수업 명 : 시스템프로그래밍

과제 이름 : Proxy 2-1

학 과: 컴퓨터정보공학부

담당 교수님: 김태석 교수님

분 반: 월5, 수6

학 번: 2023202043

성 명: 최은준

0. Introduction

이 과제에서는 TCP 기반의 proxy server와 client를 구현하는 것이다.

클라이언트는 사용자가 입력한 URL을 서버에 전송하고, 서버는 해당 URL을 SHA-1 해시 방식으로 변환한다. 그다음 캐시 디렉터리 내에 저장하거나 기존에 존재하는지 확인하여 HIT 또는 MISS 여부를 판별한다. 서버는 각각의 클라이언트 요청에 대해 독립된 프로세스로 처리하며, 처리 결과 및 로그를 logfile에 기록한다.

1. Flow Chart

1-1. proxy server #2-1 코드 작성 순서도

위 코드 순서도의 동작 방식은 다음과 같다.

1. 클라이언트 연결 수락 대기
서버 프로그램은 소켓을 생성하고, 지정된 포트(8080번)에서 클라이언트 연결 요청을 대기한다.
연결이 수락되면 클라이언트의 요청을 처리하기 위해 fork()를 통해 자식 프로세스를 생성하여 클라이언트의 요청을 처리하고, 부모 프로세스는 클라이언트 소켓을 닫고 다음 연결을 대기한다.

2. 클라이언트 요청 처리
자식 프로세스는 클라이언트로부터 URL 문자열을 수신한다.
받은 URL은 SHA1 해시 함수를 통해 해시 문자열로 변환하여 해시값의 앞 3자리로 디렉터리를 만들고, 나머지 문자열로 캐시 파일명을 구성한다.

동일한 캐시 파일이 존재할 경우 HIT으로 처리, 로그 기록 후 클라이언트에게 응답하고, 그렇지 않을 경우 MISS로 처리, 파일을 생성하고 로그 기록, 클라이언트에게 응답한다.
로그는 ./logfile/logfile.txt파일에 남기며, 처리된 해시값과 원본 URL 모두 기록된다.

3. 클라이언트 종료 및 로그 기록
클라이언트가 "bye" 문자열을 전송하면 자식 프로세스는 실행 시간, 총 HIT/MISS 횟수를 로그 파일에 기록하고 종료된다.

4. 서버 종료 처리
서버는 클라이언트 연결을 계속 대기하며, 사용자가 서버를 강제 종료할 경우 SIGCHLD시그널을 무시하여 좀비 프로세스 발생을 방지한다.

2. Pseudo code

2-1. proxy server #2-1 알고리즘

```
Function main():
    Create server socket
    Bind socket to IP and PORT
    Listen for client connections
    Ignore SIGCHLD to prevent zombie processes

    Loop:
        Accept incoming client connection
        Fork a new process
            If child process:
                Close server socket
                Call handle_client(client_socket, client_address)
            Else if parent process:
                Close client socket
            Else:
                Print fork error
    End Loop
    Close server socket
End Function
```

```
Function handle_client(client_socket, client_address):
    Initialize buffer and URL/hash variables
    Initialize hit and miss counters
    Record start time

    Create 'cache' and 'logfile' directories if not exist
    Print client connection message

    Loop:
        Read URL string from client
        If read fails or input is "bye":
            Break loop

        Store input_url from client
        Compute SHA1 hash of input_url → hashed_url
        Extract first 3 characters of hashed_url → dir_name

        Create cache subdirectory based on dir_name
        Construct full cache file path

        If cache file exists:
            Increment hit counter
            Log HIT (hashed path and original URL)
            Send "HIT" to client
        Else:
            Increment miss counter
            Create new cache file with placeholder content
            Log MISS with input_url
            Send "MISS" to client
    End Loop

    Record end time
    Log termination info (PID, run time, hit/miss count)
    Print client disconnection message
    Close client socket
    Exit process
End Function
```

```
Function sha1_hash(input_url, hashed_url):
    Compute SHA1 hash of input_url
    Convert hash to hex string → hashed_url
    Return hashed_url
End Function
```

```
Function is_cache_hit(path, filename):
    Construct full file path from path and filename
    Return whether file exists
End Function
```

```
Function write_log(status, info, pid):
    Open logfile for appending
    Get current timestamp
    Write HIT or MISS log entry based on status
    If HIT and info is not a path, also log original URL
    Close logfile
End Function
```

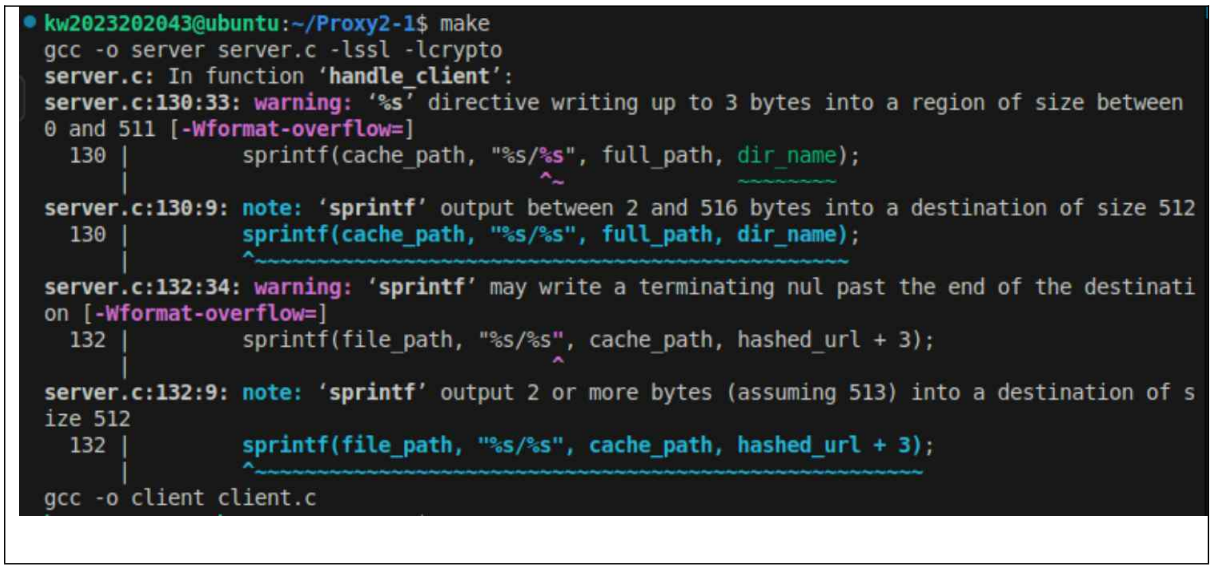
```
Function write_termination_log(pid, run_time, hit, miss):
    Open logfile for appending
    Write termination log with PID, time, hit/miss count
    Close logfile
End Function
```

2-1. proxy client #2-1 알고리즘

```
Function main():
    Create client socket

    Set up server address with:
        - IPv4 address (127.0.0.1)
        - Port number (8080)

    Connect to the server using the socket and address
    If connection fails, terminate with error

    Loop:
        Prompt user to input URL
        Read input into message buffer
        Send message to server

        If input is "bye":
            Break loop

        Read server response (HIT or MISS)
        Print server response
    End Loop

    Close client socket
    Exit program
End Function
```

```
Function error_handling(message):
    Print system error message for 'message'
    Terminate program with error status
End Function
```

3. 결과화면



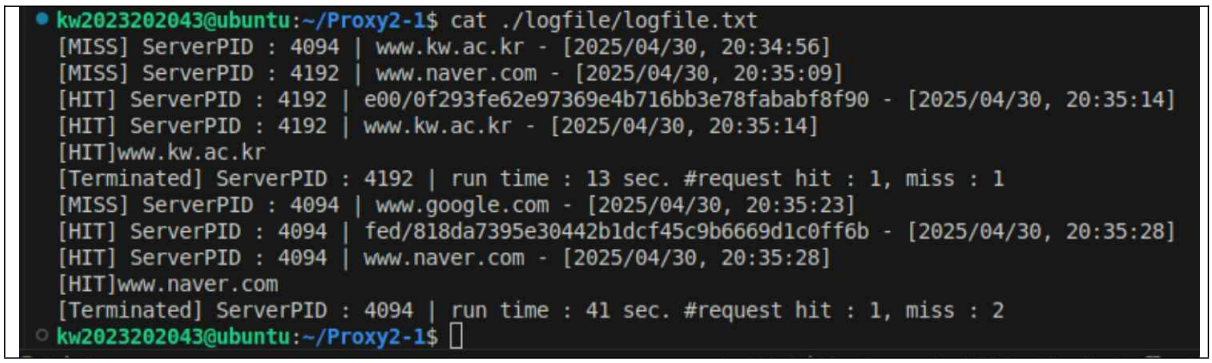
코드 작성 후 make 한 뒤 client와 server가 생성된 것을 확인할 수 있다.

server에서 client를 기다리고, client가 연결되면 자식 process를 생성하여 클라이언트 요청을 처리한다. 사용자가 URL을 입력하고, 만약 동일한 cache file이 존재할 경우 HIT, 아닐 경우 MISS로 처리하여 HIT인지 MISS인지 판별한다.

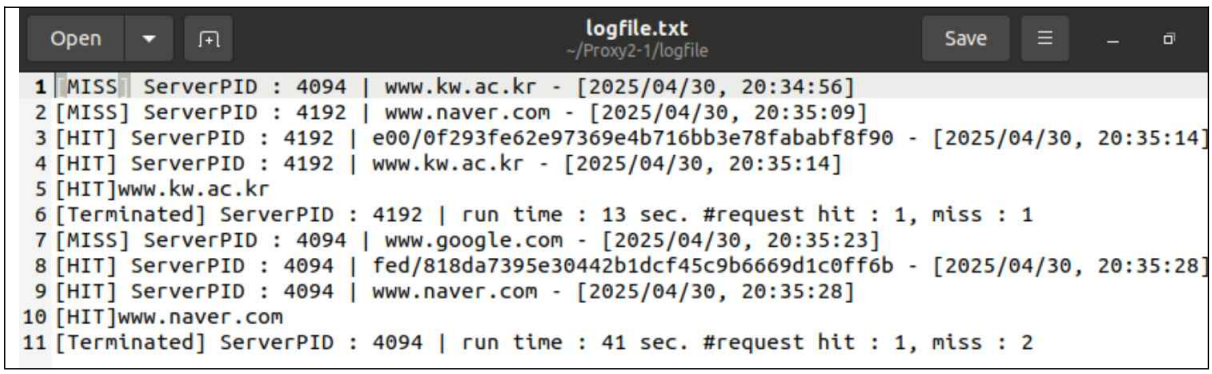
만약 bye를 입력하면 실행 시간과 총 HIT, MISS 횟수를 log file에 기록한다. server는 계속 다른 client 연결을 대기하며 전체 client 요청 처리를 log file에 기록한다.

log는 ./logfile/logfile.txt file에 남기며 처리된 해시값과 url 등이 기록된다. server는 사용자가

server를 강제 종료할 경우 SIGCHLD 신호를 무시하여 zombie pocess 발생을 방지한다.



cat ./logfile/logfile.txt는 현재 디렉토리 기준(./)로그 파일의 내용을 출력하는 명령어다. 처리된 해시값과 원본 URL, 실행 시간, 총 HIT/MISS 횟수가 logfile에 기록된 것을 알 수 있다.



위와 같이 logfile에 잘 기록된 것을 확인할 수 있다.

4. 고찰

이번 과제를 통해 TCP 소켓 프로그래밍, 프로세스 생성(fork)을 활용한 동시 처리 방식에 대해 실습할 수 있었다. 특히 클라이언트는 connect 함수를 통해 서버에 명시적으로 연결을 요청하고, 서버는 accept 함수를 통해 클라이언트의 연결을 수락히야 클라이언트 요청을 독립적인 프로세스로 처리함으로써 TCP 통신의 흐름을 배울 수 있었다. 로그 파일에 기록과 처리를 남기면서 디버깅과 프로그램 흐름을 파악하는 데 수월했다. 마지막으로, fork를 이용한 자식 프로세스 분기, SIGCHLD 시그널 무시를 통한 좀비 프로세스 방지, 정상 종료 시 로그 기록 등으로 시스템 프로그래밍의 개념을 배울 수 있었다.