Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**«ОБРАБОТКА СТРОК (РАБОТА С ТЕКСТОВЫМИ ДАННЫМИ)»**

**ОТЧЕТ**

**по лабораторной работе №12**

**дисциплины**

**«Операционные системы»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Мизин Глеб Егорович  2 курс, группа ПИЖ-б-о-21-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2022 г.

**Цель работы**: практическое знакомство со способами эффективной обработки текста при помощи интерфейса командной строки и набора стандартных утилит

**Задание №1**: Используя утилиты hexdump и strings, вывести на экран содержимое одного из перечисленных ниже файлов из каталога /bin. Позиция файла для распечатки определяется номером бригады. Имена файлов для выполнения задания 1: tar, sort, sed, ping, vi, unlink, uname, touch, sleep, sty.

Команда hexdump предназначена для вызова одноименной утилиты, осуществляющей вывод содержимого бинарных файлов. Ключ -C используется для представления битов в 16-ричном фомате (hex)

string же используется когда нам нужно найти текстовые строки в бинарном файле

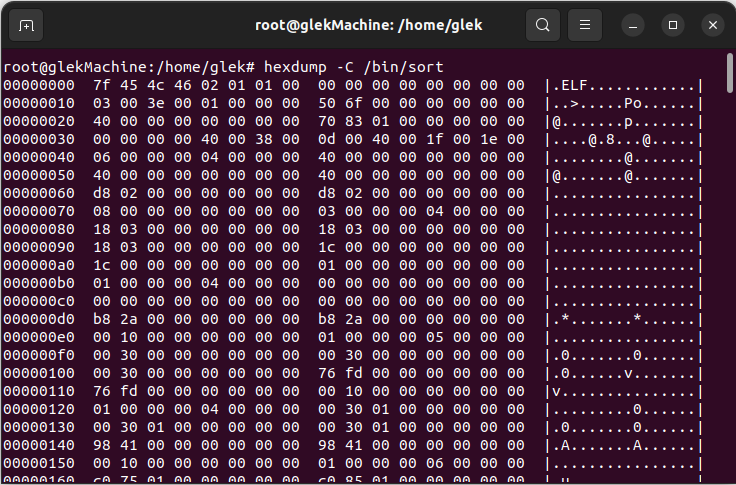


Рисунок 1 – Вывод на экран командой hexdump

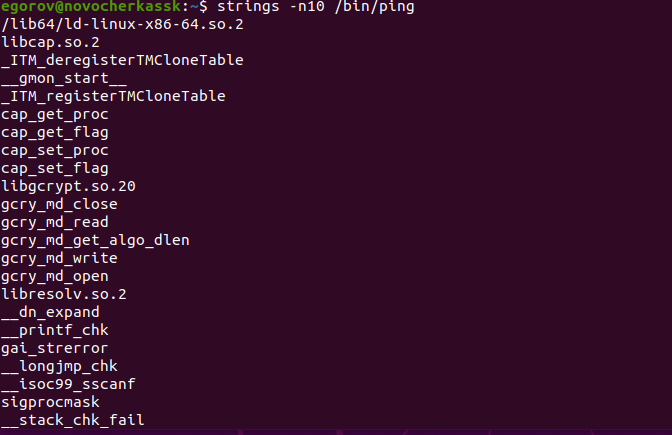


Рисунок 2 – Вывод на экран командой strings

**Задание №2**: подсчитать общее количество файлов (каталогов) в одном из перечисленных ниже каталогов. Каталог для подсчета количества определяется номером бригады. Имена каталогов для выполнения задания 2: /bin, /etc, /lib, /proc, /usr, /var, /dev, /sbin, /sys, /root

Для решения этой задачи была использована утилита wc (word count) утилита, выводящая число переводов строк, слов и байт для каждого указанного файла и итоговую строку, если было задано несколько файлов.

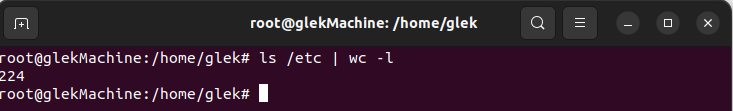


Рисунок 3 – Подсчёт файлов в каталоге /etc

**Задание №3**: найти общее количество процессов, выполняющихся в системе в данный момент.

При помощи команды ps с ключом -a получим все процессы и так же при помощи команды wc посчитаем их количество.



Рисунок 4 – Общее количество процессов

**Задание №4**: вывести список выполняющихся процессов, в именах которых присутствует слово manager и отсутствует слово grep

Так же при помощи команды ps получим информацию о процессах при помощи команды grep найдём все процессы со словом manager, а при помощи grep -v исключим все процессы c, словом, grep



Рисунок 5 – Процесс подходящий по заданному условию

**Задание №5**: создать текстовый файл, содержащий набор строк вида:

123

178

176

755

713

873

С помощью утилиты grep найти строки, в которых есть цифра 7, после которой находится одна из цифр — 1, 3 или 5

Создадим файл при помощи команды echo с папметром -e для того, чтобы писать каждый символ после \n с новой строки.

Далее при помощи grep укажем первый символ, а именно 7, а после него в [] укажем символы, которые могут быть за первым указанным символом

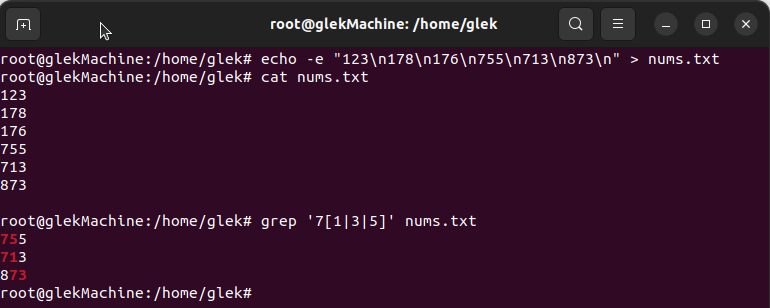


Рисунок 6 – Создание файла и выборка подходящих значений

**Задание №6**: создать текстовый файл, содержащий набор строк вида:

starfish

starless

samscripter

stellar

microsrar

ascender

sacrifice

scalar

С помощью утилиты grep найти строки, начинающиеся на букву s и заканчивающиеся на букву r

Создадим файл так же, как и в задании №5, но для поиска будем использовать команду grep с границами слова \b

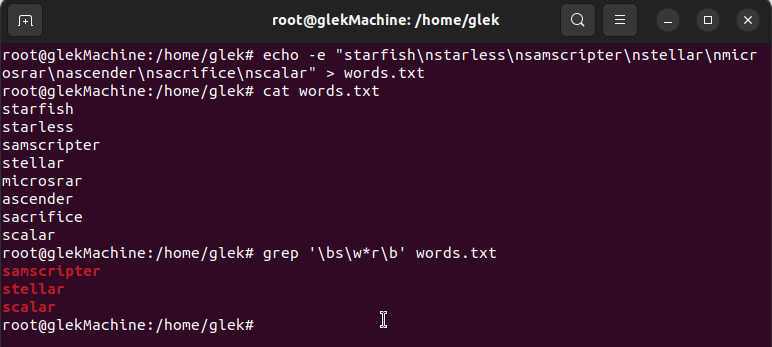


Рисунок 7 – Создание файла и выборка подходящих слов

**Задание №7**: создать текстовый файл, содержащий простейшие адреса электронной почты вида username@website.com. С помощью утилиты grep найти строки, содержащие правильные простейшие адреса. Проверить возможность использования более сложного регулярного выражения для распознавания адресов, содержащих другие допустимые символы.

Для выполнения данного задания к команде grep допишем 2 ключа -E и -o первый ключ будет интерпретировать наш запрос как регулярное выражение а второй выдавать только те результаты, которые имеют полное сходство с регулярным выражением. Так же будем использовать \b и [].

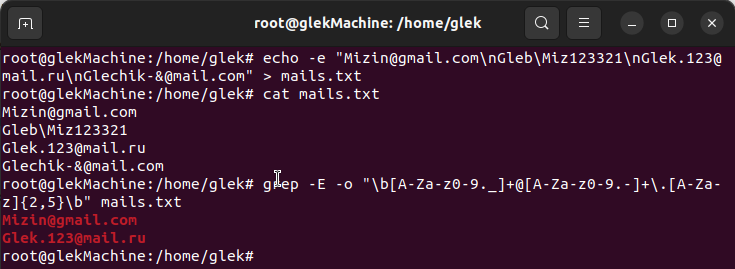


Рисунок 8 – Выбор подходящих данных

**Задание №8**: на произвольном примере продемонстрировать работу утилиты tr. Создать текстовый файл, содержащий допустимые и недопустимые IPадреса, например:

127.0.0.1

255.255.255.255

12.34.56

123.256.0.0

1.23.099.255

0.79.378.111

С помощью утилиты grep и руководства man найти строки, содержащие

допустимые четырехбайтовые IP адреса.

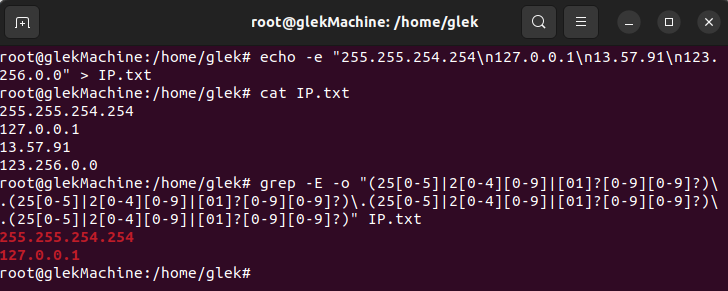


Рисунок 9 – Выбор подходящих данных

**Задание №9**: создать текстовый файл, содержащий корректные и некорректные номера телефонов ведомственной АТС объемом 399 номеров, номера с 000 до 399 – корректные, 0, 400, 900 –некорректные. С помощью утилиты grep и руководства man найти строки, содержащие допустимые номера телефонов.

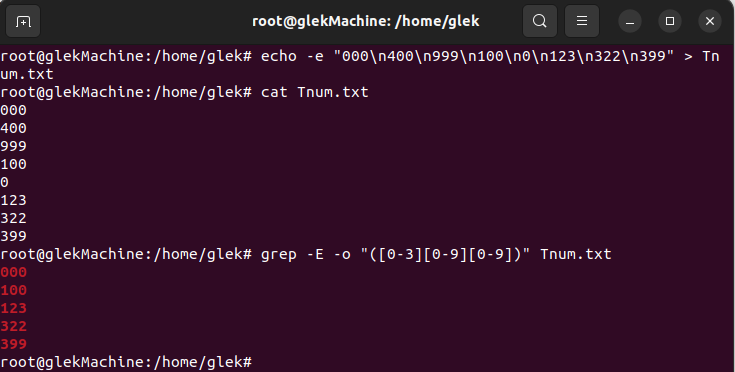


Рисунок 10 – Выбор подходящих данных

**Вывод**: ознакомились со способами эффективной обработки текста при помощи интерфейса командной строки и набора стандартных утилит

**Контрольные вопросы**:

**1. Вывод на экран содержимого нетекстового файла с помощью утилит**

**hexdump и strings.**

Если содержимое нетекстового файла все-таки желательно

просмотреть (то есть превратить в текст), можно воспользоваться

утилитой **hexdump** с ключом -С, которая выдает содержимое файла в виде

шестнадцатеричных ASCII-кодов, или **strings**, показывающей только те

части файла, которые могут быть представлены в виде текста.

[student@localhost root]$ hexdump -C /bin/cat | less

[student @localhost root]$ strings -n3 /bin/cat | less

**2. Стандартный ввод, вывод, стандартный вывод ошибок.**

**Cтандартный вывод** (standard output, stdout) - это поток данных,

открываемый системой для каждого процесса в момент его запуска и

предназначенный для данных, выводимых процессом.

Некоторые утилиты умеют выводить не только на терминал, но и в

файл. Например, info при указании ключа "-o" с именем файла выведет

текст руководства в файл, вместо того, чтобы отображать его на мониторе.

Даже если разработчиками программы не предусмотрен такой ключ,

известен и другой способ сохранить вывод программы в файле вместо

того, чтобы выводить его на монитор: **поставить знак ">"** и указать после

него имя файла.

**Стандартный ввод** (standard input, stdin) - поток данных,

открываемый системой для каждого процесса в момент его запуска и

предназначенный для ввода данных.

Для передачи данных на вход программе может быть использован

стандартный ввод (сокращенно - stdin). При работе с командной строкой

стандартный ввод - это символы, вводимые пользователем с клавиатуры.

Стандартный ввод можно перенаправить при помощи командной

оболочки, подав на него данные из некоторого файла. Символ "<" служит

для перенаправления содержимого файла на стандартный ввод

программе.

**3. Конвейер и канал.**

Нередко возникают ситуации, когда нужно обработать вывод одной

программы какой-то другой программой. Для решения подобной задачи в

bash предусмотрена возможность перенаправления вывода можно не

только в файл, но и непосредственно на стандартный ввод другой

программе. В Linux такой способ передачи данных называется конвейер.

В bash для перенаправления стандартного вывода на стандартный

ввод другой программе служит **символ "|".**

**4. Фильтры.**

Если программа и вводит данные, и выводит, то ее можно

рассматривать как трубу, в которую что-то входит и из которой что-то

выходит. Обычно смысл работы таких программ заключается в том, чтобы

определенным образом обработать поступившие данные. В Linux такие

программы называют фильтрами: данные проходят через них, причем

что-то "застревает" в фильтре и не появляется на выходе, а что-то

изменяется, что-то проходит сквозь фильтр неизменным.

**5. Структурные единицы текста. Подсчет количества единиц текста.**

Работу в системе Linux почти всегда можно представить как работу с

текстами. Поиск файлов и других объектов системы - это получение от

системы текста особой структуры - списка имен. Операции над файлами -

создание, переименование, перемещение, а также сортировка,

перекодировка и прочее - замену одних символов и строк другими либо в каталогах, либо в самих файлах. Работая с текстом в Linux, нужно

принимать во внимание, что текстовые данные, передаваемые в системе,

структурированы. Большинство утилит обрабатывает не непрерывный

поток текста, а последовательность единиц. В текстовых данных в Linux

выделяются следующие структурные единицы: **Строки, Поля, Символы**

**6. Элементарные регулярные выражения.**

Регулярными выражениями называются особым образом

составленные наборы символов, выделяющие из текста нужное сочетание

слов или символов, которое соответствует признакам, отраженным в

регулярном выражении. Иными словами, регулярное выражение — это

фильтр для текста.

**7. Знакозаменяющие метасимволы.**

Не все символы можно использовать прямо по назначению.

Посмотрите, например, на конструкцию, которая описывалась в

предыдущем разделе. Допустим, требуется найти в каком-то файле строки,

содержащие следующий набор символов: abc[def. Можно предположить,

что регулярное выражение будет составлено по принципам, описанным

выше, но это неверно. Открывающая квадратная скобка — это один из

символов, который несет для программы, работающей с регулярными

выражениями, особый смысл (который был рассмотрен ранее). Такие

символы называются метасимволами.

**8. Метасимволы количества повторений в регулярных выражениях.**

Количество символов, которые должны быть в

конечном тексте, не всегда известно (примером может быть распознавание

имени веб-сайта), поэтому при помощи вышеописанных конструкций и

метасимволов написать действительно универсальные регулярные

выражения невозможно. Рассмотрим еще одну группу символов, которые

помогут решить подобные проблемы. Они используются сразу после

символа, метасимвола либо конструкции, количество вхождения которых

они должны описать.

Пример: **“?”** - Указывает обработчику регулярных выражений на то, что предыдущий

символ, метасимвол или конструкция могут вообще не существовать в

конечном тексте либо присутствовать, но иметь не более одного

вхождения.

**9. Группировка выражений в регулярных выражениях.**

Особым приемом при составлении регулярных выражений является

группировка нескольких его составляющих в одну единицу. Рассмотрим

пример, в котором требуется выделить из текста обычный телефонный

номер в его записи без кода города, когда весь номер разбивается на три

части, между которыми ставится дефис. Для этого подойдет такое

регулярное выражение:

\d{l,3}-\d{l,3}-\d{l,3}

**10.Использование зарезервированных символов в регулярных**

**выражениях.**

Выше упоминалось, что некоторые символы имеют для программы,

работающей с регулярными выражениями, особый смысл. Это, например,

косая черта, точка, круглая, фигурная и квадратная скобки, звездочка и т.

д. Однако не исключено, что в целевом выражении также могут быть эти

символы, и их наличие нужно будет определить в регулярном. В данном

случае эти символы нужно указать особым образом («защитить» их). При

этом перед нужным символом ставят косую черту, то есть, чтобы указать

наличие в конечном тексте символа звездочки, в регулярном выражении на

соответствующем месте следует написать \\*.

**11.Подсчет количества элементов текстового файла.**

Часто бывает необходимо подсчитать количество определенных

элементов текстового файла. Для подсчета строк, слов и символов служит

стандартная утилита - wc (от англ. "word count" - "подсчет слов").

Используя текстовый вывод утилит, можно посчитать свои файлы:

Пример 3. Подсчет файлов при помощи find и wc

[student@localhost root]$ find . | wc -l

**12.Назначение утилит head, tail, cut.**

Иногда пользователя интересует только часть из тех данных,

которые собирается выводить программа. Утилита **head** нужна, чтобы

вывести только первые несколько строк файла. Не менее полезна утилита

**tail** (англ. "хвост"), выводящая только последние строки файла. Если же

пользователя интересует только определенная часть каждой строки файла -

поможет утилита **cut**.

**13.Назначение и использование утилиты grep.**

Зачастую пользователю нужно найти только упоминания чего-то

конкретного среди данных, выводимых утилитой. Обычно эта задача

сводится к поиску строк, в которых встречается определенное слово или

комбинация символов. Для этого подходит стандартная утилита grep,

которая может искать строку в файлах, а может работать как фильтр:

получив строки со стандартного ввода, она выведет на стандартный

вывод только те строки, где встретилось искомое сочетание символов. В

следующем примере анализируются процессы bash, которые выполняются

в системе.

[student@ localhost root]$ ps aux | grep bash

**14.Выполнение транслитерации.**

Для замены одних символов другими предназначена утилита tr

(сокращение от англ. "translate" - "преобразовывать, переводить"),

работающая как фильтр. В примере 7.7 утилита применена прямо по

назначению – с ее помощью выполнена транслитерация - замена латинских

символов близкими по звучанию русскими

**15.Назначение потокового редактора sed.**

Помимо простой замены отдельных символов, возможна замена

последовательностей (слов). Специально для этого предназначен

потоковый редактор sed (сокращение от англ. «stream editor»). Он работает

как фильтр и выполняет редактирование поступающих строк: замену

одних последовательностей символов другими, причем можно заменять и

регулярные выражения.

Например, с помощью sed можно сделать более понятным для

непривычного читателя список файлов, выводимый ls