*Лабораторная работа №5*

**Обработка текстовых потоков в ОС Linux**

**Цель работы:** освоение инструментария работы с текстовыми потоками.

**1. Методические рекомендации**

Основным интерфейсом в операционных системах Linux является консольный интерфейс с текстовым вводом и выводом данных. Это определяет подход к управлению объектами операционной системы в их текстовом отображении. Например, состояние процессов отображается в виде набора текстовых файлов в псевдофайловой системе /proc, сведения о событиях в системе хранятся в текстовых файлах журналов, настройки отдельных пакетов в текстовых конфигурационных файлах. Это делает необходимым для решения дальнейших задач управления операционной системы освоение инструментария работы с текстовыми потоками.

**1.1. Управление вводом-выводом команд (процессов)**

У любого процесса по умолчанию всегда открыты три файла:

- stdin (стандартный ввод, клавиатура),

- stdout (стандартный вывод, экран),

- stderr (стандартный вывод сообщений об ошибках на экран).

Эти и любые другие открытые файлы могут быть перенаправлены. В данном случае термин "перенаправление" означает: получить вывод из файла (команды, программы, сценария) и передать его на вход в другой файл (команду, программу, сценарий). Дескрипторы файлов открытых по умолчанию:

0 = stdin

1 = stdout

2 = stderr

***команда > файл*** – перенаправление стандартного вывода в файл, содержимое существующего файла удаляется.

***команда >> файл*** – перенаправление стандартного вывода в файл, поток дописывается в конец файла.

***команда1 | команда2*** – перенаправление стандартного вывода первой команды на стандартный ввод второй команды = образование конвейера команд.

***команда1 $(команда2)*** – передача вывода команды 2 в качестве параметров при запуске команды 1. Внутри скрипта конструкция $(команда2) может использоваться, например, для передачи результатов работы команды 2 в параметры цикла for … in.

**1.2. Работа со строками (внутренние команды bash)**

***${#string}*** – выводит длину строки (string – имя переменной);

***${string:position:length}*** – извлекает $length символов из $string, начиная с позиции $position. Частный случай: ***${string:position}*** извлекает подстроку из $string, начиная с позиции $position.

***${string#substring}*** – удаляет самую короткую из найденных подстрок $substring в строке $string. Поиск ведется с начала строки. $substring – регулярное выражение (см. п. 1.5).

***${string##substring}*** – удаляет самую длинную из найденных подстрок $substring в строке $string. Поиск ведется с начала строки. $substring – регулярное выражение.

***${string/substring/replacement}*** – замещает первое вхождение $substring строкой $replacement. $substring – регулярное выражение.

***${string//substring/replacement}*** – замещает все вхождения $substring строкой $replacement. $substring – регулярное выражение.

**1.3. Работа со строками (внешние команды)**

Для каждой команды доступно управление с помощью передаваемых команде параметров.

***sort*** – сортирует поток текста в порядке убывания или возрастания, в зависимости от заданных опций.

***uniq*** – удаляет повторяющиеся строки из отсортированного файла.

***cut*** – извлекает отдельные поля из текстовых файлов (поле – последовательность символов в строке до разделителя).

***head*** – выводит начальные строки из файла на stdout.

***tail*** – выводит последние строки из файла на stdout.

***wc*** – подсчитывает количество слов/строк/символов в файле или в потоке

***tr*** – заменяет одни символы на другие.

**1.4. Полнофункциональные многоцелевые утилиты**

***grep*** – многоцелевая поисковая утилита, использующая регулярные выражения.

***grep pattern [file...]*** – утилита поиска участков текста в файле(ах), соответствующих шаблону pattern, где pattern может быть как обычной строкой, так и регулярным выражением.

***Sed*** – неинтерактивный "потоковый редактор". Принимает текст либо с устройства stdin, либо из текстового файла, выполняет некоторые операции над строками и затем выводит результат на устройство stdout или в файл. Sed определяет, по заданному адресному пространству, над какими строками следует выполнить операции. Адресное пространство строк задается либо их порядковыми номерами, либо шаблоном. Например, команда 3d заставит sed удалить третью строку, а команда /windows/d означает, что все строки, содержащие "windows", должны быть удалены. Наиболее часто используются команды p – печать (на stdout), d – удаление и s – замена.

***awk*** – утилита контекстного поиска и преобразования текста, инструмент для извлечения и/или обработки полей (колонок) в структурированных текстовых файлах. Awk разбивает каждую строку на отдельные поля. По умолчанию поля – это последовательности символов, отделенные друг от друга пробелами, однако имеется возможность назначения других символов в качестве разделителя полей. Awk анализирует и обрабатывает каждое поле в отдельности.

**1.5. Регулярные выражения**

Регулярные выражения – это набор символов и/или метасимволов, которые наделены особыми свойствами. Их основное назначение – поиск текста по шаблону и работа со строками. При построении регулярных выражений используются нижеследующие конструкции (в порядке убывания приоритета), некоторые из которых могут быть использованы только в расширенных версиях соответствующих команд (например, при запуске grep с ключом -E).

|  |  |
| --- | --- |
| **с** | Любой неспециальный символ с соответствует самому себе |
| **\с** | Указание убрать любое специальное значение символа с (экранирование) |
| **^** | Начало строки |
| **$** | Конец строки; выражение "^$" соответствует пустой строке. |
| **.** | Любой одиночный символ, за исключением символа перевода строки |
| **[...]** | Любой символ из ...; допустимы диапазоны типа а-z; возможно объединение диапазонов, например [a-z0-9] |
| **[^...]** | Любой символ не из ...; допустимы диапазоны |
| **\n** | Строка, соответствующая n-му выражению\(...\) |
| **r\*** | Ноль или более вхождений символа r |
| **r+** | Одно или более вхождений символа r |
| **r?** | Ноль или одно вхождение символа r |
| **\<...\>** | Границы слова |
| **\{ \}** | Число вхождений предыдущего выражения. Например, выражение "[0-9]\{5\}" соответствует подстроке из пяти десятичных цифр |
| **r1r2** | За r1 следует r2 |
| **r1|r2** | r1 или r2 |
| **(r)** | Регулярное выражение r; может быть вложенным |

**1.6. Классы символов POSIX**

|  |  |
| --- | --- |
| **[:class:]** | альтернативный способ указания диапазона символов. |
| **[:alnum:]** | соответствует алфавитным символам и цифрам. Эквивалентно выражению [A-Za-z0-9]. |
| **[:alpha:]** | соответствует символам алфавита. Эквивалентно выражению [A-Za-z]. |
| **[:blank:]** | соответствует символу пробела или символу табуляции. |
| **[:cntrl:]** | соответствует управляющим символам |
| **[:digit:]** | соответствует набору десятичных цифр. Эквивалентно выражению [0-9]. |
| **[:lower:]** | соответствует набору алфавитных символов в нижнем регистре. Эквивалентно выражению [a-z]. |
| **[:space:]** | соответствует пробельным символам (пробел и горизонтальная табуляция). |
| **[:upper:]** | соответствует набору символов алфавита в верхнем регистре. Эквивалентно выражению [A-Z]. |
| **[:xdigit:]** | соответствует набору шестнадцатиричных цифр. Эквивалентно выражению [0-9A-Fa-f]. |

**2. Примеры обработки текстовых потоков**

**Конвейеризация с использованием оператора |**

Передаваемые командам параметры могут служить входными данными этих команд, а выходные данные могут выводиться на ваш терминал. Многие команды обработки текста (фильтры) могут получать входные данные либо из стандартного потока ввода, либо из файла. Чтобы передать вывод команды1 на вход команде2 (выступающей в качестве фильтра), необходимо соединить эти две команды оператором конвейеризации ввода/ вывода (|). Ниже показано, как перенаправить вывод команды echo на вход команде sort, которая сортирует полученный список слов:

*echo -e "apple\npear\nbanana"|sort*

У любой из этих команд могут иметься опции или аргументы. С помощью оператора | можно также перенаправить вывод второй команды на вход третьей команде и так далее. Построение длинных конвейеров из команд, каждая из которых имеет свой ограниченный функционал – это распространенный в Linux и UNIX прием, используемый для решения поставленных задач. Иногда аргументом команды может являться не имя файла, а знак дефиса (-); это означает, что входные данные следует принимать со стандартного устройства ввода, а не из файла.

**Перенаправление вывода с помощью оператора >**

Помимо возможности создавать конвейеры из нескольких команд и выводить результаты на экран терминала, иногда возникает необходимость сохранить вывод в файл. Для этого используется оператор перенаправления вывода (>).

Перенаправим с помощью оператора > вывод команды echo в файл с именем text1.

*echo -e "1 apple\n2 pear\n3 banana" > text1*

Поскольку весь вывод перенаправляется в файл, он не отображается на экране.

**Команда cat**

После того, как вы создали файл test1, вы можете просмотреть его содержимое. Для вывода содержимого файла на стандартное устройство вывода используется команда cat (сокращенно от concatenate – объединять):

*cat text1*

Если не указать имя файла (или поставить вместо имени файла дефис), то команда cat принимает входные данные со стандартного устройства ввода. Давайте используем эту возможность (а также перенаправление вывода) для создания еще одного текстового файла:

*cat >text2*

Введите:

9<Tab>plum

3<Tab>banana

10<Tab>apple

Здесь <Tab> - клавиша табуляции.

Команда cat продолжает считывать данные с устройства stdin до тех пор, пока не будет достигнут конец файла. Чтобы обозначить конец файла, нажмите комбинацию клавиш Ctrl-d (удерживайте Ctrl и нажмите d).

С помощью cat вы можете объединить несколько файлов (в данном случае файлы text1 и text2) и вывести их содержимое на экран:

*cat text\**

**Команда wc**

Команда wc выводит количество содержащихся в файле строк и слов, а также размер файла в байтах:

*wc text\**

Различные опции позволяют вам управлять выводом команды wc или отображать другую информацию, например, максимальную длину строки.

**Команды head и tail**

Позволяют отображать либо первую часть файла (заголовок), либо последнюю (хвост). Их можно использовать в качестве фильтров или передавать им в качестве аргумента имя файла. По умолчанию эти команды отображают 10 первых (или последних) строк файла или потока. В примере совместно используются команды dmesg (отображение информации о загрузке системы):

*dmesg | tail*

*dmesg | tail -n15 | head -n 6*

**Команды sort**

Команда sort сортирует входные данные с использованием схемы упорядочивания локали (LC\_COLLATE) системы. Команда sort также может объединять уже отсортированные файлы и определять, является ли файл отсортированным или нет. Пример использования команды sort для сортировки двух текстовых файлов после замены в файле text1 пробелов на символы табуляции:

*cat text1 | tr ' ' '\t' | sort - text2*

Поскольку сортировка выполняется на основе символьных значений, вы можете удивиться, увидев результаты. К счастью, команда sort может выполнять сортировку не только на основе символьных, но также и на основе числовых значений. Вы можете указать требуемый метод сортировки для всей записи или для каждого поля. Если вы не указываете разделитель полей, то используются пробелы или символы табуляции. Во втором примере сортировка первого поля выполняется по числовым значениям, а сортировка второго поля – с использованием схемы упорядочивания (в алфавитном порядке):

*cat text1|tr ' ' '\t'|sort -u -k1n -k2 - text2*

В этом примере использована опция -u для удаления повторяющихся строк. Заметьте, что в списке все равно присутствуют две строки со словом "apple", поскольку проверка уникальности выполнялась по всем ключам сортировки (в нашем случае это k1n и k2).

***Задание:*** Подумайте, какие команды нужно изменить или добавить в конвейер в последнем примере, чтобы исключить дублирование слова 'apple'.

**Команда uniq**

Можно управлять удалением повторяющихся строк с помощью другой команды – uniq. В обычном режиме команда uniq работает с отсортированными файлами и удаляет последовательные повторяющиеся строки из любого файла независимо от того, отсортирован он или нет. Также эта команда может игнорировать заданные поля. В примере выполняется сортировка двух текстовых файлов по второму полю (имя фрукта), после чего удаляются строки, в которых повторяются значения второго поля (т. е. при проверке мы не обращаем внимания на первое поле):

*cat text1|tr ' ' '\t'|sort -k2 - text2|uniq -f1*

***Задание:*** В этом примере сортировка выполнялась с использованием схемы упорядочивания, поэтому команда uniq оставила запись "10 apple", а не "1 apple". Вы можете добавить сортировку первого поля по числовым значениями и посмотреть, что изменится в этом случае.

**Команда cut, paste и join**

Команда cut извлекает поля из текстовых файлов. Символом-разделителем по умолчанию является символ табуляции. В примере команда cut используется для разделения двух столбцов файла text2, а затем в качестве разделителя выходных данных используется пробел, что является необычным способом преобразования символов табуляции в пробелы:

*cut -f1-2 --output-delimiter=' ' text2*

**Команда paste**

Команда paste вставляет (склеивает) строки из двух или более файлов, размещая их рядом (подобно тому, как команда pr объединяет файлы с помощью опции -m):

*paste text1 text2*

В этом примере показана простейшая операция, тем не менее, команда paste может вставлять данные из одного или нескольких файлов различными способами.

**Команда join**

Команда join, которая объединяет файлы на основе совпадения полей. Файл должен быть отсортирован по объединяемому полю. Поскольку файл text2 не отсортирован по числовым значениям, то можно отсортировать его, а затем объединить с помощью команды join две строки с одинаковым значением поля, по которому выполняется объединение (в нашем примере это первое поле, содержащее значение 3):

*sort -n text2|join -j 1 text1 -*

Объединение не будет выполняться для числовых полей до тех пор, пока все поля не будут иметь одинаковую длину. Мы использовали опцию -j 1 для объединения по первому полю в каждом файле. Для каждого файла можно указать отдельное поле, по которому будет выполняться объединение. Например, можно объединить поле 3 в одном файле с полем 10 другого файла. Давайте создадим еще один файл, text5, выполнив сортировку файла text1 по второму полю (имя фрукта), а затем заменив пробелы на символы табуляции. Если теперь мы отсортируем файл text2 по второму полю и объединим его с файлом text5 по этому же полю, то получим два совпадения (apple и banana):

*sort -k2 text1|tr ' ' '\t'>text5*

*sort -k2 text2 | join -1 2 -2 2 text5 -*

**Редактор Sed**

Sed (stream editor) – это потоковый редактор. Является чрезвычайно мощным инструментом, а круг решаемых им задач ограничен лишь вашим воображением. Sed может работать как фильтр или принимать входные данные из файла. Вывод осуществляется на стандартное устройство вывода. Sed загружает строки из входных данных в область шаблонов, применяет к ее содержимому команды редактирования и передает ее на стандартное устройство вывода. Sed может объединять в области шаблонов несколько строк; результат может быть записан в файл, может быть записан частично, а может быть не записан вообще.

Для поиска и выборочной замены текста в области шаблонов, а также для определения строк, над которыми необходимо выполнять те или иные команды редактирования, sed использует синтаксис регулярных выражений. Временным хранилищем текста служит буфер удержания. Буфер удержания может заместить собой область шаблонов, может быть добавлен к области шаблонов, а может обмениваться с ней данными. Хотя в sed имеется ограниченное число команд, их использование совместно с регулярными выражениями и буфером удержания открывает безграничные возможности. Набор команд sed обычно называется сценарием sed.

В примерах показаны три простых сценария sed. В первом сценарии используется команда s (substitute – замена) для замены в каждой строке символа 'a' в нижнем регистре на этот же символ в верхнем регистре:

*sed 's/a/A/' text1*

В первом примере выполняется замена только первого символа 'a', поэтому во втором примере добавляем флаг 'g' (global – глобальный), благодаря которому, будет выполняться замена всех найденных вхождений этого символа:

*sed 's/a/A/g' text1*

В третьем примере используем команду d (delete – удалить) для удаления строки:

*sed '2d;$s/a/A/g' text1*

В примере использован адрес 2, чтобы показать, что необходимо удалить только строку с этим номером. Мы разделяем команды точкой с запятой (;) и используем глобальную замену символов 'a' на 'A', как это было сделано во втором примере.

Помимо работы с отдельными строками, sed может работать с диапазонами строк. Начало и конец диапазона разделяются запятой (,) и могут определяться в виде номера строки, регулярного выражения или знака доллара ($), означающего конец файла. Зная адрес или диапазон адресов, вы можете сгруппировать несколько команд, заключив их в фигурные скобки { и }; таким образом, эти команды будут работать только с теми строками, которые указаны в диапазоне. Рассмотрим два примера глобальной замены, которая применяется только к последним двум строкам нашего файла:

*sed -e '2,${' -e 's/a/A/g' -e '}' text1*

*sed -e '/pear/,/bana/{' -e 's/a/A/g' -e '}' text*

Также приведен пример использования опции -e для добавления нескольких команд в сценарий.

В следующем примере сначала используется команда = для вывода номеров строк, а затем выполняется фильтрация полученного вывода с помощью sed (в результате мы получим такой же эффект, как от использования команды nl для нумерации строк):

*sed '=' text2*

В следующем примере с помощью команды = выводятся номера строк, затем с помощью команды N вторая строка ввода считывается в область шаблонов и, наконец, между двумя строками в области шаблонов удаляется символ новой строки (/n):

*sed '=' text2|sed 'N;s/\n//'*

**3. Задания для самостоятельной работы**

1. Реализуйте примеры п. 2. Там, где обозначены задания – выполните их.

2. Напишите скрипт, решающий задачу: подсчитать общее количество слов в файлах, находящихся в директории /var/log/ и имеющих расширение log. Описать решение задачи.

3. Напишите скрипт, выводящий в файл в алфавитном порядке имена всех файлов с расширением «sh», находящихся в домашнем каталоге и подкаталогах домашнего каталога, с указанием каталога их размещения.