시스템프로그래밍

과 제 명 : Proxy 2-2

교 수 님 : 최상호 교수님

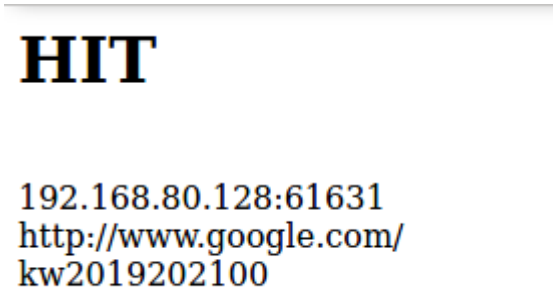
강의시간 : 월요일

학 번 : 2019202100

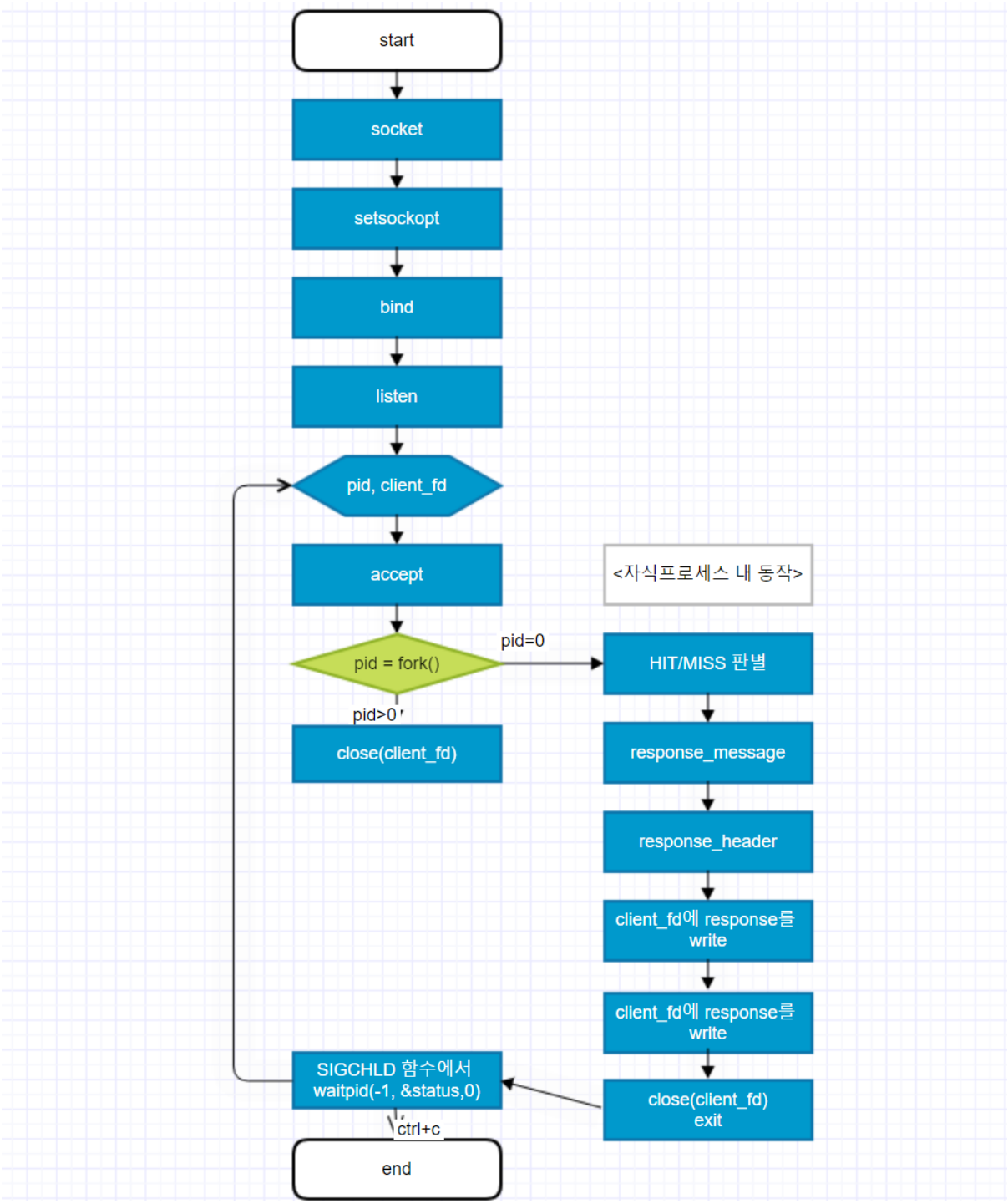
이 름 : 하주영

Introduction

웹 브라우저가 HTTP request를 보내면 서버가 HTTP response를 보내도록 구현하는 과제이다. 동작은 다음과 같다. 첫번째는 client의 browser에서 포트번호와 ip를 설정한다. 두번째로 client의 browser는 모든 HTTP request를 서버로 전송한다. HTTP request 정보 중 host(url) 정보를 추출하여 HIT/MISS를 판별한다. (#1-2 과제) 네번째로 response 정보(response header+response data)를 client에게 전송하고 웹 브라우저에서 "HIT or MISS₩n ip:port₩n kw본인학번"을 아래와 같이 출력한다.



Flow Chart


start
socket
setsockopt
bind
listen
pid, client_fd
accept
pid = fork()
pid=0
pid>0
close(client_fd)
<자식프로세스 내 동작>
HIT/MISS 판별
response_message
response_header
client_fd에 response를 write
client_fd에 response를 write
close(client_fd) exit
SIGCHLD 함수에서 waitpid(-1, &status,0)
ctrl+c
end

Pseudo Code

```
int main() {
 struct sockaddr_in server_addr, client_addr;
 int socket_fd, client_fd;
 socket_fd = socket(PF_INET, SOCK_STREAM, 0);
 server_addr.sin_family = AF_INET;
 server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 server_addr.sin_port = htons(PORTNO);

 int opt = 1;
 setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
 bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr))
 listen(socket_fd, 5);
 signal(SIGCHLD,(void *)handler);
 while(1) {
     struct in_addr inet_client_address;
     char response_header, response_message;
     client_fd=accept(socket_fd,(struct sockaddr *)&client_addr, &len);

     printf("[%s:%d] client was connected.\n",
                      inet_ntoa(inet_client_address), client_addr.sin_port);

     pid = fork();
     if(pid == 0) {
         read(client_fd, buf, 1024);
         HIT/MISS 판별
         response_message에 1. hit or miss 2. ip:port 3. url 4. kw학번을 저장
         response_header에 "HTTP/1.0 200 OK\r\n"
                               "Server:2018 simple web server\r\n"
                               "Content-length:%lu\r\n"
                               "Content-type:text/html\r\n\r\n",
strlen(response_message) 저장
         write(client_fd, response_header, strlen(response_header));
```

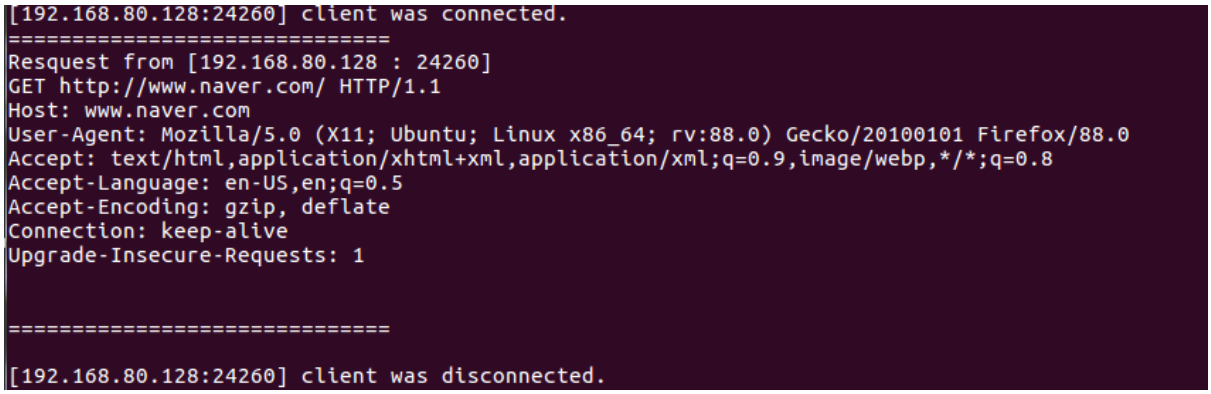
```
                    write(client_fd, response_message,
strlen(response_message));


        printf("[%s:%d] client was disconnected.\n",
                                    inet_ntoa(inet_client_address),
client_addr.sin_port);
       close(client_fd);
       exit(0);
    else if(pid > 0) { close(client_fd);
 }//endwhile
}//endmain
```

Result Screen

1. Web browser
   - 첫번째 naver을 쳤을 때 miss가 나오며 ip:port, url, kw학번을 같이 출력한다.
   - 두번째 naver을 쳤을 때 hit이 나오며 ip:port, url, kw학번을 같이 출력한다.

2. Terminal

- http request을 terminal 창에 출력한다.



Consideration

1. read(client_fd, buf, BUFFSIZE)

지난 #2-1 과제와 달라서 헷갈렸던 부분은 자식 프로세스구역의 read(client_fd, buf, BUFFSIZE) 이다.

#2-1 과제에서는 ./Client로 client에 연결되면 "bye"를 입력할 때까지 수많은 url을 입력하더라도 꺼지지 않았지만 이번 과제#2-2에서는 웹 브라우저에서 url을 입력하면 client의 요청과 server의 응답이 거의 동시에 이뤄지며 client가 종료된다.

따라서 #2-1에서는

```
if(pid==0) {
  while( read(client_fd, buf, BUFFSIZE) ) {
    if(strncmp(buf, "bye", 3) == 0) {
      break;
```

```
    }
  …
}
```

위에 코드처럼 read를 while문 안에 넣어서 if문이 참이 되기 전까지 read를 받는다.

하지만 #2-2 에서는

```
if(pid == 0) {
  read(client_fd, buf, BUFFSIZE);
  …
}
```

위에 코드처럼 read를 #2-1과는 다르게 한 번만 수행하면 된다.

2. setsockopt()

이 함수는 bind 에러를 방지하는 함수이다. bind() error란 프로그램을 종료 후에 다시 실행하면 bind()에서 에러가 생기는 현상으로 TIME_WAIT state가 되는 것과 같다.

따라서 setsockopt()를 이용해 bind()에 의해 생기는 TIME_WAIT 현상을 막을 수 있다.

```
int opt = 1;

setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

Conclusion

Web browser와 server가 HTTP 요청과 응답을 주고받는 메커니즘에 대해 배웠다. HTTP란 표준 웹 전송 프로토콜이며 web browser가 연결을 요청하면 server는 요청을 수락한다. url 검색창에 url을 입력하면 http request message를 server에 전송한다. Server는 해당 url을 실행하기 위해 함수(프로그램)를 실행하여 결과를 response message형태로 client에게 전송한다.

http message에는 2가지의 종류가 있으며 request message와 response message의 차이점과 각각에 대해 이해하게 되었다.