

## Лабораторная работа №2. Обработка текстовых потоков в ОС Linux

### Рассматриваемые вопросы

1. Понятие стандартного ввода и стандартного вывода процесса
2. Перенаправление стандартного вывода в файл
3. Связь процессов по вводу/выводу
4. Использование вывода процесса как параметра другого процесса
5. Регулярные выражения и фильтрация текстовых потоков

### Управление вводом-выводом команд (процессов)

У любого процесса по умолчанию всегда открыты три файла – **stdin** (стандартный ввод, клавиатура), **stdout** (стандартный вывод, экран) и **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).

**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**.

### 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 самой короткой из найденных подстроки **\$substring** в строке **\$string**.

Поиск ведется с начала строки. **\$substring** – регулярное выражение (см. ниже).

**\${string##substring}** – удаляет самую длинную из найденных подстроки **\$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.

### Работа со строками (внешние команды)

*Для каждой команды доступно управление с помощью передаваемых команде параметров. Рекомендуем ознакомиться с документацией по этим командам с помощью команды **man**.*

**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.

*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 "поточный 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.

**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**).

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

### Классы символов POSIX

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

### Задание на лабораторную работу

1. Создайте свой каталог в директории **/home/user/** Все скрипты и файлы для вывода результатов создавайте внутри этого каталога или его подкаталогов. (**mkdir lab2**)
2. Напишите скрипты, решающие следующие задачи:
  - i) Создать файл **errors.log**, в который поместить все строки из всех доступных для чтения файлов директории **/var/log/**, начинающиеся с последовательности символов (**==**), без указания имени файла, в котором встретилась строка. Вывести на экран те строчки из получившегося файла, которые содержат полные имена каких-либо файлов.
  - ii) Создать **full.log**, в который вывести строки файла **/var/log/Xorg.0.log**, содержащие предупреждения и информационные сообщения, заменив маркеры предупреждений и информационных сообщений на слова **Warning:** и **Information:**, чтобы в получившемся файле сначала шли все ошибки, а потом все предупреждения. Вывести этот файл на экран.
  - iii) Создать файл **emails.lst**, в который вывести через запятую все адреса электронной почты, встречающиеся во всех файлах директории **/etc**.
  - iv) Найти в директории **/bin** все файлы, которые являются сценариями, и вывести на экран полное имя файла с интерпретатором, наиболее часто используемым в этих сценариях (только полное имя файла).
  - v) Вывести список пользователей системы с указанием их **UID**, отсортировав по **UID**. Сведения о пользователях хранятся в файле **/etc/passwd**. В каждой строке этого файла первое поле – имя пользователя, третье поле – **UID**. Разделитель – двоеточие.
  - vi) Подсчитать общее количество строк в файлах, находящихся в директории **/var/log/** и имеющих расширение **log**.
  - vii) Вывести три наиболее часто встречающихся слова из **man** по команде **bash** длиной не менее четырех символов.
3. Предъявите скрипты преподавателю и получите вопрос или задание для защиты лабораторной работы.
4. После защиты лабораторной работы удалите созданный каталог со всем его содержимым (**rm -R lab2**)