2-1

금3,4

과목명	시스템프로그래밍
담당교수	김태석 교수님
학과	컴퓨터정보공학부
학년	3학년
학번	2019202009
이름	서여지
제출일	21.04.28 (수)

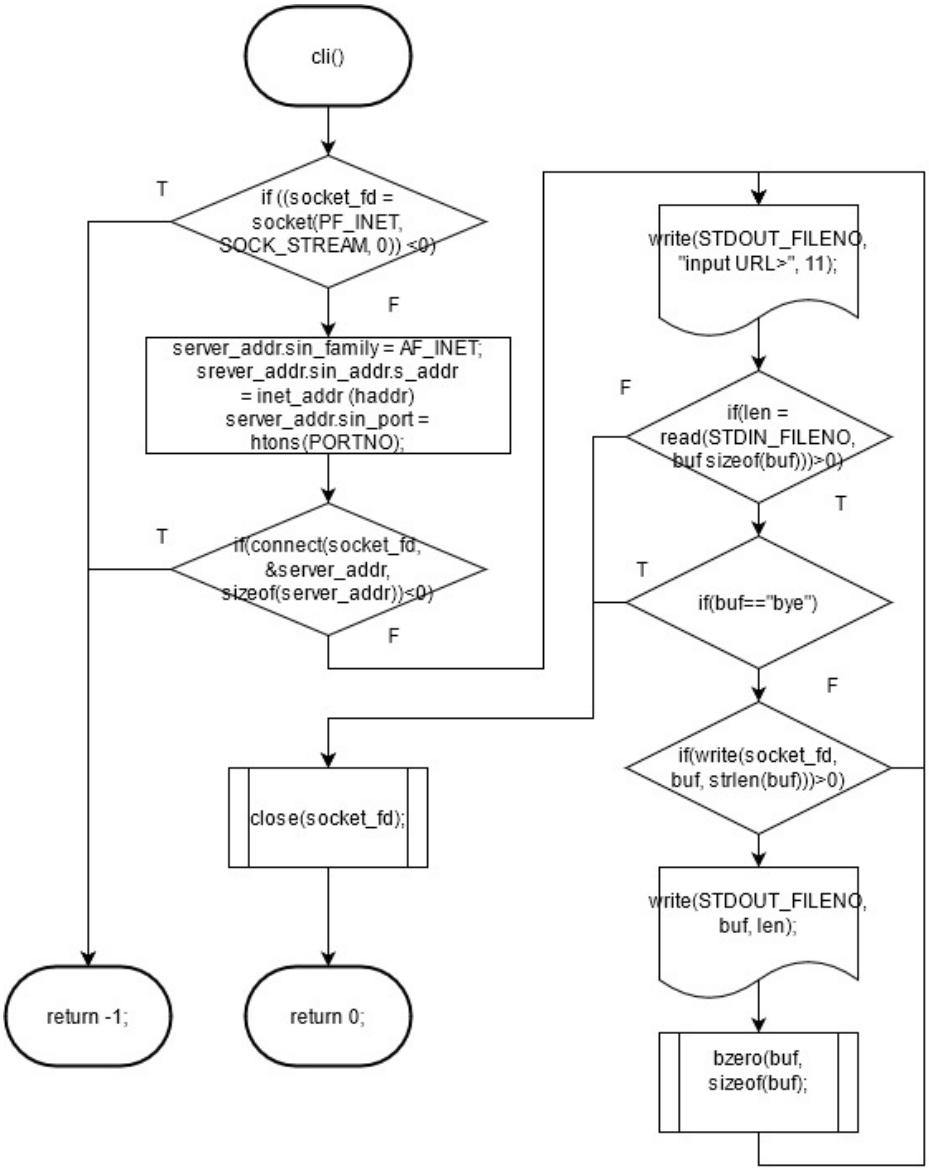
KWANGWOON UNIVERSITY

1. Introduction //과제 소개 – 5줄 내외(background 제외)

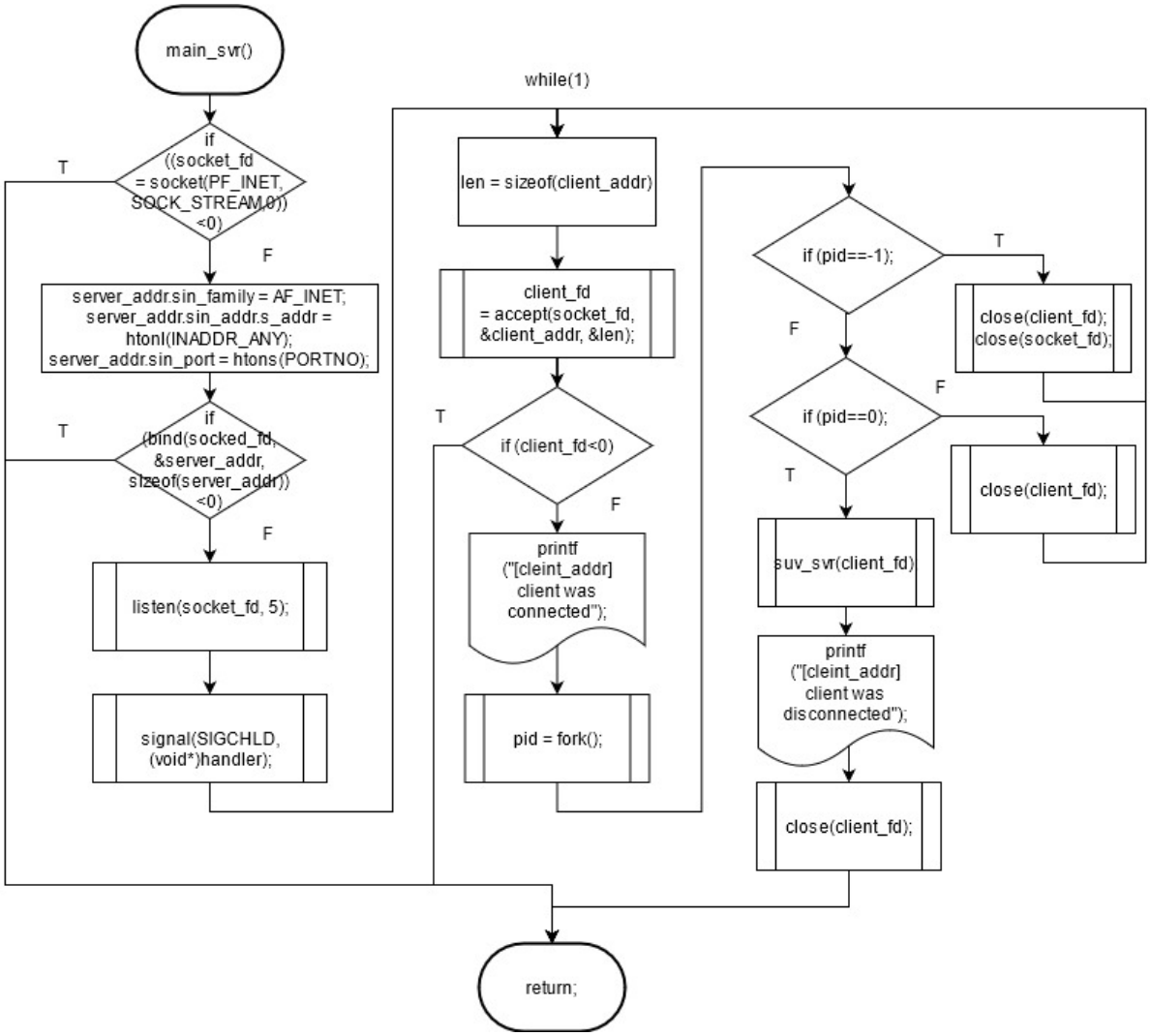
이번 과제는 지금까지 구현한 proxy_cache를 server와 client로 나누어 작성하는 것이다. 이때 server는 다수의 client를 처리할 수 있어야한다. server와 client는 socket을 이용하여 통신한다. client가 URL을 보내면 server의 sub process에서 cache를 조회한 후, 결과를 다시 client로 전송한다. client는 표준출력으로 결과를 출력하고, server sub process는 logfile에 결과를 출력한다.

2. Flow Chart //코드 작성 순서도

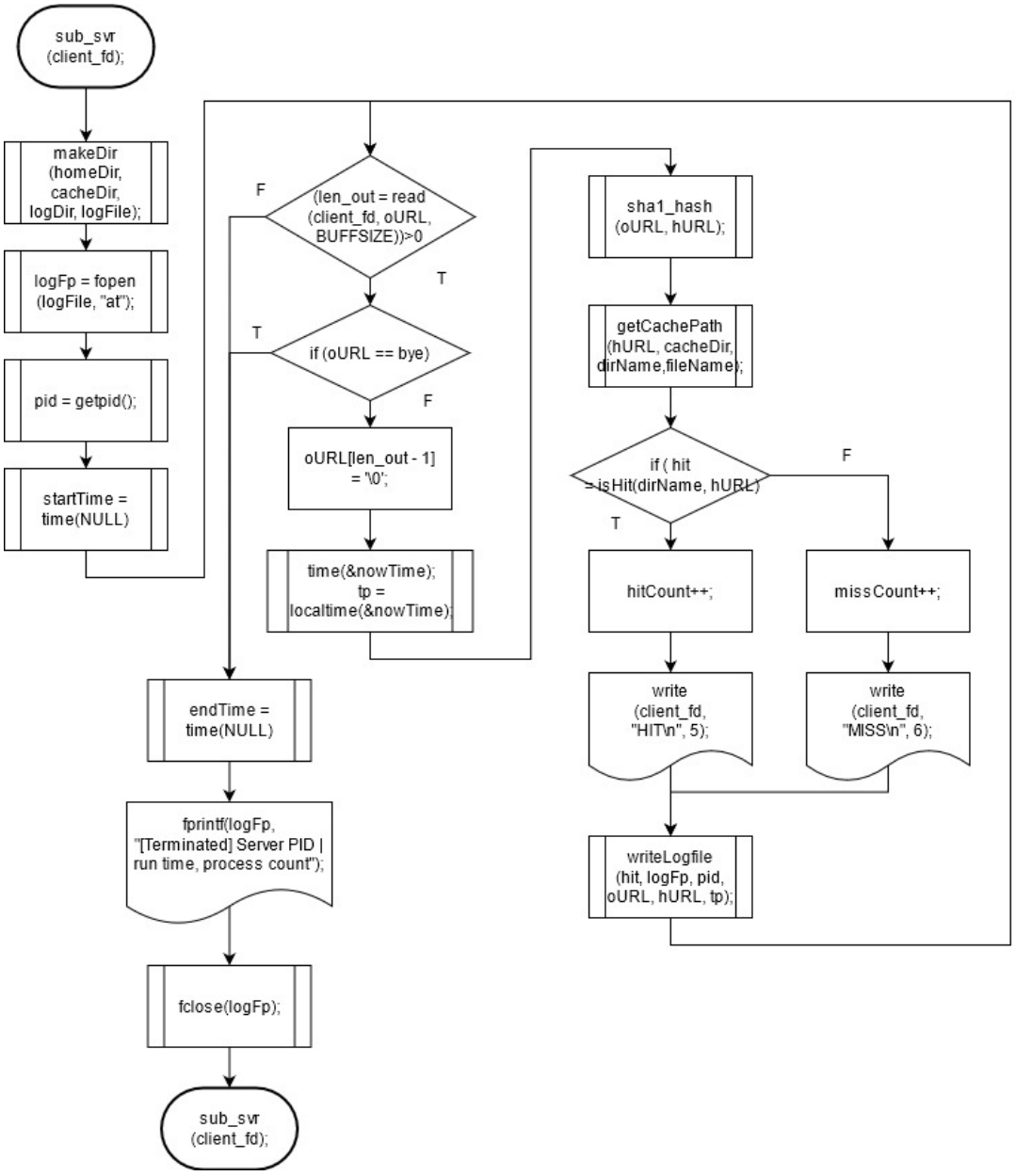
(1) cli



(2) main_svr


main_svr()
if ((socket_fd = socket(PF_INET, SOCK_STREAM,0)) <0)
T
F
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(PORTNO);
if (bind(socked_fd, &server_addr, sizeof(server_addr)) <0)
T
F
listen(socket_fd, 5);
signal(SIGCHLD, (void*)handler);
while(1)
len = sizeof(client_addr)
client_fd = accept(socket_fd, &client_addr, &len);
if (client_fd<0)
T
F
printf ("[cleint_addr] client was connected");
pid = fork();
if (pid==-1);
T
F
close(client_fd); close(socket_fd);
if (pid==0);
F
T
close(client_fd);
suv_svr(client_fd)
printf ("[cleint_addr] client was disconnected");
close(client_fd);
return;

(3) sub_svr



3. Pseudo code //알고리즘

```
int cli (void){
      socket_fd = socket(PF_INET, SOCK_STREAM, 0
      if (socket error) return -1      // can't create socket

      buf = 0; server_addr = 0;
      server_addr.sin_family = AF_INET;
      server_addr.sin_addr.s_addr = inet_addr(haddr);
      server_addr.sin_port = htons(PORTNO);

      connect(socket_fd, server_addr, sizeof(server_addr))
      if (connect error) return -1      // can't connect
```

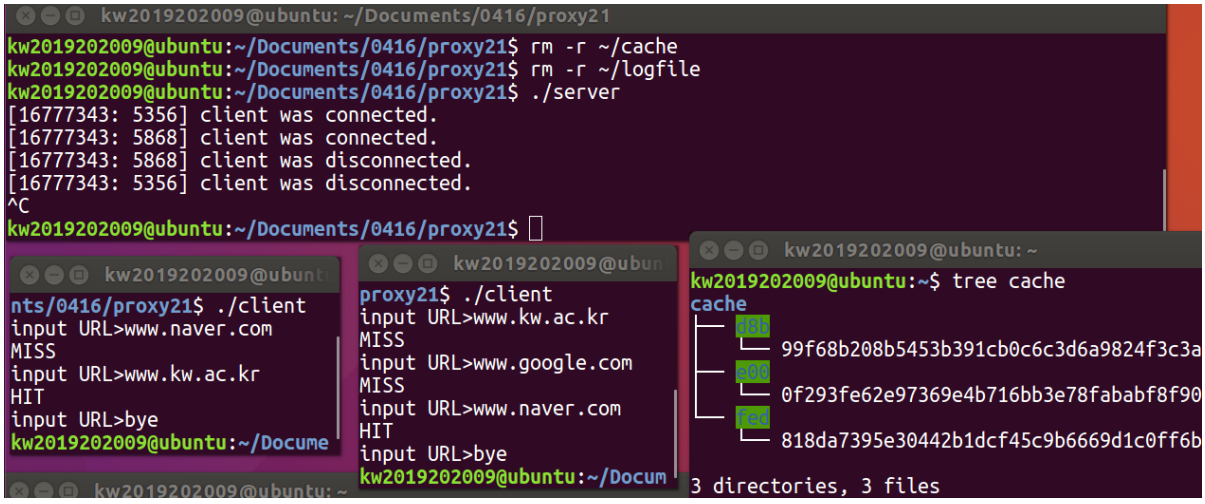
```
    write(STDOUT_FILENO, “input URL>”);
    while((len = read(STDIN_FILENO, buf)) {
        if (buf == “bye”) break;
        if(write(socket_fd, buf, strlen(buf))>0) {
            len = read(socket_fd, sizeof(buf));
            if(len >0) {
                write(STDOUT_FILENO, buf);
                buf = 0;
            }
        }
        write (STDOUT_FILENO, “input URL>”);
    }
    close(socket_fd);
    return 0;
}
```

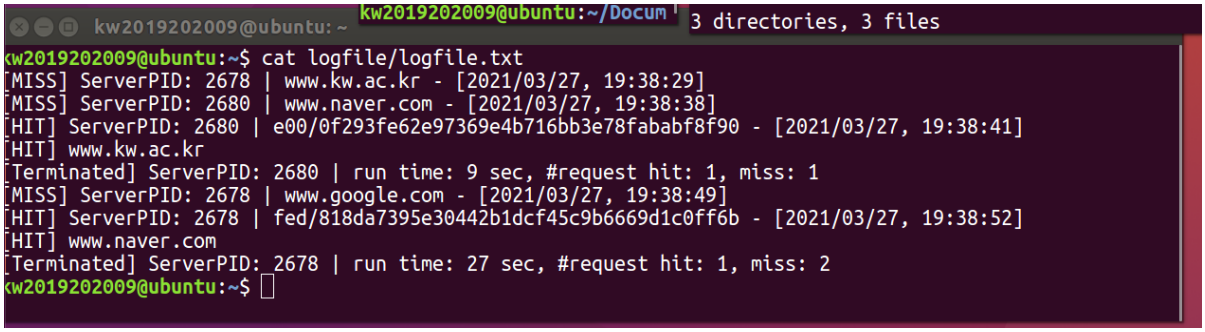
```
void main_svr(void) {
    socket(PF_INET, SOCK_STREAM);
    if (socket error ) return;      // can’t open socket

    server_addr = 0;
    server_addr.sin_family = AF_INET;
    srever_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    srever_addr.sin_port = htons(PORTNO);

    bind(socket_fd, &server_addr, sizeof(server_addr))
    if( bind error ) return;      // can’t bind address
    listen(socket_fd, 5);
    signal(SIGCHID, (void*)handler);

    while(true){
        client_addr = 0;
        len = sizeof(client_addr);
        client_fd = accept(socket_fd, &server_addr, &len)
        if(client_fd <0)      return;      // accept err
        write (“[client addr] client was connected/n”);
        pid = fork();
        if (fork err) {
            close(client_fd);
            close (socket_fd);
            continue;
        }
        if (sub process) {
            sub_svr(client_fd);
            write(“[client addr] client was disconnected₩n”);
            close(client_Fd);
            return;
        }
        close(client_fd);      //main process
    }
    close(socket_fd);
    return;
}
```

```
void sub_svr(int client_fd) {
    makeDir(honeDir, cacheDir, logDir, logfile);
    logFp = fopen(logfile, "at");
    pid = getpid();
    startTime = time (NULL);

    while((len_out = read(client_fd, oURL, BUFFSIZE))>0{
        if( oURL == "bye") break;
        oURL[-1] = NULL;
        time(&nowTime);
        tp = localtime(&nowTime);

        sha1_hash(oURL, hURL);
        getCachePath(hURL, cacheDir, dirName, filename);

        if(hit) {
            hitCount++;
            write(client_fd, "HIT₩n", 5);
        }
        else if(miss) {
            missCount++;
            create(filename, 0644);
            write(client_fd, "MISS₩n",6);
        }
        writeLogfile(hit, logFp, pid, oURL, hURL, tp);
    }
    endTime = time(NULL);
    fprintf(logFp, "[Terminated] ServerPID | run time, process count₩n");
    fclose(logFp);
    return;
}
```

4. 결과화면 //수행한 내용을 캡쳐 및 설명

강의자료에 제시된 예시를 이용하여 결과를 확인하였다. server와 client간 통신이 정상적으로 이루어졌고, stdout과 logFile이 적절하게 출력되었다.





5. 결론 및 고찰

예제의 signal함수의 동작을 오해해서 실습코드를 이해하는 것에 어려움을 겪었다. signal함수는 호출되었을 때 해당 시그널을 대기하는 것이 아니라 이후 시그널이 들어왔을 때의 동작을 미리 지정해놓는 역할을 한다. 따라서 실습코드에서는 하나의 자식프로세스가 종료된 뒤 handler 함수가 호출되어 종료된 자식 프로세스의 state를 전달받는 동작을 수행한다.

처음 코드를 작성한 후 실행했을 때 client에서 버퍼에 입력한 내용에 '₩n'이 포함되어 문제가 발생했다. server sub process에서 버퍼를 읽은 뒤 마지막 자리에 NULL문자를 추가하는 것으로 문제를 해결했다.

server sub process에서 logfile을 작성할 때 fprintf함수를 호출한 순서대로 기록되는 것이 아니라 먼저 종료된 client의 내용이 우선 출력되는 문제를 겪었다. 버퍼의 내용을 프로세스 종료시점에 한 번에 기록하기 때문에 문제가 발생한 것으로 추측되었다. 버퍼의 내용을 비우는 fflush함수를 추가하여 문제를 해결하였다.

6. reference //과제를 수행하면서 참고한 내용을 구체적으로 기록

signal, c++ reference, https://www.cplusplus.com/reference/csignal/signal/