Собес ЦК

1. Работа в терминале

Задание 2

Найдите в архиве файл с правами на исполнение. Выведите его название.

```
#!/bin/bash # bash-скрипт
mkdir unzipped
tar -xf find-exe.tar.gz -C unzipped # распаковать архив в папку
dir=$(ls unzipped | head -1) # имя первой папке в распакованном архиве
find $dir -type f -executable # найти исполняемый файл
```

Задание 3

Посчитайте суммарное количество слов во всех файлах, имена которых содержат слово, указанное в файле `target.word`.

```
#!/bin/bash
unzip -qq word-count.zip -d unzipped
cd unzipped
chmod u+x target.word # добавить разрешение на исполнение
directory=$(ls -d */) # сохраняет поддиректории
word=$(cat target.word)
function search_files() {
    for file in "$1"/*; do # проходит по всем файлам и поддиректориям директории (1
аргумент)
        if [[ -d "$file" ]]; then # если является директорией
            search_files "$file" "$word"
        elif [[ -f "$file" && "$file" == *"$word"* ]]; then # если это файл и
содержит word в названии
            count=$(wc -w < "$file") # сохраняет кол-во слов в файле
            echo "File: $file, Word Count: $count"
            total=$((total + count))
        fi
    done
}
total=0
search_files "$directory" "$word"
echo "Total Word Count: $total"
```

Задание 4

Отсортируйте файлы в директории по возрастанию времени модификации. Посчитайте шестнадцатеричный SHA-256 хеш конкатенации данных этих файлов в этом порядке (без каких-либо дополнительных символов).

```
#!/bin/bash
mkdir unzipped_dir
unzip -qq mod-sort.zip -d unzipped_dir
cd unzipped_dir
files=$(ls -t) # сохраняет содержимое директории, сортируя по времени изменения
concatenated_data=""
for file in $files do
  if [ -f $file ]
  then
    concatenated_data=$(cat $file)$concatenated_data
  fi
done
hash=$(echo -n $concatenated_data | sha256sum | cut -d ' ' -f 1)
# сохраняет переменную без перевода на новую строку
# переводит в sha-256 хеш
# выводит без пробелов ???
echo $hash
```

Задание 5

Посчитайте количество телефонных номеров в формате E.164 в данном файле. Номера могут начинаться со знака "+" и не могут начинаться с 0.

```
#!/bin/bash
file_path="phone-numbers"
count=0
while IFS= read -r line; do # ifs - не учитывать символы пробела и новой строки как
разделители; read -r - прочитать строку и записать в переменную
    words=($line)
    for word in "${words[@]}"; do
        if [[ "$word" =~ ^\+?[1-9][0-9\s]{1,14}$ ]]; then # *
            ((count++))
        fi
    done
done < "$file_path"
echo $count

*
^\+?[1-9][0-9\s]{1,14}$
^ - начало строки
\+? - может быть “+”, но не обязательно
[1, 9] - одна цифра
[0-9\s]{1,14} - последовательность от 1 до 14 цифр или пробелов
$ - конец строки
```

Задание 6

Скачайте все файлы по указанным ссылкам. Посчитайте суммарный размер PDF-документов среди скачанных файлов в байтах.

```
#!/bin/bash
mkdir downloaded_dir
wget -q -i ./wget-pdfs -P downloaded_dir
```

```
cd downloaded_dir
total_size=0
downloaded_files=$(find -type f)
for file in $downloaded_files
do
    if [[ $(file -b --mime-type "$file") == "application/pdf" ]] # проверяет, что
тип файла - "application/pdf"
    then
        file_size=$(stat -c%s "$file") # находит размер файла в байтах
        total_size=$((total_size + file_size))
    fi
done
echo $total_size
```

2. GIT

2. commit-one-file-staged

```
git commit -m “file A.txt” A.txt
```

3. ignore-them

```
cat >> .gitignore
*.exe
*.o
*.jar
libraries/
*.swp
*.swo
^C
git add file.txt
git add .gitignore
git commit -m “file.txt and .gitignore commited”
```

4. merge-conflict

```
git merge another-piece-of-work
vim equation.txt
2+3=5  # HEAD + another-piece-of-work
:wq
git add equation.txt
git commit -m merged
```

5. save-your-work

```
git stash
vim bug.txt
#delete line with bug
git add bug.txt
git commit -m "fixed bug"
git stash pop
vim bug.txt
#add "Finally, finished it!"
git add bug.txt
git add program.txt
git commit -m "finished"
```

6.change-branch-history

```
git rebase hot-bugfix # переместить текущую ветку на ветку hot-bugfix
```

7. forge-date

```
git commit --amend --no-edit --date="1987.10.19" #--amend - меняет данные коммита,
--no-edit - не меняет сообщение коммита
```

8. fix-old-typo

```
git rebase -i # interactive rebase
#pick -> edit
vim file.txt
#wordl -> world
git add file.txt
git commit --amend -m "Add Hello world"
git rebase --continue
vim file.txt
#resolve conflict: Hello wordl -> Hello world
git add file.txt
git commit
git rebase --continue
```

9. commit-lost

```
git reflog # история действий
#find name of lost commit in list
git reset --hard HEAD@{1}.
```

10. too-many-commits

```
git rebase -i HEAD~2
#in latest commit: pick -> s
```

11. executable

```
git add --chmod+=x script.sh
git commit --amend -m "Execute permission"
```

12. commit-parts

```
git add -p file.txt # добавление только выбранных частей
#add only lines with "task 1"
git commit -m "task1 commit"
git add file.txt
git commit -m "left lines commit"
```

13. invalid-order

```
git rebase -i HEAD~2
#change order of lines
```

14. search-improved

```
git bisect start
git bisect bad
git bisect good 43d649
# repeat until answer's not found:
{
   #output: Bisecting ... [x]
   git checkout x
   ./faulty-check
   echo $?
   #if 0: y=good, if 1: y=bad
   git bisect y
}
```

1. Curl

Сделайте с помощью утилиты curl запрос на http://10.8.0.1:8080/980a921f6bc0 с методом PATCH, передав query-параметр do со значением 09b636af4a94, заголовок X-Access-Token со значением 9fab0333d789 и JSON-объект с единственным полем text со значением 1d242712fd53 в теле запроса.

```
curl -X PATCH http://10.8.0.1:8080/980a921f6bc0?do=09b636af4a94 -H "X-Access-Token:
9fab0333d789" -H "Content-Type: application/json" -d "{\"text\": \"1d242712fd53\"}"
```

2. HTTP

Откройте в браузере http://10.8.0.1/. Чтобы получить секрет, нажмите кнопку на этой странице 10 000 раз. Автоматизируйте этот процесс.

```
var button = document.querySelector("button");

function clickButton() {
    var stop = 100;
    setInterval(function() {
        button.click();
    }, stop);
}
clickButton();
```

3. TCP

Установите TCP соединение с хостом 10.8.0.1 на порт 1234. Выполните предлагаемое задание.

```
import telnetlib

host = "10.8.0.1"
port = 1234

telnet = telnetlib.Telnet(host, port)
while True:
    try:
        out = telnet.read_until(b"\n", timeout=5)
        try:
            a = out.decode().replace("\n", "")
            if len(a) == 1:
                continue
            else:
                s = eval(out)
        except(SyntaxError, IndexError):
            continue
        telnet.write(str(s).encode())
    except EOFError:
        break
print(a)
```

4. DNS

Узнайте TXT-запись домена 3f5f66f45b82.digital-culture-networking.melnikov.ch

```
nslookup -type=TXT 3f5f66f45b82.digital-culture-networking.melnikov.ch
```

5. **UDP**

Хост 10.8.0.1 раз в 10 секунд проверяет, отвечает ли хост 10.8.0.214 на пинги (запросы ICMP Echo Request), и отправляет UDP-датаграмму ему на порт 666, если да. Добавьте этот адрес к своему VPN-интерфейсу и получите датаграмму.

```
поменяла в ip-адрес vpn на 10.8.0.214
через приложение wireshark нашла пакет, который передается по протоколу udp на порт
666
```

4. Базы данных

Задание 1

Выведите кол-во пользователей, у кого не установлен пароль (то есть passwordHash равен NULL).

```
SELECT COUNT(*) FROM User WHERE passwordHash IS NULL;
# COUNT(*) - считает кол-во строк
```

Задание 2

Выведите идентификаторы авторов — пользователей, которые опубликовали хотя бы один пост. Идентификаторы отсортируйте по возрастанию.

```
SELECT id FROM `User` WHERE EXISTS (SELECT * FROM Post WHERE Post.userId =
User.id) ORDER BY id;
# SELECT * - выбирает строки
```

Задание 3

Выведите таблицу со всеми постами из двух столбцов — «название поста» (postTitle) и «имя автора» (authorName). Отсортируйте данные по возрастанию времени публикации поста, а при совпадении времени — по ID поста.

```
SELECT Post.title AS postTitle, User.name AS authorName
FROM Post
INNER JOIN User ON Post.userId = User.id # объединяет строки из 2 таблиц
ORDER BY Post.creationTime, Post.id;
```

Задание 4

Для каждого автора (как в задании 2) выведите название его первого поста — поста с минимальным временем публикации, а если таких несколько — с минимальным ID. Отсортируйте список по возрастанию времени публикации, при равенстве — по ID поста.

```
SELECT title FROM Post
WHERE id IN (SELECT MIN(id) FROM Post
WHERE creationTime IN (SELECT MIN(creationTime) FROM Post GROUP BY userId)
GROUP BY userId) ORDER BY creationTime, id;
```

Задание 5

Выведите все даты (не времена, а именно даты, то есть дни), в которые был опубликован хотя бы один пост. Для каждой даты выведите количество постов, опубликованных в этот день. Отсортируйте данные по возрастанию даты.

```
SELECT Date(creationTime) AS postCreationDate, COUNT(creationTime) AS postCount
FROM Post
GROUP BY DATE(creationTime);
```

Задание 6

Добавьте в базу данных новую таблицу (сущность) Comment. Сущность должна быть связана с Post и User отношениями много-к-одному. У сущности должны быть столбцы id, postId, userId, text и creationTime. Используйте те же типы данных, которые используются в таблице Post. Корректно укажите внешние ключи.

```
CREATE TABLE `Comment` (
    `id` BIGINT NOT NULL AUTO_INCREMENT,
    `postId` BIGINT NOT NULL,
    `userId` BIGINT NOT NULL,
    `text` LONGTEXT NOT NULL,
    `creationTime` DATETIME NOT NULL,
    PRIMARY KEY (`id`),
    INDEX `index_Post_creationTime` (`creationTime`)
) ENGINE = InnoDB;

ALTER TABLE Comment ADD CONSTRAINT `fk_Comment_userId` FOREIGN KEY (userId)
REFERENCES User (id);
ALTER TABLE Comment ADD CONSTRAINT `fk_Comment_postId` FOREIGN KEY (postId)
REFERENCES Post (id);

INSERT INTO `Comment` (`postId`, `userId`, `text`, `creationTime`) VALUES (6, 2,
"Morning certainly listen production statement.", "2005-03-18 21:34:49"), ...
```

Задание 7

Отсортируйте посты по убыванию количества комментариев. Выведите ID постов в этом порядке. Если количество комментариев одинаково, сначала выведите посты с меньшим ID.

```
SELECT Post.id
FROM Post
LEFT JOIN Comment ON Post.id = Comment.postId # берет каждую запись из таблицы
"Post" и пытается найти соответствующие записи в таблице "Comment"
GROUP BY Post.id
ORDER BY COUNT(postId) DESC, Post.id;
```

5. VIM

0.

```
2j
28l
i
hello vim
```

(2 вниз, 28 направо, Insert mode, "hello vim")

1)
Спускаемся к началу функции
Shift + v
Выделяем строки, нажимаем d

2.
`4j`
`$`
`a`
`;`
(4 вниз, в конец строки, Insert mode, ";")

3)
Shift+4
14 j
4 h
i
печатаем next

4.
`13j`
`32l`
`a`
`Backspace`
`t`
(13 вниз, 32 направо, Insert mode, удалить символ, "t")

5)
23j
3w
i
удаляем лишнее

6.
`G`
`dd`
(в конец файла, удалить строку)

7)
/vehicla
Enter
i
добавляем букву

8.
`:%s/Emacs/Vim/g`
`Enter`
(заменить во всем файле "Emacs" на "Vim")

9)
/vim
enter
Shift+n # последнее вхождение
Shift+y

Shift+g
P

10.
`:%s/\(ohn\|ames\|acob\)//g`
`Enter`
(заменить во всем файле "ohn" или "ames" или "acob" на "")

11)
4j
Shift+y
P

12.
`6j`
`5w`
v
`5e`
`d`
`4j`
`3h`
`p`
(6 вниз, на начало 6 слова, Visual mode, на конец 5 слова, удалить, 4 вниз, 3 влево, вставить)

13)
?t
Enter
Shift+r # repalace
st

14.
`5j`
`Ctrl+a`
(5 вниз, увеличить число на 1)

15)
w
Ctrl v
Shift g
Ctrl x

16.
`:%s#// ##g`
`Enter`
(заменить во всем файле "// " на "")

6. Отладка и профилирование

Отладка
Поставить брейкпоинт на конец main
Запустить отладку
В консоли найти значение переменной ans

Профилирование
Запустить профилирование
В консоли найти процессы, записать по возрастанию

7. Системы сборки

Создать файл pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>dc-maven</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-math3</artifactId>
            <version>3.6.1</version>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.10.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.2.2</version>
                <configuration>
                    <includes>
                        <include>**/TestMath.java</include>
                        <include>**/CloudTest.java</include>
                    </includes>
                </configuration>

            </plugin>
```

```
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
            </plugin>
        </plugins>
    </build>

</project>
```

Закоммитить его в гитхаб
Зайти в actions, выбрать new workflow -> java with maven
Появится файл .yml, поменять название на test.yaml, положить в .github/workflows

```
name: test-maven
on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]
jobs:
  test-maven:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - name: Set up JDK 11
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'temurin'
        cache: maven
    - name: Build with Maven
      run: mvn -B package --file pom.xml
    - name: Run test
      run: mvn test --file pom.xml
    - name: Update dependency graph
      uses:
advanced-security/maven-dependency-submission-action@571e99aab1055c2e71a1e2309b9691de18d6b7
d6
```

8. Безопасность и криптография

Задание hash

Найдите хеш SHA224 строки Prevent identify white better

```
import hashlib
input_string = "Prevent identify white better"
hash_object = hashlib.sha224(input_string.encode())
hash_hex = hash_object.hexdigest()
print(hash_hex)
```

Задание encoding

В результате нескольких последовательных кодирований алгоритмами base64 и base32 получилась следующая строка:

```
онлайн декодеры много раз подряд
```

Задание ssl

Укажите серийный номер TLS-сертификата для сайта telegram.org. Ответом является HEX-строка.

```
echo | openssl s_client -servername telegram.org -connect telegram.org:443 | openssl x509 -noout -serial
#  использует OpenSSL для установки SSL/TLS-соединения с сервером telegram.org на порту 443. Параметр -servername указывает серверное имя, а параметр -connect указывает адрес сервера и порт.
# берет вывод и передает его на вход команде openssl x509, которая используется для работы с сертификатами. Флаг -noout указывает, что нам не нужно выводить сам сертификат, а -serial просит вывести только серийный номер сертификата.
```

Задание rsa1
Познакомьтесь с алгоритмом RSA. Наш приватный ключ — n = 1002905049463, d = 117988593713, зашифрованное сообщение — 518245687138. Расшифруйте сообщение и введите число в качестве ответа.

Для расшифровки используется формула: $C^d \ mod \ n$, где C - зашифрованное сообщение

Задание rsa2
Мы зашифровали сообщение алгоритмом RSA с использованием паддинга PKCS#1 OAEP (RSAES-OAEP). Расшифруйте его.

✨онлайн декодер✨

Задание ssh
Сгенерируйте пару SSH-ключей. Подключитесь с использованием этой пары ключей к серверу по адресу ctddev.ifmo.ru:32337 с именем пользователя 408775 и получите ответ.

```
ssh-keygen -t rsa -b 4096 -C "<email>" # генерация пару ключей
ssh -i <путь к приватному ключу> 408775@ctddev.ifmo.ru -p 32337 # устанавливает соединение
```

Задание pgp
Напишите нам письмо на 408775@i.nsychev.ru — но не простое, а зашифрованное. Вставьте зашифрованный текст письма (он должен начинаться строкой -----BEGIN PGP MESSAGE----- и заканчиваться -----END PGP MESSAGE-----) прямо в тело сообщения.

✨онлайн декодер✨

9. Виртуализация и контейнеры

1. **first**
   Просто запустите контейнер — он выведет первый ключ в лог.

   ```
   docker run 6664104beceeadd70927ef905438169ce998e263778f7bc9dfc5b4760f1db6f3
   ```

2. **args**
   Запустите контейнер с командой give_args_key c13867cc1423a2f20a4a57ae3385b99a. В логе вы увидите другой ключ.

   ```
   docker run 6664104beceeadd70927ef905438169ce998e263778f7bc9dfc5b4760f1db6f3
   give_args_key c13867cc1423a2f20a4a57ae3385b99a
   ```

3. **env**

Запустите контейнер, передав ему переменную окружения I_NEED_KEY=370eeb6ff0d20b20125907ffa183adad. В логе вы увидите ещё один ключ.

```
docker run -e I_NEED_KEY=370eeb6ff0d20b20125907ffa183adad
6664104beceeadd70927ef905438169ce998e263778f7bc9dfc5b4760f1db6f3
```

4. **port**
   Запустите контейнер и пробросьте порт 33725 вашего компьютера на порт 8000 контейнера. Зайдите в браузере на http://localhost:33725, чтобы увидеть ключ.

```
docker run -d -p 33725:8000
6664104beceeadd70927ef905438169ce998e263778f7bc9dfc5b4760f1db6f3
```

5. **volume**
   Создайте пустую директорию на компьютере. Запустите контейнер, примонтировав эту директорию в него по пути /place. Контейнер положит ключ в эту директорию.

```
docker run -v "$(pwd)"/docker-dir:/place 6664104becee
```

6. **file**
   Запустите контейнер. Извлеките из него файл /file.png.

```
docker cp 4a4c76250475304e561dd502c9f1a02c13414dcc8f3c640f952711515a6e98bd:/file.png
file.png
```

7. **run**
   Запустите контейнер. В работающем контейнере выполните ls /var/. Ключ — название файла в директории.

```
docker exec -i -t 4b0b5b29b2fd232f203ef64ad13c8c7f78be58b5b422a86c0fb35ed70ce91231
ls /var
```

8. **lost**
   Когда мы собирали контейнер, мы добавили ключ в переменную окружения LOST_KEY, но её кто-то очистил. Посмотрите, как мы собирали контейнер и найдите переменную.

```
docker image history
6664104beceeadd70927ef905438169ce998e263778f7bc9dfc5b4760f1db6f3
```

9. **qemu** 🍓 💘 💅 ✨ 💐

1. обновить пакеты и wsl до последней версии
2. скачать qemu и связанные пакеты
3. скачать образы
4. установить параметры ядра и сетевых мостов с помощью sysctl
5. запустить libvirtd
6. запустить сеть

7. дать разрешение qemu использовать виртуальный мост (записать "allow bridge00" в файл bridge.conf, настроить права доступа)
8. настроить права доступа для qemu-bridge-helper
9. создать мак адрес для ppc64
10. запустить виртуальную машину ppc64, войти
11. загрузить и поместить client в root
12. декодировать токен
13. создать мак адрес для s390x
14. запустить виртуальную машину s390x, войти
15. загрузить и поместить server в root
16. вывести информацию о сетевом интерфейсе (ifconfig) и скопировать ip-адрес (inet addr)
17. запустить server
18. запустить client с ip-адресом и токеном

Все команды:

```
Открываем CMD
wsl --update
wsl --shutdown
Закрываем CMD

Открываем Ubuntu(Окно 1)
Открываем Ubuntu(Окно 2)
Открываем Ubuntu(Окно 3)

Окно 1:
sudo apt update
sudo apt upgrade

sudo apt install qemu
sudo apt install qemu-kvm
sudo apt install qemu-system-common
sudo apt install qemu-system-ppc64
sudo apt install qemu-system-s390x
sudo apt install qemu-kvm libvirt-daemon-system
sudo apt install libvirt-daemon
sudo apt install bridge-utils

wget https://storage.yandexcloud.net/ct-itmo-intro-public/virt/alpine-ppc64le.qcow2
wget https://storage.yandexcloud.net/ct-itmo-intro-public/virt/alpine-s390x.qcow2

sudo sysctl net.ipv4.ip_forward=1
sudo sysctl net.bridge.bridge-nf-call-iptables=1
sudo sysctl net.bridge.bridge-nf-call-ip6tables=1

sudo systemctl enable libvirtd.service
sudo systemctl start libvirtd.service

sudo virsh net-autostart --network default
sudo virsh net-start --network default

sudo mkdir /etc/qemu
cd /etc/qemu
sudo touch bridge.conf
sudo nano bridge.conf
allow virbr0
^O + Enter + ^X
```

```
sudo chown root:root /etc/qemu/bridge.conf
sudo chmod 0640 /etc/qemu/bridge.conf
sudo chmod u+s /usr/lib/qemu/qemu-bridge-helper

Окно 2:
mac_ppc64=$(printf '52:54:00:%02x:%02x:%02x\n' $((RANDOM%256)) $((RANDOM%256))
$((RANDOM%256)))

qemu-system-ppc64 -nographic -netdev bridge,id=hn0,br=virbr0 -device
virtio-net-pci,netdev=hn0,id=nic1,mac=$mac_ppc64 -drive
file=your_virtual_machine_image.img,format=qcow2

Логин: root, пароль пустой
apk add git
git clone https://github.com/4EZZZZ/files
cd files
rm server
mv client /root/
cd /root/
rm -fr files
touch token
echo 'Токен с сайта ЦК' > token
cat token | base64 -d > decoded
chmod 777 client
chmod 777 decoded

Окно 3:
mac_s390x=$(printf '52:54:00:%02x:%02x:%02x\n' $((RANDOM%256)) $((RANDOM%256))
$((RANDOM%256)))

qemu-system-s390x -nographic -netdev bridge,id=hn0,br=virbr0 -device
virtio-net-pci,netdev=hn0,id=nic1,mac=$mac_s390x -drive
file=your_virtual_machine_image.img,format=qcow2

Логин: root, пароль пустой
apk add git
git clone https://github.com/4EZZZZ/files
cd files
rm client
mv server /root/
cd /root/
rm -fr files
chmod 777 server
ifconfig
С eth0 копируем ip-адрес со второй строки с inet addr
./server

Окно 2:
./client 'inet addr' decoded

Копируем ключ и вставляем на сайт ЦК, вместе с собранными командами
```

✨ Радуемся ✨