به نام خدا

گزارش پروژه Ping

امیرحسین حیدری

9821373

محمد عباس آبادی

9823483

این کد یک ابزار پینگ ساده در ترمینال است که به کمک زبان برنامه‌نویسی Python نوشته شده است. در ادامه، یک توضیح برای این کد آورده شده است:

۱. مقدمه:

این پروژه یک ابزار پینگ ساده در ترمینال با استفاده از زبان برنامه‌نویسی Python است. پینگ یک ابزار شبکه است که برای ارسال بسته‌های کنترلی به یک دستگاه مقصد و اندازه‌گیری زمان پاسخ از آن استفاده می‌شود تا بفهمیم آدرس مورد نظر قابل دسترسی است یا خیر که اینکار توسط پروتکل ICMP انجام می شود.

۲. توابع اصلی:

- is_valid_ip:

این تابع بررسی می‌کند که یک رشته ورودی یک آدرس IP معتبر است یا خیر. بررسی شامل اعتبارسنجی تعداد بخش‌ها و محدوده مقادیر هر بخش است. در صورت معتبر بودن همان آدرس را پینگ کرده و در غیر این صورت رشته را به DNS resolver می دهیم تا نام دامنه را تبدیل به آدرس IP کند.

```python
def is_valid_ip(ip):
    con = True
    bytes = ip.split('.')
    if len(bytes) != 4:
        con = False
    for i in range(len(bytes)):
        if re.search("[0-9]+",bytes[i]) is None:
            con = False
        if int(bytes[i]) > 255 or int(bytes[i]) < 0:
            con = False
    return con
```

- create_socket:

این تابع یک سوکت ICMP ایجاد می‌کند که برای ارسال پیام‌های پینگ به دستگاه مقصد استفاده می‌شود.

```
def create_socket():
    icmp = socket.getprotobyname("icmp")
    try:
        my_socket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    except socket.error as e:
        errno, msg = e.args
        if errno == 1:
            msg = msg + (
                " - Note that ICMP messages can only be sent from processes"
                " running as root."
            )
            raise socket.error(msg)
        raise
    return my_socket
```

- resolve_destination:

این تابع یک نام میزبان یا آدرس IP را از طریق رزولوشن DNS به IP تبدیل می‌کند.

```
def resolve_destination(destination):
    try:
        answers = dns.resolver.resolve(destination, 'A')
        destination_ip = answers[0].address
    except (dns.resolver.NoAnswer, dns.resolver.NXDOMAIN):
        print('Invalid hostname or IP address.')
        sys.exit(1)
    except (dns.resolver.LifetimeTimeout):
        print("DNS resolution timeout")
        sys.exit(1)
```

- calculate_checksum:

این تابع یک دایجست از دیتا با الگوریتم checksum ایجاد می کند که در هدر ICMP کاربرد دارد. در این الگوریتم دیتای ورودی را که شامل چندین بایت است به صورت ۱۶ بیتی جمع کرده در آخر carry را به آن اضافه کرده و همه بیت ها را invert می کنیم و عددی ۱۶ بیتی بازمی گردانیم.

```python
def calculate_checksum(data):
    checksum = 0
    count_to = (len(data) // 2) * 2
    for count in range(0, count_to, 2):
        checksum += (data[count + 1] << 8) + data[count]
    if count_to < len(data):
        checksum += data[len(data) - 1]
    checksum &= 0xFFFFFFFF
    checksum = (checksum >> 16) + (checksum & 0xFFFF)
    checksum += (checksum >> 16)
    return ~checksum & 0xFFFF
```

- send_ping_request:

این تابع یک بسته ICMP پینگ را به دستگاه مقصد ارسال کرده و زمان ارسال را ذخیره می‌کند. ابتدا هدر های مربوط به بسته پینگ را مشخص کرده و سپس توسط سوکت باز شده به آن میفرستیم. هدر ها شامل موارد زیر میباشد:

Type

Code

Checksum

Identifier

Sequence

رشته BBHHH! معادل تایپ های زیر است که به ترتیب استفاده شده است:

```
struct Foo {
  unsigned char a;
  unsigned char b;
  unsigned short c;
  unsigned short d;
  unsigned short e;
 }
```

```python
def send_ping_request(destination_ip, my_socket):
    icmp_type = 8
    icmp_code = 0
    icmp_checksum = 0
    icmp_identifier = 12345
    icmp_sequence = 1
    icmp_payload = b'0'
    timeout = 3

    icmp_header = struct.pack('!BBHHH', icmp_type, icmp_code, icmp_checksum, icmp_identifier, icmp_sequence)
    icmp_checksum = socket.htons(calculate_checksum(icmp_header + icmp_payload))
    icmp_header = struct.pack('!BBHHH', icmp_type, icmp_code, icmp_checksum, icmp_identifier, icmp_sequence)

    t = time.time()
    my_socket.sendto(icmp_header + icmp_payload, (destination_ip, 0))
    my_socket.settimeout(timeout)
    return t
```

- receive_ping_reply:

این تابع تا زمان تایم اوت منتظر پاسخ‌های پینگ از دستگاه مقصد می ماند و زمان دریافت را محاسبه می‌کند. اگر کد و تایپ بسته دریافتی معادل کد و تایپ دریافت پینگ باشد به معنای پینگ شدن است.

```python
def receive_ping_reply(my_socket, start_time):
    try:
        while True:
            received_packet, _ = my_socket.recvfrom(1024)
            receive_time = time.time()
            icmp_header = struct.unpack('!BBHHH', received_packet[20:28])
            if icmp_header[0] == 0 and icmp_header[1] == 0:
                t = f'{int((receive_time - start_time) * 1000)}ms'
                print(f'Ping successful: time={t}')
                return t
    except socket.timeout:
        print('Request Timed out.')
```

- ping:

این تابع یک پینگ به یک دستگاه مقصد ارسال می‌کند و زمان پاسخ را دریافت می‌کند.

```python
def ping(destination):
    if is_valid_ip(destination):
        destination_ip = destination
    else:
        destination_ip = resolve_destination(destination)
    icmp_socket = create_socket()
    start_time = send_ping_request(destination_ip, icmp_socket)
    return receive_ping_reply(icmp_socket, start_time)
```

۳. اجرای اصلی:

در این بخش، ورودی‌های کاربر از طریق خط فرمان (Command Line Interface) با استفاده از ماژول argparse پردازش می‌شوند. تعداد بسته‌های پینگ و مقصد از کاربر گرفته می‌شود.

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Ping tool in the terminal.')
    parser.add_argument('destination', help='IP address or domain to ping')
    parser.add_argument('packet_count', help='Number of packets to ping')
    destination , packet_count = parser.parse_args().destination , int(parser.parse_args().packet_count)
    print(f"Pinging {destination} with {packet_count*8} bytes of data:")
    lost = 0
    sumation = 0
    for i in range(packet_count):
        temp = ping(destination)
        if temp == None :
            lost += 1
        else:
            sumation += int(temp[:-2])

    print("PACKETS** sent:",packet_count,"received:",packet_count-lost,"lost:", lost,f"({lost/packet_count*100}%)loss")
    if packet_count!=lost:
        print("Approximate round trip times in milli-seconds:\n\tAverage = ",sumation/(packet_count-lost),"ms")
```

۴. نتایج:

پس از ارسال تعداد مشخصی از درخواست‌های پینگ، اطلاعات آماری نظیر تعداد بسته‌های ارسالی، تعداد بسته‌های دریافتی، تعداد بسته‌های گم‌شده و درصد از دست رفته شده نمایش داده می‌شود. در صورت موفقیت، زمان میانگین پاسخ نیز نمایش داده می‌شود.

۵. نکات پایانی:

این پروژه از کتابخانه‌های مختلف از جمله socket، struct، time، argparse، sys و dns.resolver برای ارتباط شبکه، ساختار داده، مدیریت زمان و پردازش آرگومان‌ها استفاده می‌کند. همچنین، کدها و توابع به گونه‌ای طراحی شده‌اند که از قابلیت خوانایی بالا برخوردار باشند. همچنین شامل موارد امتیازی همچون استفاده از گیت در پیاده سازی و همچنین دیگر موارد امتیازی است.