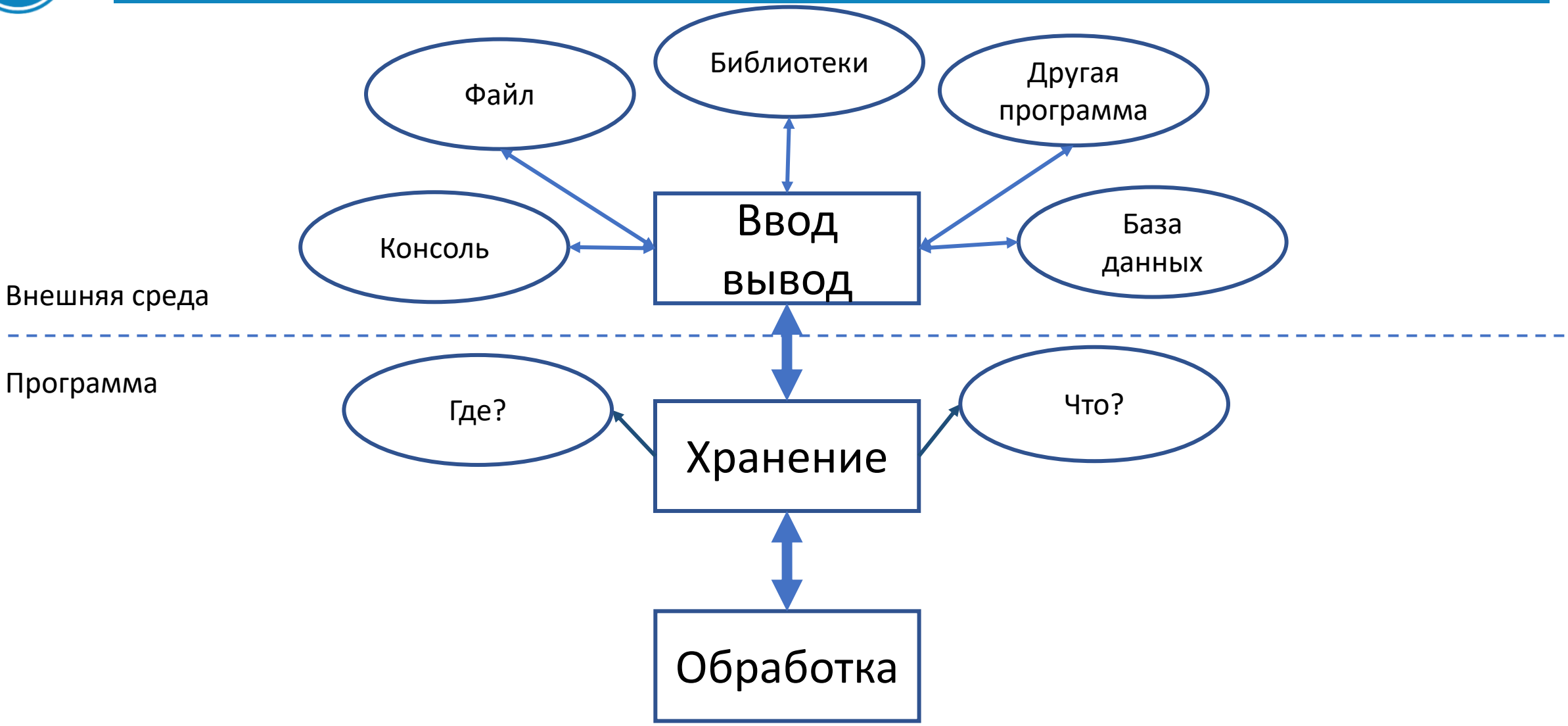
Факультатив по программированию на языке C

Занятие 2
Основы работы с командной строкой Linux
Основы языка C

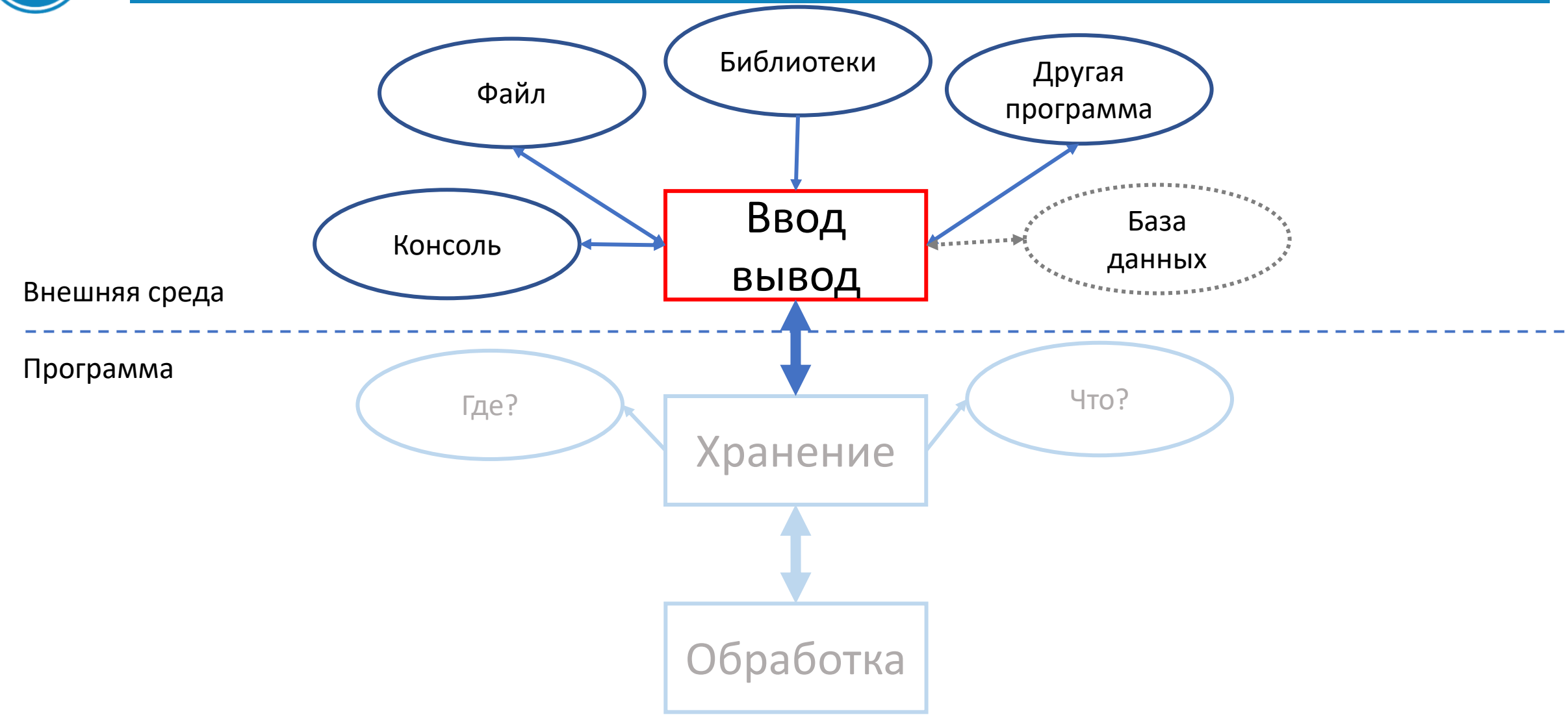
План занятий

№	Тема	Описание
1	Введение в курс	Языки программирования. Основы работы с Linux.
2	Основы языка С	Написание и компиляция простейших программ с использованием gcc. Правила написания кода. Разбиение программы на отдельные файлы. Make файлы
3	Ввод данных. Библиотеки	Работа со вводом/выводом. Статические и динамические библиотеки. Компиляция.
4	Язык ассемблера	Основы анализа программ на языке ассемблер.
5	Хранение данных. Память	Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы.
6	Хранение данных.	Стек, куча. Типы данных. Преобразования типов. Gdb и отладка Хранение различных типов данных. Указатели. Передача аргументов в функцию по указателю.
7	Обработка данных	Безопасные функции. Битовые операции – сдвиги, логические операции. Битовые поля.
8	Программирование под встраиваемые ОС	Работа с микрокомпьютером Raspberry Pi

Дерево языка



Дерево языка



Работа с командной строкой

```
sab@SAB:.../Lesson1$
```

№	Команда	Описание	Пример
0	man	Описание работы команды	man ls
1	pwd	Показать текущее местонахождение	~/SAB$ pwd /home/user/SAB
2	ls	Позволяет просмотреть содержимое текущего каталога	~/SAB$ ls 1 1.txt
3	cd <путь к директории>	Перейти в другую директорию	~$ cd ~ /SAB/2 (полный путь) или ~/SAB$ cd 2 (короткий путь)
4	mkdir <название директории>	Создание директории	~/SAB$ mkdir 1
5	touch <название файла>	Создание файла	~/SAB$ touch 1.txt
6	nano <название файла>	Редактирование файла	~/SAB$ nano 1.txt
7	cp <что_копировать куда_копировать>	Копирование файла	~/SAB/1$ cp 1.txt ~/SAB/2
8	cp -r <путь_к_папке путь_к_новому_месту>	Копирование директории	~/SAB/1$ cp -r 1 ~/SAB/2
9	mv <что_переместить куда_переместить>	Переместить файл	~/SAB/1$ mv 1.txt ~/SAB/2
10	rm <название файла>	Удалить файл	~/SAB/1$ rm 1.txt
11	rm -r <название файла>	Удалить директорию	~/SAB/1$ rm -r 1

Потренируемся

1. Откройте терминал Linux
2. Создайте директорию. Назовите ее «Task1»
3. Войдите внутрь директории
4. Создайте внутри нее еще две директории – «Task1_1» и «Task1_2»
5. Войдите внутрь директории Task1_1
6. Создайте внутри ее файл с названием «File_1» и еще одну директорию «Task1_1_1»
7. Откройте созданный файл и запишите туда любую информацию. Закройте его
8. Скопируйте «File_1» и «Task1_1_1» в директорию «Task1_2»
9. Опуститесь на уровень ниже и удалите директорию Task1_1
10. Из данной директории переместите «File_1» в директорию «Task1»

Перенаправление ввода и вывода

Ввод и вывод распределяется между тремя стандартными потоками:

- **stdin** — стандартный ввод (клавиатура),- - 0
- **stdout** — стандартный вывод (экран), - 1
- **stderr** — стандартная ошибка (вывод ошибок на экран). - 2

< file — использовать файл как источник данных для стандартного потока ввода.
> file — направить стандартный поток вывода в файл (перезапись)
2> file — направить стандартный поток ошибок в файл (перезапись)
>>file — направить стандартный поток вывода в файл (добавление)
2>>file — направить стандартный поток ошибок в файл. (добавление)
&>file или **>&file** — направить с.п. вывода и с.п. ошибок в файл.

Перенаправление ввода и вывода

< file — использовать файл как источник данных для стандартного потока ввода.
> file — направить стандартный поток вывода в файл (перезапись)
2> file — направить стандартный поток ошибок в файл (перезапись)
>>file — направить стандартный поток вывода в файл (добавление)
2>>file — направить стандартный поток ошибок в файл. (добавление)
&>file или **>&file** — направить с.п. вывода и с.п. ошибок в файл.

```
sab@SAB: /$ ps > 1.txt
sab@SAB: /$ cat 1.txt
sab@SAB: /$ ps >> 1.txt
sab@SAB: /$ ps qq > 1.txt
sab@SAB: /$ ps qq 2> 1.txt
```

Grep

Grep (*global regular expression printer) – утилита командной строки, позволяющая производить поиск строки в файле.*

grep [*ключи*] шаблон [*имя_файла ...*]

Ключ	Описание
-c	Выдает только количество строк, содержащих выражение.
-h	Скрывает вывод названия файла, в котором было обнаружено вхождение. Используется при поиске по нескольким файлам.
-i	Игнорирует регистр символов при поиске.
-l	Выдает только имена файлов, содержащих сопоставившиеся строки.
-n	Выдает перед каждой строкой ее номер в файле (строки нумеруются с 1).
-s	Скрывает выдачу сообщений о не существующих или недоступных для чтения файлах.
-v	Выдает все строки, за исключением содержащих выражение.
-E	Поиск с использованием регулярных выражений
-o	Вывод только обнаруженных символов
-r	Рекурсивный поиск

Grep (примеры)

```
sab@SAB: /$ grep Hello file.txt
```

Ищем строку ***Hello*** в файле ***file.txt***

```
sab@SAB: /$ grep -r Hello .
```

Ищем строку ***Hello*** во ***всех*** файлах текущей директории

```
sab@SAB: /$ grep -c Hello file.txt
```

Ищем число вхождений строки ***Hello*** в файле ***file.txt***

Grep (пример с директориями)

Задача:

Необходимо быстро найти пароль от телефона среди множества других паролей.

```
sab@SAB: /$ ./script.sh 5 5 5 100
```

Генерируем 5 директорий, в каждой 5 поддиректорий, в каждой 5 файлов, в каждом из которых 100 строк. В одной из них содержится строка: Password_phone:XXXXXX

```
sab@SAB: /$ grep -R "Password_phone" ./Files
```

Производим поиск по директории *Files*

```
sab@SAB: /$ rm -rf Files/
```

Удаляем созданные директории

Регулярные выражения

Регулярные выражения - инструмент для поиска текста по шаблону.

Метасимвол	Описание работы
****	начало буквенного спецсимвола
^	указывает на начало строки
$	указывает на конец строки
*****	указывает, что предыдущий символ может повторяться 0 или больше раз
+	указывает, что предыдущий символ должен повторится больше один или больше раз
?	предыдущий символ может встречаться ноль или один раз
{n}	указывает сколько раз (n) нужно повторить предыдущий символ
{N,n}	предыдущий символ может повторяться от N до n раз
.	любой символ кроме перевода строки
[az]	любой символ, указанный в скобках
x|y	символ x или символ y
[^az]	любой символ, кроме тех, что указаны в скобках
[a-z]	любой символ из указанного диапазона
[^a-z]	любой символ, которого нет в диапазоне
[:alpha:]	является алфавитным символом
[:digit:]	является числом

Регулярные выражения (примеры)

Поиск содержимого файлов, начинающихся с символов A или B

```
sab@SAB: /$ grep -re ‘^[AB]’ .
```

Поиск количества строк в каждом файле, содержащие подряд две буквы «в»

```
sab@SAB: /$ egrep -rc "[в]{2}" .
```

Поиск содержимого файлов, содержащих слова «Вы или вы»

```
sab@SAB: /$  grep -re ‘[Вв]ы' .
```

Поиск в файле IPv4 адресов (*для любителей реальной практики*)

```
sab@SAB: /$ grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' ip.txt
```

Задача на подумать…

Поиск в файле IPv4 адресов (*для любителей реальной практики*)

```
sab@SAB: /$ grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' ip.txt
```

Необходимо ввести ограничение на *ip* адреса

Адреса:

123.321.234.712	-	(>255)
999.999.999.999	-	
192.168.5.5	+	
10.0.0.4	+	
10.000.000.04	-	(не более 1 нуля подряд)

Pipes (Каналы)

Каналы используются для перенаправления потока из одной программы в другую.

```
sab@SAB: /$ ps | grep p
sab@SAB: /$ ps > 1.txt; ls >> 1.txt; cat 1.txt | grep a
```

Несколько полезных команд Linux

hexdump — показывает шестнадцатеричное представление данных, поступающих на стандартный поток ввода.
cat — считывает данные со стандартного потока ввода и передает их на стандартный поток вывода.
sudo - запуск программы от имени других пользователей, а также от имени суперпользователя.
bc – калькулятор
python3 – среда разработки Python

```
sab@SAB: /$ echo "10*10" | bc
sab@SAB: /$ hexdump 1.txt
```