Отчет по Лабораторной работе №9

«Обработка строк. Работа с текстовыми данными»

Группа 2-МВ-4

Ярошевская Д.А.

1. Используя утилиты hexdump и strings, вывести на экран содержимое файла tar из каталога /bin.

С помощью команды **hexdump –C** выводим содержимое файла в виде шестнадцатеричных ASCII-кодов. В правой стороне экрана – текстовое представление данных, где непечатные символы заменены точками.

```
dari@Ubuntu:~$ hexdump -C /bin/sort
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  03 00 3e 00 01 00 00 00  f0 69 00 00 00 00 00 00  |..>......i......|
00000020  40 00 00 00 00 00 00 00  00 c3 01 00 00 00 00 00  |@...............|
00000030  00 00 00 00 40 00 38 00  0d 00 40 00 1e 00 1d 00  |....@.8...@.....|
00000040  06 00 00 00 04 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000050  40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |@.......@.......|
00000060  d8 02 00 00 00 00 00 00  d8 02 00 00 00 00 00 00  |................|
00000070  08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  |................|
00000080  18 03 00 00 00 00 00 00  18 03 00 00 00 00 00 00  |................|
00000090  18 03 00 00 00 00 00 00  1c 00 00 00 00 00 00 00  |................|
000000a0  1c 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
000000b0  01 00 00 00 04 00 00 00  00 00 00 00 00 00 00 00  |................|
```

С помощью команды **strings** выводим те части файла, которые могут быть представлены в виде текста. Наименьшая длина строки передается ключом **–n**.

```
dari@Ubuntu:~$ strings -n10 /bin/sort
/lib64/ld-linux-x86-64.so.2
libpthread.so.0
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
pthread_join
pthread_mutex_init
pthread_sigmask
pthread_cond_signal
pthread_cond_init
pthread_cond_wait
pthread_mutex_lock
__errno_location
pthread_mutex_unlock
pthread_create
__stpcpy_chk
__printf_chk
clearerr_unlocked
dcngettext
```

2. Подсчитать общее количество файлов (каталогов) в каталоге /etc.

Используем конвейер: вывод по одному файлу каталога в строке с помощью **ls -1** перенаправляем через | команде **wc –l**, которая считает число строк.

```
dari@Ubuntu:~$ ls -1 /etc | wc -l
227
```

3. Найти общее количество процессов, выполняющихся в системе в данный момент.

Аналогично перенаправляем на подсчет список процессов, который получаем с помощью **ps aux**.

```
dari@Ubuntu:~$ ps aux | wc -l
197
```

4. Вывести список выполняющихся процессов, в именах которых присутствует слово manager и отсутствует слово grep.

Из всех процессов (**ps aux**) с помощью команды **grep manager** ищем процессы с заданным словом, а с помощью **grep –v grep** исключаем те, где заданное слово отсутсвует.

```
dari@Ubuntu:~$ ps aux | grep manager | grep -v grep
root         119  0.0  0.0      0     0 ?        I<   11:44   0:00 [charger_manager]
```

5. Создать текстовый файл, содержащий набор строк вида:

123
178
176
755
713
873
180

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

С помощью редактора nano создаю и открываю текстовый файл, куда записываю строки с числами.

```
dari@Ubuntu:~$ nano task5.txt
```

Сохраняю изменения и закрываю редактор (**Ctrl+X**).

С помощью **grep “7[1|3|5]”** выбираем только те строки, где обязательно есть цифра 7, а за ней – 1, 3 или 5.

```
dari@Ubuntu:~$ grep "7[1|3|5]" task5.txt
755
713
873
```

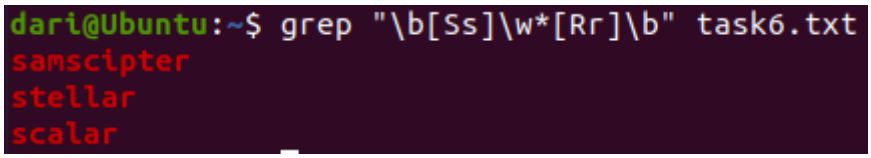
6. Создать текстовый файл, содержащий набор строк вида:

starfish
starless
samscripter
stellar
microsrar
ascender
sacrifice
scalar

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

Аналогично создаем текстовый файл.

Используем grep и регулярное выражение **“\b[Ss]\w*[Rr]\b”** (метасимвол **\b** ставится там, где слово должно начинаться или заканчиваться, **[Ss]** и **[Rr]** – заданные буквы в любом регистре, **\w** замещает любые символы, ***** означает, что предшествующий символ может как быть, так и нет.



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.

Создаем файл с адресами, через cat просматриваем.

Используем команду **grep –E** (расширенные регулярные выражения) и выражение ‘**\b[A-Za-z0-9._%=?^+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}\b**', где

\b – граница слова,

[A-Za-z0-9._%=?^+-] – имя пользователя с допустимыми символами**,**

[A-Za-z0-9.-] – доменное имя**,**

\. – обязательная точка после**,**

[A-Za-z]{2,6} – домен верхнего уровня от 2 до 6 символов.

```
dari@Ubuntu:~$ cat task7.txt
user@mail.com
invalid@address
hiiiii
680
@nope.com
test.user@domain.org
meow;@meow.ru

dari@Ubuntu:~$ grep -E -o '\b[A-Za-z0-9._%=?^+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}\b' task7.txt
user@mail.com
test.user@domain.org
```

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.

Создаем файл, просматриваем. Снова используем **grep –E** и выражение для проверки корректности IP-адреса, в котором 4 раза повторяется блок вида (25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?), разделенный обязательной точкой **‘\.’** . Блок охватывает все допустимые значения (25[0-5] — значения 250–255, 2[0-4][0-9] — значения 200–249; [01]?[0-9][0-9]? — значения 0–199).

```
dari@Ubuntu:~$ cat task8.txt
127.0.0.2
255.255.255.255
12.34.56
123.256.0.0
1.23.099.255
0.79.378.111
dari@Ubuntu:~$ grep -E '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][
0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][
0-9]|[01]?[0-9][0-9]?)' task8.txt
127.0.0.2
255.255.255.255
1.23.099.255
```

С помощью **tr –s** можно удалить дублирующиеся символы.

```
dari@Ubuntu:~$ echo "heeelloooo" | tr -s "eo"
hello
```

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.**

Создаем текстовый файл с номерами телефонов. Корректные – от 000 до 399. Используем **grep –E '\b[0-3][0-9]{2}\b'**, где **[0-3]** — первая цифра от 0 до 3; **[0-9]{2}** — две любые цифры после неё.