

Задание 1

Необходимые знания

- `&` в `shell`
- Как работает команда `cat`
- Как работает команда `clear`
- Как работает команда `wc`

Задание

- Запустите скрипт `background.sh` в фоновом режиме.
- Создайте текстовый файл и выведите его содержимое на экран терминала.
- Очистите окно терминала.
- Посчитайте количество символов в файле.

1. `&` в `shell`

Командная оболочка для управления системой и запуска скриптов.

Применение: автоматизация задач, работа с файлами, настройка серверов.

2. Команда `cat`

Выводит содержимое файла в терминал или объединяет несколько файлов.

Пример: `cat file.txt` покажет текст файла.

3. Команда `clear`

Очищает окно терминала от предыдущих команд и вывода.

Аналог: `Ctrl + L` (работает в большинстве терминалов).

4. Команда `wc`

Считает строки, слова и символы в файле или введенном тексте.

Ключи:

`-l` — строки,

`-w` — слова,

`-m` — символы.

```

● @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ ./background.sh &
[1] 21926
1 sec
● @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ 2 sec
3 sec
4 sec
5 sec
6 sec
7 sec
8 sec
9 sec
10 sec
Done!
echo "Веном" > venom.txt
[1]+  Done                  ./background.sh
● @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ cat venom.txt
Веном
○ @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ █

```

Clear сработал

```

● @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ wc -m venom.txt
6 venom.txt
○ @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ █

```

Задание 2

Необходимые знания

- Как работает команда `grep`
- Как работает редирект в `bash`
- Что такое `pipe` в `bash`
- Специальные девайсы в *nix системах (`/dev/null` , `/dev/full` и т.д.)

Задание

- С помощью команды `grep` , используя `pipe` и редирект (`>`), запишите в файл `with_cake.txt` все строки из файла `cake_rhymes.txt` , в которых есть слово `cake` .
- Сделайте команду `rm` "тихой", используя редирект в `/dev/null` .

1. Команда `grep`

Поиск текста в файлах или выводе команд по шаблону (регулярным выражениям).

Пример:

`grep "error" log.txt` — найти строки с "error" в файле.

2. Редирект в Bash (>, >>, <)

Перенаправление ввода/вывода:

- > — перезапись файла (echo "text" > file.txt).
- >> — добавление в конец файла (echo "text" >> file.txt).
- < — передача файла как ввода (sort < file.txt).

3. Pipe (|) в Bash

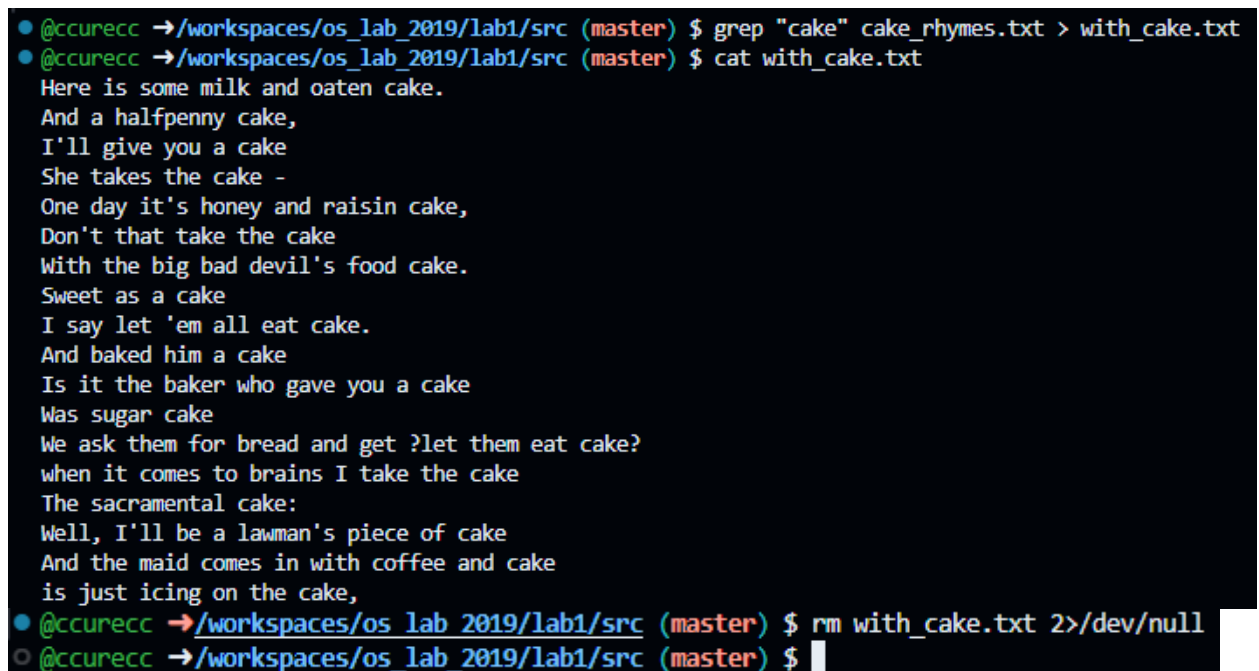
Передача вывода одной команды на вход другой.

Пример:

cat file.txt | grep "hello" | wc -l — подсчитать строки с "hello".

*4. Специальные устройства в nix (/dev/null, /dev/full и др.)

- /dev/null** — "чёрная дыра": удаляет всё, что в него записывают.
Пример: command > /dev/null — скрыть вывод команды.
- /dev/full** — при записи сразу возвращает ошибку "No space left".
Пример: echo "test" > /dev/full → ошибка.
- /dev/zero** — генерирует нулевые байты.
- /dev/random//dev/urandom** — генератор случайных чисел.



```
@ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ grep "cake" cake_rhymes.txt > with_cake.txt
@ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ cat with_cake.txt
Here is some milk and oaten cake.
And a halfpenny cake,
I'll give you a cake
She takes the cake -
One day it's honey and raisin cake,
Don't that take the cake
With the big bad devil's food cake.
Sweet as a cake
I say let 'em all eat cake.
And baked him a cake
Is it the baker who gave you a cake
Was sugar cake
We ask them for bread and get ?let them eat cake?
when it comes to brains I take the cake
The sacramental cake:
Well, I'll be a lawman's piece of cake
And the maid comes in with coffee and cake
is just icing on the cake,
@ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ rm with_cake.txt 2>/dev/null
@ccurecc →/workspaces/os_lab_2019/lab1/src (master) $
```

/dev/null — игнорирует, все что в него записывают

2 — stderr вывод ошибок

Задание 3

Необходимые знания

- Права в linux
- Команда `chmod`
- Переменные окружения
- Команда `date` в bash
- Что такое `shebang`

Задание

- Сделайте файл `hello.sh` исполняемым, выполните его.
- Напишите `bash` скрипт, который выводит текущий путь, текущую дату и время, а также содержимое переменной окружения `PATH`.

1. Права в Linux

Система контроля доступа к файлам и папкам.

Три типа прав:

- **r (read)** — чтение
- **w (write)** — запись
- **x (execute)** — выполнение

Три категории пользователей:

- **owner** (владелец)
- **group** (группа)
- **others** (остальные)

Пример:

-rwxr-xr-- — владелец может читать/писать/исполнять, группа — читать/исполнять, остальные — только читать.

2. Команда chmod

Изменяет права доступа к файлам/папкам.

Способы задания прав:

1. Цифровой (octal):

```
chmod 755 script.sh # rwxr-xr-x
```

2. Символьный:

`chmod u+x script.sh` # добавить исполнение для владельца

`chmod o-w file.txt` # запретить запись для остальных

3. Переменные окружения

Динамические переменные, влияющие на поведение программ и оболочки.

Примеры:

PATH — список путей для поиска исполняемых файлов.

HOME — домашняя директория пользователя.

Управление:

`export VAR=value` # установить переменную

`echo $PATH` # вывести значение

`unset VAR` # удалить переменную

4. Команда date в Bash

→ Выводит или устанавливает системную дату и время.

Примеры:

`date` # текущая дата и время

`date +%Y-%m-%d` # форматированный вывод (2024-05-10)

`date -d "tomorrow"` # дата завтрашнего дня

5. Shebang (#!)

Указание интерпретатора для скрипта в первой строке.

Примеры:

`#!/bin/bash` # запускать через Bash

`#!/usr/bin/python3` # запускать через Python 3

Как работает:

При запуске скрипта (например, `./script.sh`) система читает shebang и выполняет команду:

`/bin/bash script.sh`

```
[Предварительный просмотр] README.md | $ update.sh | ≡ with_cake.txt | $ script.sh U X
```

```
lab1 > src > $ script.sh
```

```
1 echo "Текущий путь: $(pwd)"
```

```
2
```

```
3 echo "Текущая дата и время: $(date)"
```

```
4
```

```
5 echo "Содержимое переменной PATH: $PATH"
```

Задание 4* (повышенной сложности)

🔗 Необходимые знания

- Как работать с аргументами в `bash` скриптах
- Как работает команда `od`
- Специальные устройства в *nix системах
- Как работает редирект

Задание

- Напишите скрипт `average.sh`, который выводит количество и среднее арифметическое его входных аргументов.
- С помощью `bash` и `dev/random` создайте файл `numbers.txt` из 150 случайных чисел.
- "Скормите" скрипту `average.sh` значения из файла `numbers.txt`.

1. Аргументы в Bash-скриптах

Передача данных в скрипт при запуске.

Доступ к аргументам:

- `$0` — имя скрипта.
- `$1, $2, ..., $9` — 1-й, 2-й, ..., 9-й аргумент.
- `$@` — все аргументы.
- `$#` — количество аргументов.

2. Команда `od` (Octal Dump)

→ Выводит содержимое файла в восьмеричном, шестнадцатеричном или другом формате.

Основные флаги:

`-c` — символы (ASCII).

`-x` — шестнадцатеричный формат.

`-o` — восьмеричный формат.

*3. Специальные устройства в `linux`

Виртуальные файлы для взаимодействия с ядром системы.

Основные устройства:

`/dev/null` — "чёрная дыра" (удаляет все записанные данные).

`echo "текст" > /dev/null` # Ничего не произойдет

`/dev/zero` — генерирует нулевые байты.

`dd if=/dev/zero of=file.bin bs=1M count=10` # Создаст 10 МБ нулей

/dev/random//dev/urandom — генераторы случайных чисел.

```
• @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ touch avarage.sh
• @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ touch numbers.txt
• @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ for i in {1..150}; do
  >   echo $((RANDOM % 1000)) >> numbers.txt
  > done
```

[Предварительный просмотр] README.md X \$ update.sh with_cake.txt numbers.txt U X

lab1 > src > numbers.txt

```
1 259
2 815
3 213
4 423
5 546
6 249
7 628
8 888
9 393
10 671
11 853
12 856
13 653
14 138
15 976
```

[Предварительный просмотр] README.md \$ update.sh with_cake.txt numbers.txt U \$ avarage.sh U X

lab1 > src > \$ avarage.sh

```
1 #!/bin/bash
2
3 # Проверяем, есть ли аргументы
4 if [ $# -eq 0 ]; then
5     echo "Нет входных аргументов."
6     exit 1
7 fi
8
9 # Подсчет количества аргументов
10 count=$#
11 sum=0
12
13 # Подсчет суммы входных аргументов
14 for num in "$@"; do
15     sum=$((sum + num))
16 done
17
18 # Вычисление среднего арифметического
19 average=$(echo "scale=2; $sum / $count" | bc)
20
21 # Вывод результата
22 echo "Количество аргументов: $count"
23 echo "Среднее арифметическое: $average"
```

```
• @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ chmod +x avarage.sh
• @ccurecc →/workspaces/os_lab_2019/lab1/src (master) $ ./avarage.sh $(cat numbers.txt)
Количество аргументов: 150
Среднее арифметическое: 527.51
```