**Практическое занятие №1. Введение, основы работы в командной строке**

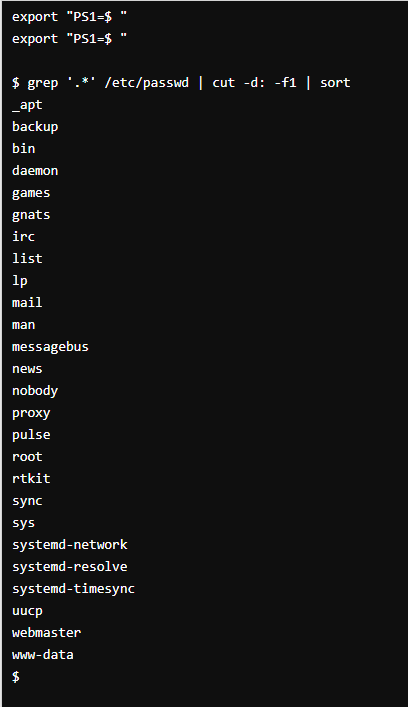
П.Н. Советов, РТУ МИРЭА

Научиться выполнять простые действия с файлами и каталогами в Linux из командной строки. Сравнить работу в командной строке Windows и Linux.

**Задача 1**

Вывести отсортированный в алфавитном порядке список имен пользователей в файле passwd (вам понадобится grep).

grep '.\*' /etc/passwd | cut -d: -f1 | sort



**Задача 2**

Вывести данные /etc/protocols в отформатированном и отсортированном порядке для 5 наибольших портов, как показано в примере ниже:

[root@localhost etc]# cat /etc/protocols ...

142 rohc

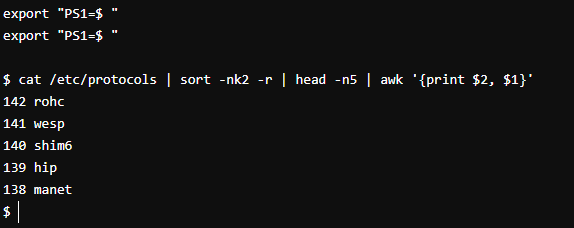
141 wesp

140 shim6

139 hip

138 manet

cat /etc/protocols | sort -nk2 -r | head -n5 | awk '{print $2, $1}'



**Задача 3**

Написать программу banner средствами bash для вывода текстов, как в следующем примере (размер баннера должен меняться!):

[root@localhost ~]# ./banner "Hello from RTU MIREA!"

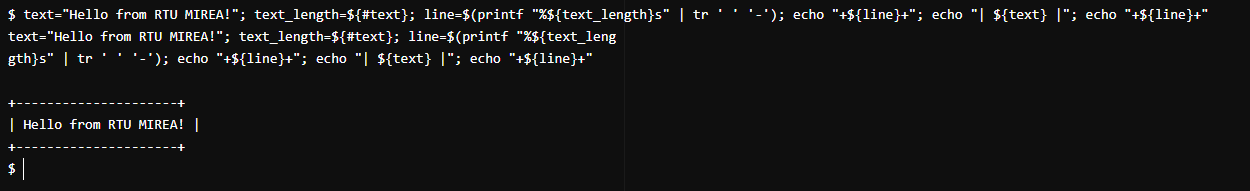
+-----------------------+

| Hello from RTU MIREA! |

+-----------------------+

Перед отправкой решения проверьте его в ShellCheck на предупреждения.

text="Hello from RTU MIREA!"; text\_length=${#text}; line=$(printf "%${text\_length}s" | tr ' ' '-'); echo "+${line}+"; echo "| ${text} |"; echo "+${line}+"

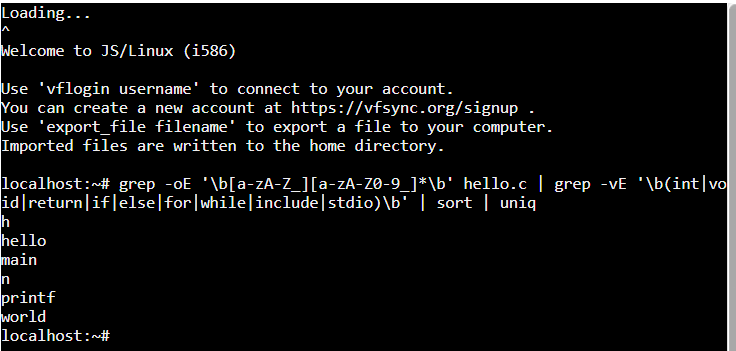


**Задача 4**

Написать программу для вывода всех идентификаторов (по правилам C/C++ или Java) в файле (без повторений).

Пример для hello.c:

h hello include int main n printf return stdio void world



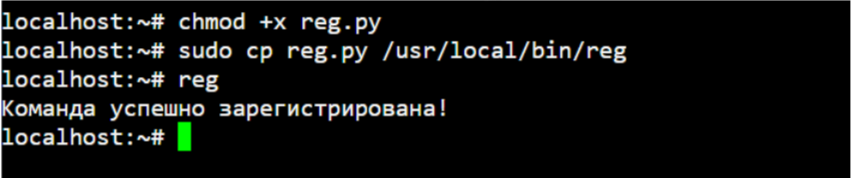
**Задача 5**

Написать программу для регистрации пользовательской команды (правильные права доступа и копирование в /usr/local/bin).

Например, пусть программа называется reg:

./reg banner

В результате для banner задаются правильные права доступа и сам banner копируется в /usr/local/bin.



**Задача 6**

Написать программу для проверки наличия комментария в первой строке файлов с расширением c, js и py.

import os

def check\_comment(file\_path): with open(file\_path, 'r') as file: first\_line = file.readline().strip()

if file\_path.endswith('.c'):

if first\_line.startswith('//') or first\_line.startswith('/\*'):

return f"{file\_path}: комментарий найден"

else:

return f"{file\_path}: комментарий не найден"

elif file\_path.endswith('.js'):

if first\_line.startswith('//'):

return f"{file\_path}: комментарий найден"

else:

return f"{file\_path}: комментарий не найден"

elif file\_path.endswith('.py'):

if first\_line.startswith('#'):

return f"{file\_path}: комментарий найден"

else:

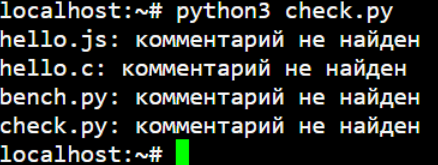
return f"{file\_path}: комментарий не найден"

else:

return f"{file\_path}: неподдерживаемый формат"

**Перебираем все файлы с заданными расширениями в текущем каталоге**

for filename in os.listdir('.'): if filename.endswith(('.c', '.js', '.py')): result = check\_comment(filename) print(result)



**Задача 7**

Написать программу для нахождения файлов-дубликатов (имеющих 1 или более копий содержимого) по заданному пути (и подкаталогам).

import os import hashlib from collections import defaultdict

def hash\_file(file\_path): """Возвращает хеш файла.""" hasher = hashlib.md5() # Используем MD5 для хеширования try: with open(file\_path, 'rb') as f: while chunk := f.read(8192): hasher.update(chunk) return hasher.hexdigest() except Exception as e: print(f"Ошибка при открытии файла {file\_path}: {e}") return None

def find\_duplicates(directory): """Находит дубликаты файлов в заданном каталоге.""" file\_hashes = defaultdict(list)

for root, \_, files in os.walk(directory):

for file in files:

file\_path = os.path.join(root, file)

file\_hash = hash\_file(file\_path)

if file\_hash:

file\_hashes[file\_hash].append(file\_path)

# Выводим дубликаты

duplicates = {hash\_value: paths for hash\_value, paths in file\_hashes.items() if len(paths) > 1}

return duplicates

if **name** == "**main**": path\_to\_search = input("Введите путь для поиска дубликатов: ") duplicates = find\_duplicates(path\_to\_search)

if duplicates:

print("Найденные дубликаты:")

for hash\_value, paths in duplicates.items():

print(f"\nХеш: {hash\_value}")

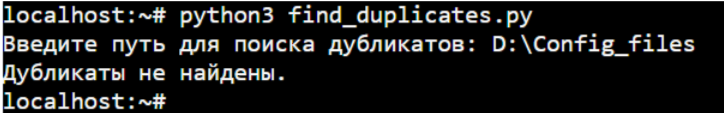
for path in paths:

print(f" {path}")

else:

print("Дубликаты не найдены.")

Мы создали 2 файла: 12.txt и 22.txt имеющие следующие строки: "12345" и "123456"



**Задача 8**

Написать программу, которая находит все файлы в данном каталоге с расширением, указанным в качестве аргумента и архивирует все эти файлы в архив tar.

 import os import tarfile import argparse

def find\_files\_with\_extension(directory, extension): """Находит все файлы с заданным расширением в указанном каталоге.""" matched\_files = [] for root, \_, files in os.walk(directory): for file in files: if file.endswith(extension): matched\_files.append(os.path.join(root, file)) return matched\_files

def create\_tar\_archive(files, archive\_name): """Создает архив tar из списка файлов.""" with tarfile.open(archive\_name, 'w') as tar: for file in files: tar.add(file, arcname=os.path.relpath(file, start=os.path.dirname(files[0])))

if **name** == "**main**": parser = argparse.ArgumentParser(description="Архивирует файлы с заданным расширением.") parser.add\_argument("directory", help="Путь к каталогу для поиска файлов.") parser.add\_argument("extension", help="Расширение файлов для архивации (например, .txt).") parser.add\_argument("archive\_name", help="Имя выходного архива (например, archive.tar).")

args = parser.parse\_args()

files\_to\_archive = find\_files\_with\_extension(args.directory, args.extension)

if files\_to\_archive:

create\_tar\_archive(files\_to\_archive, args.archive\_name)

print(f"Архив '{args.archive\_name}' успешно создан с {len(files\_to\_archive)} файлами.")

else:

print("Файлы с указанным расширением не найдены.")

