ПРАКТИЧНА РОБОТА 4
РОБОТА З ФАЙЛАМИ

Завдання

Создать функцию get_int_vlan_map, которая обрабатывает конфигурационный файл коммутатора и возвращает два объекта:

* словарь портов в режиме access, где ключи номера портов, а значения access VLAN:

{'FastEthernet0/12':10,
 'FastEthernet0/14':11,
 'FastEthernet0/16':17}

* словарь портов в режиме trunk, где ключи номера портов, а значения список разрешенных VLAN:

{'FastEthernet0/1':[10,20],
 'FastEthernet0/2':[11,30],
 'FastEthernet0/4':[17]}

Функция ожидает в качестве аргумента имя конфигурационного файла.Проверить работу функции на примере файла config_sw.txt

config_sw.txt

```
!
version 15.0
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname S3
!
boot-start-marker
boot-end-marker
!
enable secret 4 06YFDUHH61wAE/kLkDq9BGho1QM5EnRtoyr8cHAUg.2
!
no aaa new-model
system mtu routing 1500
!
!
no ip domain-lookup
!
!
!
!
!
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
```

```
vlan internal allocation policy ascending
!
!
!
!
!
!
!
interface FastEthernet0/1
 shutdown
!
interface FastEthernet0/2
 shutdown
!
interface FastEthernet0/3
 switchport trunk allowed vlan 99,10,30
 switchport mode trunk
 !
interface FastEthernet0/4
 switchport trunk allowed vlan 99,10,30
 switchport mode trunk
 !
interface FastEthernet0/5
 shutdown
!
interface FastEthernet0/6
 shutdown
!
interface FastEthernet0/7
 shutdown
!
interface FastEthernet0/8
 switchport trunk allowed vlan 99
 switchport mode trunk
!
interface FastEthernet0/9
 switchport trunk allowed vlan 99
 switchport mode trunk
!
interface FastEthernet0/10
 shutdown
!
interface FastEthernet0/11
 shutdown
!
interface FastEthernet0/12
 shutdown
!
interface FastEthernet0/13
 shutdown
!
interface FastEthernet0/14
 shutdown
!
interface FastEthernet0/15
 shutdown
!
```

```
interface FastEthernet0/16
 shutdown
!
interface FastEthernet0/17
 shutdown
!
interface FastEthernet0/18
 switchport access vlan 10
 switchport mode access
!
interface FastEthernet0/19
 switchport access vlan 10
 switchport mode access

!
interface FastEthernet0/20
 shutdown
!
interface FastEthernet0/21
 shutdown
!
interface FastEthernet0/22
 shutdown
!
interface FastEthernet0/23
 switchport access 30
 switchport mode access

!
interface FastEthernet0/24
 switchport access 30
 switchport mode access

!
interface GigabitEthernet0/1
 shutdown
!
interface GigabitEthernet0/2
 shutdown
!
interface Vlan1
 no ip address
!
interface Vlan99
 ip address 192.168.99.13 255.255.255.0
!
ip http server
ip http secure-server
!
!
banner motd ^C
   Unauthorized Access Prohibited.^C
!
line con 0
 password 7 045802150C2E
 logging synchronous
 login
```

```
line vty 0 4
 password 7 110A1016141D
 login
line vty 5 15
 password 7 070C285F4D06
 login
!
end
```

2. РОБОТА З ФАЙЛАМИ

Навчальною метою розділу є ознайомлення студентів з роботою з файлами.
В результаті вивчення даного розділу студенти повинні знати:
− режими відкриття файлу;
− методи роботи з файлами;
уміти:
− відкривати файл в різних режимах;
− зчитувати та записувати дані в файл.

При роботі з мережним обладнанням Cisco (і не тільки) адміністратор мережі створює конфігурацію, яка визначає необхідну функціональність пристрою. Конфігураційні файли – це прості не структуровані текстові файли. Робота з ними розглядається в цьому розділі.

2.1. Відкриття та закриття файлу

Для відкриття файлу використовується метод *open()*, який повертає файловий об'єкт *file*, до якого потім можна застосовувати різні методи для роботи з ним.

open(им'я_файлу/шлях, режим_доступу)

Для зазначення режиму доступу використовують наступні символи (табл.4.1):

Таблиця 4.1

Режими відкриття файлу

Режим	Дія
'r'	відкрити файл для читання (значення за замовчуванням)
'r+'	відкрити файл для читання та запису
'w'	відкрити файл для запису (якщо файл існує, він його вміст видаляється, інакше створюється новий)
'w+'	відкрити файл для читання та запису (якщо файл існує, він його вміст видаляється, інакше створюється новий)
'a'	відкрити файл для додавання запису, при цьому нові дані будуть додані в кінець файлу
'a+'	відкрити файл для читання та додавання запису, при цьому нові дані будуть додані в кінець файлу
'x'	відкрити файл з метою створення, якщо файл існує, то виклик функції *open()* завершиться з помилкою;
'b'	бінарний режим
't'	текстовий режим (за замовчуванням)

У файлового об’єкта є атрибут *mode,* який повертає режим доступу до файлу, при цьому файл повинен бути відкритий.

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.mode
'r'
```

Атрибут *name* повертає ім’я файлу.

```
>>> f.name
'startup-config.txt'
```

Для закриття файлу використовується метод *close().* Атрибут *closed* дозволяє перевірити, закритий файл чи ні. Повертає *true* якщо файл закритий, інакше *false*:

```
>>> f.closed
False
>>> f.close()
>>> f.closed
True
```

2.2. Читання даних з файлу

Читання даних з файлу здійснюється за допомогою методів:

- *read(розмір)* - зчитує вміст файлу в рядок;
- *readline()* - зчитує файл порядково;
- *readlines()* - зчитує рядки з файлу і створює з них список.

Метод *read(розмір)* зчитує з файлу певну кількість символів, передане в якості аргументу. Якщо використовувати цей метод без аргументів, то буде зчитаний весь файл.

Приклад читання даних з файлу startup-config.txt, в якому записана така конфігурація:

```
startup-config.txt
!
version 12.4
no service timestamps log datetime msec
no service timestamps debug datetime msec
service password-encryption
```

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.read()
'!\nversion 12.4\nno service timestamps log datetime msec\nno service timestamps debug datetime msec\nservice password-encryption'
>>> f.read()
''
>>>f.close()
```

При повторному читанні файлу в 3 рядку, відображається порожній рядок, так як при першому виклику методу *read()* зчитується весь файл і курсор залишається в кінці його.

В якості аргументу методу *read()* можна передати кількість символів, яке потрібно зчитати.

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.read(15)
'!\nversion 12.4\n'
>>> f.close()
```

Метод *readline()* дозволяє зчитати рядок з відкритого файлу.

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.readline()
'!\n'
>>> f.readline()
'version 12.4\n'
>>> f.close()
```

Порядкове зчитування можна організувати за допомогою оператора for.

```
>>> f = open ("startup-config.txt" , "r" )
>>> for line in f:
        print (line)
!

version 12.4

no service timestamps log datetime msec

no service timestamps debug datetime msec

service password-encryption
>>> f.close()
```

Метод *readlines()* дозволяє зчитати рядки з файлу і зберегти їх в список.

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.readlines()
['!\n', 'version 12.4\n', 'no service timestamps log datetime msec\n', 'no service timestamps debug datetime msec\n', 'service password-encryption']
>>> f.close()
```

2.3. Запис даних в файл

Для запису в файл одного рядка використовують метод *write(рядок)*. При успішному запису він поверне кількість записаних символів. Наприклад, записати в файл config.txt список рядків з конфігурацією інтерфейса.

```
>>> conf=["interface Serial0/3/0", "ip address 192.168.3.161 255.255.255.252", "clock rate 128000"]
```

Відкриття файлу config.txt в режимі для запису:

```
>>> f_w=open("config.txt",'w')
```

Перетворення списку команд в один великий рядок за допомогою методу *join*, де роздільник символ нового рядку :

```
>>> conf_str='\n'.join(conf)
```
Запис рядку в файл:
```
>>> f_w.write(conf_str)
80
```
Після завершення роботи з файлом, його необхідно закрити:
```
>>> f.close()
```
Після відкриття файлу config.txt в ньому будуть наступні дані:
```
interface Serial0/3/0
ip address 192.168.3.161 255.255.255.252
clock rate 128000
```
Метод *writelines()* в якості аргументу очікує список рядків.
```
>>> conf=["interface Serial0/3/0", "ip address 192.168.3.161 255.255.255.252", "clock rate 128000"]
>>> f_w=open("config.txt",'w')
>>> f_w.writelines(conf)
>>> f_w.close()
```
Після відкриття файлу config.txt в ньому будуть наступні дані:
```
interface Serial0/3/0ip address 192.168.3.161 255.255.255.252clock rate 128000
```
В результаті, всі рядки зі списку, записалися в один рядок файлу, так як в кінці рядків не було символу ‘\n’. Щоб додати символ нового рядка кожному елементу списку, можна обробити список в циклі:
```
>>> conf2=[]
>>> for line in conf:
        conf2.append(line+'\n')


>>> conf2
['interface Serial0/3/0\n', 'ip address 192.168.3.161 255.255.255.252\n', 'clock rate 128000\n']
```
Якщо новий список conf2 записати заново в файл, то в ньому вже буде перенесення рядків.

2.4. Додаткові методи для роботи з файлами

При першому виклику методу *read()* зчитується весь файл і курсор залишається в його кінці. Повторне читання повертало порожній рядок, тому кожного разу доводилося відкривати файл заново, щоб знову його зчитати. Керувати положенням курсора можна за допомогою методу *seek()*. Метод *seek (позиція)* виставляє позицію в файлі.
```
>>> f = open ("startup-config.txt" , "r" )
>>> f.read()
'!\nversion 12.4\nno service timestamps log datetime msec\nno service timestamps debug datetime msec\nservice password-encryption'
>>> f.read()
''
```

```
>>> f.seek(0)
0
>>> f.read()
'!\nversion 12.4\nno service timestamps log datetime msec\nno service timestamps debug datetime msec\nservice password-encryption'
>>> f.close()
```

Метод *tell()* повертає поточну позицію "умовного курсора" у файлі. Наприклад, якщо зчитати десять символів, то "курсор" буде встановлено в позицію 10.

```
>>> f = open ("startup-config.txt" , "r" )
>>> f.read(10)
'!\nversion '
>>> f.tell()
11
>>> f.close()
```

2.5. Конструкція with

Коли з файлом потрібно працювати порядково, краще використовувати оператор *with*. При його використанні немає необхідності закривати файл. При завершенні роботи з ним ця операція буде виконана автоматично.

Щоб у виводы між рядками файлу не було зайвих порожніх рядків, можна використовувати *метод rstrip():*

```
>>> with open ( "startup-config.txt" , "r" ) as f:
        for line in f:
            print (line.rstrip())
!
version 12.4
no service timestamps log datetime msec
no service timestamps debug datetime msec
service password-encryption
>>> f.closed
True
```

З конструкцією *with* можна використовувати всі методи, які розглядалися до цього.

Додаткова література

https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files