

CTFight Automation USSR  
Практика по автоматизации

# Automation USSR

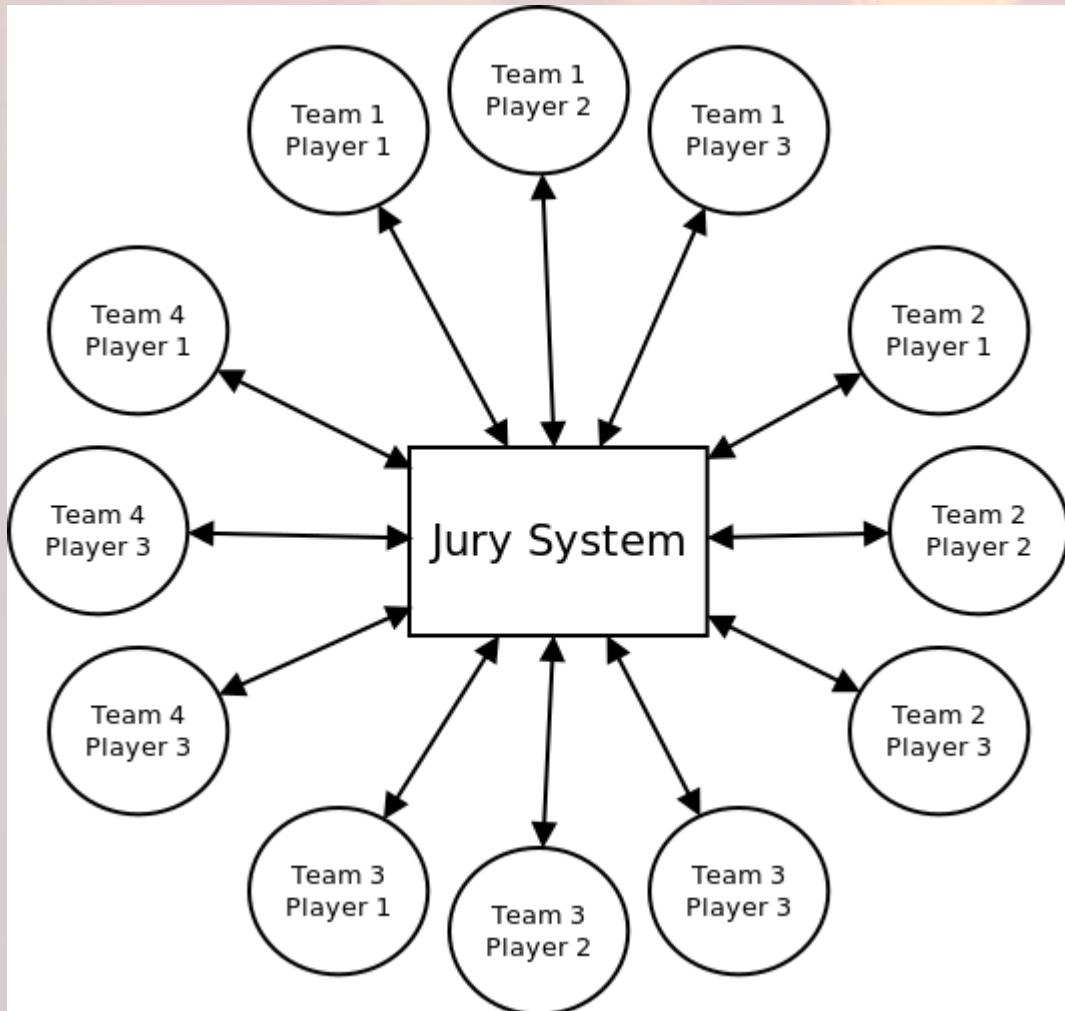


Evgenii Sopov (17.12.2015)

# План

- Лекция (30-60 мин)
- Практикуемся (2 часа)
- Подведение итогов (30-60 мин)

# Классический подход к сдаче флагов на игре

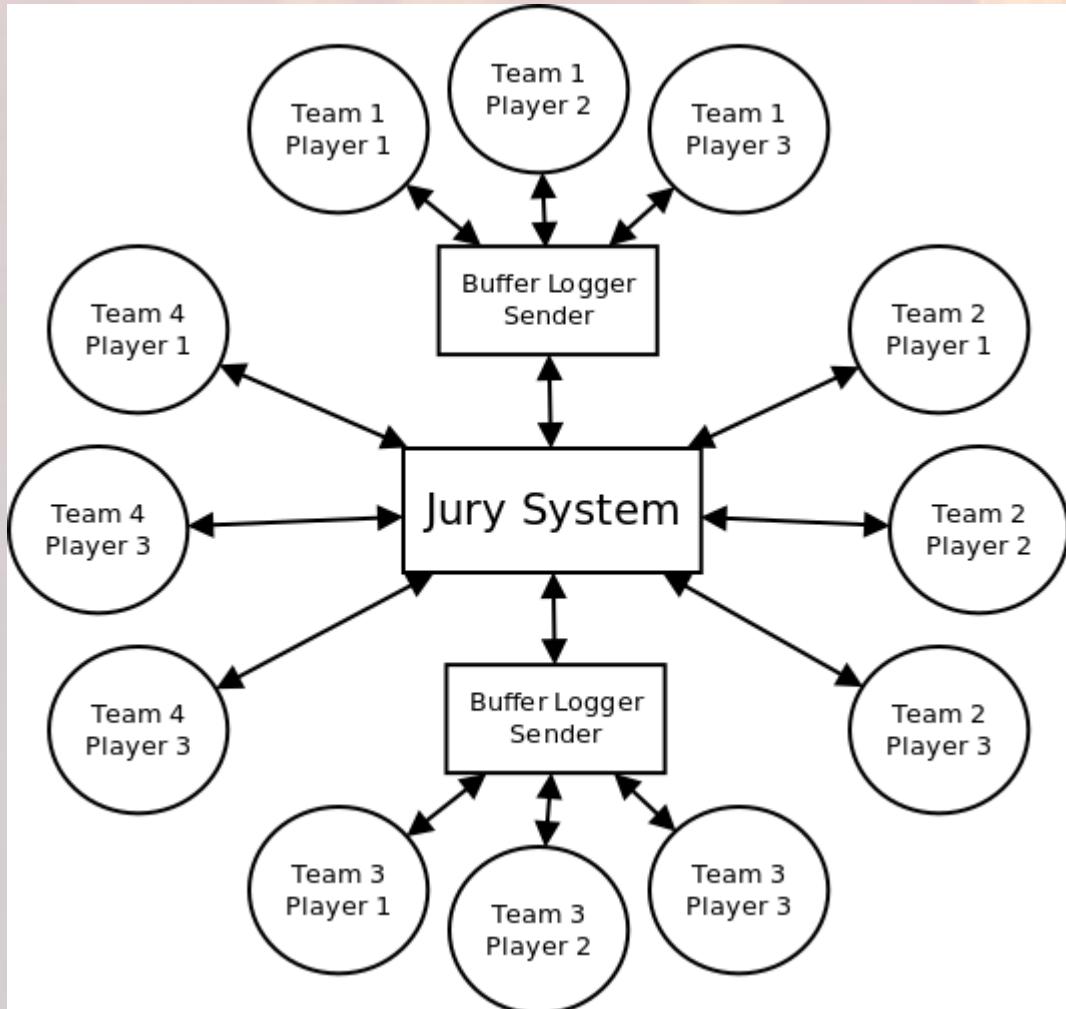


В классическом  
подходе: игроки  
взаимодействуют  
непосредственно с  
жюрийной системой

# Классический подход к сдаче флагов на игре (недостатки)

- Невозможно(или сложно) отследить какие флаги кем были сданы
- Повторная сдача флагов отнимает время
- Если жюрийная система была недоступна то найденные флаги за это время не были сданы.
- Нагрузка на жюрийную систему и внешний канал.

# Новый подход к сдаче флагов



В новом подходе игроки взаимодействуют с промежуточным буффером который находится на стороне команды. Он логирует и отправляет флаги.

# Новый подход к сдаче флагов (Достоинства)

- Логирование флагов, сбор статистики
- Если жюриная система будет недоступна, то есть возможность отложенной сдачи флагов.
- Высокий отклик от буфера
- Не отправляются дубликаты в жюриную систему

# Тезис

Один из аспектов победы –  
умение  
правильно и хорошо  
автоматизировать атаку на сервисы  
противника.

# Советы

- Атаку лучше всего проводить вдвоем или втроем.
- На изучение сервиса тратьте около 30 минут, прежде чем перейти к другому сервису.
- Страйтесь не паниковать и “не бегать” от одного сервиса к другому наводя панику в команде.
- Тихонько сидите и разбираетесь с сервисом.
- Делайте время от времени перерывы.
- Незабывайте документировать – что бы потом составить интересный writeup и поделиться им с командой после игры.

# План действий

- Выбирать сервис
- Узнать список host'ов или ip'шниками противников
- Добыть исходники (или бинари)
- Изучить протокол работы сервиса
  - Либо изучить исходники
  - Либо “руками” разобрать что да как работает
- Найти где лежат флаги, попытаться сдать любой флаг
- Найти уязвимость (если есть)
- Написать автоматизацию (незабывайте о правилах:  
Dos-атаки запрещены)
- Отдыхаете а ваш скрипт работает

# Проблемы

- Для новичков (я тоже когда то им был) все кажется достаточно сложным из-за нехватка информации и инструкций
- На текущий момент не так много игр такого типа проходят онлайн
- Практически никто из новичков ни разу нормально не писал автоматизацию. Хотя ничего сверхестественного в этом нет.
- 12-24 часов игры мало для того что бы освоить базовые навыки написания автоматизированных скриптов самостоятельно.

# Решение

- Поднят тестовый упрощенный сервис
- Вся необходимая информация здесь:  
<http://automation-ussr.sea-kg.com/>
- Сам сервис располагается на там же сервере.
- Сервис будет поднят на протяжении некоторого времени, что бы вы могли неторопясь попробовать и понять как лучше автоматизировать.

# Некоторые полезные утилиты

- “nmap” – сканер портов
- “telnet” - сетевой клиент
- “nc” - сетевой клиент
- “nano” - консольный текстовый редактор

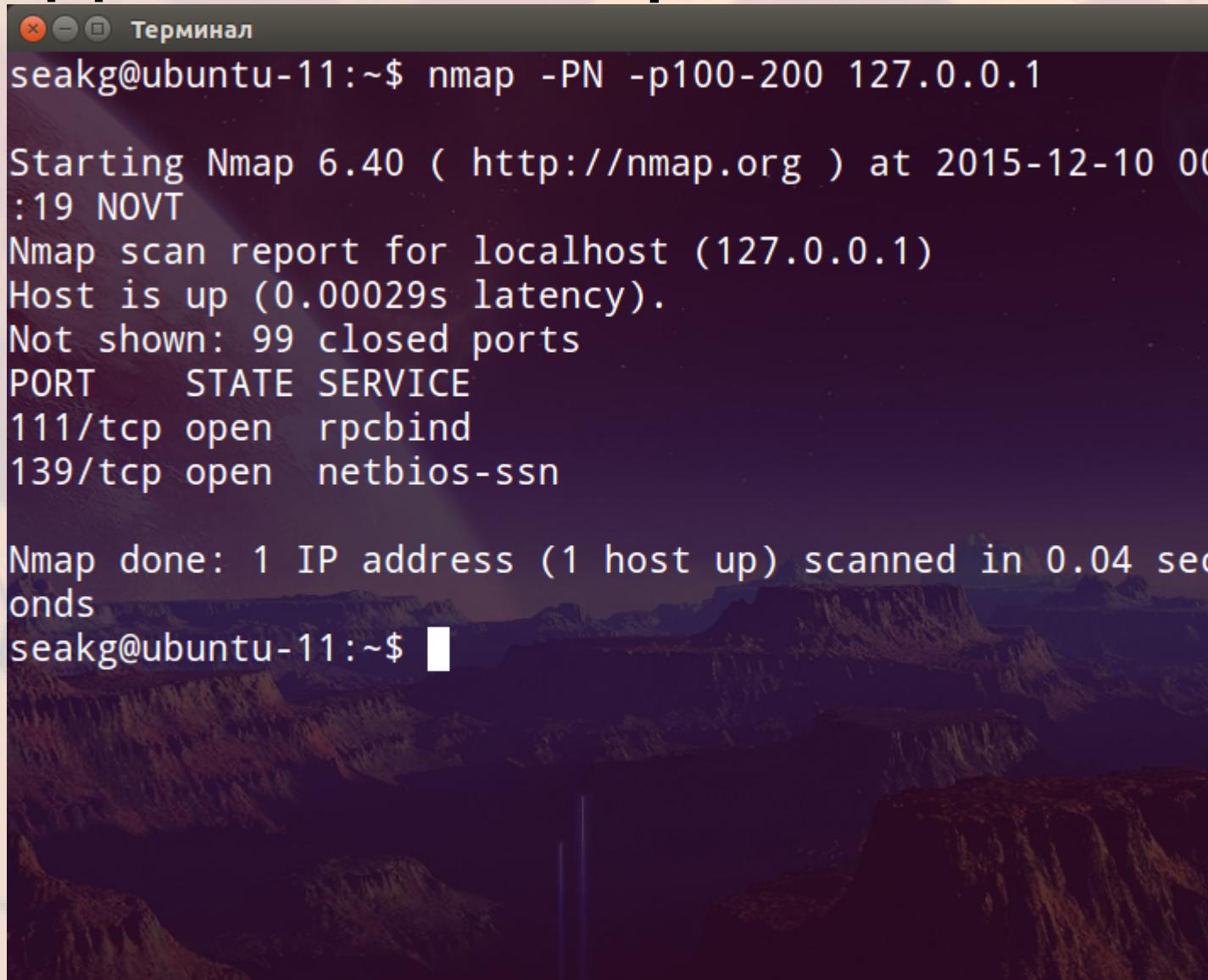
# “nmap” сканирование на стандартные порты

```
Терминал
seakg@ubuntu-11:~$ nmap -PN 127.0.0.1

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-10 00:18 NOVT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00025s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3306/tcp  open  mysql
4445/tcp  open  upnotifyp
6667/tcp  open  irc

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
seakg@ubuntu-11:~$
```

# “nmap” сканирование на определенные порты с 100 по 200

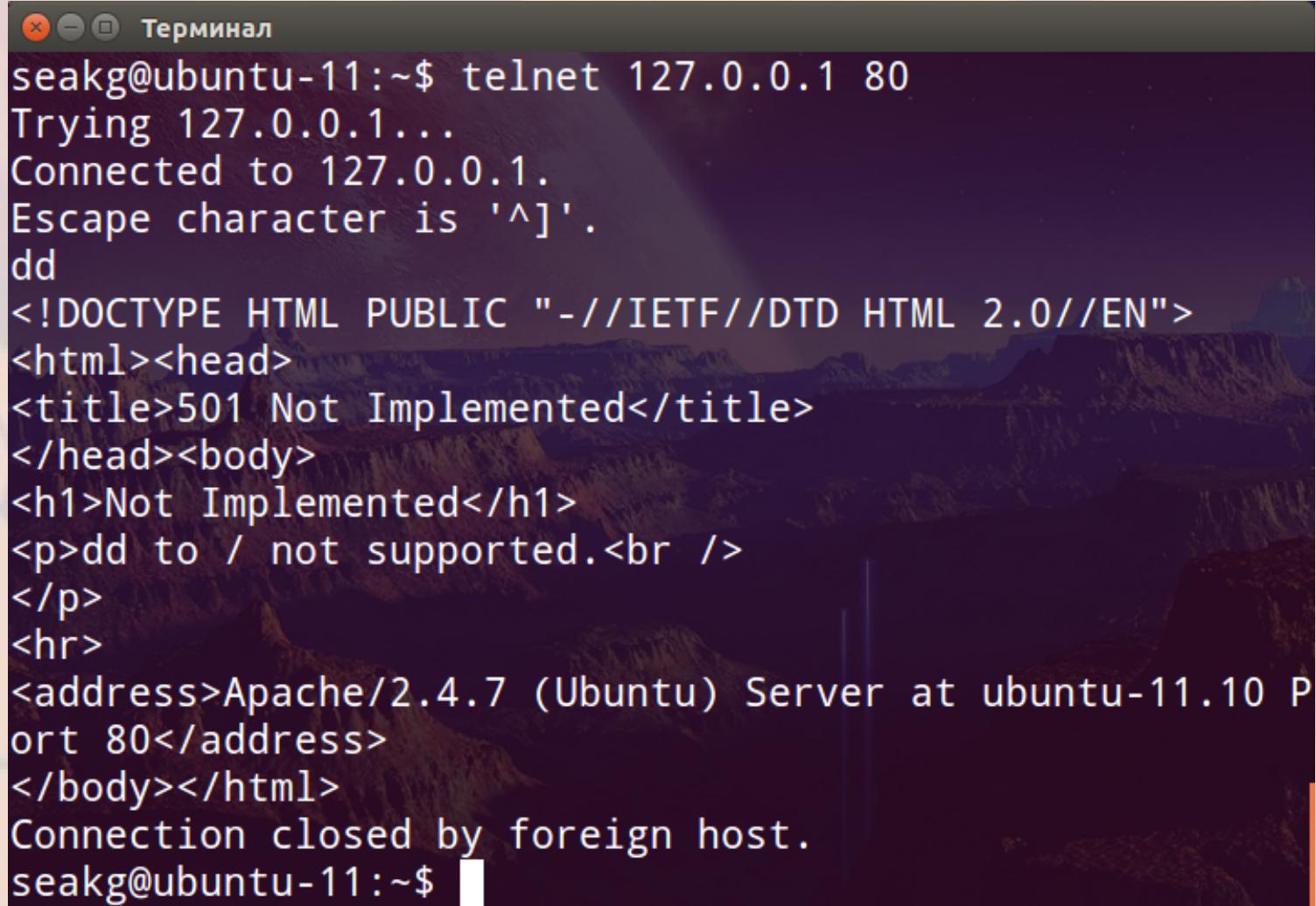
A screenshot of a terminal window titled "Терминал" (Terminal) on an Ubuntu 11.04 desktop. The window displays the output of an Nmap scan. The command entered was "nmap -PN -p100-200 127.0.0.1". The output shows the following:

```
seakg@ubuntu-11:~$ nmap -PN -p100-200 127.0.0.1
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-10 00:19 NOVT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00029s latency).
Not shown: 99 closed ports
PORT      STATE SERVICE
111/tcp    open  rpcbind
139/tcp    open  netbios-ssn

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
seakg@ubuntu-11:~$
```

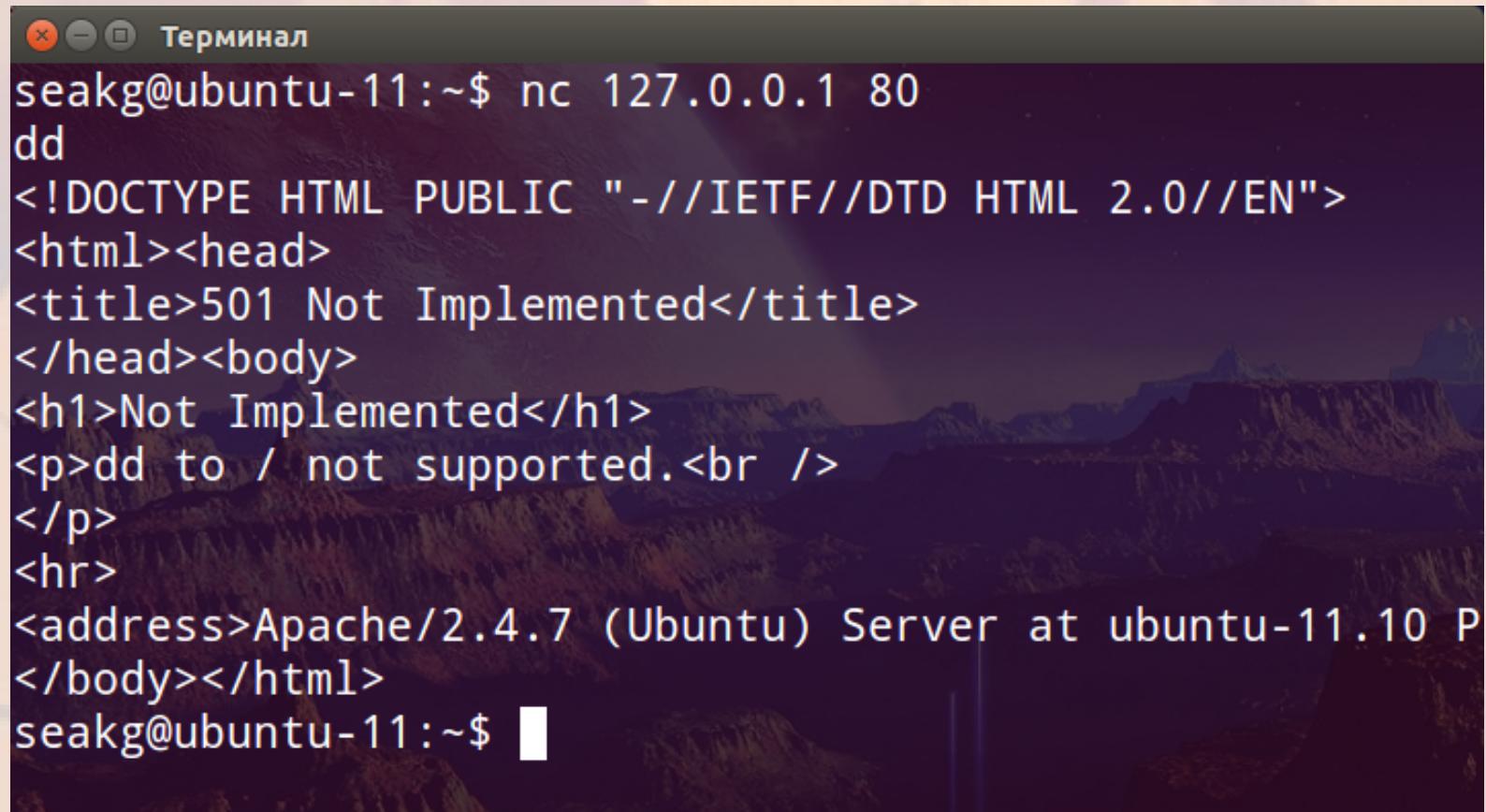
The terminal has a dark background with a mountain landscape watermark.

“telnet” подключаемся к 127.0.0.1  
на 80 порт (web server) и  
отправляем “dd” ему



```
seakg@ubuntu-11:~$ telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
dd
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Not Implemented</title>
</head><body>
<h1>Not Implemented</h1>
<p>dd to / not supported.<br />
</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at ubuntu-11.10 P
ort 80</address>
</body></html>
Connection closed by foreign host.
seakg@ubuntu-11:~$
```

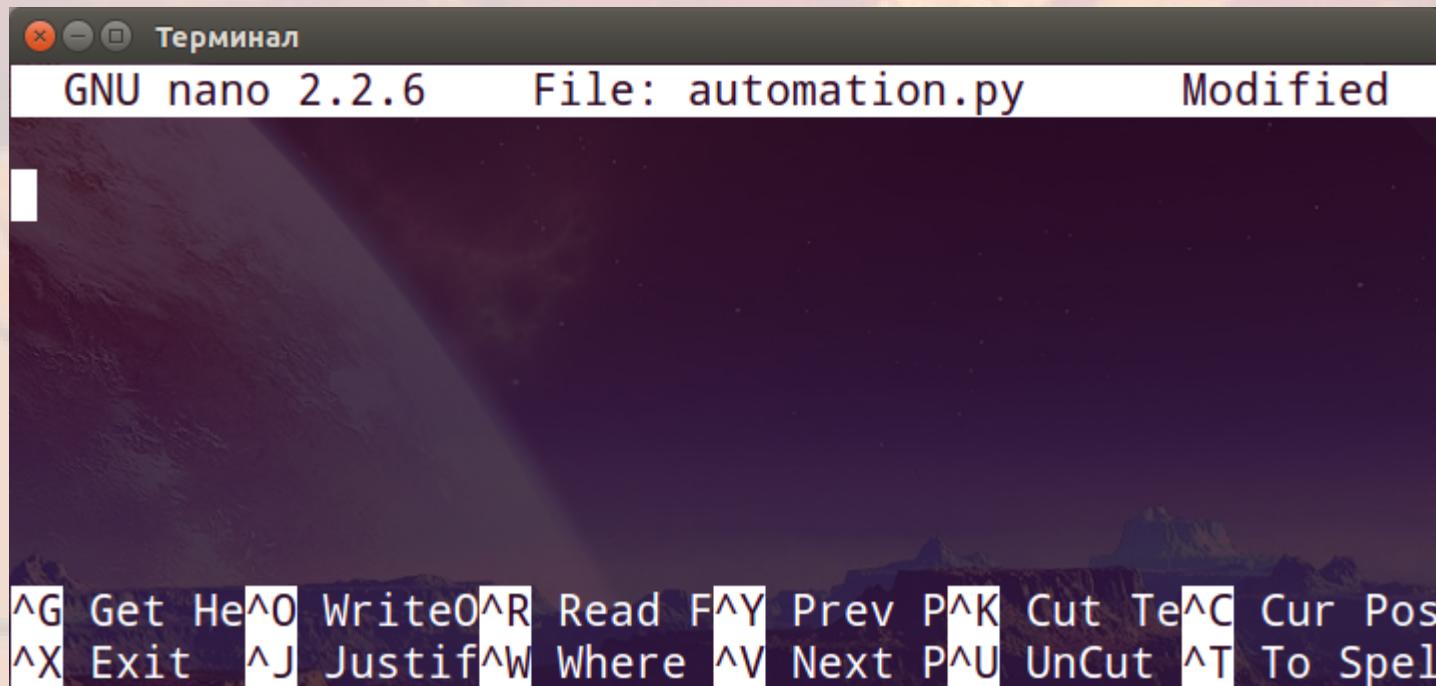
“nc” подключаемся к 127.0.0.1 на 80 порт (web server) и отправляем “dd” ему



```
seakg@ubuntu-11:~$ nc 127.0.0.1 80
dd
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Not Implemented</title>
</head><body>
<h1>Not Implemented</h1>
<p>dd to / not supported.<br />
</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at ubuntu-11.10 P
</body></html>
seakg@ubuntu-11:~$
```

# “nano”

- “nano filename” - Открыть файл для редактирования
- Ctrl+O – Сохранить (Записать изменения в файл)
- Ctrl+X – Выйти из редактора



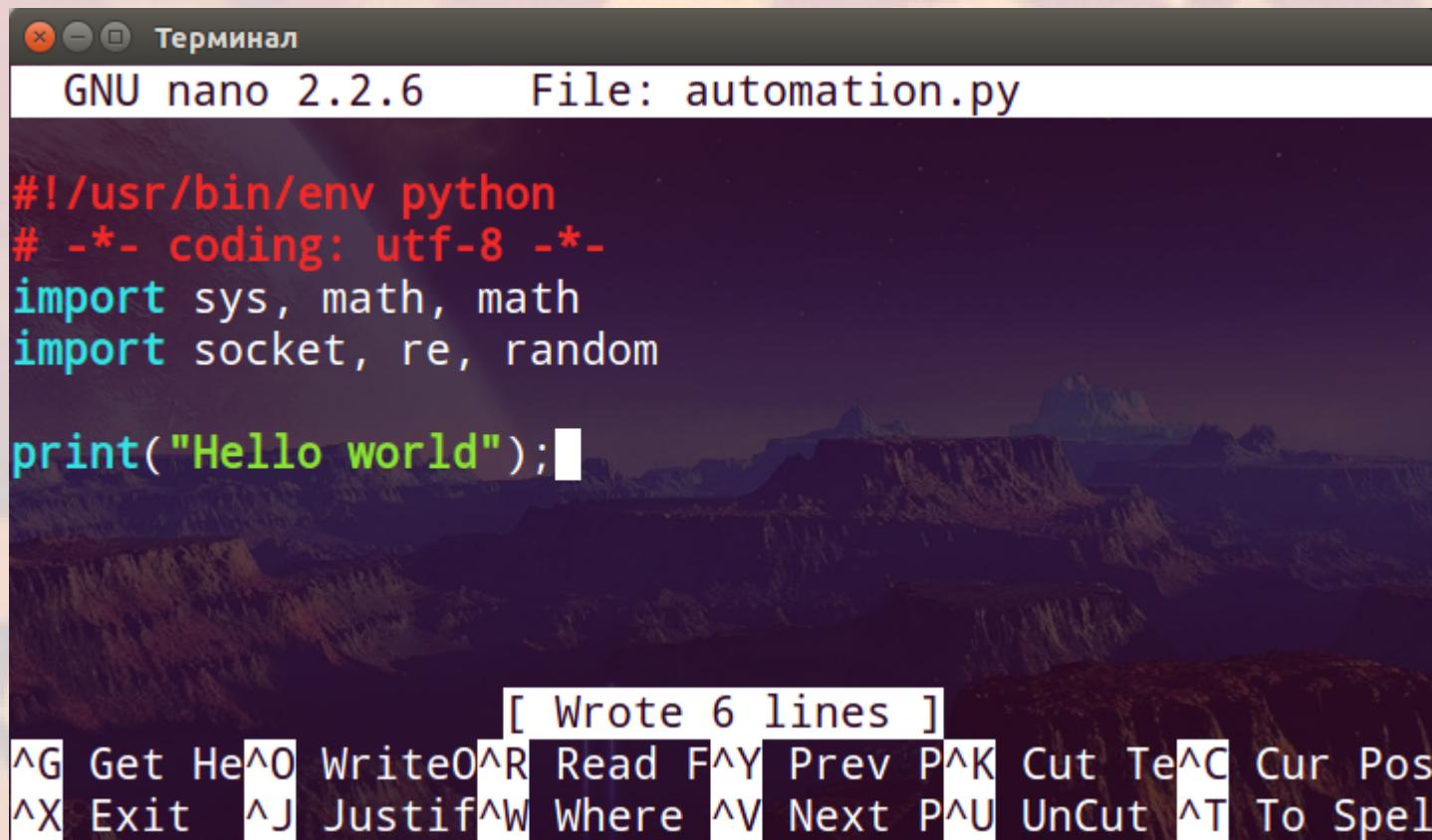
# Автоматизация (на чем?)

- “python” (рассмотрим на его примере)
- “bash”
- “c++”
- “php”
- “nodejs”
- “C#”
- Все что умеет работать с сетью

# Знакомство с Python

- Создаем файл командой “touch automation.py”
- Даем права на выполнения “chmod +x automation.py”
- Закидываем код (см следующий слайд)
- Выполнить скрипт: “./automation.py”

# Знакомство с Python (Привет Мир!)



The screenshot shows a terminal window titled "Терминал" (Terminal) running "GNU nano 2.2.6". The file being edited is "automation.py". The code in the editor is:

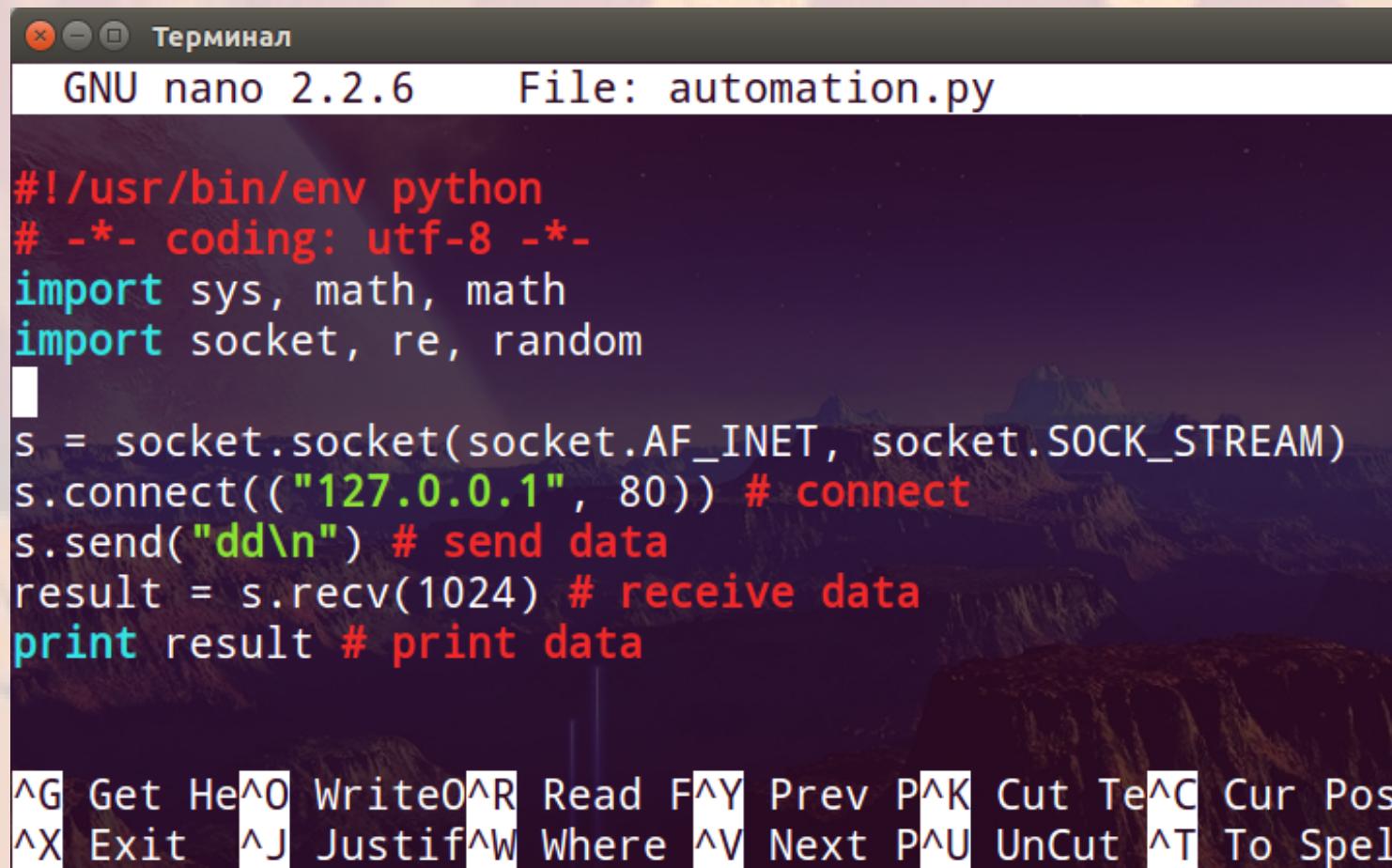
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys, math, math
import socket, re, random

print("Hello world");
```

At the bottom of the screen, a status message "[ Wrote 6 lines ]" is displayed above a series of keyboard shortcuts:

^G	Get He	^O	Write	0	^R	Read	F	^Y	Prev	P	^K	Cut	T	^C	Cur Pos
^X	Exit	^J	Justif	^W	Where	^V	Next	P	^U	UnCut	^T	To Spel			

# Знакомство с Python (сеть)



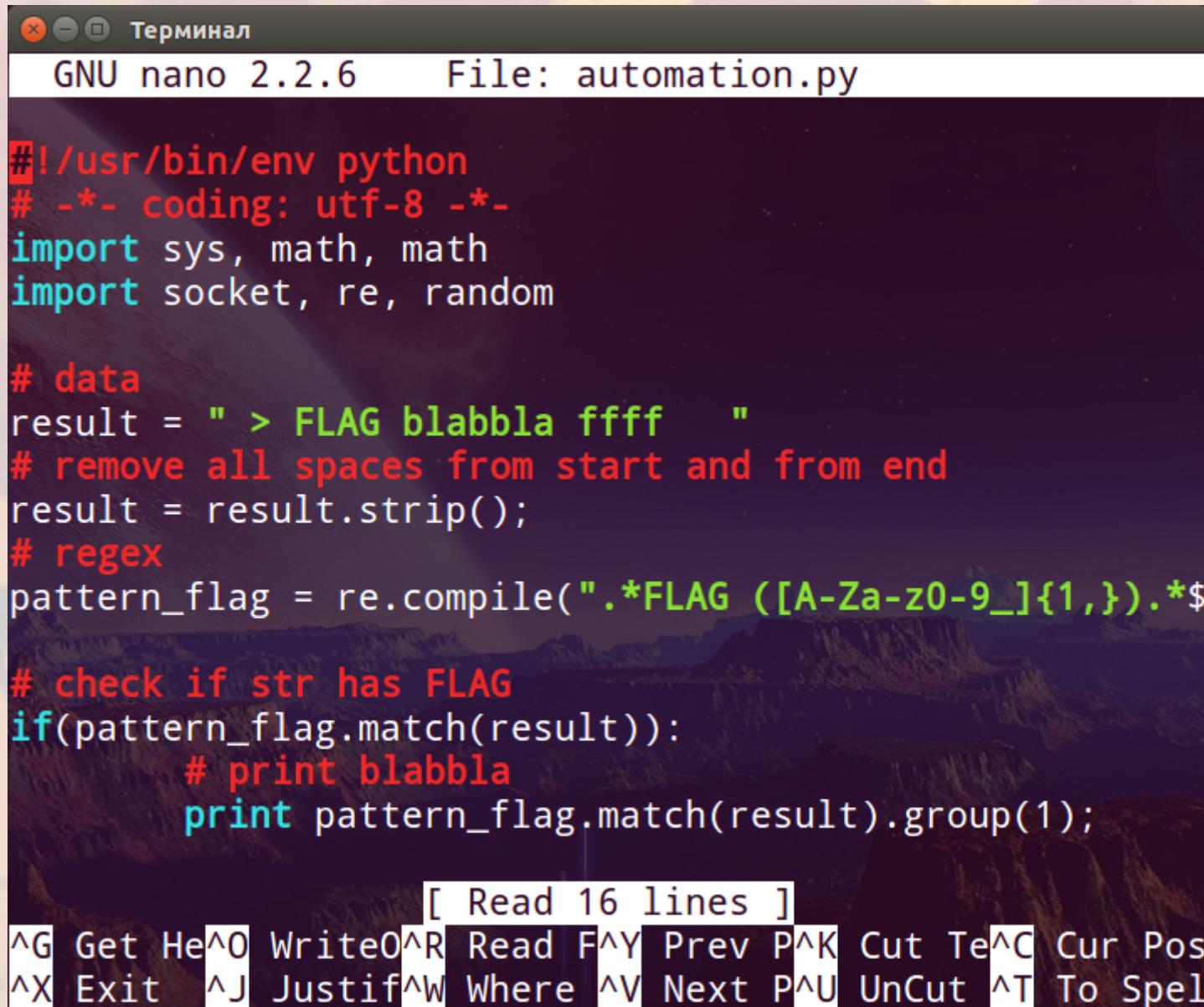
The screenshot shows a terminal window titled "Терминал" (Terminal) with the title bar "GNU nano 2.2.6 File: automation.py". The window displays the following Python script:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys, math, math
import socket, re, random

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1", 80)) # connect
s.send("dd\n") # send data
result = s.recv(1024) # receive data
print result # print data
```

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts for navigating the file.

# Знакомство с Python (парсинг данных)



The screenshot shows a terminal window titled "Терминал" (Terminal) with the file "automation.py" open in GNU nano 2.2.6. The code in the terminal is as follows:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys, math, math
import socket, re, random

# data
result = " > FLAG blabbla ffff "
# remove all spaces from start and from end
result = result.strip();
# regex
pattern_flag = re.compile(".*FLAG ([A-Za-z0-9_]{1,}).*$")

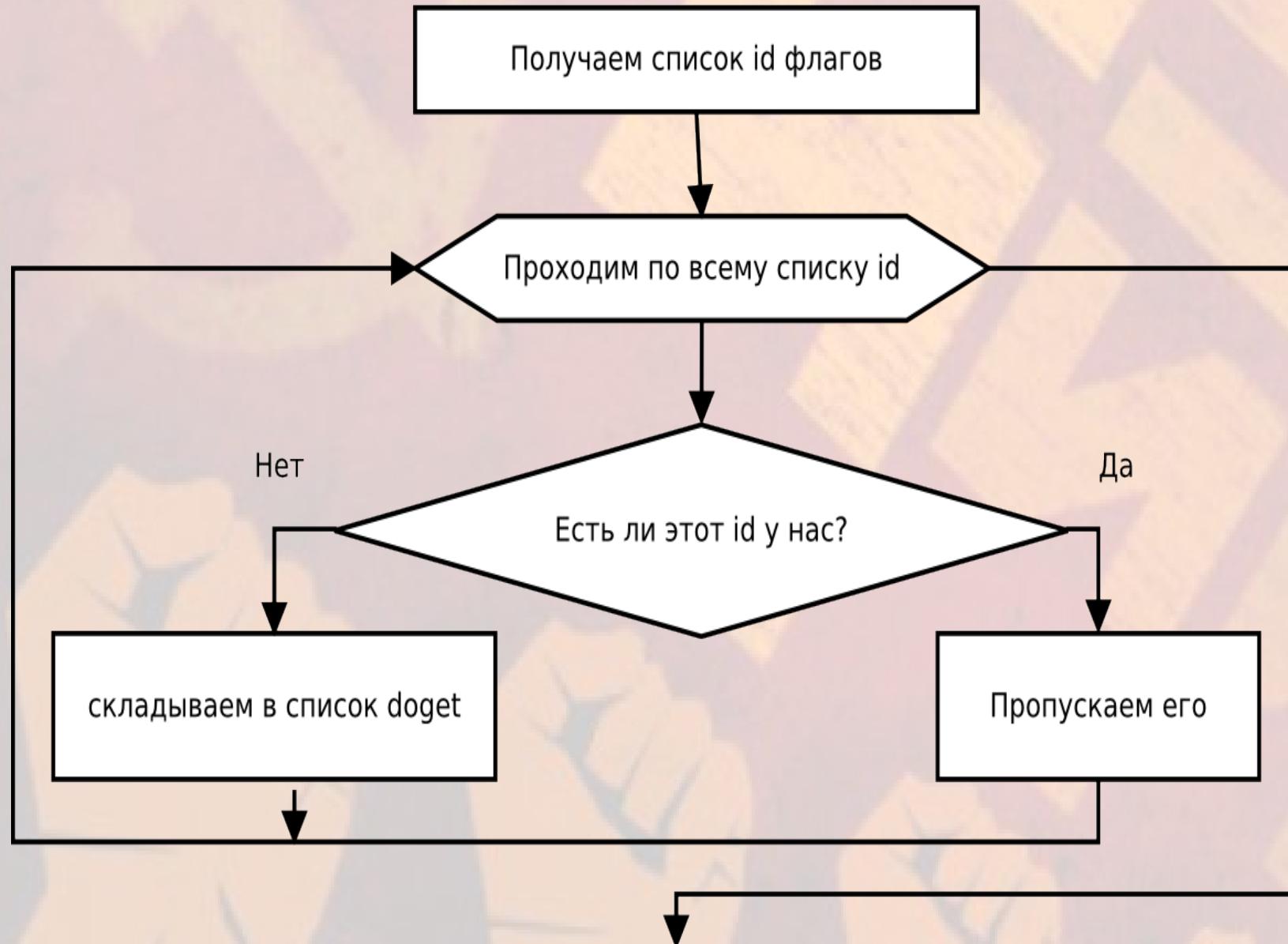
# check if str has FLAG
if(pattern_flag.match(result)):
    # print blabbla
    print pattern_flag.match(result).group(1);

[ Read 16 lines ]
^G Get He^O Write0^R Read F^Y Prev P^K Cut Te^C Cur Pos
^X Exit ^J Justif^W Where ^V Next P^U UnCut ^T To Spel
```

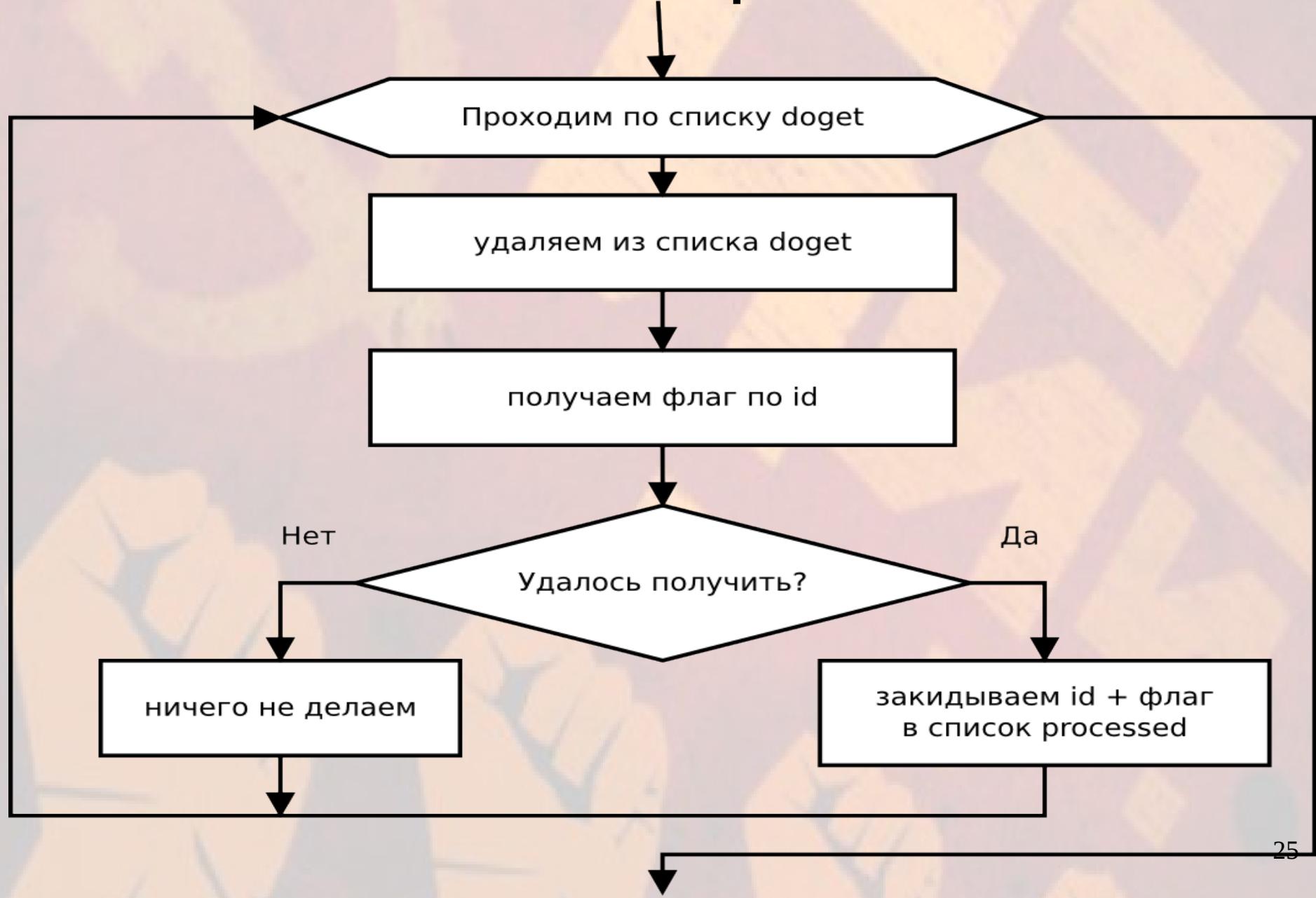
# Алгоритм работы скрипта автоматизации

- Условно его можно разделить на несколько блоков-обработчиков
- Блок первый: получение списка id флагов с атакуемого сервиса
- Блок второй: получение флагов по id
- Блок третий: сдача флагов в жюрийную систему
- В каждом блоке лучше всего использовать промежуточный список (список файлов или база данных) который будет храниться не в памяти скрипта

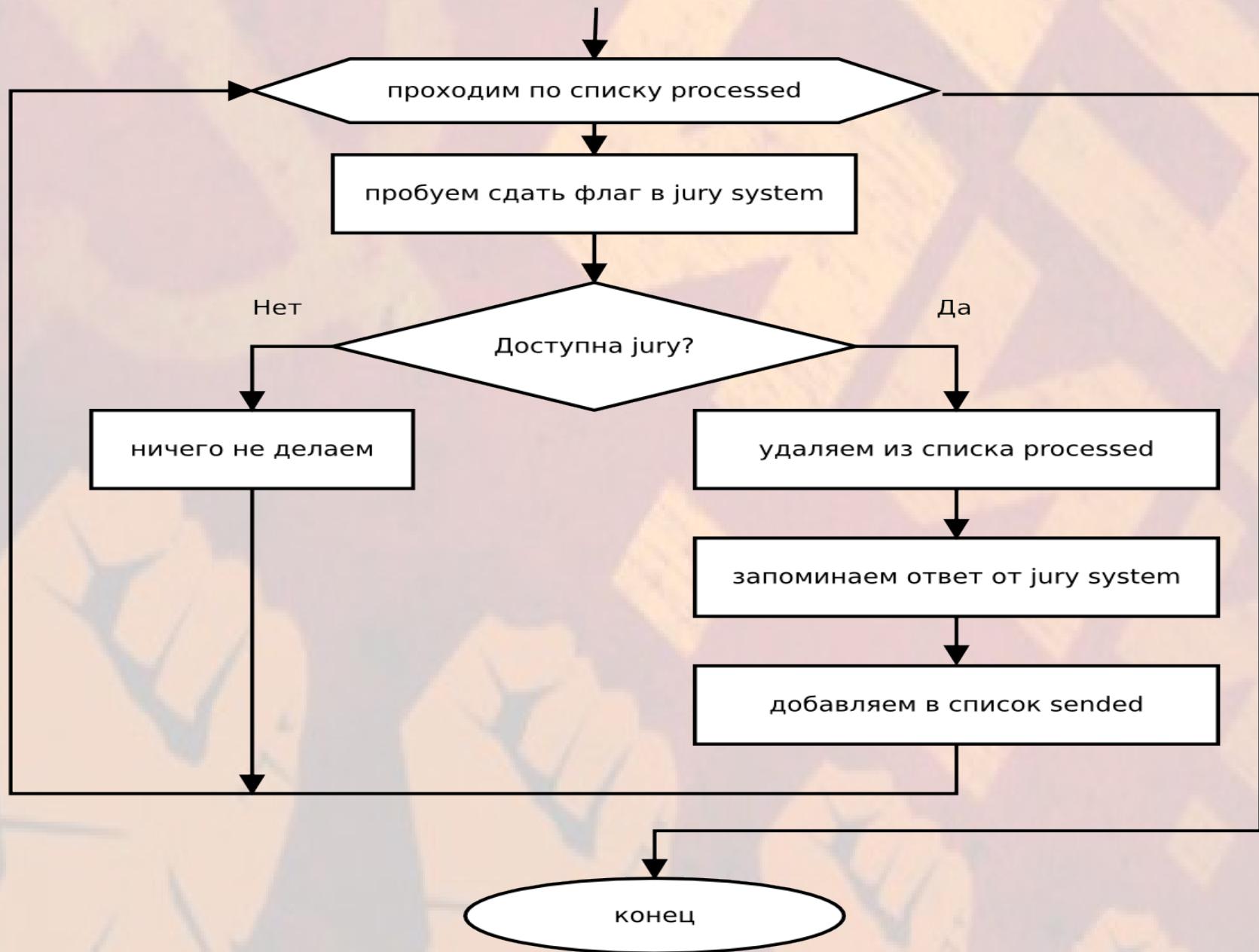
# Блок первый



## БЛОК ВТОРОЙ



# БЛОК ТРЕТИЙ



# Начало скрипта

```
seakg@seakg-notebook: ~/develop/ctfight-automation-ussr.git
GNU nano 2.2.6          Файл: automate_example.py

#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys, traceback, math, socket, re, random, os
import httplib2
from os import listdir
from os.path import isfile, join
host="80.89.147.43"
port=4445
user_token="FaNKQTWyqy"
#jury_sys="http://localhost/automation-ussr/"
jury_sys="http://automation-ussr.sea-kg.com/"

^G Помощь ^O Записать ^R ЧитФайл ^Y ПредСтр ^K Вырезать ^C ТекПозиц
^X Выход ^J Выровнять ^W Поиск ^V СледСтр ^U ОтмВырезк ^T Словарь
```

# Блок первый (получение id)

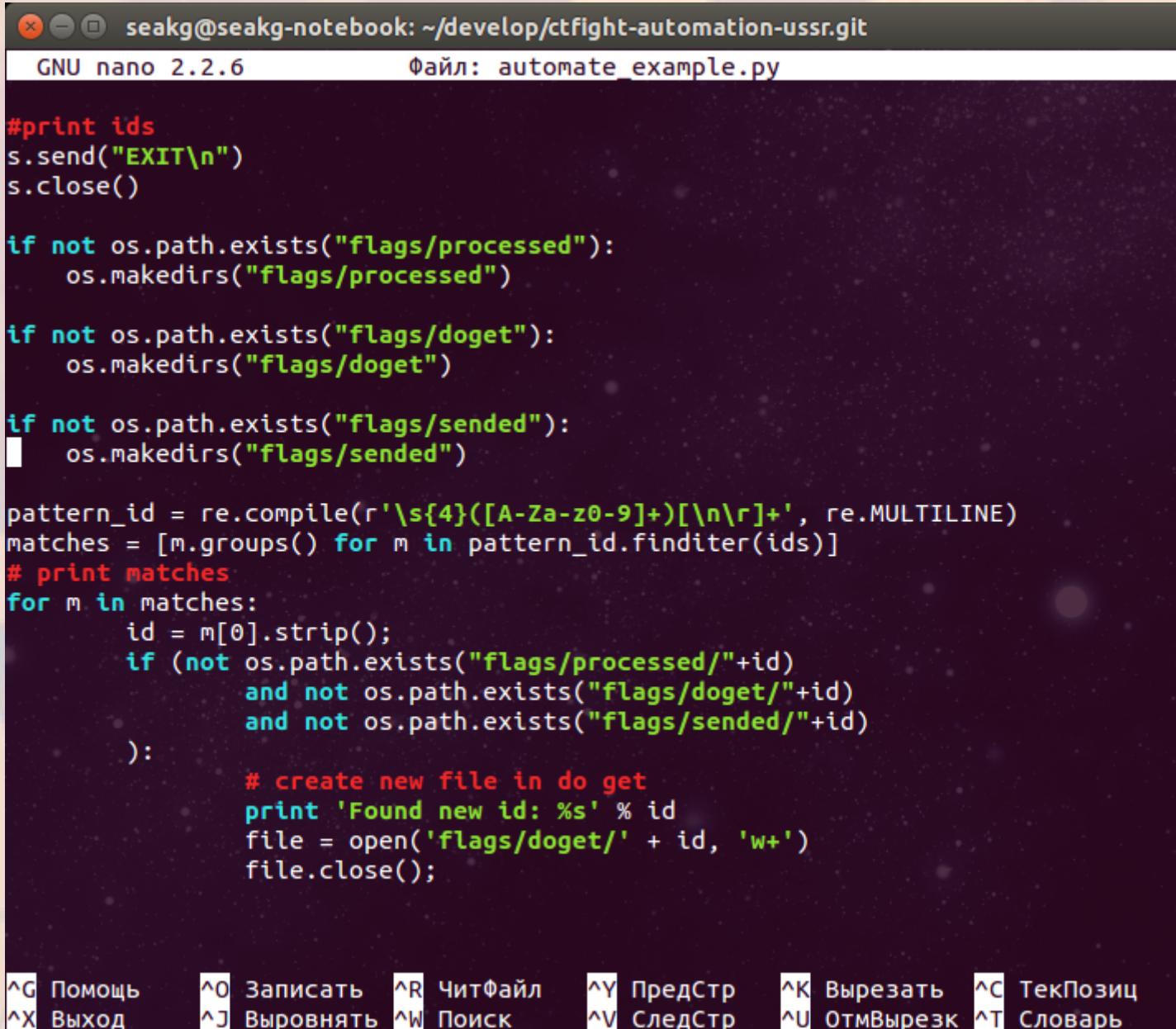
```
seakg@seakg-notebook: ~/develop/ctfight-automation-ussr.git
GNU nano 2.2.6          Файл: automate_example.py

#jury_sys="http://localhost/automation-ussr/"
jury_sys="http://automation-ussr.sea-kg.com/"

# search new flag id
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
s.recv(2048)
s.send("LIST\n")
ids=""
# search for example: "FOUND 4 ITEM(S)"
pattern_found = re.compile(".*FOUND \d+ ITEM\(\S+\).*", re.MULTILINE)
while(True):
    part = s.recv(2048)
    # print part
    ids = ids + part
    matches = [m.groups() for m in pattern_found.finditer(ids)]
    if(len(matches) > 0):
        break;
#print ids
s.send("EXIT\n")
s.close()

^G Помощь ^O Записать ^R ЧитФайл ^Y ПредСтр ^K Вырезать ^C ТекПозиц
^X Выход ^J Выровнять ^W Поиск ^V СледСтр ^U ОтмВырезк ^T Словарь
```

# Блок первый (получение флагов)



The screenshot shows a terminal window titled "seakg@seakg-notebook: ~/develop/ctfight-automation-ussr.git" running "GNU nano 2.2.6". The file being edited is "automate\_example.py". The code is a Python script designed to collect flags from a system. It starts by printing IDs and exiting. Then it checks for three directories: "flags/processed", "flags/doget", and "flags/sended". If any of these do not exist, it creates them. The script then uses a regular expression to find IDs in a log file. For each ID found, it checks if a corresponding directory in the "flags/" structure does not exist. If it doesn't, it creates the directory and prints a message indicating a new ID was found. Finally, it opens a file at the path "flags/doget/" + id and writes to it.

```
#print ids
s.send("EXIT\n")
s.close()

if not os.path.exists("flags/processed"):
    os.makedirs("flags/processed")

if not os.path.exists("flags/doget"):
    os.makedirs("flags/doget")

if not os.path.exists("flags/sended"):
    os.makedirs("flags/sended")

pattern_id = re.compile(r'\s{4}([A-Za-z0-9]+)[\n\r]+', re.MULTILINE)
matches = [m.groups() for m in pattern_id.finditer(ids)]
# print matches
for m in matches:
    id = m[0].strip();
    if (not os.path.exists("flags/processed/" + id)
        and not os.path.exists("flags/doget/" + id)
        and not os.path.exists("flags/sended/" + id)):
        # create new file in do get
        print 'Found new id: %s' % id
        file = open('flags/doget/' + id, 'w+')
        file.close();
```

^G Помощь ^O Записать ^R ЧитФайл ^Y ПредСтр ^K Вырезать ^C ТекПозиц  
^X Выход ^J Выровнять ^W Поиск ^V СледСтр ^U ОтмВырезк ^T Словарь

# БЛОК ВТОРОЙ

```
seakg@seakg-notebook: ~/develop/ctfight-automation-ussr.git
GNU nano 2.2.6          Файл: automate_example.py      Изменён

# processed all flags in folder doget
ids_doget = [f for f in listdir("flags/doget/") if isfile(join("flags/doget/", $

pattern_data = re.compile(".*DATA (.*)")
pattern_fail = re.compile(".*FAIL .*")
if(len(ids_doget) > 0):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    s.recv(2048)
    for id_doget in ids_doget:
        os.remove("flags/doget/" + id_doget)
        # print id_doget
        s.send("GET " + id_doget + "\n")
        response = s.recv(1024).strip();
        if(not pattern_data.match(response) and not pattern_fail.match($
            response = s.recv(1024).strip()

        if(pattern_fail.match(response)):
            print response

        if(pattern_data.match(response)):
            flag = pattern_data.match(response).group(1).strip();
            print("FLAG: " + id_doget + " => " + flag);
            file = open('flags/processed/' + id_doget, 'w+')
            file.write(flag);
            file.close();
    s.send("EXIT\n");
    s.close();

^G Помощь ^O Записать ^R ЧитФайл ^Y ПредСтр ^K Вырезать ^C ТекПозиц
^X Выход ^J Выровнять ^W Поиск ^V СледСтр ^U ОтмВырезк ^T Словарь
```

# БЛОК ТРЕТИЙ

```
seakg@seakg-notebook: ~/develop/ctfight-automation-ussr.git
GNU nano 2.2.6          Файл: automate_example.py

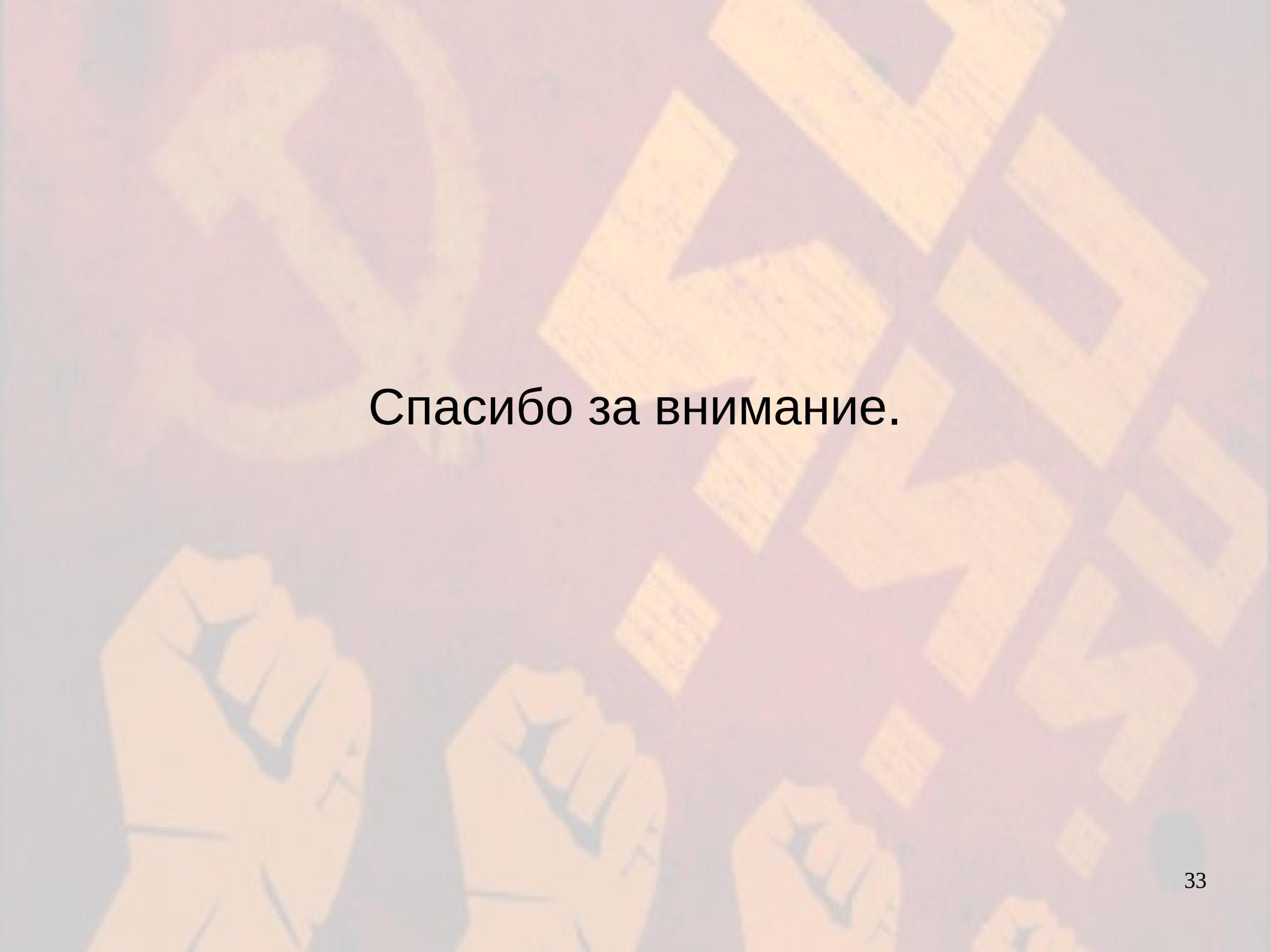
ids_processed = [f for f in listdir("flags/processed/") if isfile(join("flags/processed/", f))]

if(len(ids_processed) > 0):
    for id_processed in ids_processed:
        file = open('flags/processed/' + id_processed, 'r')
        flag=file.read().replace('\n', '')
        file.close();
        s_url=jury_sys + "flag.php?token=" + user_token + "&flag=" + flag
        resp, content = httplib2.Http().request(s_url)
        if(resp.status == 200):
            print("FLAG SENDED: " + id_processed + " => " + flag);
            os.remove("flags/processed/" + id_processed)
            file = open('flags/sended/' + id_processed, 'w+')
            file.write(flag);
            file.close();
```

^G Помощь ^O Записать ^R ЧитФайл ^Y ПредСтр ^K Вырезать ^C ТекПозиц  
^X Выход ^J Выровнять ^W Поиск ^V СледСтр ^U ОтмВырезк ^T Словарь

# Правила тренировки

- “ip” - вам скажут
- Большая часть информации находится на сайте
- Регистрация: сохраняем свой токен – он пригодится при сдачи флагов
- Не стесняемся задавать вопросы
- Скрипты оставляем на память



Спасибо за внимание.