**프로그램 실행 순서**

if \_\_name\_\_ == "\_\_main\_\_":

* 이 구문은 다른 파일에서 임포트할 때에는 \_\_name\_\_ 이 파일명이 되고, 특정 코드 블록이 직접 실행될 때에만 \_\_name\_\_ == "\_\_main\_\_" 이 되어 실행되도록 하기 위해 사용됨.

**log\_data = []**

**[EVENT 처리 Procedure]**

1. # CSV File로 로그 저장 스레드 시작.

***threading.Thread***(target=**save\_logs\_to\_csv,** daemon=True).start()

“”” SyslogUDPHandler()클레스 내 handle()함수를 통하여 log\_data[]에 저장된 data를 csv file로 저장. “””

* 1. # CSV 파일로 로그 데이터를 저장하는 함수

**def save\_logs\_to\_csv():**

valid\_logs = [log for log in log\_data if **preprocess\_log**(**log**['raw\_message']) is not None]

* 1. # 로그 데이터 전처리 함수

**def preprocess\_log(log):**

**return** {

                "pri": pri,

                "timestamp": timestamp,

                "host": host,

                "process": process,

                "message": message

            }

1. # Syslog 서버 시작

try:

***with socketserver.UDPServer***((HOST, PORT), **SyslogUDPHandler)** as server:

        print(f"Syslog server started on {HOST}:{PORT}")

        server.serve\_forever()

* socketserver.BaseRequestHandler를 상속받는”SyslogUDPHandler” 클래스를 통하여 수신한 패킷을 처리한다.

1. # SyslogUDPHandler 클래스

class **SyslogUDPHandler**(socketserver.BaseRequestHandler):

**data**, socket = self.request # self.request[0]은 데이터.

**message** = **data**.strip().decode('utf-8')

1. # 메시지 분류 및 이벤트 핸들러 호출

event\_type, interface = **classify\_message**(message)

if event\_type:

    dev\_ip\_addr = self.client\_address[0]  # syslog 수신 주소를 사용.

**handle\_event**(event\_type, message, dev\_ip\_addr, interface)

1. # 메시지 분류 함수

def **classify\_message**(message):

    interface = None

    return event\_type, interface

1. # 이벤트 처리 및 파일 저장 함수

def **handle\_event**(event\_type, message, dev\_ip\_addr, interface=None):

* 1. # Event 별 조회 및 설정 CLI 수행.

if event\_type == 'link\_down':

1. # Event 발생 시 Transceiver csv file에 정보 업데이트

            network\_state['link\_status'] = 'down'

            network\_state['link\_event'] += 1

get\_outputs = **execute\_cli\_command**(device\_params, get\_commands, "get")

set\_outputs = **execute\_cli\_command**(device\_params, set\_commands, "set")

* 1. # 로그 데이터를 텍스트 파일로 저장

**save\_log\_to\_text\_file**(file\_base\_name, event\_type, message, combined\_output)

**[TRANSCEIVER DATA Collection Procedure]**

1. # CSV File로 로그 저장 스레드 시작.

threading.Thread(target=**read\_cli\_and\_write\_to\_csv**, args=(m\_box\_ip,), daemon=True).start()

1. # Transceiver 정보 수집

def read\_cli\_and\_write\_to\_csv(m\_box):

1. # # Transceiver 정보 수집 CLI 실행 및 정규표현식으로 값 추출

def gather\_network\_stats(child):

results = defaultdict(list)

1. # # 추출된 Transceiver 정보 csv file로 저장.

 write\_to\_csv(filename, data\_to\_write)