

Семинар #4: Linux. Поток, перенаправление. Практика.

Как сдавать задачи

Для сдачи ДЗ вам нужно создать репозиторий на GitLab (если он ещё не создан) под названием `devtools-homework`. Структура репозитория должна иметь вид:

```
├── seminar4_pipelines/
│   ├── 01.sh
│   ├── 02.sh
│   └── ...
└── ...
```

Для каждой задачи нужно создать 1 файл. То есть всего в этом задании будет 13 файлов. Подзадачи нужно оформлять в следующем формате:

```
# Subtask a
echo Hello world
# Subtask b
printf "Hello world\n"
# Subtask c
...
```

Следующие задачи: 4a, 4b, 4c, 10a, 10b, 11f, – можно не решать. Они нужны для ознакомления.

Для решения задач: 1l, 4d, 4e нужно ответить на вопросы задачи в файле решения.

Файлы для задания находятся в репозитории в папке `seminar4_pipelines/practice/files`.

Вам нужно скачать эти файлы перед выполнением задания.

Задача 1. Стандартный выход `stdout`

- a. **Привет мир**
Введите команду, которая будет печатать на экран "Hello world". Используйте команду `echo`.
- b. **Привет мир 2**
Введите команду, которая будет печатать "Hello world". Используйте команду `printf`. Вывод должен быть аналогичен предыдущей команде, то есть в конце строки нужно печатать перевод на новую строку.
- c. **Where am I?**
Введите команду, которая будет печатать на экран текущую директорию, в которой вы находитесь.
- d. **Who am I?**
Введите команду, которая будет печатать на экран ваше имя пользователя.
- e. **Время и дата**
Введите команду, которая будет печатать текущие время и дату.
- f. **Календарь**
Введите команду, которая будет печатать календарь текущего месяца.
- g. **Информация о системе**
Введите команду, которая будет печатать полную информацию о системе. Используйте команду `uname -a`.
- h. **cat**
Выполните команду, которая печатает на экран содержимое файла `happiness.txt`.
- i. **tac**
Выполните команду, которая печатает на экран строки файла `happiness.txt` в обратном порядке.
- j. **Начало**
Выполните команду, которая печатает на экран первые 5 строк файла `happiness.txt`.

k. **Конец**

Выполните команду, которая печатает на экран последние 5 строк файла `happiness.txt`.

l. **Строки, слова и символы**

Используйте программу `wc`, чтобы найти количество строк, слов и символов в файле `happiness.txt`. Что делают опции `-l`, `-w` и `-c` этой команды?

m. **less**

Используйте команду `less` для просмотра файла `invisible_man.txt`.

Стрелочки вверх/вниз, `PageUp/PageDown`, `Ctrl-B/Ctrl-F` – перемещение, `q` – выйти из `less`, `g/G` – начало/конец файла, `/строка` – поиск строки.

n. **Опции ls**

- (a) Введите команду `ls`, чтобы напечатать файлы в текущей директории.
- (b) Введите команду `ls -l`, чтобы напечатать файлы в текущей директории. Один файл на строку.
- (c) Введите команду `ls -a`, чтобы напечатать все файлы в текущей директории, в том числе скрытые.
- (d) Введите команду `ls -l`. Подробная информация о файлах в текущей директории.
- (e) Введите команду, которая бы печатала все файлы подробно, но так, чтобы они были отсортированы по их размеру.
- (f) Введите команду, которая бы печатала все файлы подробно, но так, чтобы они были отсортированы по дате последнего изменения.
- (g) Введите команду, которая бы печатала все файлы подробно, но так, чтобы они были отсортированы по их расширению.
- (h) Введите команду, которая бы печатала все файлы текущей директории, а потом все файлы поддиректорий и так далее.

Задача 2. Перенаправление stdout в файл

a. **Запись в файл**

Введите команду, которая будет создавать файл `2a.txt` и записывать туда строку `"Sapere Aude"`. Если файл с таким именем уже существует, то сначала всё его содержимое должно удаляться, а потом уже производиться запись.

b. **Запись времени в файл**

Введите команду, которая будет создавать файл `2b.txt` и записывать туда текущее время. Если файл с таким именем уже существует, то сначала всё его содержимое должно удаляться, а потом уже производиться запись. Текущее время можно напечатать командой:

```
date +%H:%M:%S
```

c. **Дозапись времени в файл**

Введите команду, которая будет создавать файл `2c.txt` и записывать туда текущее время. Если файл с таким именем уже существует, время должно записываться в новую строку в конец файла.

d. **Сортировка строк**

Используйте программу `sort`, чтобы отсортировать все строки в файле `names.txt`. Сохраните отсортированные имена в файле `2d.txt`.

e. **Обращение каждой строки**

Используйте программу `rev`, чтобы обратить каждую строку в файле `names.txt`. Сохраните обращенный файл в имена в файле `2e.txt`.

f. **Конкатенация**

Используйте программу `cat`, чтобы конкатенировать два файла: `dream.txt` и `happiness.txt`. Результат конкатенации запишите в файл `2f.txt`.

g. **Три строки**

Напишите команду, которая будет печатать в файл `2g.txt` три строки:

```
first
second
third
```

Решите эту задачу пятью способами:

- i. С использование опции `-e` команды `echo`.
- ii. С использование команды `printf`.
- iii. С помощью объединения нескольких команд `echo` без использования subshell (круглых скобок). Поместите всю команду в одну строку.
- iv. С помощью объединения нескольких команд `echo` с использованием subshell. Поместите всю команду в одну строку.
- v. С помощью heredoc.

Задача 3. Синтаксис подстановки команд `$(...)`

a. **Ссылка**

В файле `ref.txt` хранится название некоторого файла. Вам нужно выполнить команду, которая читает файл `ref.txt`, считывает оттуда название другого файла и печатает содержимое этого файла на экран.

b. **Привет user**

Введите команду, которая будет создавать файл `3b.txt` и записывать туда строку "Hello `<user>`", где вместо `<user>` будет записано ваше имя пользователя.

c. **alice-1759761949.txt**

Выполните команду (одна строка), которая бы создавала файл по имени `<user>-<seconds>.txt`, где вместо `<user>` команда должна подставить имя пользователя, а вместо `<seconds>` команда должна подставить количество секунд, прошедших с 1 января 1970 года (команда `date +%s`). Команда должна записывать в новый файл календарь за текущий месяц.

Задача 4. Стандартный вход `stdin`

a. **cat из stdin в stdout**

Запустите программу `cat`, не указывая файл. В результате `cat` будет требовать ввести текст через стандартный вход (`stdin`). После ввода строки и нажатия `Enter`, `cat` будет печатать на экран строку, которую вы ввели. Введите строки:

```
one
two
three
```

После каждой строки `cat` будет печатать её же в стандартный выход `stdout`. Для завершения ввода введите `Ctrl-D`.

b. **cat из stdin в файл**

Запустите программу `cat` не указывая названия файла как аргумента, а просто перенаправив вывод в файл:

```
$ cat > 4b.txt
```

В результате `cat` будет требовать ввести текст через стандартный вход. Введите строки:

```
one
two
three
```

и нажмите `Ctrl-D`. Посмотрите на содержимое файла `4b.txt`.

c. **Сортировка строк из stdin**

Запустите программу `sort`, не указав файл в качестве аргумента. В результате `sort` будет требовать ввести текст через стандартный вход. Введите:

```
one
two
three
four
```

и нажмите `Ctrl-D`. Эти строки должны напечататься в алфавитном порядке.

d. **Программы, которые читают из stdin**

Протестируйте, какие из следующих программ читают из `stdin`, если им не передать аргументы, а какие не читают:

```
echo ls wc cp mkdir touch tac head tail rev
```

e. **Аргумент или перенаправление**

Что делают следующие команды и чем они отличаются?

```
$ cat dream.txt
$ cat < dream.txt
$ wc dream.txt
$ wc < dream.txt
```

f. **Перенаправление из файла в stdin**

Введите команду `bash` (одну строку), которая будет вычислять и печатать на экран количество строк файла `invisible_man.txt` в следующем формате:

```
file has <number> lines
```

где вместо `<number>` будет поставляться количество строк.

Задача 5. Pipe (из `stdout` одной программы в `stdin` другой)

a. **Полное обращение**

Выполните команду (одна строка `bash`), которая делает следующее:

- Сортирует строки файла `names.txt`.
- Обращает полученные отсортированные строки. Первая строка должна стать последней, вторая – предпоследней и так далее.
- Обращает каждую строку этого же файла.
- Сохраняет результат в файл `totally_reversed.txt`.

Первые строки результирующего файла должны иметь следующий вид:

```
yrahcaZ
araY
reivaX
...
```

Используйте программы `sort`, `tac` и `rev`.

b. **Удаление дубликатов**

Файл `logs.txt` содержит список сайтов, при этом некоторые строки повторяются. Необходимо выполнить команду (одна строка), которая:

- Отбирает только уникальные строки.
- Сортирует их в алфавитном порядке.
- Сохраняет результат в файл `uniq_logs.txt`.

В итоговом файле должно остаться ровно 20 строк.

c. **Количество файлов**

Выполните команду, которая будет печатать количество файлов в текущей директории.

d. **Просмотр файлов**

Напишите команду, для просмотра подробной информации (`ls -l`) о всех файлах из директории `/usr/bin` с использованием команды `less`.

e. **Самые часто посещаемые сайты**

Введите команду (одна строка), которая будет печатать топ-5 самых часто встречающихся строк в `logs.txt`.

Задача 6. Перенаправление из stderr

Файл `printerr.py` – это python скрипт, который печатает одну строку в `stdout` и одну строку в `stderr`. Если запустить этот скрипт:

```
$ python printerr.py
```

то оба сообщения напечатаются на экран, так как по умолчанию `stdout` и `stderr` выводятся на экран терминала.

a. **Перенаправление в файлы**

Выполните команду, которая будет перенаправлять поток `stdout` в файл `out.txt`, в поток `stderr` в файл `err.txt`.

b. **Перенаправление в /dev/null**

Выполните команду, которая будет печатать на экран только поток `stderr`, а поток `stdout` игнорировать.

c. **Перенаправление в один файл**

Выполните команду, которая будет перенаправлять оба потока в файл `all.txt`.

d. **Pipe stderr**

По умолчанию pipe работает с `stdout`. Выполните команду, которая будет перенаправлять поток `stderr` на вход программе `rev`. Поток `stdout` должен игнорироваться. В результате на экране должно быть напечатано:

```
.rredts ot seog sihT
```

Задача 7. Wildcards

В директории `wild` содержится набор различных файлов. Выполните команды, которые делают следующее:

a. Печатает на экран все файлы (то есть имена всех файлов).

b. Печатает все файлы, с именами, начинающимися на `article`.

c. Печатает все файлы, с расширением `.txt`.

d. Печатает все файлы, с именем в формате `article?.txt`, где вместо вопроса расположена цифра.

e. Печатает все файлы, с именем в формате `article?.txt`, где вместо вопроса расположена заглавная буква.

f. Печатает все файлы, начинающиеся на `backup` и имеющие в своём названии дату, соответствующую октябрю 2024-го года.

g. Печатает все файлы, начинающиеся на `backup` и имеющие в своём названии дату, соответствующую 2023-му или 2025-му году.

h. Печатает все файлы, начинающиеся на букву `a` или на букву `c`.

i. Печатает все файлы, имена которых содержат хотя бы один символ подчёркивания.

j. Печатает все файлы, имена которых содержат хотя бы одну цифру.

k. Удаляет все файлы, заканчивающиеся на `.bin`.

l. Создаёт папку `logs` и копирует туда все файлы, заканчивающиеся на `.log`.

m. Создаёт файл `filenames.txt` и записывает туда имена всех файлов, которые начинаются на `a` или на `c`.

Задача 8. Brace expansion

- a. Создайте папку `brace` и перейдите в неё. Создайте файлы с именами:

```
hello.txt hello.log hello.xml hello.md hello.json hello.cfg
```

Создать все файлы нужно, используя только одну строку. Сделайте эту строку как можно короче, используя brace expansion.

- b. Создайте файлы с именами:

```
alice1.txt alice2.txt alice3.txt bob1.txt bob2.txt bob3.txt casper1.txt casper2.txt casper3.txt
```

Создать все файлы нужно, используя только одну строку. Сделайте эту строку как можно короче, используя brace expansion.

- c. Выполните команду, которая бы печатала на экран все числа от 1 до 100 с шагом 5. Используйте brace expansion.

- d. Создайте файлы с именами:

```
file001.txt file002.txt file003.txt ... file100.txt
```

Создать все файлы нужно, используя только одну строку. Сделайте эту строку как можно короче, используя brace expansion.

- e. Создайте следующую структуру файлов в папке `brace`:

```
brace/
├── dir_a/
│   ├── 1.txt
│   ├── 2.txt
│   └── 3.txt
├── dir_b/
│   ├── 1.txt
│   ├── 2.txt
│   └── 3.txt
└── dir_c/
    ├── 1.txt
    ├── 2.txt
    └── 3.txt
```

Создать все файлы нужно, используя только одну строку. Сделайте эту строку как можно короче, используя brace expansion.

- f. Перейдите в папку `wild` и напечатайте имена всех файлов с расширениями `.jpg` или `.png`,

Задача 9. find и locate

В этой задаче, под термином "все файлы в директории" подразумевается рекурсивный поиск файлов, то есть все файлы в этой директории плюс все файлы в поддиректориях плюс все файлы в поддиректориях поддиректорий и так далее. Если не сказано иначе, при выполнении команд данной задачи, если это необходимо, вывод в `stderr` нужно игнорировать и не отображать его на экран, а перенаправлять в `/dev/null`.

- a. **Печать всех файлов**

Перейдите в директорию `files`. Напечатайте все файлы в текущей директории, используя программу `find`.

- b. **Печать файлов папки `/etc`**

Выполните команду, которая бы печатала все файлы внутри папки `/etc`.

- c. **Поиск файла**

Напечатайте все файлы по имени `README.md` в директории `/usr`.

- d. **Поиск файлов с определённым расширением**
Напечатайте все файлы с расширением `.sh` (скрипты) в директории `/usr`.
- e. **Поиск директорий**
Напечатайте все директории, внутри папки `/opt`. Все ошибки, возникшие при выполнении этой команды перенаправьте в файл `errors.log`.
- f. **Поиск больших файлов**
Напечатайте рекурсивно все файлы внутри папки `/usr`, чей размер составляет больше 50-ти мегабайт.
- g. **Поиск и копирование**
Создайте папку `scripts`. Выполните команду, которая будет копировать в эту папку все файлы с расширением `.sh` из директории `/usr`. Если есть несколько файлов с одинаковым названием, то можно скопировать любой. Используйте опцию `-exec`.
- h. **Поиск и отображение подробной информации**
Найти все файлы с расширением `.conf` в директории `/etc` и вывести подробную информацию о каждом из них с помощью команды `ls -l`. Используйте опцию `-exec ... {} +`.
- i. **Команда locate**
Установите команду `locate`:

```
$ sudo apt install plocate  
$ sudo updatedb
```

Если `apt` нет, то используйте пакетный менеджер вашей системы вместо `apt`.
Напечатайте все файлы `README.md` в системе с помощью `locate`.
- j. **Сравнение скорости find и locate**
Напечатайте все файлы с расширением `.sh` (скрипты) в директории `/usr`, используя `locate`. Сравните скорость работы `find` и `locate` при выполнении аналогичных операций с помощью команды `time`.

Задача 10. xargs (из stdin в аргументы другой программы)

В этой задаче печать в `stderr` должна игнорироваться.

- a. **Программа xargs**
Перейдите в папку `files` и выполните команду:

```
$ xargs cp
```

В результате команда будет требовать ввести текст через стандартный вход. Введите строку:

```
dream.txt copy.txt
```

и нажмите `Ctrl-D`. Программа `xargs` считала строку из стандартного входа и сама вызовет программу `cp` передав ей `"dream.txt copy.txt"` в качестве аргументов. Затем программа `cp` скопирует файл `dream.txt` в файл `copy.txt`.
- b. **Программа xargs вместе с pipe**
Перейдите в папку `files` и выполните команду:

```
$ echo "dream.txt copy2.txt" | xargs cp
```

В результате выполнения этой команды файл `dream.txt` должен быть скопирован в файл `copy2.txt`.
- c. **Количество строк**
Выполните команду, которая бы считала количество строк во всех файлах папки `scripts`.
- d. **Массовое изменение прав**
Выполните команду, которая будет забирать права на выполнение у все скриптов из папки `scripts`. Проверьте результат с помощью `ls -l`. Выполните команду, которая наоборот будет давать права на выполнение у всех скриптов из папки `scripts`.

e. **Количество строк**

Выполните команду, которая бы считала количество строк во всех файлах папки `/etc` и рекурсивно всех файлах всех поддиректорий. Используйте опции `-print0` и `-0`, чтобы корректно обработать файлы, содержащие пробелы в названиях.

f. **Подробная информация об определённых файлах**

Найдите все файлы в папке `/etc`, в названии которых встречается дефис и выведите подробную информацию о всех этих файлах. Для просмотра используйте `less`.

g. **Самые большие скрипты**

Введите команду, которая будет печатать топ-5 самых больших файлов по размеру с расширением `.sh` в директории `/usr`.

Задача 11. `grep`

В этой задаче печать в `stderr` должна игнорироваться.

a. **Поиск в файле**

Выполните команду, которая будет выводить на экран все строки в файле `urls.txt`, которые содержат подстроку `"data"`.

b. **Поиск в файле с исключением**

Выполните команду, которая будет выводить на экран все строки в файле `urls.txt`, которые **не** содержат подстроку `"https"`.

c. **Поиск во всех файлах директории**

Выполните команду, которая бы искала строку `"debug"` во всех строках всех файлов директории `/etc`. Для каждого совпадения команда должна печатать название файла, номер строки и саму строку.

d. **Файлы, содержащие строку**

Выполните команду, которая бы печатала все файлы, в директории `/etc`, содержащие строку `"config"`

e. **Количество во всех файлах директории**

Выполните команду, которая бы печатала суммарное количество подстрок `"config"` во всех строках всех файлов директории `/etc`.

f. **`grep` через `stdin`**

Выполните команду

```
$ grep "hello"
```

В результате команда будет требовать ввести текст через стандартный вход. Введите строки:

```
hello, alice
i am bob
ok, hello, bob
nice to meet you
```

затем нажмите `Ctrl-D`. Команда должна вывести все строки из `stdin`, содержащие подстроку `"hello"`.

g. **Список файлов**

Напечатайте все файлы из папки `wild`, которые имеют расширение `.log`. Используя команды `ls` и `grep`.

h. **Поиск в истории**

Напечатайте все команды в истории (команда `history`), которые содержат подстроку `"find"`.

i. **Поиск `ip`-адресов**

Выполните команду, которая бы искала все записи `ip`-адресов во всех строках всех файлов директории `/etc`. Используйте `grep -E`.

Задача 12. tee (одновременный вывод в stdout и файлы)

- a. **Hello Tee**
Выполните команду (одну строку), которая выводит на экран строку "first", а также записывает в файл `output.txt` такую же строку.
- b. **Вторая строка**
Выполните команду (одну строку), которая выводит на экран строку "second", а и также добавляет такую же строку в конец файла `output.txt`.
- c. **tee в несколько файлов**
Выполните команду (одну строку), которая печатает на экран текущее время и дату (команда `date`), а также записывает такой же вывод в файлы `out1.txt`, `out2.txt`, ... , `out9.txt`.
- d. **tee вместе с grep**
Используйте команду `ls -l` для просмотра содержимого папки `/etc`. Передайте вывод этой команды в `grep`, чтобы отфильтровать только строки, содержащие "conf" в именах файлов. Результат фильтрации одновременно сохраните в файле `conf_files.txt` и отобразите на экране.

Задача 13. Поиски в java-проекте

Клонируйте репозиторий для библиотеки `jsoup`:

```
git clone https://github.com/jhy/jsoup.git
```

Репозиторий должен создаться в папке `jsoup`.

- a. **Все файлы проекта**
Выполните команду, которая будет печатать все файлы проекта. Используйте команду `find`.
- b. **Все файлы с определённым расширением**
Выполните команду, которая будет печатать все файлы проекта, с расширением `.html`.
- c. **Количество строк в файлах**
Выполните команду, которая будет печатать количество строк в каждом файле проекта.
- d. **Поиск в файлах**
Найдите и выведите все строки во всех файлах репозитория, содержащие слово `parser`. Используйте команду `grep`.
- e. **Поиск в файлах - вывод только названий**
Найдите и выведите названия всех файлов проекта, в которых содержится слово `parser`.
- f. **Поиск в файлах без учёта регистра**
Найдите все строки во всех файлах, содержащие слово `connect`, игнорируя регистр (то есть, найдите "Connect", "connect", "CONNECT" и т. д.).
- g. **Поиск в файлах с определённым расширением**
Найдите все строки в файлах с расширением `.java`, содержащие слово `connect`, игнорируя регистр. Для каждого совпадения напечатайте файл и номер строки.
- h. **Поиск с контекстом**
Найдите и выведите все строки во всех файлах репозитория, содержащие слово `StringBuilder`. Для каждого совпадения выведите 3 строки до совпадения и 3 строки после совпадения. Используйте `less` для удобного просмотра результата.
- i. **Поиск объявлений классов**
Найдите все объявления классов, используя регулярные выражения (`grep -E`). Для нахождения всех классов можно использовать следующее регулярное выражение:

```
"^\\s*(public|protected|private|abstract|final)?\\s*class\\s+[A-Za-z_][A-Za-z0-9_]*"
```


Для каждого совпадения напечатайте ещё 15 строчек кода после совпадения.
- j. **Поиск методов с определённым названием**
Используйте регулярные выражения для поиска во всем репозитории всех объявлений методов, которые начинаются со слова `is` (например, `public boolean isAttribute()`).

Необязательные задачи (никак ни учитываются)

Задача 1. Системные вызовы для работы с файлами

В следующих задачах нужно использовать системные вызовы `open`, `close`, `read`, `write`, `dup2`.

a. **Копирование**

Напишите программу на C – простой аналог программы `cp`.

b. **Конкатенация**

Напишите программу на C – простой аналог программы `cat`.

c. **tee**

Напишите программу на C – простой аналог программы `tee`.

Что нужно добавить

- Объединение команд.
- Скобочки $()$ - сабшелл.
- Подстановка процессов $<(\dots), >(\dots)$.
- Использование дефиса $(-)$