



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
ЗАДАЧА 1	3
ЗАДАЧА 2	3
ЗАДАЧА 3	4
ЗАДАЧА 4	5
ЗАДАЧА 5	6
ЗАДАЧА 6	7
ЗАДАЧА 7	9
ЗАДАЧА 8	10
ЗАДАЧА 9	11
ЗАДАЧА 10	12

ПРАКТИЧЕСКАЯ РАБОТА №1 — ОСНОВЫ РАБОТЫ В КОМАНДНОЙ СТРОКЕ

Задача 1

Условие. Вывести отсортированный в алфавитном порядке список имен пользователей 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
```

```
bratushkadan@miscellaneous:~$ grep -o -E "^\\w{1,}" /etc/passwd | sort
_apt
backup
bin
bratushkadan
daemon
games
gnats
irc
list
lp
mail
man
messagebus
news
nobody
proxy
root
sshd
sync
sys
syslog
systemd
systemd
systemd
systemd
tcpdump
uucp
uucidd
www
bratushkadan@miscellaneous:~$ █
```

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}'

[bratushkadan@miscellaneous:~$ cat /etc/protocols | sort -k 2nr | head -n 5 | awk '{print $1, $2}'
rohc 142
wesp 141
shim6 140
hip 139
manet 138
bratushkadan@miscellaneous:~$ █
```

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 $@

```

```

> ./03.sh "Diam tempor sit ipsum sanctus lorem, amet eos accusam consetetur stet justo dolores, kasd."
+-----+
| Diam tempor sit ipsum sanctus lorem, amet eos accusam consetetur stet justo dolores, kasd. |
+-----+

conf-management/1 on [ main [?]
> ./03.sh "Roses are red"
+-----+
| Roses are red |
+-----+

conf-management/1 on [ main [?]
> ./03.sh "MIREA - RTU"
+-----+
| MIREA - RTU |
+-----+

```

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
```

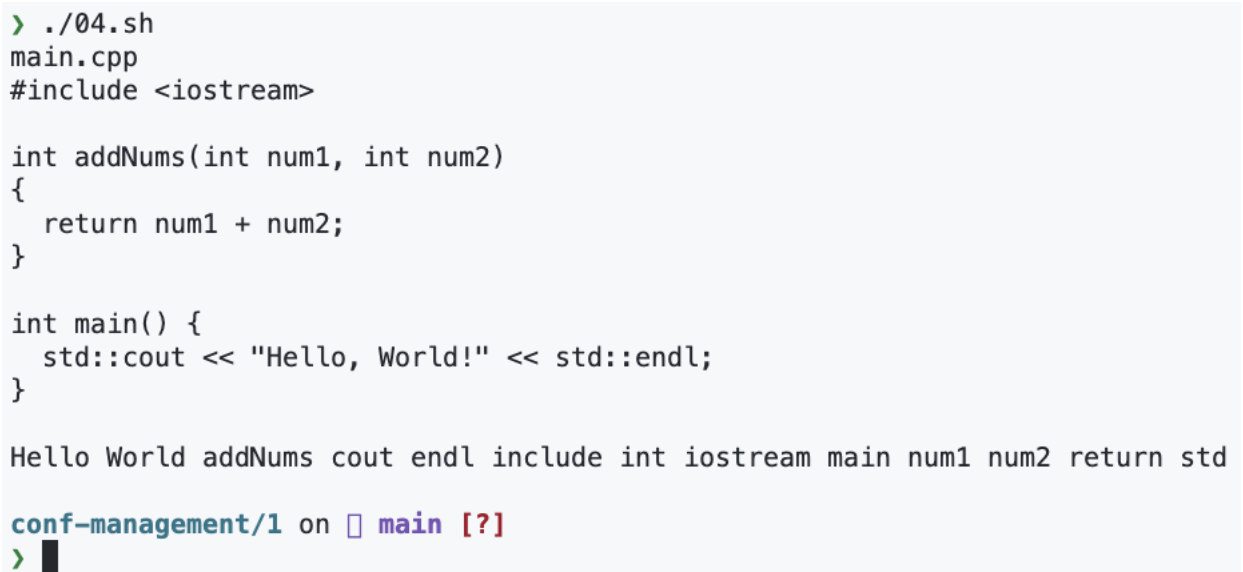
Решение. Комментарии поясняют значение непонятных кусков кода.

Листинг 1.4

```
#!/bin/bash

echo "main.cpp"
cat ./artifacts/main.cpp
echo ""

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



```
> ./04.sh
main.cpp
#include <iostream>

int addNums(int num1, int num2)
{
    return num1 + num2;
}

int main() {
    std::cout << "Hello, World!" << std::endl;
}

Hello World addNums cout endl include int iostream main num1 num2 return std

conf-management/1 on main [?]
> █
```

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.

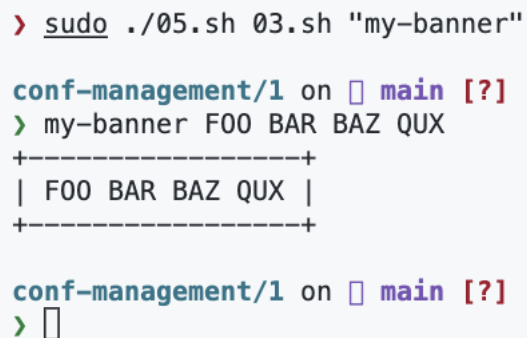
Решение. Комментарии поясняют значение непонятных кусков кода.

Листинг 1.5

```
#!/bin/bash

filename=$1
command_name=$2

cp $1 /usr/local/bin/$2
chmod 755 /usr/local/bin/$2
```



```
> sudo ./05.sh 03.sh "my-banner"

conf-management/1 on [?] main [?]
> my-banner F00 BAR BAZ QUX
+-----+
| F00 BAR BAZ QUX |
+-----+

conf-management/1 on [?] main [?]
>
```

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 комментария 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

```



```

> ./06.sh

----- 1.py
# print("Hello, world!") # prints hello world

a = 13 + 91
-----
1.py contains a comment

----- 1.js
process.stdout.write('Hello, World!', /* Write to stdout */)
-----
1.js contains a comment

----- 2.js
function a(b, c) { // add numbers
    return b + c;
}
-----
2.js contains a comment

conf-management/1 on ☐ main [?]
> █

```

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

```

```

[bratushkadan@miscellaneous:~$ ls -R duplicates
duplicates:
1.js 2.js f

duplicates/f:
3.js f2

duplicates/f/f2:
1.js
[bratushkadan@miscellaneous:~$ cat duplicates/1.js
console.log(1)

[bratushkadan@miscellaneous:~$ cat duplicates/2.js
// !

[bratushkadan@miscellaneous:~$ cat duplicates/f/3.js
#!/usr/bin/node

[bratushkadan@miscellaneous:~$ cat duplicates/f/f2/1.js
console.log(1)

[bratushkadan@miscellaneous:~$ find ./duplicates ! -empty -type f -exec md5sum {} + | sort | uniq -w32 -dD
24d28edbeedcdf3289b271bffd8c0466 ./duplicates/1.js
24d28edbeedcdf3289b271bffd8c0466 ./duplicates/f/f2/1.js
[bratushkadan@miscellaneous:~$ █

```

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)

```

```

> ./08.sh artifacts js

conf-management/1 on [?]main [?]
> tar -ztfv archived.tar.gz
-rw-r--r-- 0 bratushkadan 593637566 61 Sep 27 02:19 artifacts/1.js
-rw-r--r-- 0 bratushkadan 593637566 54 Sep 27 01:48 artifacts/2.js

conf-management/1 on [?]main [?]
> ./08.sh artifacts py

conf-management/1 on [?]main [?]
> tar -ztfv archived.tar.gz
-rw-r--r-- 0 bratushkadan 593637566 59 Sep 27 02:27 artifacts/1.py

conf-management/1 on [?]main [?]
> ./08.sh . sh

conf-management/1 on [?]main [?]
> tar -ztfv archived.tar.gz
-rwxr-xr-x 0 bratushkadan 593637566 52 Sep 26 18:20 ./01.sh
-rw-r--r-- 0 bratushkadan 593637566 330 Sep 26 19:25 ./02.sh
-rwxr-xr-x 0 bratushkadan 593637566 417 Sep 26 19:59 ./03.sh
-rwxr-xr-x 0 bratushkadan 593637566 715 Sep 26 19:52 ./04.sh
-rwxr-xr-x 0 bratushkadan 593637566 94 Sep 26 20:09 ./05.sh
-rwxr-xr-x 0 bratushkadan 593637566 680 Sep 27 02:31 ./06.sh
-rwxr-xr-x 0 bratushkadan 593637566 158 Sep 27 02:41 ./07.sh
-rwxr-xr-x 0 bratushkadan 593637566 67 Sep 27 03:35 ./08.sh
-rwxr-xr-x 0 bratushkadan 593637566 12 Sep 26 17:06 ./09.sh
-rwxr-xr-x 0 bratushkadan 593637566 12 Sep 26 17:06 ./10.sh

conf-management/1 on [?]main [?]
> █

```

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
```

```
[>] ./09.sh foo.txt bar.txt
      |      |
      // comments

=====
      |      |
      // comments

conf-management/1 on [?]main [?]
> █
```

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"
```

```
[> ls dir-with-files
empty1          empty3.rs          empty6.h          not-an-empty-file.txt
empty2.js       empty5-.cs         non-empty-4.rb

conf-management/1 on [?]main [?]
[> ./10.sh dir-with-files
empty2.js
empty3.rs
empty1
empty6.h
empty5-.cs

conf-management/1 on [?]main [?]
> █
```

Рисунок 1.10 — результат работы программы