

Конвейер — последовательность из одной или более команд, разделенных символом |. Формат конвейера следующий:

```
[time [-p]] [!] command [ | command2 ... ]
```

Стандартный вывод `command` подключается к стандартному вводу команды `command2`. Это подключение производится до выполнения любых перенаправлений.

Если конвейеру предшествует зарезервированное слово `!`, то код завершения конвейера равен логическому отрицанию кода завершения последней команды. Иначе код завершения конвейера

равен коду завершения последней команды. Интерпретатор ожидает завершения всех команд до того как вернет значение.

Если конвейеру предшествует зарезервированное слово `time`, то после завершения выполнения конвейера будет выведена информация о времени выполнения конвейера и о затраченном времени процессора в режимах пользователя и системы.

Каждая команда в конвейере выполняется как отдельный процесс (т.е. в подоболочке).

5 Переменные окружения

У каждого процесса имеется область памяти называемая *программным окружением* (program environment) — это набор строк, заканчивающихся нулевым символом. Эти строки называются *переменными окружения*. Каждая строка имеет вид: `имя переменной = значение`. Имя переменной может состоять из алфавитно-цифровых символов и знака подчеркивания. Цифра не может быть первым символом имени. Присвоение значения переменной в оболочке производится следующим образом:

```
Имя = Значение
```

Для того, чтобы значение переменной передавалось процессам порождаемым оболочкой, следует использовать встроенную команду `export`. Следующие две команды помечают переменные `VAR` и `TST` как экспортируемые и присваивают переменной `TST` значение `/usr/doc`:

```
export VAR
export TST=/usr/doc
```

Для того, чтобы просмотреть значения переменных окружения можно использовать команду `set`, которая выводит значения всех переменных окружения.

Для того, чтобы получить значение переменной, перед ее именем указывается знак доллара. Такое выражение будет заменяться интерпретатором на значение переменной. Например, команда `echo` выводит в стандартный вывод свои аргументы, следующее выражение:

```
echo TST=$TST
```

выведет на экран TST=/usr/doc (при условии, что значение переменной TST – /usr/doc).

6 Основные команды для работы с файлами

`cd` [*каталог*]
Меняет текущий каталог на указанный. Если параметр опущен, то текущим становится домашний каталог.

`ls [-alFR]` [*файл ...*]
Выводит список файлов в указанном (или текущем) каталоге. Ключ -a заставляет выводить все файлы, ключ -l служит для вывода подробной информации о файлах, ключ -F приводит к тому, что к именам каталогов добавляется символ '/', к именам ссылок '@', к именам выполняемых файлов '*'. При использовании ключа -R выводится список файлов не только указанного каталога, но и его подкаталогов.

`touch` *файл ...*
Меняет время доступа и изменения файла. Если файл не существовал, то он будет создан.

`mkdir` *каталог*
Создает каталог.

`rmdir` *каталог*
Удаляет каталог.

`cp [-rp]` *файл1 файл2*
`cp [-rp]` *файл ... каталог*

Копирует один файл в другой или копирует файлы в указанный каталог. Ключ -R предназначен для копирования каталогов, ключ -p позволяет сохранять владельцев файлов, режим доступа и время доступа и изменения.

`rm [-r]` *файл ...*
Удаляет файлы. Ключ -r позволяет удалять каталоги.

`mv` *файл1 файл2*
`mv` *file ... directory*
Перемещает один файл в другой или перемещает файлы в заданный каталог.

`cat [` *файл ...*`]`
Объединяет содержимое указанных файлов и выводит на стандартный вывод.

`find` *путь выражение*
Команда предназначена для поиска файлов. Находит файлы для которых значение выражения "истина". Для определения выражений могут использоваться следующие примитивы:

-name *шаблон*
возвращает значение истина если файл соответствует шаблону.

-nouser возвращает значение истина, если идентификатор пользователя не определен.

-nogroup
возвращает значение истина, если идентификатор группы не определен.

-perm *режим*
возвращает значение истина, если файл имеет заданный режим доступа. Режим может задаваться как в символьной форме (-,+,=), так и в числовой.

-type *тип*
значение истина, если файл имеет указанный тип. (b – блок-ориентированное устройство, c – байт-ориентированное устройство, d – каталог, f – регулярный файл, p – канал).

-links *n*
значение истина, если файл имеет указанное число ссылок.

-user *пользователь*
значение истина если файл принадлежит указанному пользователю.

-group *группа*
значение истина если файл принадлежит указанной группе.

-size *[+|-]размер[c]*
истина, если размер файла в блоках (байтах, если используется размер**c**) равен (больше, если используется +; меньше, если используется -) заданному.

-atime *[+|-]d*
истина, если доступ к файлу производился между (d-1)*24 и d*24 часов назад (+ более d*24 часов назад, - менее (d-1)*24 часов назад). Аналогично **-mtime** для времени изменения содержимого файлов и **-ctime** для времени изменения статуса файлов.

-exec *программа [аргументы];*
исполнение программы для каждого найденного файла. Имя программы и аргументы состоящие только из двух символов `{}` будут заменены именем найденого файла. Заключительному знаку `;` должен предшествовать `\`.

-ok *программа [аргументы];*
аналогична предыдущей команде, но выводит запрос на подтверждение.

-print выводит имя найденного файла.

Параметры могут объединяться следующим образом:
`(выражение)`, группировка выражений
`!выражение`, отрицание выражений
`выражение1 [-a] выражение2`, логическое И
`выражение1 -o выражение2`, логическое ИЛИ.

7 Практическое задание

1. Войдите в систему под выданной пользовательской учетной записью.
2. Смените пароль пользователя. Для этого следует воспользоваться командой `passwd`. Команда последовательно запросит текущий пароль, новый пароль и подтверждение нового пароля.
3. Создайте в домашнем каталоге при помощи команды `touch` файл отчета с именем `lab1.txt`.
4. При помощи команды `set` просмотрите значения переменных окружения.
5. Выведите в файл lab1.txt значения переменных окружения PATH, LANG, HOME.
6. Просмотрите полученный файл при помощи команды `less`.
7. Последовательно перейдите в каталоги /bin, /usr, /etc, /usr/bin. Выполните в каждом каталоге команду `ls` с различными ключами (-a, -l, -F, -R). Если вывод команды `ls` не умещается на экране, то можно воспользоваться командой `less` (напр. `ls -l | less`).
8. Перейдите обратно в домашний каталог (команда `cd`).
9. Изучите команды `uname` и `date`. Просмотрите справку об этих командах (`man uname`, `man date`).
10. Сравните вывод команд `date` и `LANG=C date`.
11. Добавьте в конец файла отчета, используя перенаправление вида >>, информацию выводимую командами `uname -a` и `date`.
12. Создайте в домашнем каталоге подкаталоги `test1` и `test2`.
13. Скопируйте файл `/home/labs/text.txt` в каталог `test2`. Добавьте в файл отчета вывод команды `ls -R`.
14. Переместите файл `text.txt` из каталога `test2` в каталог `test1`. Снова добавьте в файл отчета вывод команды `ls -R`.
15. Изучите команду `find`. Добавьте в файл отчета все подкаталоги каталога /usr/share содержащие в своем имени сочетание букв ”ru” (используйте параметры -type и -name).
16. Воспользуйтесь командой `rmdir` для удаления каталогов созданных при выполнении пункта 12.
17. Добавьте в конец файла отчета две строки. Первая должна содержать текст: "Лабораторная работа No 1". Вторая должна содержать Ваши имя и фамилию.