|  |  |  |
| --- | --- | --- |
|  | https://www.mirea.ru/bitrix/templates/unlimtech/images/logo.png |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

|  |
| --- |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) |

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | Отчет по выполнению практических работ | | по дисциплине | | **«**Конфигурационное управление**»** | |

|  |  |
| --- | --- |
|  | |
| Выполнил студент | Братушка Д.О. |
| Группа | ИКБО-24-21 |

Москва 2022

Содержание

[Практическая работа №1 — основы работы в командной строке 3](#_Toc115116190)

[Задача 1 3](#_Toc115116191)

[Задача 2 3](#_Toc115116192)

[Задача 3 4](#_Toc115116193)

[Задача 4 5](#_Toc115116194)

[Задача 5 6](#_Toc115116195)

[Задача 6 7](#_Toc115116196)

[Задача 7 8](#_Toc115116197)

[Задача 8 8](#_Toc115116198)

[Задача 9 9](#_Toc115116199)

[Задача 10 10](#_Toc115116200)

# Практическая работа №1 — основы работы в командной строке

## Задача 1

Условие. Вывести отсортированный в алфавитном порядке список именпользователей в файле passwd (вам понадобится grep).

Решение. В задаче используется grep с флагами -o (выводит только совпадения по шаблону) и -E (принимает ввод расширенного регулярного выражения), затем мы соединяем вывод с функцией sort.

Листинг 1.1

#!/bin/bash

grep -o -E -i "^\w{1,}" /etc/passwd | sort

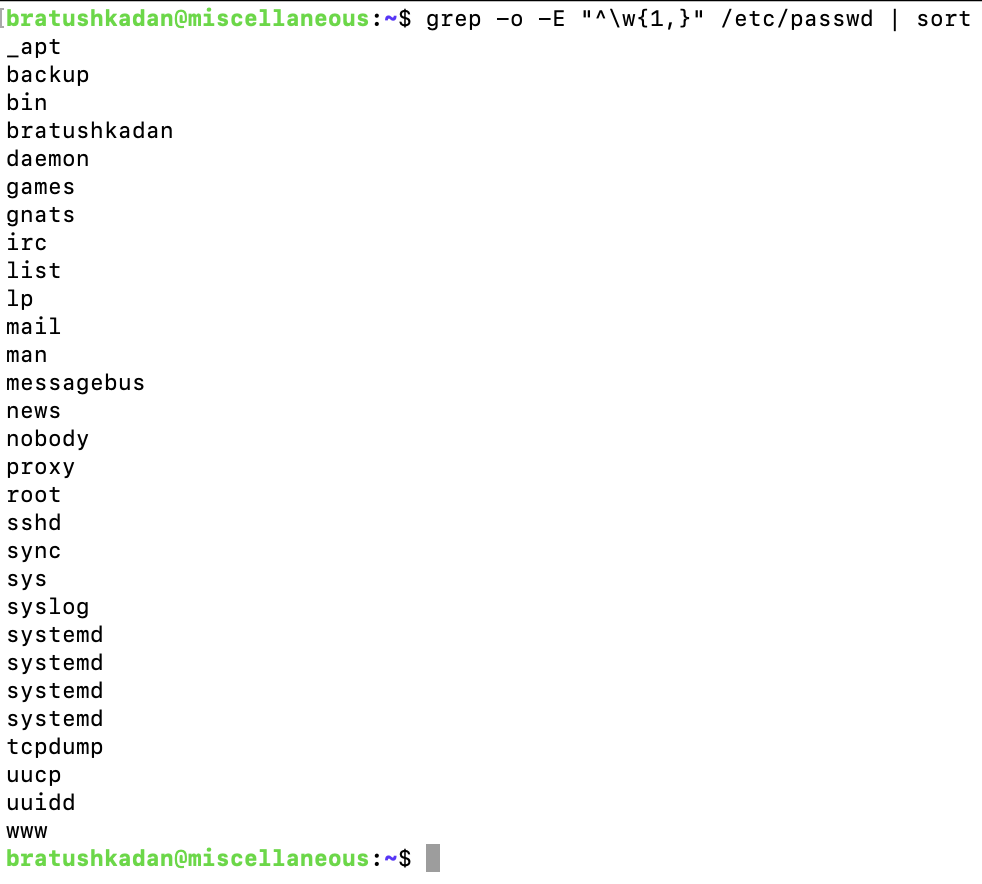


Рисунок 1.1 — результат работы программы

## Задача 2

Условие. 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

Решение. В задаче используется sort с параметром -k равным 2nr – сортировка по второму столбцу как числам в убывающем порядке, команда head -n 5 захватывает первые 5 строк, команда awk выбирает 1 и 2 столбцы.

Листинг 1.2

#!/bin/bash

# вывод файла | сортировка по второму столбцу как числам в убывающем порядке | захват первых 5 строк | печать только 1 и 2 переменных (строк)

cat /etc/protocols | sort -k 2nr | head -n 5 | awk '{print $1, $2}'

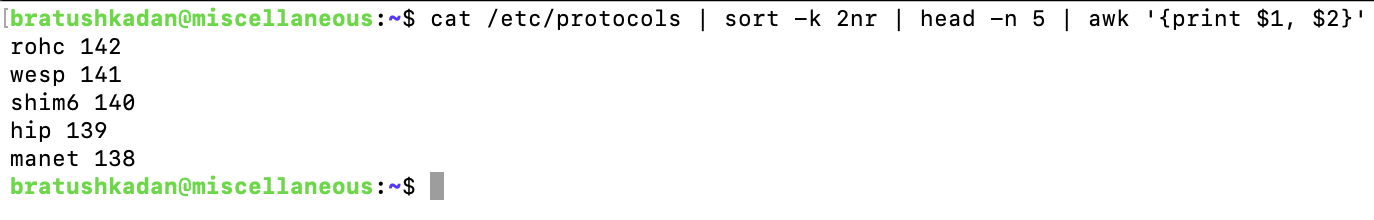


Рисунок 1.2 — результат работы программы

## Задача 3

Условие. 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! |

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

Решение. Комментарии поясняют значение непонятных кусков кода.

Листинг 1.3

#!/bin/bash

horizontal\_line() {

s="+"

# ${#@} - даёт количество элементов массива

str=$@

# ${#str} - даёт количество символов в строке

# range(1, len(str) + 2 + 1)

for i in $(seq 1 $((${#str} + 2)))

do

# конкатенация строк

s="$s-"

done

s="$s+"

echo $s

}

horizontal\_line $@

echo "| $@ |"

horizontal\_line $@

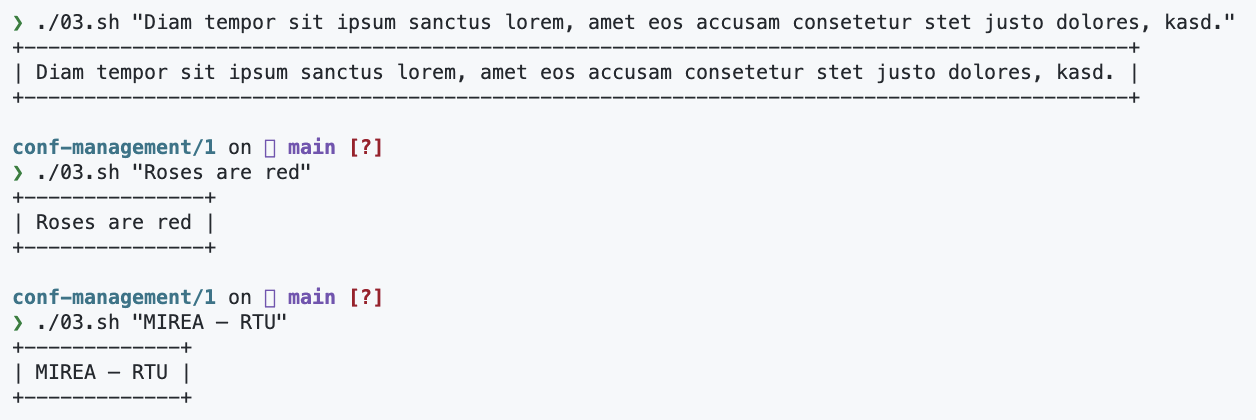


Рисунок 1.3 — результат работы программы

## Задача 4

Условие. 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

Решение. Комментарии поясняют значение непонятных кусков кода.

Листинг 1.4

#!/bin/bash

echo "main.cpp"

cat ./artifacts/main.cpp

echo ""

# tr -d "[:punct:]" - убирает все символы пунктуакции

# tr " " "\n" - разбивает текст по пустым символам на переносы строк

# sed "/^$/d" - используем stream editor для трансформации текста - убираем пустые строки

# sort -u — сортирует идентификаторы, выводя только уникальные (-u)

# tr "\n" " " - заменяет переносы строки на пустые символы

cat artifacts/main.cpp | tr "[:punct:]" " " | tr " " "\n" | sed "/^$/d" | sort -u | tr "\n" " "

echo ""

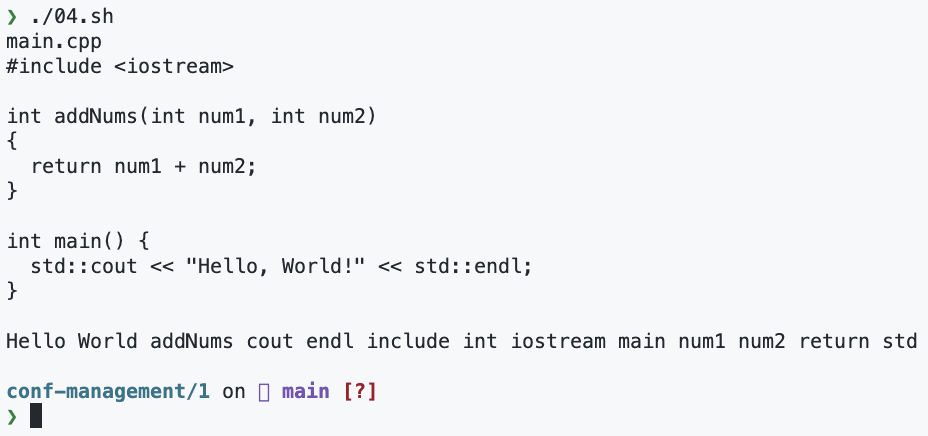


Рисунок 1.4 — результат работы программы

## Задача 5

Условие. 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.

Решение. Комментарии поясняют значение непонятных кусков кода.

Листинг 1.5

#!/bin/bash

filename=$1

command\_name=$2

cp $1 /usr/local/bin/$2

chmod 755 /usr/local/bin/$2

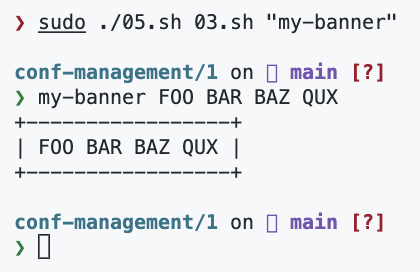


Рисунок 1.5 — результат работы программы

## Задача 6

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

Решение. Создаем функции, пытающиеся найти по регулярному выражению комментарии в каждом из ЯП в первой строчке файла. Создаем вспомогательные функции box для визуального разделения. Функция contains\_<язык>\_comment выводит, является ли строка, возвращаемая функцией поиска тела комментария на первой строке файла пустой.

Листинг 1.6

#!/bin/bash

box() {

echo ""

echo "----- $1"

cat ./artifacts/$1

echo "-----"

}

py\_comment() {

cat ./artifacts/$1 | head -1 | grep -o -E "#.+$"

}

js\_comment() {

cat ./artifacts/$1 | head -1 | grep -o -E "\/\\*.+\\*\/" || cat ./artifacts/$1 | head -1 | grep -o -E "\/\/.+$"

}

contains\_py\_comment() {

box $1

if [[ -z $(py\_comment $1) ]]; then

echo "$1 doesn't contain a comment"

else

echo "$1 contains a comment"

fi

}

contains\_js\_comment() {

box $1

if [[ -z $(js\_comment $1) ]]; then

echo "$1 doesn't contain a comment"

else

echo "$1 contains a comment"

fi

}

contains\_py\_comment 1.py

contains\_js\_comment 1.js

contains\_js\_comment 2.js

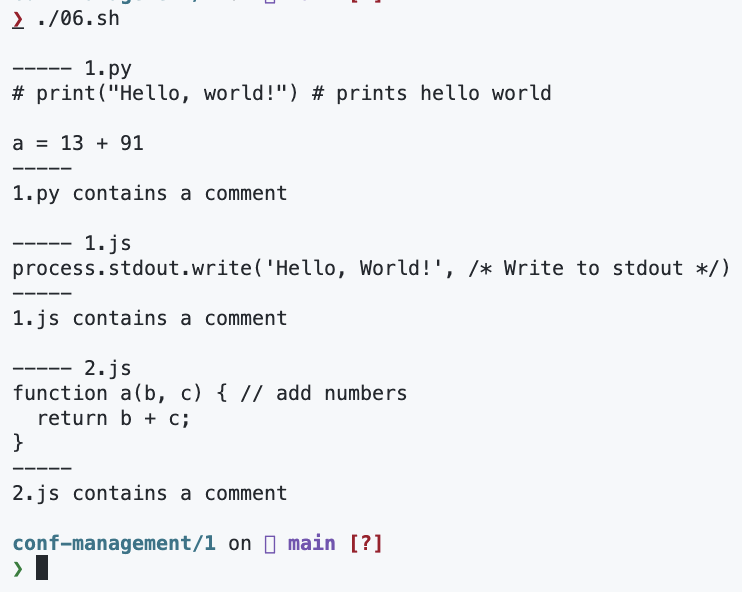


Рисунок 1.6 — результат работы программы

## Задача 7

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

Решение. find находит все непустые файлы, высчитывает для них md5-хеш сумму, сортирует файлы, команда uniq отбрасывает файлы, имеющие одинаковую хеш-сумму (содержимого)

Листинг 1.7

#!/bin/bash

## сработало только на Ubuntu, не работает на MacOS

find . ! -empty -type f -exec md5sum {} + | sort | uniq -w32 -dD

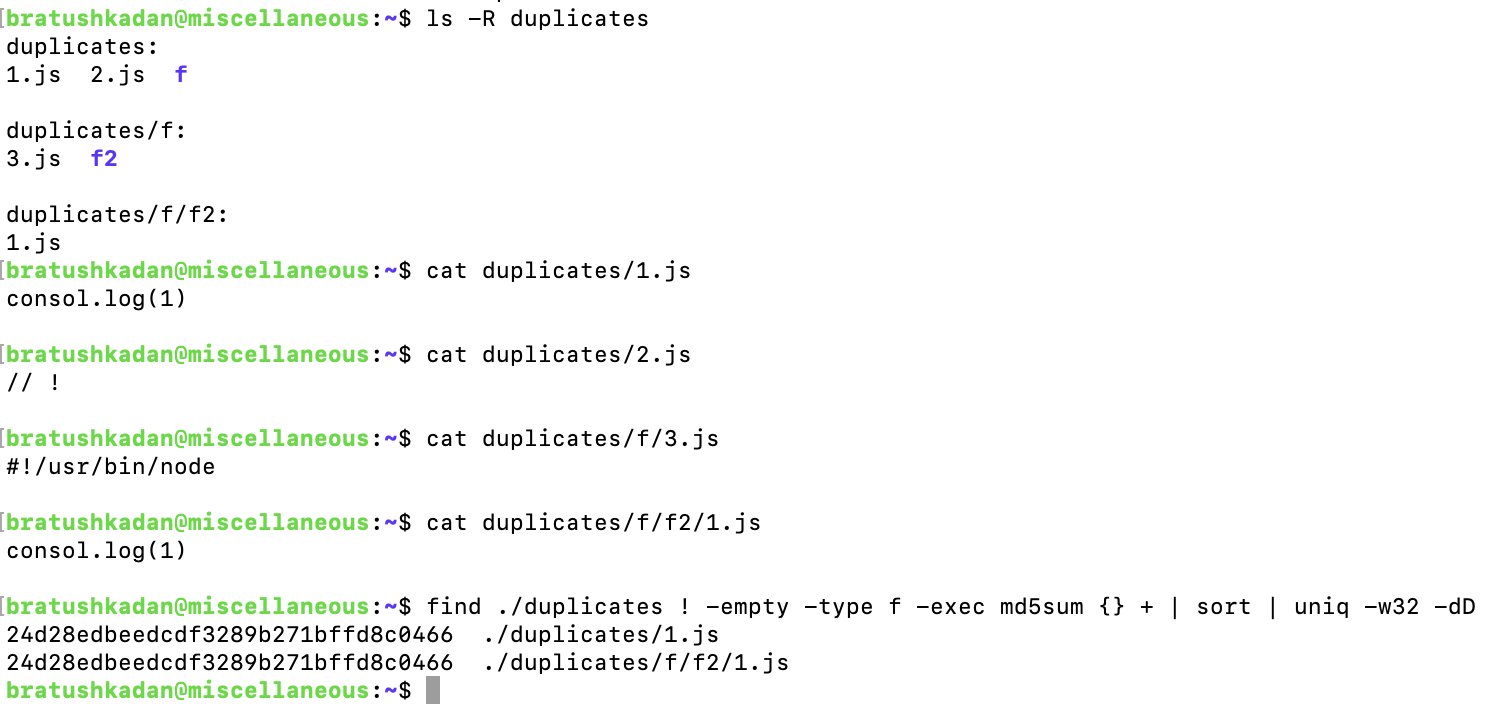


Рисунок 1.7 — результат работы программы

## Задача 8

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

Решение. Архивируем содержимое директории с указанным расширением в архив формата .tar.gz.

Листинг 1.8

#!/bin/bash

dir=$1

ext=$2

tar -zcf archived.tar.gz $(ls $1/\*.$2)

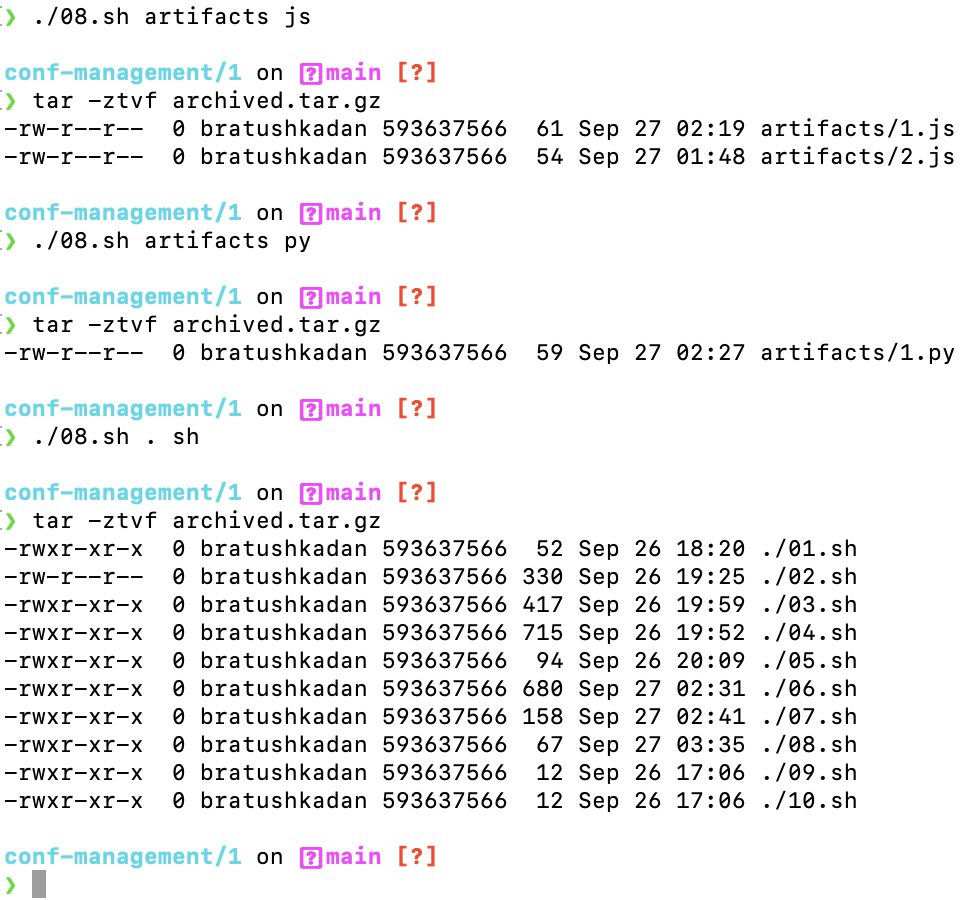


Рисунок 1.8 — результат работы программы

## Задача 9

Условие. Написать программу, которая заменяет в файле последовательности из 4 пробелов на символ табуляции. Входной и выходной файлы задаются аргументами.

Решение. Используем sed – трансформатор потоков – для замены всех вхождений 4 пустых символов на символ табуляции и записываем результат в файл с именем, передаваемым через параметр.

Листинг 1.9

#!/bin/bash

in=$1

out=$2

cat $in | sed "s/ /\t/g" > $out

cat $in

echo ""

printf '=%.0s' {1..40}

echo ""

cat $out

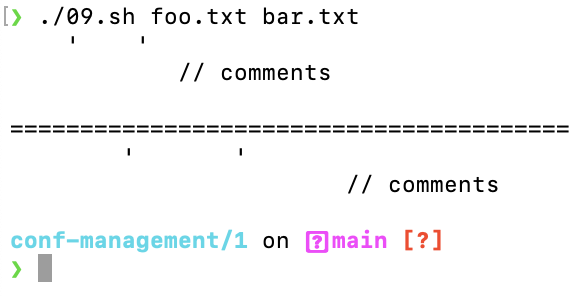


Рисунок 1.9 — результат работы программы

## Задача 10

Условие. Написать программу, которая выводит названия всех пустых текстовых файлов в указанной директории. Директория передается в программу параметром.

Решение. find находит все пустые файлы, сортирует названия.

Листинг 1.10

#!/bin/bash

find $1 -empty -type f | grep -o -E "\/[^\/]+$" | sed "s/^\///g"

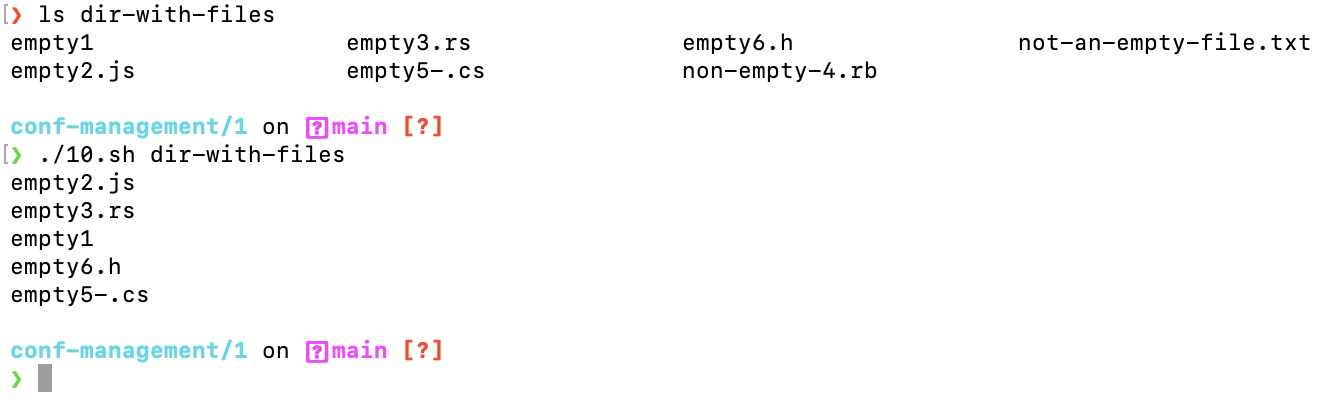


Рисунок 1.10 — результат работы программы