수업 명 : 시스템프로그래밍

과제 이름 : Proxy 2-2

학 과: 컴퓨터정보공학부

담당 교수님: 김태석 교수님

분 반: 월5, 수6

학 번: 2023202043

성 명: 최은준

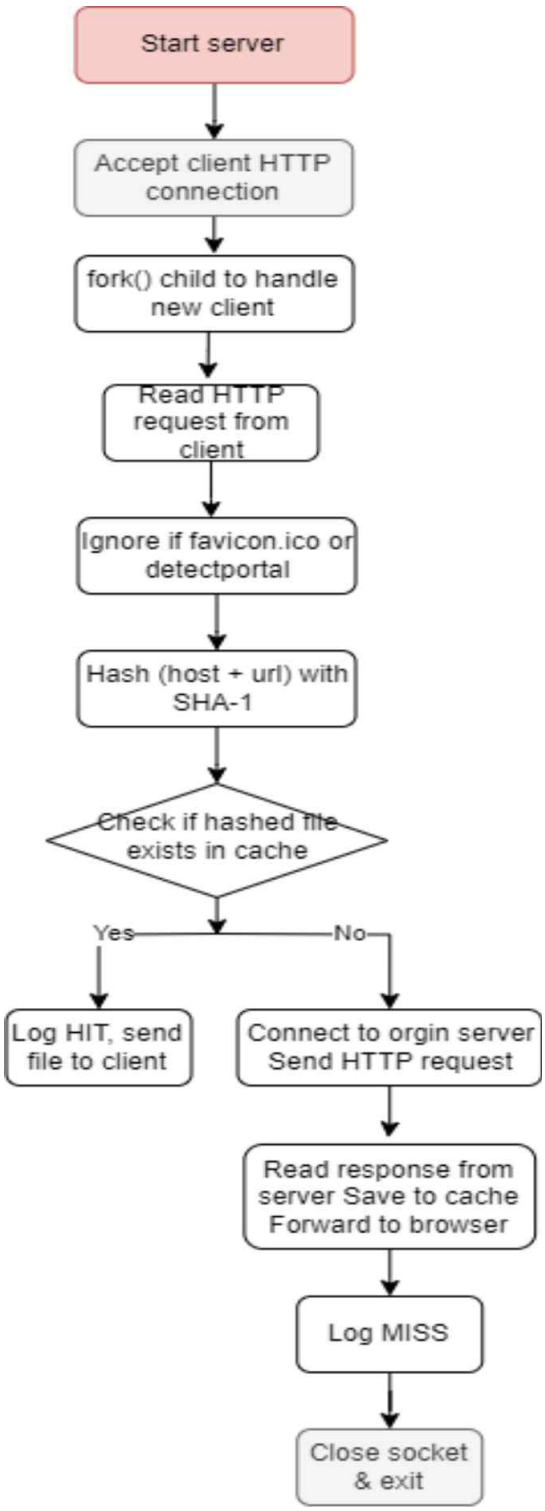
0. Introduction

이 과제에서는 웹 브라우저로부터의 HTTP 요청을 처리하는 프록시 캐시 서버를 구현하는 것이다.
프록시 서버는 클라이언트인 웹 브라우저로부터 요청을 수신한 후, 해당 요청에 대한 캐시 파일이 존재하는지 검사하고, 캐시가 존재하는 경우에는 이를 브라우저로 전송, 존재하지 않는 경우에는 원 서버로부터 데이터를 가져와 클라이언트에 전달함과 동시에 캐시에 저장한다.

서버는 fork()을 통해 동시에 여러 클라이언트의 요청을 처리할 수 있으며, favicon.ico, detectportal과 같은 자동 요청은 캐시 및 로그에서 제외한다.

1. Flow Chart

1-1. proxy server #2-2 코드 작성 순서도



위 코드 순서도의 동작 방식은 다음과 같다.

1. 서버는 포트 39999에서 대기하며, 클라이언트 연결 수락 시 fork()를 사용해 자식 프로세스 생성

2. 브라우저로부터 HTTP GET 요청 수신

3. favicon.ico와 같은 자동 요청은 무시

4. host + url문자열을 SHA1 해시 → 캐시 디렉터리 및 파일 경로 결정

5. 파일이 이미 존재하면 (HIT): 로그 기록 후 파일 내용 전송

6. 존재하지 않으면 (MISS):

6-1. 원 서버에 접속하여 요청

6-2. 응답을 브라우저에 전송하고 캐시에 저장

6-3. MISS 로그 기록

7. 연결 종료 및 자식 프로세스 종료

2. Pseudo code

2-1. proxy client #2-2 알고리즘

```
Function main():
    Ignore SIGCHLD to prevent zombie processes
    Create TCP socket, bind, and listen on PORT
    While true:
        Accept client connection
        Fork a child
        If child:
            Close server socket
            Call handle_client()
        Else:
            Close client socket
End Function
```

```
Function handle_client(clnt_sock, clnt_addr):
```

```
    Read HTTP request
    If method != GET → exit
    If request is favicon or detectportal → exit
    Extract host and url
    full_url ← host + url
    hash ← sha1_hash(full_url)
    Construct cache path from hash
    If file exists:
        Log "HIT"
        Send cached file to client
    Else:
        Connect to origin server
        Send GET request
        Read response, send to client and save to cache
        Log "MISS"
    Close client socket and exit
End Function
```

```
Function sha1_hash(input):
    Perform SHA1 hash on input string
    Return 40-character hex string
End Function
```

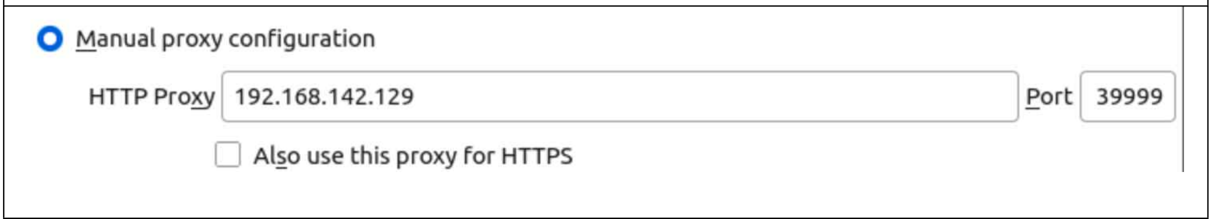
```
Function is_cache_hit(path):
    Return true if file at path exists
End Function
```

```
Function write_log(status, info, ip, port):
    Open logfile for appending
    Get current time
    Write entry: "[status] ip:port | info - [YYYY/MM/DD, HH:MM:SS]"
    Close logfile
End Function
```

```
Function send_file_to_client(sock, filepath):
    Open file
    While data remains:
        Read and send to client socket
    Close file
End Function
```

3. 결과화면

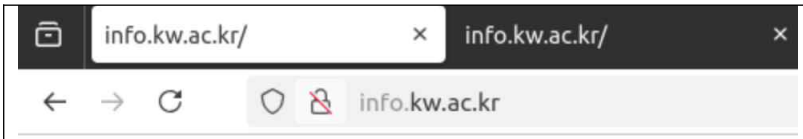
```
kw2023202043@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.142.129  netmask 255.255.255.0  broadcast 192.168.142.255
        inet6 fe80::d5dd:c95a:5333:8e37  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:da:27:8e  txqueuelen 1000  (Ethernet)
        RX packets 7002  bytes 4894020 (4.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5016  bytes 1647316 (1.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 5732  bytes 546606 (546.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5732  bytes 546606 (546.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

kw2023202043@ubuntu:~$
```



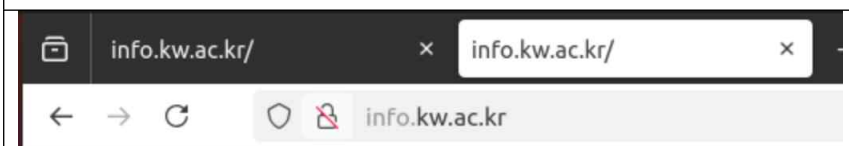
ifconfig 명령어로 IP 주소를 확인한 뒤 프록시 설정을 해준다. 이 때 포트 번호를 39999로 설정한다.

```
rm -f proxy_cache
kw2023202043@ubuntu:~/Proxy2-2$ make
gcc -o proxy_cache proxy_cache.c -lssl -lcrypto
proxy_cache.c: In function ‘handle_client’:
proxy_cache.c:169:30: warning: ‘sprintf’ may write a terminating nul past the en
d of the destination [-Wformat-overflow=]
  169 |     sprintf(full_path, "%s/%s", cache_path, hash + 3);
      |                              ^
proxy_cache.c:169:5: note: ‘sprintf’ output 2 or more bytes (assuming 513) into
a destination of size 512
  169 |     sprintf(full_path, "%s/%s", cache_path, hash + 3);
      |     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
kw2023202043@ubuntu:~/Proxy2-2$ ./proxy_cache
Proxy server is running on port 39999...
```

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



MISS
192.168.142.129:59128
http://info.kw.ac.kr/
kw2023202043



HIT
192.168.142.129:48812
http://info.kw.ac.kr/
kw2023202043

```
kw2023202043@ubuntu:~/Proxy2-2$ ./proxy_cache
Proxy server is running on port 39999...
[192.168.142.129 : 59128] client was connected
================================================
Request from [192.168.142.129 : 59128]
GET http://info.kw.ac.kr/ HTTP/1.1
Host: info.kw.ac.kr
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:136.0) Gecko/20100101 Fir
efox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i

================================================
[192.168.142.129 : 59128] client was disconnected
[192.168.142.129 : 48812] client was connected
================================================
Request from [192.168.142.129 : 48812]
GET http://info.kw.ac.kr/ HTTP/1.1
Host: info.kw.ac.kr
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:136.0) Gecko/20100101 Fir
efox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i

================================================
[192.168.142.129 : 48812] client was disconnected
^C
kw2023202043@ubuntu:~/Proxy2-2$
```

서버는 클라이언트의 접속을 기다리다가 요청이 오면 fork()를 통해 자식 프로세스를 생성하여 요청을 처리한다. http://info.kw.ac.kr에 접속하면, 서버는 해당 URL을 SHA1 해시하여 캐시 디렉토리를 확인한다. 첫 번째 요청: 캐시 파일이 존재하지 않음 → MISS(원 서버에서 리소스를 받아 캐시에 저장 후 브라우저로 전송, logfile/log.txt에 MISS 로그가 기록됨), 두 번째 요청: 캐시 파일이 이미 존재함 → HIT(캐시 파일을 브라우저에 전송, logfile/log.txt에 HIT 로그가 기록됨)

위와 같이 logfile/log.txt에는 각 요청에 대한 HIT/MISS 및 URL, IP/Port, 타임스탬프가 기록되었음을 확인할 수 있다.

4. 고찰

이번 과제를 통해 실제 웹 브라우저가 요청하는 HTML 등 다양한 리소스를 프록시 서버를 통해 처리하는 방식에 대해 실습할 수 있었다.

단순히 HIT/MISS 여부만을 응답하는 것이 아니라, 실제 리소스를 원 서버에서 받아와 브라우저로 전송하고 동시에 캐시에 저장하는 방식으로 구현하여 프록시 캐시 서버의 흐름에 대해 배울 수 있었다.

특히 host + url전체를 해시하여 캐시 키를 생성함으로써 중복 캐시 문제를 방지했다. 또한, 시그널 처리, 로그 기록, 자동 요청 필터링 등 과제에서 요구한 세부 기능 등으로 시스템 프로그래밍 개념을 배울 수 있었다.