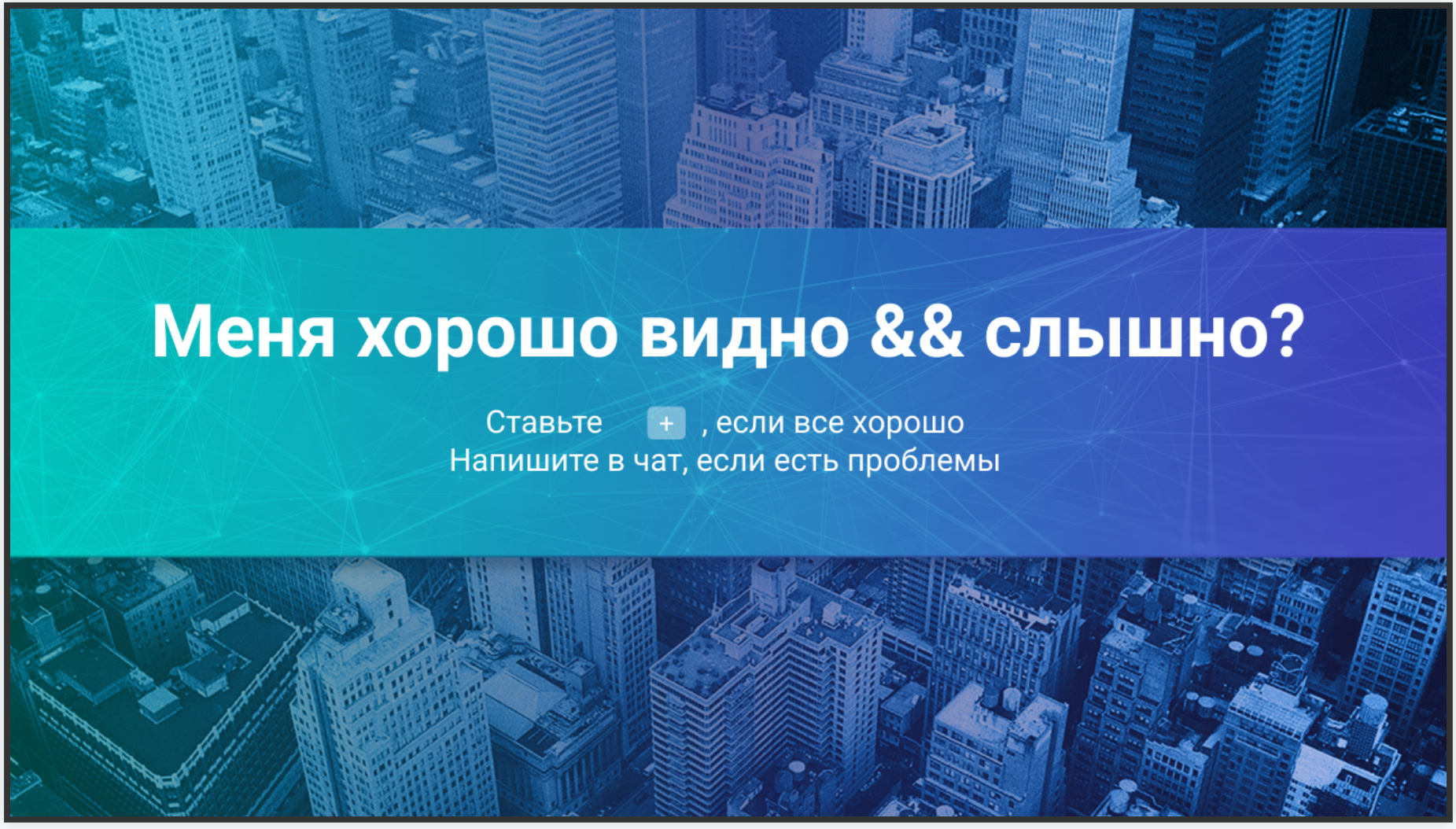

OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование


Меня хорошо видно && слышно?
Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

НЕ ЗАБЫТЬ ВКЛЮЧИТЬ ЗАПИСЬ!!!

grep sed, awk и другие

Цели

- научиться работать с утилитами фильтрами
- читать/составлять регулярные выражения
- использовать потоковые редакторы
- добавлять новые функции в систему

Зачем

В Unix/Linux все есть текст:

- исходные тексты программ, включая скрипты на Shell
- конфигурационные файлы /etc
- лог файлы
- основной формат ввода/вывода данных для программ и утилит

Утилиты фильтры текста

Что делают эти команды?

- cat tac
- head tail
- sort uniq
- wc
- grep
- rev
- paste
- cut
- tr

Утилиты фильтры текста

- cat tac - вывести файл целиком
- head tail - вывести начало и конец файла
- sort uniq - сортировка и убрать повторы в отсортированном
- wc - счётчик строк, слов и байт в тексте
- grep - поиск по образцу
- rev - строку перевернуть
- paste - объединить файлы построчно
- cut - вырезать данные из текста
- tr - замена или удаление символов

Пример использования

https://repl.it/@nixuser/textfilterexampledemo

Задание

https://repl.it/@nixuser/textfilterexample-task

Вариант решения

https://repl.it/@nixuser/textfilterexample

Как работать с pipeline?

Составить свой pipeline или читать/ модифицировать готовый

- разбиваем на части
- выполняем каждую часть в отдельности
- добавляем следующую часть
- Важно! вспоминаем аргументы и синтаксис или изучаем новые

Ключи grep

- -A, -B, -C
- -R
- -n
- -i
- -o

Регулярные выражения. Теория.

regular expression или regexp - специальные строки символов, которые задаются для поиска совпадающих фрагментов.

Элементы регулярных выражений

- **литералы** - обычные символы (буквы и цифры)
- **метасимволы** - спецсимволы
  - количество повторов
  - группировка фрагментов
  - позиция в тексте

Класс на 1 символ

- `.` (точка) - заменяет любой символ
  - Пример: `us.r.` = 'user0', 'us rX', 'us9r ' и т.д.
- [ ] символьный класс - заменяет любой символ из перечисленных в скобках
- Пример:
  - `user[0-9]` = 'user0', 'user5', но не равно 'user'
  - `-[abc-]` = '--', '-a', '-b', '-c', но не равно '--a'
- \d, \w, \S и другие - сокращенная форма записи

Группировка и обратные ссылки

Группировка - поместить выражение в скобки (выражение)

- `(\d\d\d.)`
- `(AM|PM)`
- Только в POSIX Extended RE: egrep
- `grep -e AM -e PM`

Экранирование

Поиск по метасимволу: `*, ', []`

Что делать?

`grep ***`

Квантификаторы - регулируем повторы

Квантификаторы указывают, сколько раз может повторяться символ или выражение, после которого указаны.

- `?` - необязательный символ. Пример: https? - совпадёт http и https
- `*` - любое количество символов, включая нулевое. Примеры: `.*`, `[[:digit:]]*`
- `+` - не менее одного символа. Примеры: `[a-d]+`, `(02:)+`

Интервалы (интервальные квантификаторы)

- {число} - количество повторов выражения перед интервалом
- {число1,число2}
- {число}

Примеры регулярных выражений

Читаем, модифицируем.

https://regexr.com/55ir3

Задание составить регулярное выражение

https://regexr.com/55gbs

- ищем MAC адрес
- ищем IP адрес

Поточные редакторы

Применяют если недостаточно стандартных фильтров либо заменяют программы фильтры

- изменяют текст построчно
- включены в стандарт (есть в минимальных системах типа Busybox, embedded)
- sed
- awk

sed

```
[ addr [ ,  addr ] ] cmd [ args ]
```

Команды:

- d -- удалить строку

```
who | sed -e '10 d'
who | sed -e '2,4 d'
who | sed -e '/pts/ d'
```

- s -- замена по регулярному выражению

```
who | sed -e "s/USER/user/g"
```

awk

Реализации: 'awk', 'gawk', 'mawk', 'nawk'

```
awk --version
```

Типы задач:

1. File spacing,
2. Numbering and calculations,
3. Text conversion and substitution,
4. Selective printing of certain lines,
5. Selective deleting of certain lines.

Добавим пустые строки

```
awk '1; { print "" }'    filname.ext
```

- Как читать:

```
 “pattern { action statements }“.
```

- “1" (всегда true)

Можно пропустить либо pattern либо action

- pattern тогда действие action применяется к каждой строке
- action (действие) По умолчанию выполняется ‘{ print }’.

```
awk '{ print } { print "" }'
```

Подсчет и вычисления

```
ls -l | awk 'BEGIN {sum=0} {sum=sum+$5} END {print sum}'
awk -F',' '{s+=$4}END{print s+0}'
```

```
awk 'BEGIN {printf "%.3f\n", 10 / 3}'
```

Нумерация строк (аналог nl)

```
awk '{ print FNR "\t" $0 }'   filname.ext
```

FNR - File Line Number

Реализация wc -l

```
awk 'END { print NR }'   filname.ext
```

NR - количество строк

Преобразование текста

в нижний регистр

```
awk '{ print tolower($0) }'
```

замена в каждой строке

```
awk '{gsub(/foo/,"bar")}; 1'
```

Выборка/удаление строк

- print only lines which match regular expression (emulates "grep")

```
awk '/regex/'
```

- print only lines which do NOT match regex (emulates "grep -v")

```
awk '!/regex/'
```

Выборка/удаление строк

- print any line where field #5 is equal to "abc123"

```
awk '$5 == "abc123"'
```

- remove blank lines

```
awk NF
```

- remove duplicate, consecutive lines (emulates "uniq")

```
awk 'a !~ $0; {a=$0}'
```

Получить последний элемент

```
echo 'maps.google.com' -> '.com'
```

Получить последний элемент

```
echo 'maps.google.com' | awk -F '.' '{print $NF}'
```

```
echo 'maps.google.com' | rev | cut -d'.' -f 1 | rev
```

Пример удаляем пробелы (trim)

```
msg="     Linux Administrator Courses       "
```

```
shopt -s extglob
shopt  extglob
echo "____${msg##*( )}____"
echo "____${msg%%*( )}____"
```

```
echo “$msg” | sed 's/ *$//g'
echo “$msg” | sed -e 's/[[:space:]]\+/ /g'
```

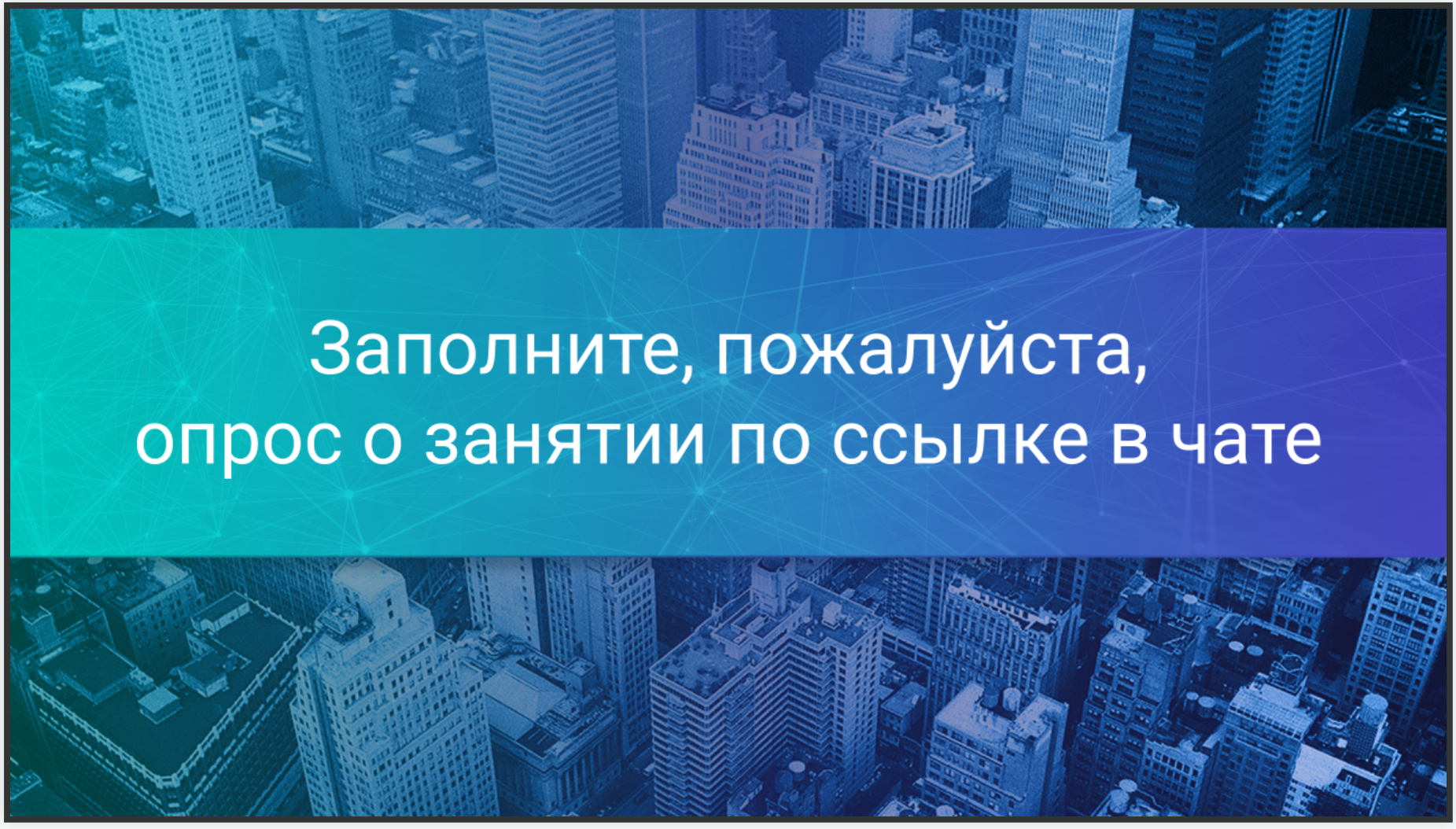
```
echo "$msg" |  awk '{gsub(/^[ \t]+| [ \t]+$/,""); print $0 }'
```

```
echo “$msg” | xargs
```

Рефлексия

Отметьте 3 пункта, которые вам запомнились с вебинара

Что вы будете применять в работе из сегодняшнего вебинара?


Заполните, пожалуйста,
опрос о занятии по ссылке в чате