Assignment#4-3

시스템 프로그래밍 실습

제출일: 6월 14일 금요일

분 반: 화요일

담당 교수: 신영주

학 번: 2015722025

학 과: 컴퓨터정보공학부

이 름: 정용훈

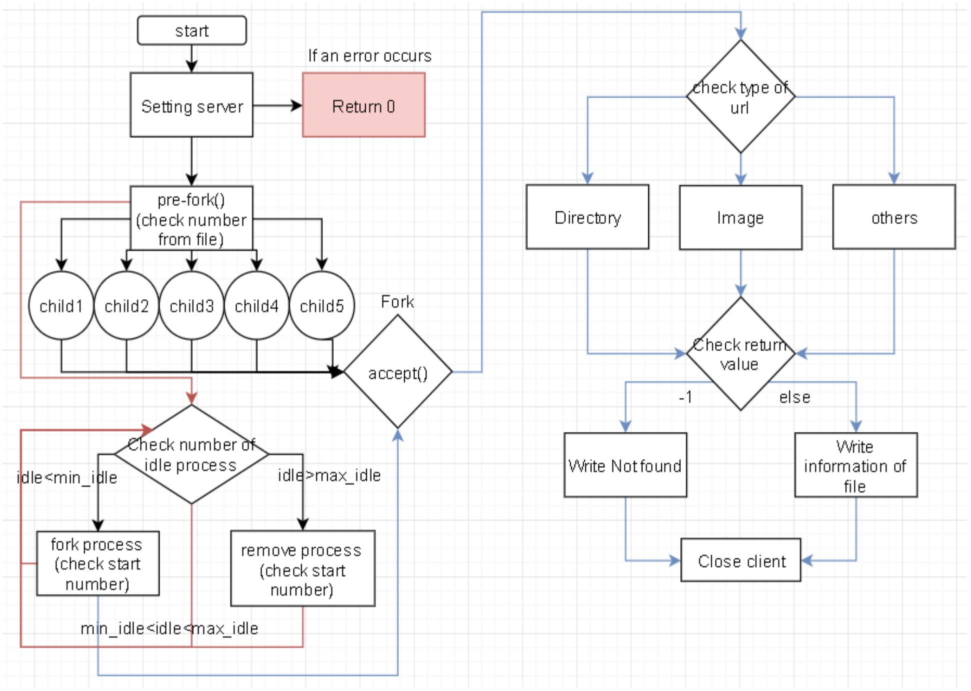
1. Introduction

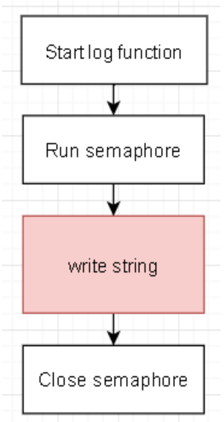
4-3과제는 지금까지 진행한 과제를 기반으로 log file을 semaphore를 통하여 작성하는 과제다. Semaphore는 process의 접근을 제어하는 기술로, log를 작성하는 함수를 process들이 공유하면서 사용하기에 semaphore를 사용하여 log에 동시 접근하는 것을 막아준다.

2. Flow Chart

이번 4-3과제의 flow chart는 변하지 않는다. 추가되는 함수는 터미널에 출력되는 string을 받아와 log file에 써주는 동작을 하는 thread함수이며, 각 thread가 사용되는 부분은 기존에 printf를 통하여 상태를 출력하는 곳 마다 함수를 call하게 된다.

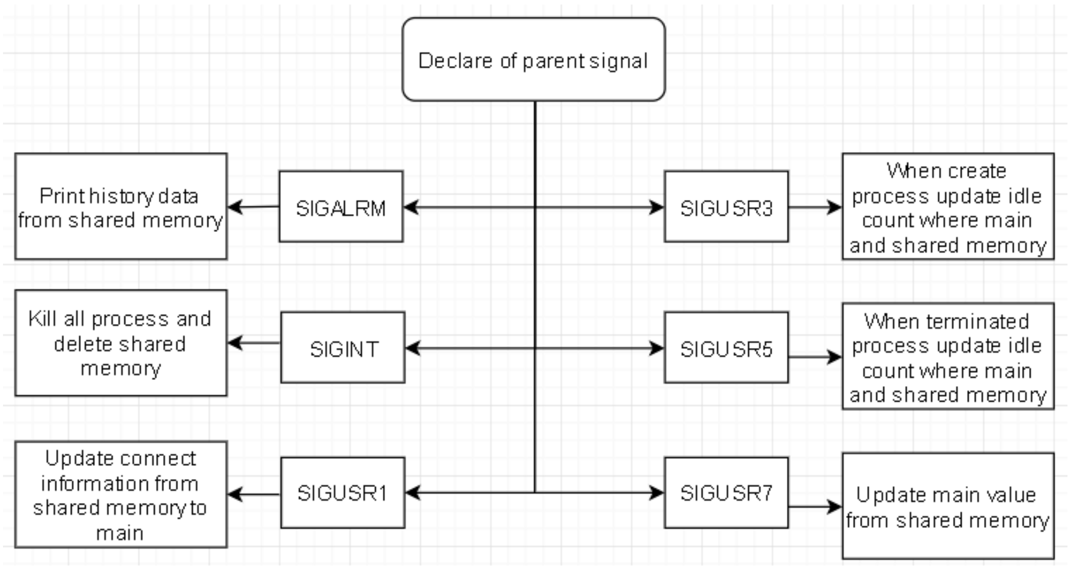
Main





다음은 메인 함수와 이번에 추가된 log file에 쓰기 위한 함수의 flowchart를 나타낸 것이다. 각 print해주는 부분마다 다음과 같이 semaphore를 사용하여 접근을 제어해주고 log file을 사용하며, 기록을 저장하는 것을 확인할 수 있다. 함수자체는 굉장히 간단하고 어렵지 않다. 사용된 나머지 함수들은 전 과제와 동일하다.

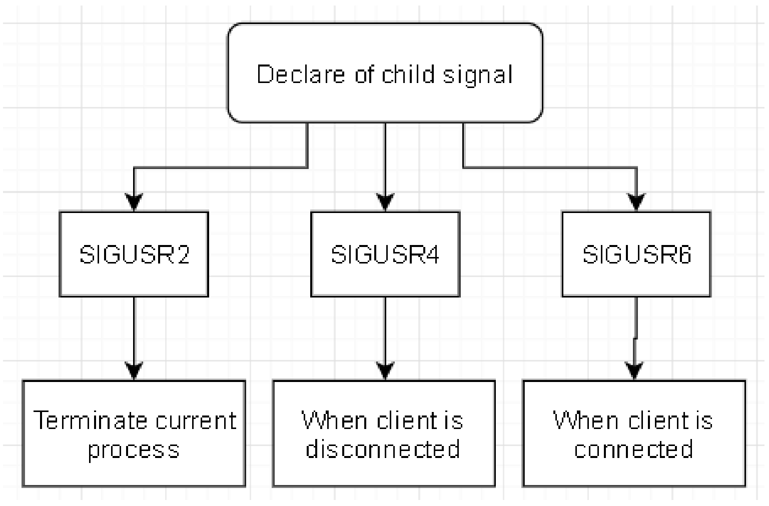
SignalHandler_parent



Declare of parent signal
SIGALRM
Print history data from shared memory
SIGINT
Kill all process and delete shared memory
SIGUSR1
Update connect information from shared memory to main
SIGUSR3
When create process update idle count where main and shared memory
SIGUSR5
When terminated process update idle count where main and shared memory
SIGUSR7
Update main value from shared memory

Parent의 signal을 정의해주는 함수로써 다음과 같은 Signal들을 정의해준다. SIGUSR1, 2가 아닌 다른 것들은 Posix에서 정의한 1~30번이 아닌 사용하지 않는 signal 50번부터 정의했으며, Signal이 많은 이유는 처음부터 체계적으로 설계하지 못한 이유가 있다.

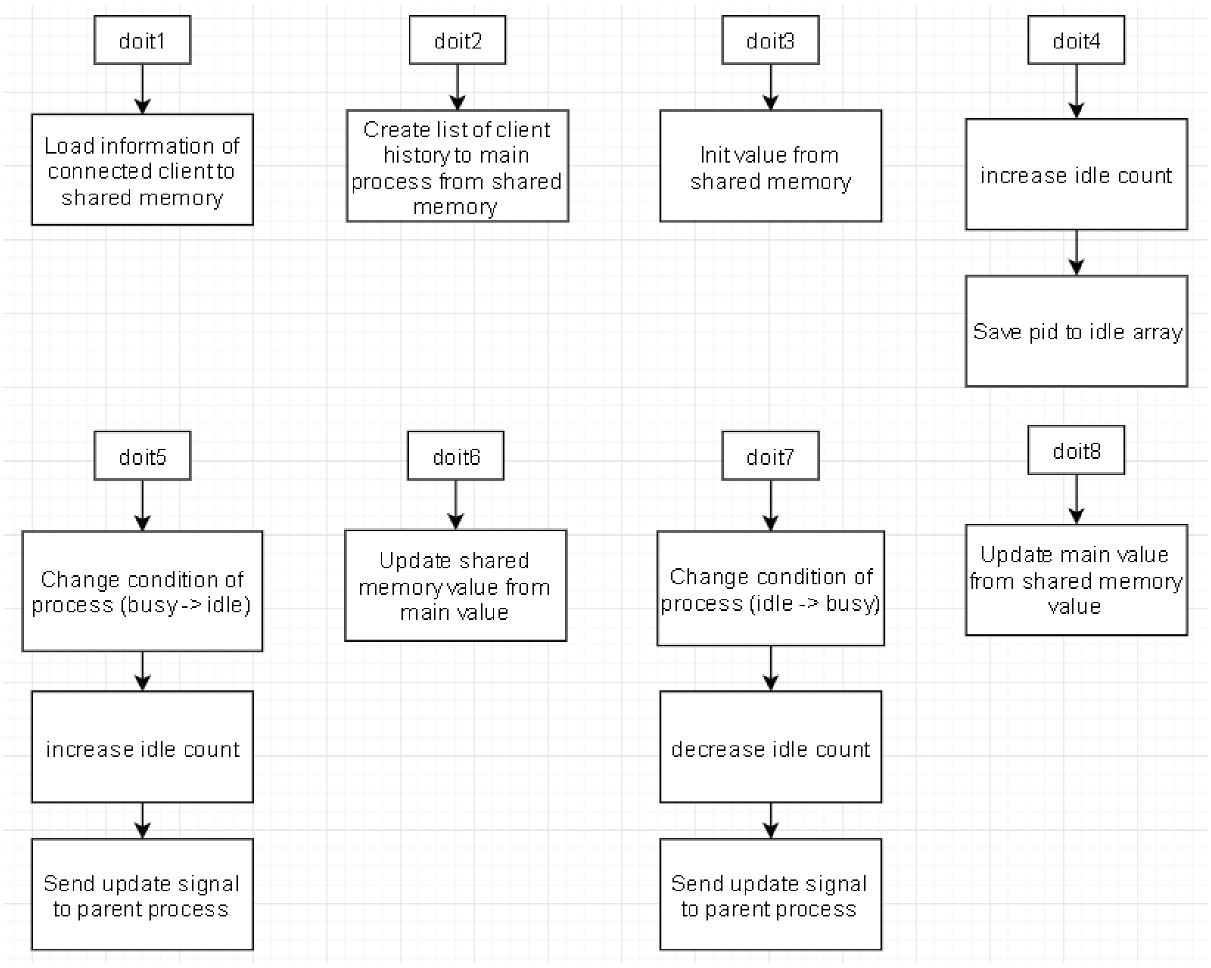
SignalHandler_child



Idle process의 수를 client가 연결되었을 때 연결이 끊어질 때 경우도 조절해주어야 하기 때문에 클라이언트가 연결되는 경우와 끊어지는 경우를 Signal 주면서 해당 Thread에서 공유메모리에 접근하여 idle수를 조절해주는 동작을 한다.

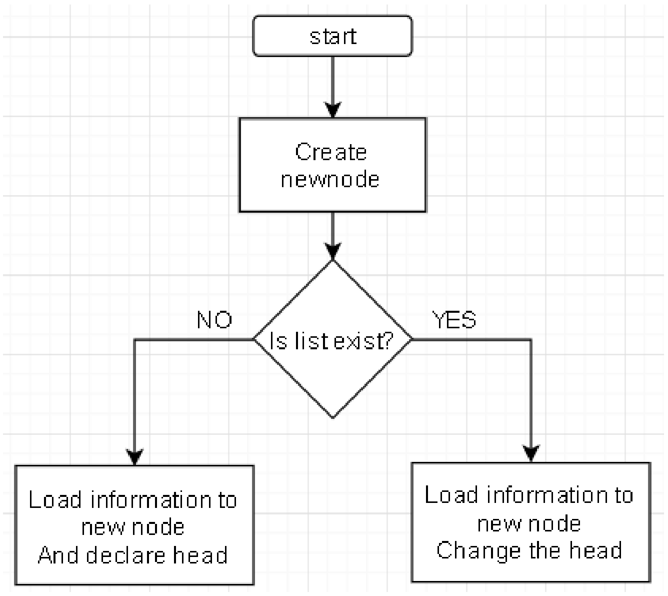
각 Thread에서 사용하는 doitN함수들

체계적으로 설계하지 못하면서 함수들이 난잡하게 여러 개가 존재합니다. 아래 사진은 각각의 thread가 사용하는 doit동작입니다.


doit1
Load information of connected client to shared memory
doit2
Create list of client history to main process from shared memory
doit3
Init value from shared memory
doit4
increase idle count
Save pid to idle array
doit5
Change condition of process (busy -> idle)
increase idle count
Send update signal to parent process
doit6
Update shared memory value from main value
doit7
Change condition of process (idle -> busy)
decrease idle count
Send update signal to parent process
doit8
Update main value from shared memory value

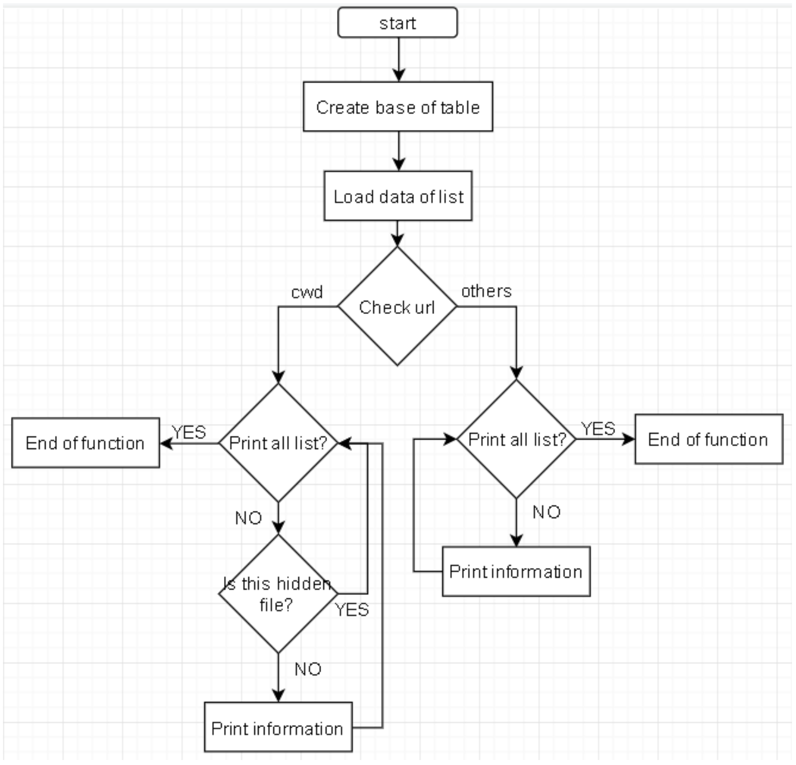
각각 함수의 종류는 다양하지만 동작 자체는 굉장히 간단하다. Doit1함수는 client가 연결되는 순간 history에 정보를 저장하기 위한 동작이고, doit2는 저장된 정보를 기반으로 main함수에서 list를 생성한다. (생성될 때는 최근 연결된 것부터 앞에 생성되므로, 시간기준 내림차순으로 생성된다.) doit3는 공유메모리의 값을 따로 초기화 시켜주기 위한 thread로 변수들의 값을 모두 0으로 만들어준다. Doit4는 fork되어 process가 생성 되는 경우 제어해주는 역할을 한다. Doit5와 doit7은 각 client들이 연결되고 끊어질 때 array와 idle count를 관리해주는 thread이며, 나머지 thread함수들은 보통 shared memory와 main의 변수를 update해주는 working을 담당하고 있다.

UpdateInfo


start
Create newnode
Is list exist?
NO
YES
Load information to new node And declare head
Load information to new node Change the head

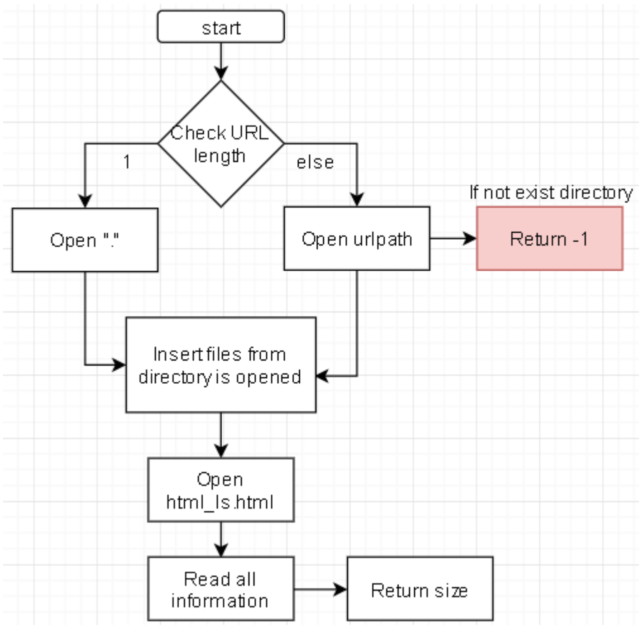
공유메모리의 history뿐 아니라 main process에서도 list를 관리해주기 때문에 update하는 함수가 필요하다.

PrintHTML(create file of html_ls.html)



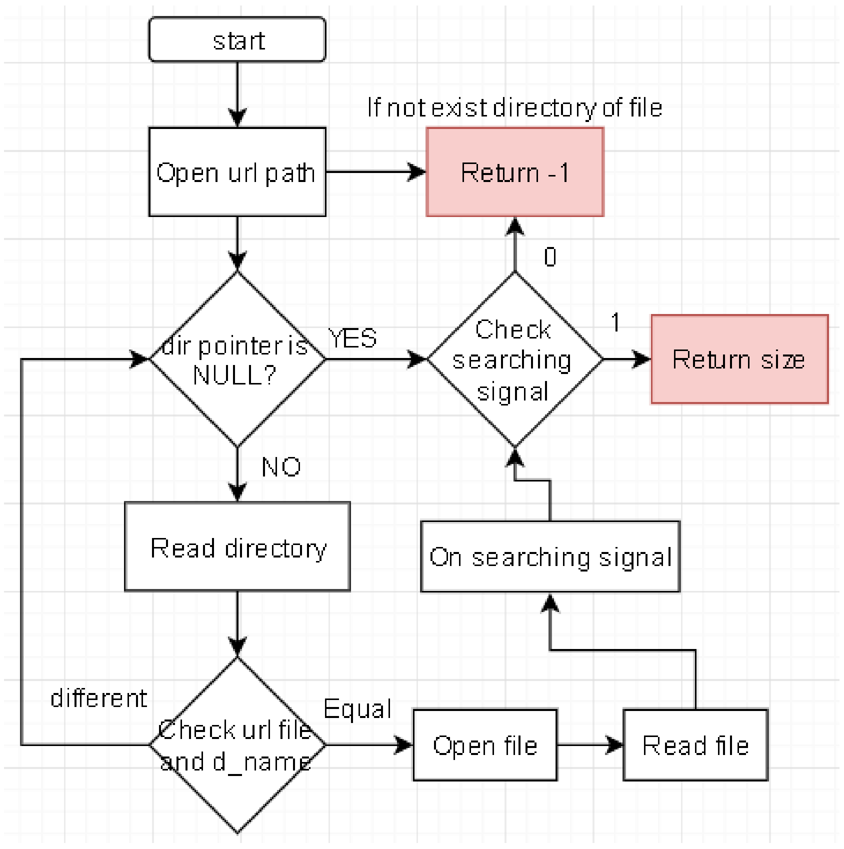
3-1 과제인 html_ls.hml을 만드는 과정으로 옵션 구현이 –a과 –al만 있기 때문에 구조만 동작 자체는 똑같지만 구조를 변경하게 되었다. 해당 함수는 Html함수에서 call하는 함수로써 아래 Html함수를 참고하면 이해하기 쉽다.

Html



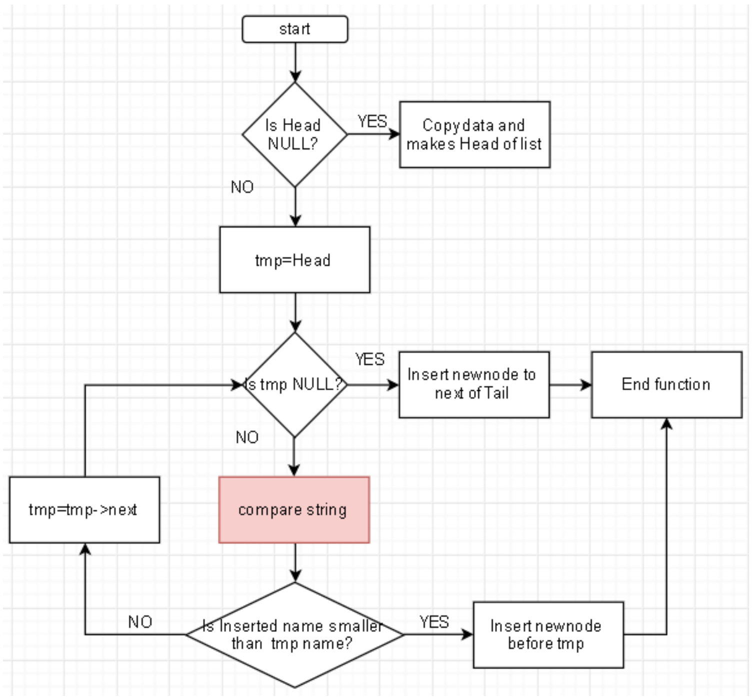
Open html을 하기 전 과정이 printHTML로써 html파일을 생성한 후 실행하게 된다.

Image & Normal (others)



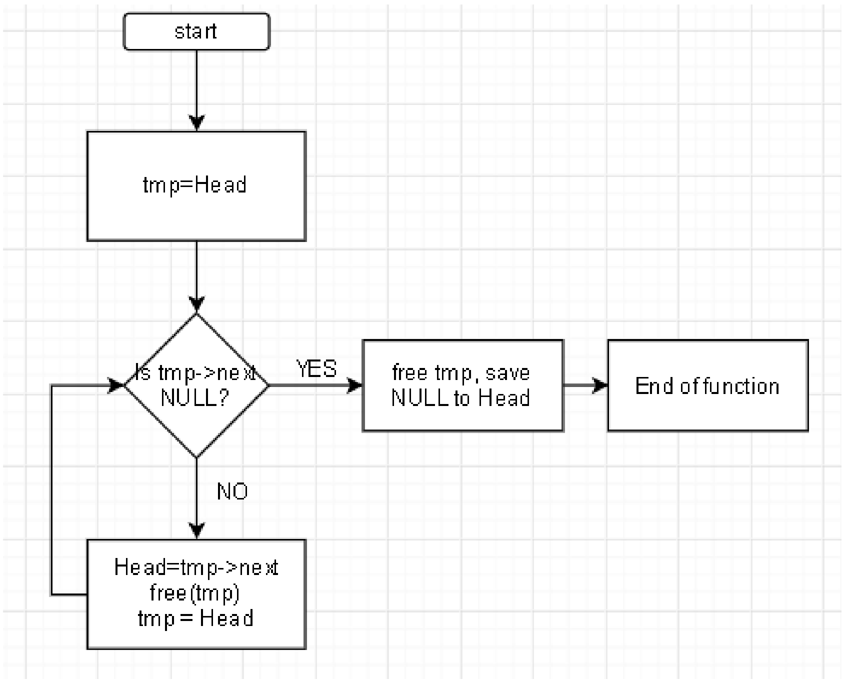
다음 함수는 Image와 나머지 다른 파일들을 처리해주는 함수로써 정보를 binary로 읽어와 write해주는 작업을 하게 된다 사실 image와 나머지 파일은 모두 binary를 통하여 읽을 수 있기 때문에 나눠줄 필요는 없다고 생각된다.

Insertnode


start
Is Head NULL?
YES
Copy data and makes Head of list
NO
tmp=Head
Is tmp NULL?
YES
Insert newnode to next of Tail
End function
NO
compare string
tmp=tmp->next
NO
Is Inserted name smaller than tmp name?
YES
Insert newnode before tmp

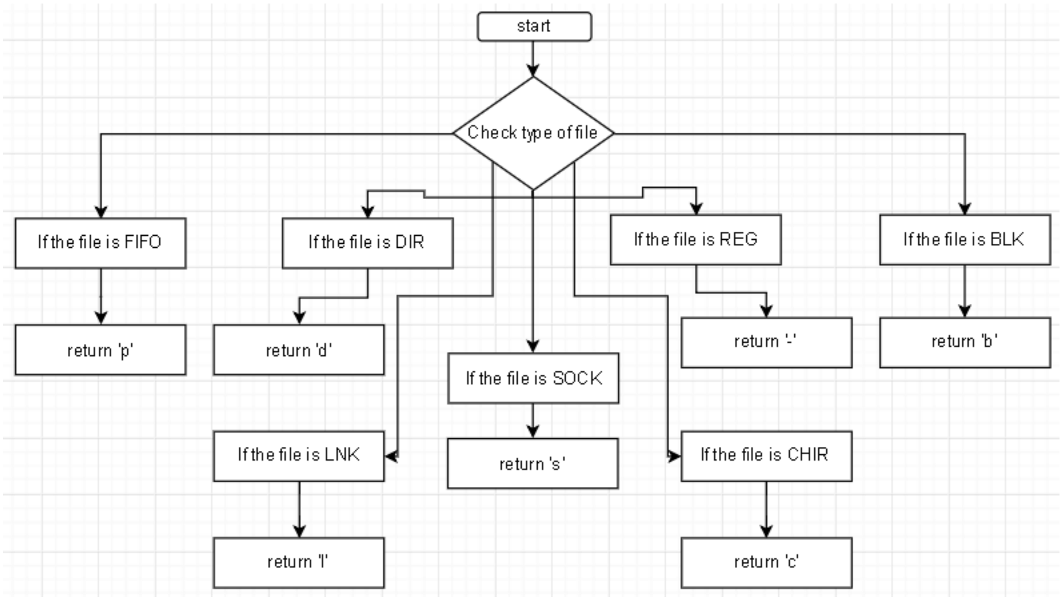
따로 sort함수를 사용하지 않고 insert를 실행 할 때부터 sort가 되며, list가 생성된다. Insert함수는 기존 함수와 동일하게 사용되므로 전체적인 변화는 없지만, S옵션을 사용면 size를 비교해야 하므로 compare를 하는 부분이 변경되게 된다. 해당 문제는 함수는 같고 조건만 바꿔주어 해결하였다.

Deletelist



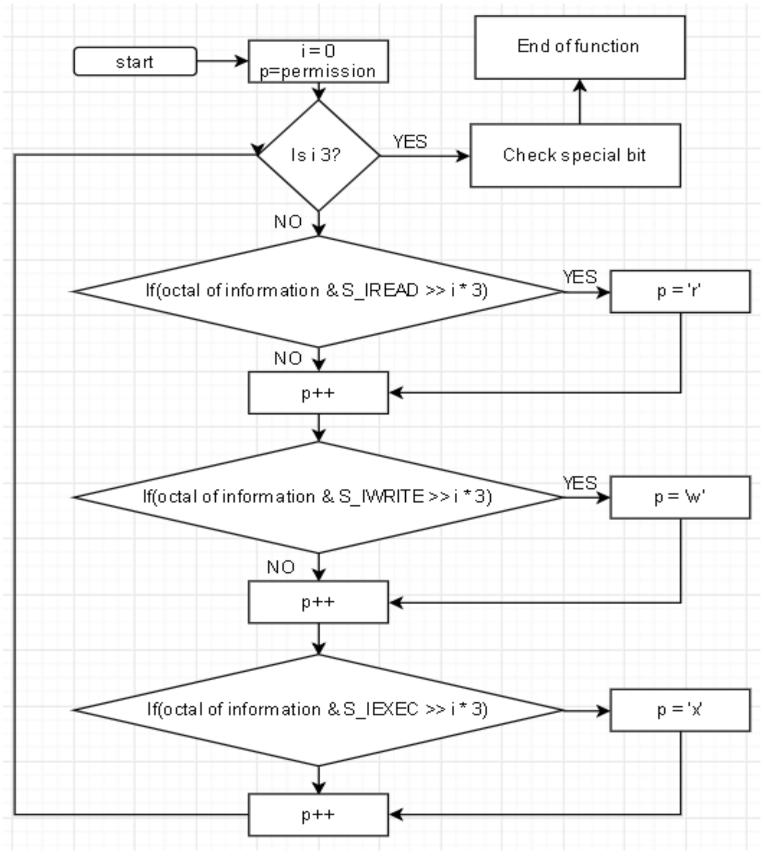
모든 정보 출력 후 linked list를 제거하는 함수다.

printType



파일의 정보를 받아와 st_mode를 통하여 파일의 type을 정의하는 함수다

printPerm


start
i = 0
p=permission
Is i 3?
YES
Check special bit
End of function
NO
If(octal of information & S_IREAD >> i * 3)
YES
p = 'r'
NO
p++
If(octal of information & S_IWRITE >> i * 3)
YES
p = 'w'
NO
p++
If(octal of information & S_IEXEC >> i * 3)
p = 'x'
p++

St_mode를 받아와 해당 파일의 permission을 확인하여, 최종적인 permission을 출력할 수 있도록 도와주는 함수다.

3. Pseudo code

Main

```
int main(int argc, char** argv)
{
            Setting signal;
            Declare value for using main function;

            ///////////////////////////Server value/////////////////////////
            struct sockaddr_in server_addr, client_addr;
            int socket_fd, client_fd;
            int len, len_out;
            int opt = 1;
            ///////////////////////////////////////////////////////////////

            load current working directory;

            ///////////////////////////For Connecting Server/////////////////////////
            setting socket;
            setting socket opt;

            memset(&server_addr, 0, sizeof(server_addr));
            server_addr.sin_family = AF_INET;
            server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
            server_addr.sin_port = htons(PORTNO);

            bind;

            listen(socket_fd, 5);
```

```
///////////////////////////Make child process/////////////////////////////////////
          for(int i=0;i<StartProcess;i++)
          {
                    if((PID=fork())>0) //parent process
                    {
                              print information start condition;
                              continue;
                    }

                    else if(PID==0) //child process
                    {
                              Declare child signal;
                              break;
                    }

                    else //error
                    {
                              Error;
                              return;
                    }
          }

         if(PID>0) //parent
         {
                   alarm(10); //alarm
                   while(1)
                   {
                            pause(); //wait parent process
                            if(Idle process smaller thaan min_idle)
                            {
                                     while(idle_porcess != start_idle_number)
                                     {
                                              if((PID=fork())>0)
                                              {
                                                       print information;
                                              }

                                              else if(PID==0)
                                              {
                                                       Declare child signal;
                                                       go to child main;
                                              }

                                              else //error
                                              {
                                                       Error;
                                                       return;
                                              }
                                     }

                            }
                    else if(idle_number > max_idle_number)
                    {
                            while(idle_number> max_process_number)
                            {
                                    delete idle process;
                                    print_current_condition;
                            }
                    }
                    else
                            continue;
            }

        }
        go to child main;
        close(socket_fd); //close socket
        return 0;
}
```

———————————————아래로 child process——————————————

```
while (1) //start server
{
        Setting variable;

        len = sizeof(client_addr);
        client_fd = accept;

        Check accessible user;
        load url;

        continue parent process;
        child process practice below code;
```

```
/////////////////////////////Declare head/////////////////////////////////////////////////
if(The signal is image file)
{
        long unsigned int filesize=0; //file size
        unsigned char image_message[3000000]={0, }; //message buffer
        filesize=Image(urlname,client_fd,image_message); //return size of file

        if(filesize==-1) //check not found
                send Not found response

        else
                send information of file

        write(client_fd, response_header, strlen(response_header)); //send header
        write(client_fd, image_message, filesize); //send entity
}
```

```
        else if(The signal is directory)
        {           long unsigned int filesize=0; //file size
                    unsigned char html_message[3000000]={0, }; //message buffer
                    filesize=Html(url,client_fd,html_message); //return size of file

                    if(filesize==-1) //check not found
                                send Not found response

                    else
                                send information of file

                    write(client_fd, response_header, strlen(response_header)); //send header
                    write(client_fd, html_message, filesize); //send entity
        }

        else //type of others
        {
                    long unsigned int filesize=0; //file size
                    unsigned char normal_message[3000000]={0, }; //message buffer
                    filesize=Normal(urlname,client_fd,normal_message); //return size of file

                                if(filesize==-1) //check not found
                                send Not found response

                    else
                                send information of file

                    write(client_fd, response_header, strlen(response_header)); //send header
                    write(client_fd, normal_message, filesize); //send entity

        }
                       close(client_fd); //close client
                       continue;

    }
    close(socket_fd); //close socket
    return 0;
}
```

각종 therad에 사용되는 doit 함수

```
void *doit1(void *vptr) //for sned history information to shared memory
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        update client information;

        unlock

        return NULL;
}
```

```
void *doit2(void *vptr)
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        shm_info=(Sh*)shm_addr;
        update main value form shared memory;

        unlock;
        return NULL;
}
```

```
void *doit3(void *vptr) //Init thread
{
        Get shared memory;
        Apply shared memory to process;

        Declare 0 to all shared memory value;

        return NULL;
}
```

```
void *doit4(void *vptr) //Just create process
{
        Get shared memory;
        Apply shared memory to process;

        lock

        load address of shared memory;
        printf("[%s] IdleProcessCount : %d₩n",time_buf,++shm_info->idle);
        ++shm_info->process;

        Save PID to main array;
        Save PID to shared memory array;

        Cur_idle=shm_info->idle;
        Cur_process=shm_info->process;

        unlock
        return NULL; //return NULL
}

void *doit5(void *vptr)
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        load address of shared memory;
        Update idle array and busy array;
        printf("[%s] IdleProcessCount : %d₩n",time_buf,++shm_info->idle); //print condition of idle
        sned SIGUSR7 signal to parent

        unlock
        return NULL; //return NULL
}

void *doit7(void *vptr)
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        load address of shared memory;
        Update idle array and busy array;
        printf("[%s] IdleProcessCount : %d₩n",time_buf,--shm_info->idle); //print condition of idle
        sned SIGUSR7 signal to parent

        unlock
        return NULL; //return NULL
}
```

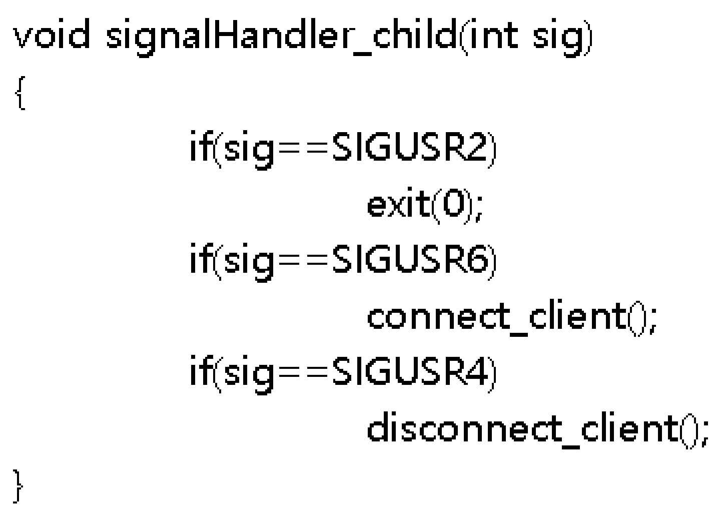
```
void *doit6(void *vptr)
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        load address of shared memory;
        printf("[%s] IdleProcessCount : %d₩n",time_buf,--shm_info->idle);
        --shm_info->process;
        Update array information from shared memory;
        Cur_idle=shm_info->idle; //update value of idle
        Cur_process=shm_info->process;

        unlock;

        return NULL;
}
```

```
void *doit8(void *vptr)
{
        Get shared memory;
        Apply shared memory to process;

        lcok;

        load address of shared memory;
        Update shared memory array information from main;
        Cur_idle=shm_info->idle; //update value of idle
        Cur_process=shm_info->process;

        unlock;
        return NULL; //return NULL
}
```

SignalHandler_parent

```
void signalHandler_parent(int sig)
{
            int status;
            if(sig == SIGALRM) //alram signal
            {
                        printHistroy();
                        alarm(10);
            }

            if(sig == SIGINT) //Check signal of ^C
            {
                        exit all process, and main process;
            }

            if(sig==SIGUSR1)
                        Update history();

            if(sig==SIGUSR3)
                        create_process_working_function();

            if(sig==SIGUSR5)
                        terminated_process_working_function();

            if(sig==SIGUSR7)
                        updateidle();
}
```

SignalHandler_child



```
void signalHandler_child(int sig)
{
            if(sig==SIGUSR2)
                        exit(0);
            if(sig==SIGUSR6)
                        connect_client();
            if(sig==SIGUSR4)
                        disconnect_client();
}
```

UpdateInfo

```
void UpdateInfo(struct sockaddr_in client_addr,int PID)
{
            create new node;

            setting time;

            if(List not exist)
            {
                        load information of client to new node;
                        Head=newnode;
            }

            else
            {
                        load information of client to new node;
                        newnode->next=Head;
                        Head-newnode;
            }
}
```

Normal & Image

```
int Normal & Image(char *urlname,int client_fd,unsigned char *normal_message)
{
            char urlpath[256]={0};
            char urlfile[256]={0};
            char cwd[256]={0};
            char Openpath[256]={0};
            char Dirpath[256]={0};
            int searching=0;

            unsigned char response_message[3000000]={0, };
            struct stat buf;
            DIR *dirp;
            struct dirent *dir;

            load current working directory;
            urlpath = urlname

            Make Open path;

////////////////////////open dir for checking image file///////////////////////////
            if(open directory==NULL)
                        return -1;

            change directory;
```

```
        do
        {
                read directory;

                If(read is NULL)
                        return -1;

                else if(urlfile is equal d_name)
                {
                        searching=1;
                        FILE *file=NULL;
                        int ch;

                        file=Open d_name file;

                        while(Repeat read file before EOF)
                                normal_message[count++]=ch;

                        close file;
                        break;
                }
        } while (1);
    change directory to home;

    if(searching==1)
        return count;

    return -1;
}
```

Html

```
int Html(char *url,int client_fd,unsigned char *html_message)
{
        DIR *dirp;
        struct dirent *dir;
        struct stat buf;
        struct group *gid;
        struct passwd *uid;
        struct tm *time;
        FILE *htmlfile;
        struct sockaddr_in server_addr, client_addr;
        struct in_addr inet_client_address;

        int total = 0,count=0;
        char response_message[3000000] = { 0, };
        char NULLpath[256]={0, };

        ////////////////////////////////////////Open DIR and wirte//////////////////////////.
        Open html file that created from printHTML function;

        if(strlen(url)==1)
        {
                Opswitch=1;
                open current directory;
                change directory;

                do
                {
                        read directory;

                        if (read is NULL)
                                break;
```

```
                                        else
                                        {
                                                    load information of file to buf;
                                                    total += buf.st_blocks / 2;
                                                    Insert node;
                                        }

                            } while (1);
            }

            else
            {
                        Opswitch=0;
                        char urlpath[256]={0,};
                        make url;

                        if(Not exist directory)
                                    return -1;
                        change working directory;

                        do
                        {
                                    read directory;

                                    if (read is NULL)
                                                break;

                          else
                          {
                                    load information of file to buf;
                                    total += buf.st_blocks / 2;
                                    Insert node;
                          }

                } while (1);
        }


        if(root path)
        {
                strcpy(response_message, "<h1>Welcome to System Programming Http</h1><br>₩n"); //copy
                fprintf(htmlfile,"<h1>Welcome to System Programming Http</h1><br>₩n"); //write to file
        }

        else
        {
                strcpy(response_message, "<h1>System Programming Http</h1><br>₩n"); //copy
                fprintf(htmlfile,"<h1>System Programming Http</h1><br>₩n"); //write to file
        }
```

```
                    practice printHTML; //create html file

                    deletelist(); //delete list
                    change working directory;
                    close html file;

                    FILE *file=NULL;
                    Open html file;

                    int ch;
                    while(Repeat read file before EOF)
                              html_message[count++]=ch;

                    close file;

                    return count;
}
```

PrintlistHTML

```
void printHTML(int client_fd, FILE* file, int flagA, int flagL, int total, char *dirpath)
{
        struct group *gid;
        struct passwd *uid;
        struct tm *time;
        char buff[256];
        char cwd[256];
        struct stat buf;
        struct dirent *dir;

        int k = 0, m = 0;
        char subpath[256] = { 0 };
        char subdirpath[256] = { 0 };
        char checkpath[256] = { 0 };
        char color[256] = { 0 };

        int createflag = 0;
        char content[BUFSIZE] = { 0, };

        load current working directory;
        For making path;

        Declare information of Directory(path, total);
        if (createflag == 0)
        {
                Declare format of table;
                createflag = 1;
        }

        for (Node* tmp=Head; tmp; tmp = tmp->next) //Repeat function for printing all list
        {
                if(Opswitch==1&&tmp->filename[0]=='.')
                        continue;
                if(!strcmp(tmp->filename,"html_ls.html"))
                        continue

                lstat(tmp->filename, &buf);
                Declare color of file;
                Make hyperlink;

                if(url's mean is current("."))
                        urlpath[strlen(urlpath)-1]='₩0';
                else
                        strcat(urlpath,tmp->filename); //make full format of url path

                uid = getpwuid(buf.st_uid);
                gid = getgrgid(buf.st_gid);

                time = load information of local time;

                if (File type is Link)
                        Make path use "->"


                Print full format;

        }

        End table and html
        return;
}
```

Nodeinsert

```
void Nodeinsert(char* name)
{
        char Ename[256];
        char Etmpname[256];
        Node* tmp = Head;
        Node* prevnode;

        if (Head of list is NULL)
        {
                Makes newnode;
                strcpy(newnode->filename, name);
                newnode->next = NULL;
                newnode->prev = NULL;

                Head = newnode;
                return;
        }

      else
      {
            while (tmp is not NULL)
            {
                  strcpy(Ename, name);
                  strcpy(Etmpname, tmp->filename);

                  if (checkstring(Ename))
                        Eliminate character of '.'
                  if (checkstring(Etmpname))
                        Eliminate character of '.'

                  if Etmpname is bigger than name)
                  {
                        if (tmp is Head)
                        {
                              Makes newnode;
                              strcpy(newnode->filename, name);
                              newnode->prev = NULL;

                              newnode->next = Head;
                              Head->prev = newnode;
                              Head = newnode;
                              return;
                        }
                        else
                        {
                                Makes newnode;
                                strcpy(newnode->filename, name);
                                newnode->next = tmp;
                                newnode->prev = tmp->prev;
                                tmp->prev->next = newnode;
                                tmp->prev = newnode;
                                return;
                        }
                }
                else continue;
        }
        Move prev to Tail;
        Makes newnode;

        strcpy(newnode->filename, name);
        newnode->next = NULL;
        prevnode->next = newnode;
        newnode->prev = prevnode;
        Tail = newnode;
        return;

    }
}
```

Function of others

```
void Eliminate(char *str, char ch)
{
        while (;befor check all character of string;str++)
        {
                if (*str == ch)
                {
                        strcpy(str, str + 1);
                        str--;
                        return;
                }
        }
}

void deletelist()
{
        Node* tmp = Head;
        for (; tmp->next != NULL;)
        {
                Head = tmp->next;
                free(tmp);
                tmp = Head;
        }
        free(tmp);
        Head = NULL;
}
```

```
char printType(mode_t mode)
{
        switch (mode & S_IFMT)
        {
        case S_IFREG:
                return('-');
        case S_IFDIR:
                return('d');
        case S_IFCHR:
                return('c');
        case S_IFBLK:
                return('b');
        case S_IFLNK:
                return('l');
        case S_IFIFO:
                return('p');
        case S_IFSOCK:
                return('s');
        }

        return('?');
}

char *printPerm(mode_t mode)
{
        int i;
        char *p;
        static char perms[10];
        p = perms;
        strcpy(perms, "---------");

        for (i = 0; i < 3; i++)
        {
                if (mode & (S_IREAD >> i * 3))
                        *p = 'r';
                p++;

                if (mode & (S_IWRITE >> i * 3))
                        *p = 'w';
                p++;

                if (mode & (S_IEXEC >> i * 3))
                        *p = 'x';
                p++;
        }
        if ((mode & S_ISUID) != 0)
                perms[2] = 's';

        if ((mode & S_ISGID) != 0)
                perms[5] = 's';

        if ((mode & S_ISVTX) != 0)
                perms[8] = 't';

        return(perms);
}
```

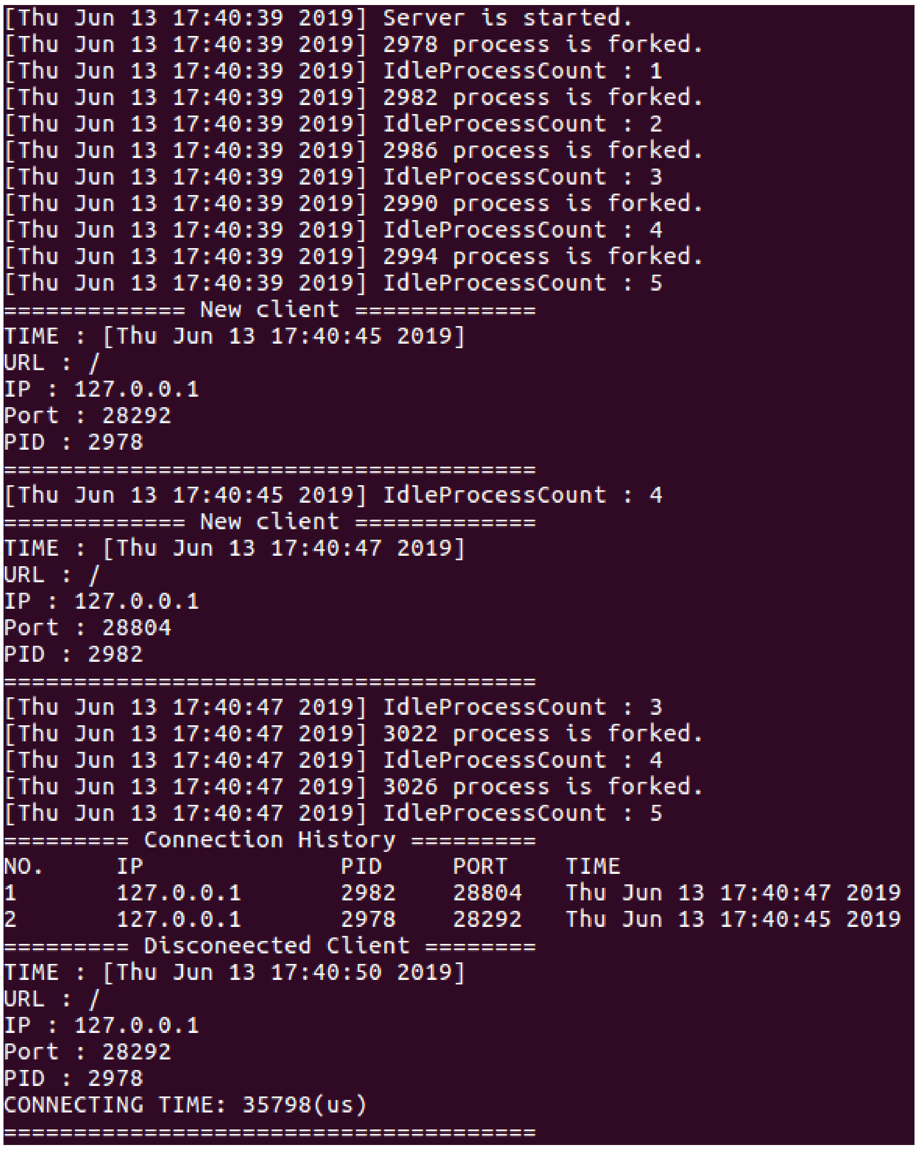
```
void printOph(long int size)
{
        int k=0,flag=0;
        double sub_size;

        sub_size=(double)size;

        for(Repeat until undivided)
        {
                if(sub_size>(double)1024)
                        sub_size/=1024;

                else
                        break;
        }

        Execute unit alignment process

        if(If the decimal point is not zero) //check first decimal point
                flag=1;

        int size_int=0;

        if(k==0)
        {

                size_int=(int)sub_size;
                printf(integer output of size_int);

        }

        else if(k==1)
        {
                if(flag==1)
                {
                        size_int=(int)sub_size;
                        printf(integer output of size_int,"K");
                }
                else
                        printf(float output of sub_size,"K");
        }

        else if(k==2)
        {
                if(flag==1)
                {
                        size_int=(int)sub_size;
                        printf(integer output of size_int,"M");
                }
                else
                        printf(float output of sub_size,"M");
        }
        else if(k==3)
        {
                if(flag==1)
                {
                        size_int=(int)sub_size;
                        printf(integer output of size_int,"G");
                }
                else
                        printf(float output of sub_size,"G");
        }

        else if(k==4)
        {
                if(flag==1)
                {
                        size_int=(int)sub_size;
                        printf(integer output of size_int,"T");
                }
                else
                        printf(float output of sub_size,"T");
        }
        return;
}
```

```
int matchfunction(char (*paname)[256], int argc)
{
        char Matchcmd[256][256]={0};
        int matchcount=0;

//////////////////////////////////////////////////////searching match command//////////////////////////////
        while(read all command)
        {
                while(read all character of one string)
                {
                        if(string has '*' or '?') //if the command has '*' or '?'
                        {
                                copy string to command vector
                                matchcount++; //increase
                                break;
                        }
                        else if(string has '[x-y]' format) //if the command has 'index'
                        {
                                copy string to command vector
                                matchcount++; //increase
                                break;
                        }
                        else if(string has '[x]' format)

                        {
                                copy string to command vector
                                matchcount++; //increase
                                break;
                        }
                }
        }
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////delete match cmd///////////////////////////////////////////////////////

        Remove extracted commands from existing vectors

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

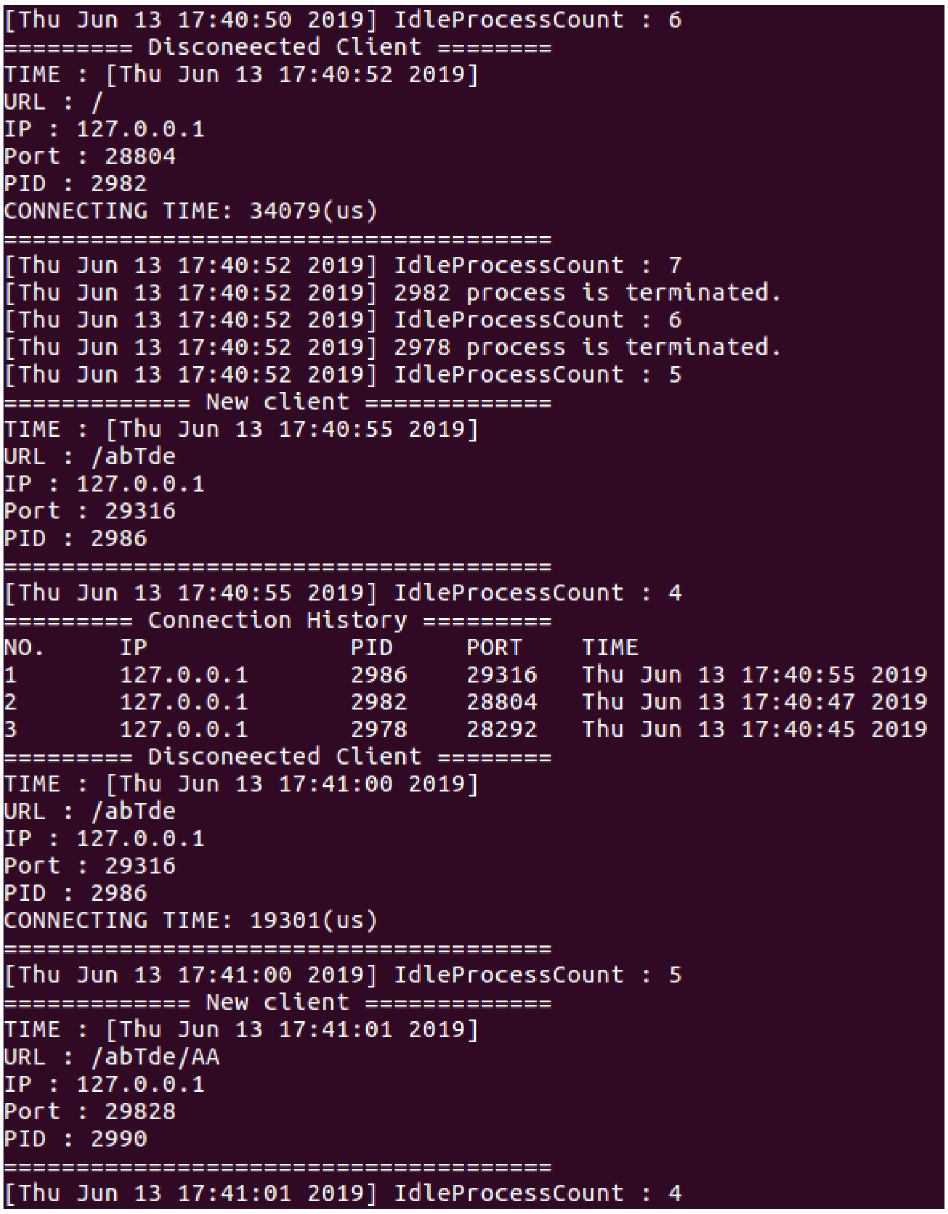
```
    int newargc=argc; //integer
    int flag=0; //integer
    struct dirent *dir;
    struct stat buf;
    DIR *dirp;
    char pathname[256][256]={0};
    char cmd[256][256]={0};

    while(checking all string) //for check match command
    {
            To separate into path and command parts
    }


    while(Repeat by match count) //for check command
    {
            int currentflag=0; //declare flag

            if(string length is 0) //check path name
                    dirp=opendir(".");
            else
            {
                    currentflag=1; //on flag
                    dirp=opendir(pathname[j]); //open pathname directory
            }
            change directory;

        if(In case there is no matching command) //Check null
        {
                Save back to original vector;
                newargc++;
                continue;
        }
        do
        {

                dir = readdir(dirp); //read file
                if (dir == NULL) //check NULL
                        break; //stop repeat functsion

                else
                {
                        if(!fnmatch(pattern[i],file name,0)&&(dir->d_name[0]!='.')) //function of match
                        {
                                Perform a function that makes it the original path;

                                Save the filename that matches the original vector;
                                newargc++;
                                flag=1; //on flag
                        }
                }

        } while (1);


        if(flag==0) //if flag is 0
        {
                Save back to original vector;
                newargc++;
        }
        flag=0; //off flag
        change pointer dirp to start
  }
  return newargc; //return new vector count

}
```

4. Result

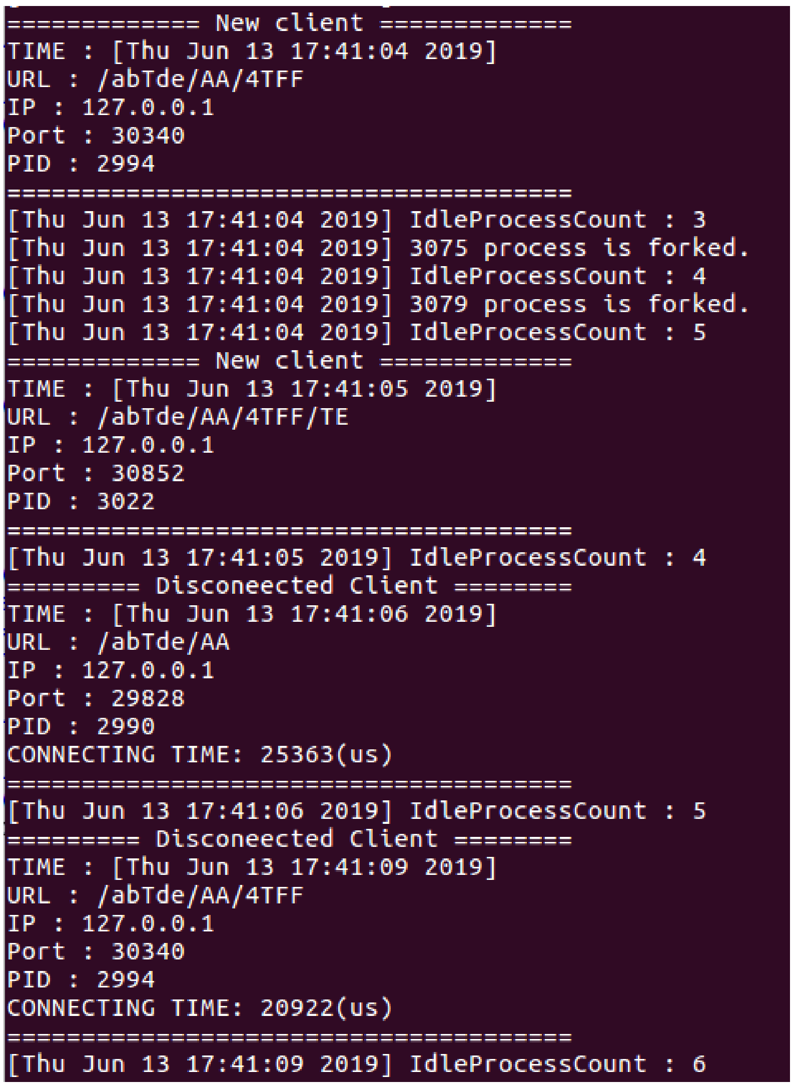
이번 과제의 결과는 Terminal에 출력되는 format과 log.txt에 출력되는 것을 확인하면 된다. 비교 결과화면은 다음과 같다.



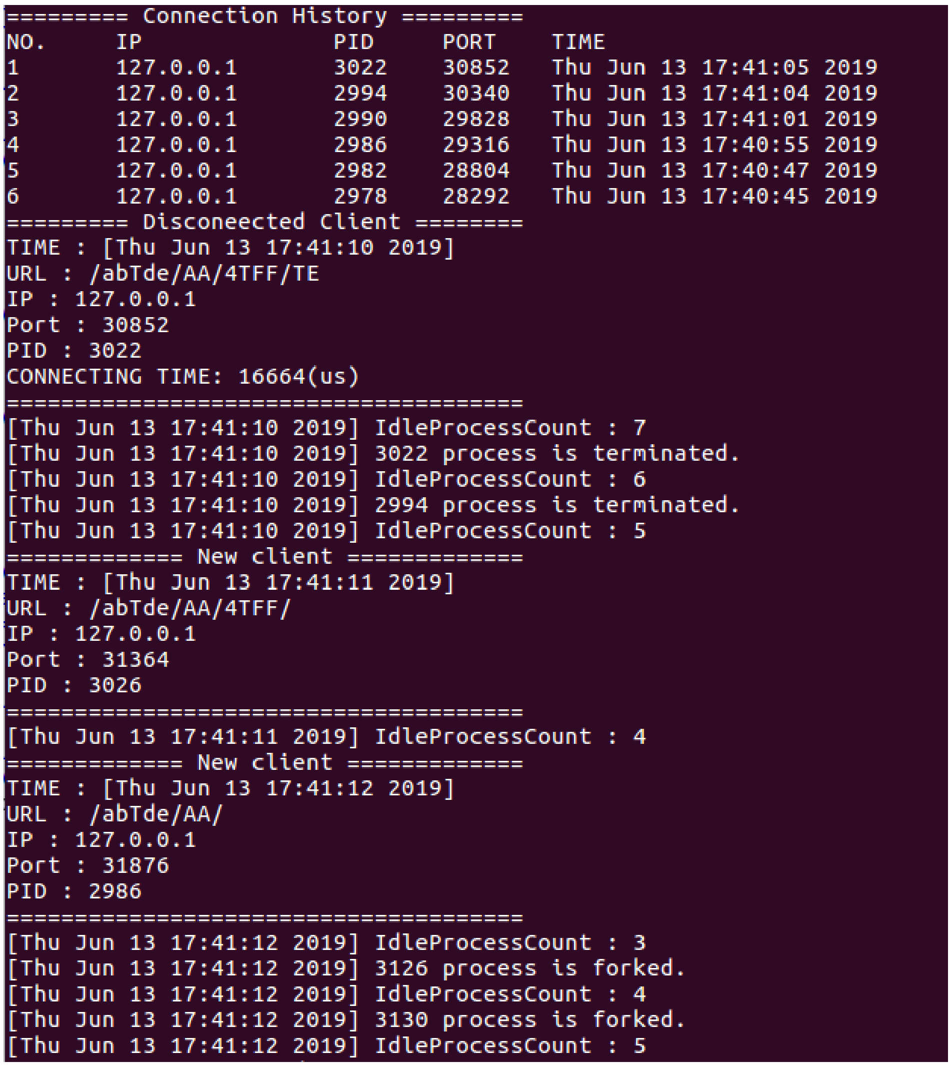
```
[Thu Jun 13 17:40:39 2019] Server is started.
[Thu Jun 13 17:40:39 2019] 2978 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 1
[Thu Jun 13 17:40:39 2019] 2982 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 2
[Thu Jun 13 17:40:39 2019] 2986 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 3
[Thu Jun 13 17:40:39 2019] 2990 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 4
[Thu Jun 13 17:40:39 2019] 2994 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:40:45 2019]
URL : /
IP : 127.0.0.1
Port : 28292
PID : 2978
======================================
[Thu Jun 13 17:40:45 2019] IdleProcessCount : 4
============= New client =============
TIME : [Thu Jun 13 17:40:47 2019]
URL : /
IP : 127.0.0.1
Port : 28804
PID : 2982
======================================
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 3
[Thu Jun 13 17:40:47 2019] 3022 process is forked.
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 4
[Thu Jun 13 17:40:47 2019] 3026 process is forked.
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 5
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
2       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:40:50 2019]
URL : /
IP : 127.0.0.1
Port : 28292
PID : 2978
CONNECTING TIME: 35798(us)
======================================
```



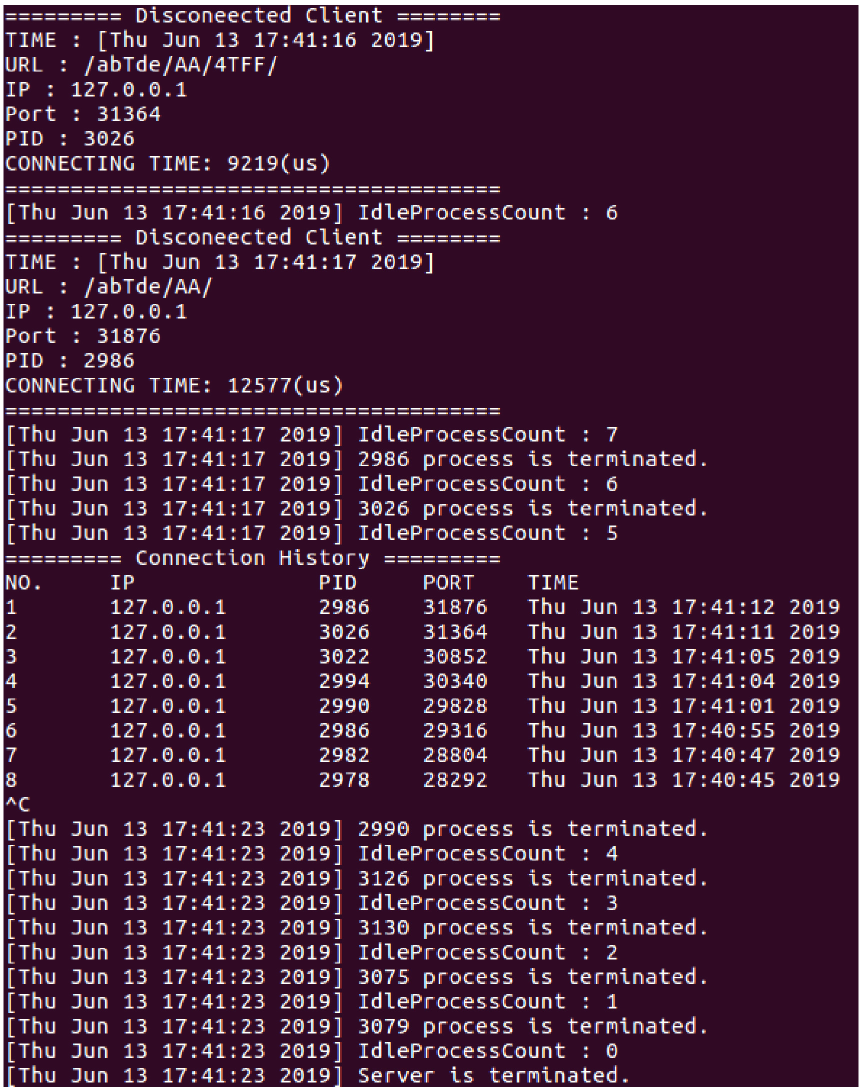
```
[Thu Jun 13 17:40:50 2019] IdleProcessCount : 6
========= Disconeected Client ========
TIME : [Thu Jun 13 17:40:52 2019]
URL : /
IP : 127.0.0.1
Port : 28804
PID : 2982
CONNECTING TIME: 34079(us)
======================================
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 7
[Thu Jun 13 17:40:52 2019] 2982 process is terminated.
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 6
[Thu Jun 13 17:40:52 2019] 2978 process is terminated.
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 5
============= New client ============
TIME : [Thu Jun 13 17:40:55 2019]
URL : /abTde
IP : 127.0.0.1
Port : 29316
PID : 2986
======================================
[Thu Jun 13 17:40:55 2019] IdleProcessCount : 4
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
2       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
3       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:00 2019]
URL : /abTde
IP : 127.0.0.1
Port : 29316
PID : 2986
CONNECTING TIME: 19301(us)
======================================
[Thu Jun 13 17:41:00 2019] IdleProcessCount : 5
============= New client ============
TIME : [Thu Jun 13 17:41:01 2019]
URL : /abTde/AA
IP : 127.0.0.1
Port : 29828
PID : 2990
======================================
[Thu Jun 13 17:41:01 2019] IdleProcessCount : 4
```



```
============= New client =============
TIME : [Thu Jun 13 17:41:04 2019]
URL : /abTde/AA/4TFF
IP : 127.0.0.1
Port : 30340
PID : 2994
======================================
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:04 2019] 3075 process is forked.
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:04 2019] 3079 process is forked.
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:41:05 2019]
URL : /abTde/AA/4TFF/TE
IP : 127.0.0.1
Port : 30852
PID : 3022
======================================
[Thu Jun 13 17:41:05 2019] IdleProcessCount : 4
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:06 2019]
URL : /abTde/AA
IP : 127.0.0.1
Port : 29828
PID : 2990
CONNECTING TIME: 25363(us)
======================================
[Thu Jun 13 17:41:06 2019] IdleProcessCount : 5
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:09 2019]
URL : /abTde/AA/4TFF
IP : 127.0.0.1
Port : 30340
PID : 2994
CONNECTING TIME: 20922(us)
======================================
[Thu Jun 13 17:41:09 2019] IdleProcessCount : 6
```

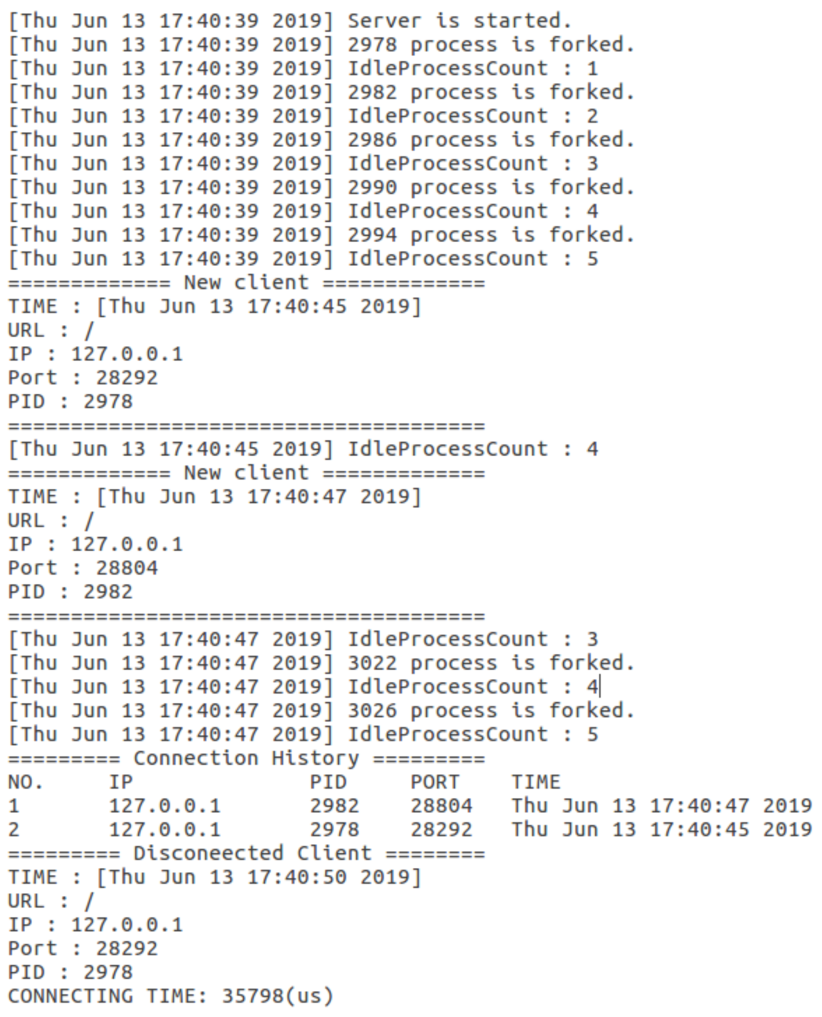


```
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       3022    30852   Thu Jun 13 17:41:05 2019
2       127.0.0.1       2994    30340   Thu Jun 13 17:41:04 2019
3       127.0.0.1       2990    29828   Thu Jun 13 17:41:01 2019
4       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
5       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
6       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:10 2019]
URL : /abTde/AA/4TFF/TE
IP : 127.0.0.1
Port : 30852
PID : 3022
CONNECTING TIME: 16664(us)
======================================
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 7
[Thu Jun 13 17:41:10 2019] 3022 process is terminated.
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 6
[Thu Jun 13 17:41:10 2019] 2994 process is terminated.
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 5
============= New client ============
TIME : [Thu Jun 13 17:41:11 2019]
URL : /abTde/AA/4TFF/
IP : 127.0.0.1
Port : 31364
PID : 3026
======================================
[Thu Jun 13 17:41:11 2019] IdleProcessCount : 4
============= New client ============
TIME : [Thu Jun 13 17:41:12 2019]
URL : /abTde/AA/
IP : 127.0.0.1
Port : 31876
PID : 2986
======================================
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:12 2019] 3126 process is forked.
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:12 2019] 3130 process is forked.
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 5
```

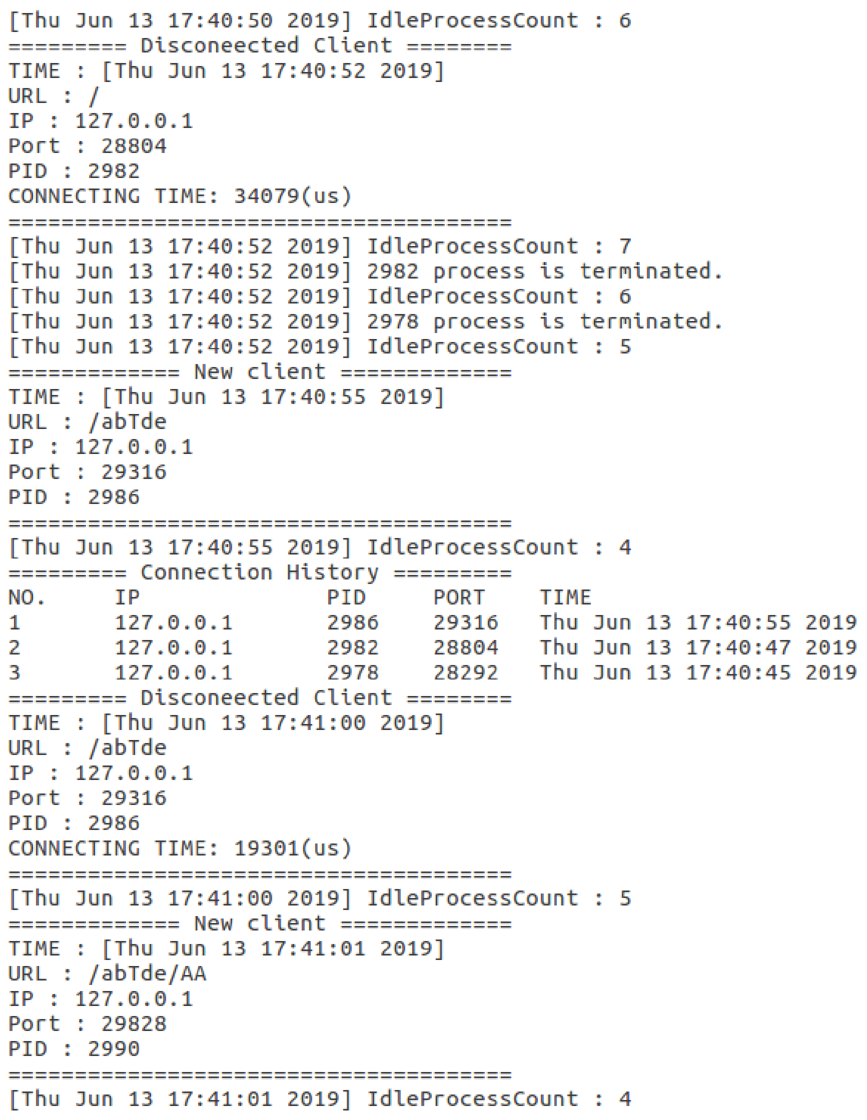


```
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:16 2019]
URL : /abTde/AA/4TFF/
IP : 127.0.0.1
Port : 31364
PID : 3026
CONNECTING TIME: 9219(us)
======================================
[Thu Jun 13 17:41:16 2019] IdleProcessCount : 6
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:17 2019]
URL : /abTde/AA/
IP : 127.0.0.1
Port : 31876
PID : 2986
CONNECTING TIME: 12577(us)
======================================
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 7
[Thu Jun 13 17:41:17 2019] 2986 process is terminated.
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 6
[Thu Jun 13 17:41:17 2019] 3026 process is terminated.
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 5
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2986    31876   Thu Jun 13 17:41:12 2019
2       127.0.0.1       3026    31364   Thu Jun 13 17:41:11 2019
3       127.0.0.1       3022    30852   Thu Jun 13 17:41:05 2019
4       127.0.0.1       2994    30340   Thu Jun 13 17:41:04 2019
5       127.0.0.1       2990    29828   Thu Jun 13 17:41:01 2019
6       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
7       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
8       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
^C
[Thu Jun 13 17:41:23 2019] 2990 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:23 2019] 3126 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:23 2019] 3130 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 2
[Thu Jun 13 17:41:23 2019] 3075 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 1
[Thu Jun 13 17:41:23 2019] 3079 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 0
[Thu Jun 13 17:41:23 2019] Server is terminated.
```

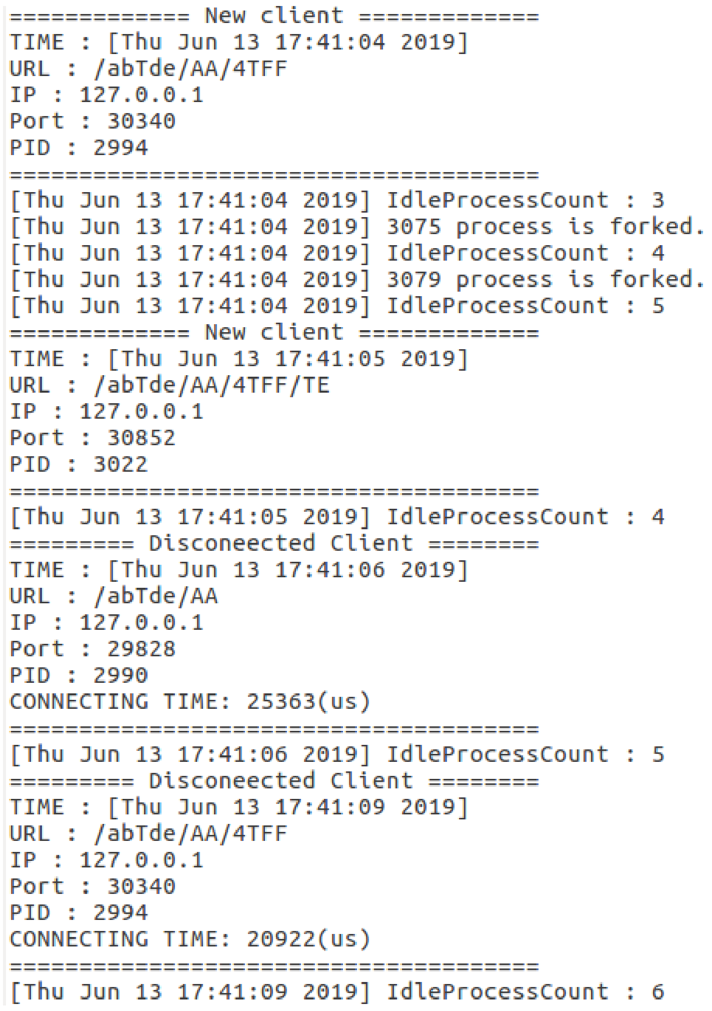
위 동작의 대한 내용은 이미 전 과제에서 설명 한데로 동작하며, 출력 format에 추가적인 요소만 들어가고 다르지 않다. 아래는 결과적으로 log에 쓰이는 결과를 보여준다.



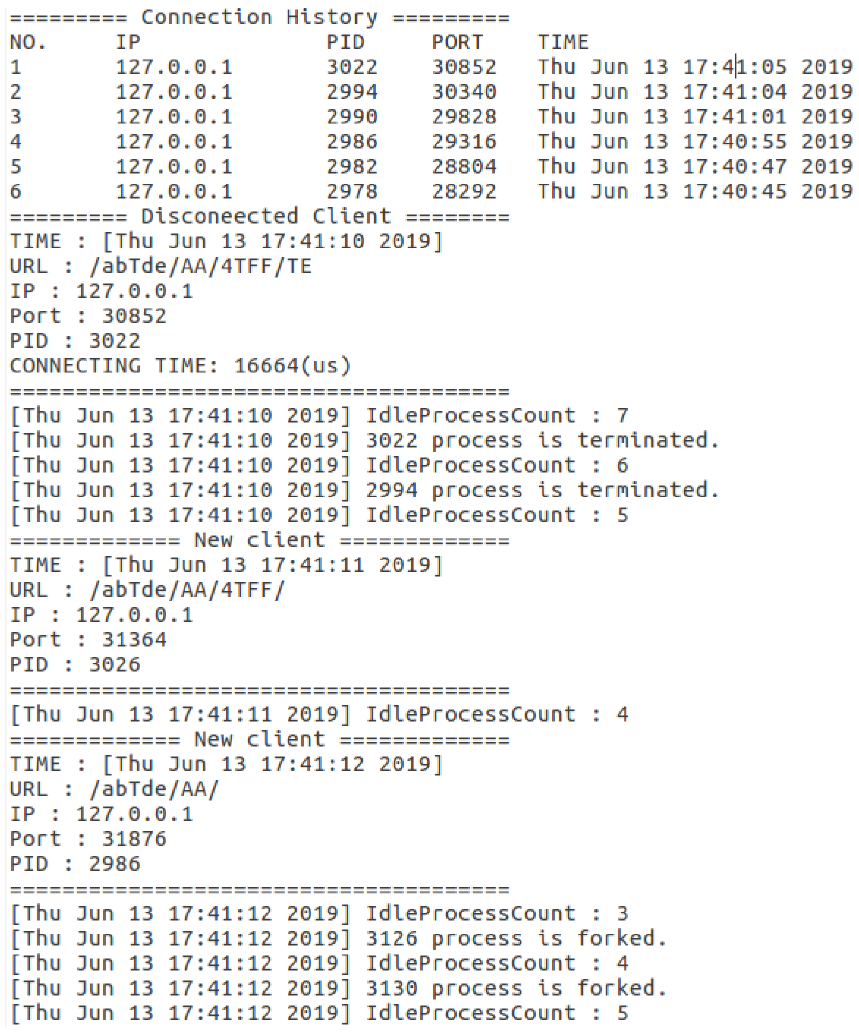
```
[Thu Jun 13 17:40:39 2019] Server is started.
[Thu Jun 13 17:40:39 2019] 2978 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 1
[Thu Jun 13 17:40:39 2019] 2982 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 2
[Thu Jun 13 17:40:39 2019] 2986 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 3
[Thu Jun 13 17:40:39 2019] 2990 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 4
[Thu Jun 13 17:40:39 2019] 2994 process is forked.
[Thu Jun 13 17:40:39 2019] IdleProcessCount : 5
============= New client ============
TIME : [Thu Jun 13 17:40:45 2019]
URL : /
IP : 127.0.0.1
Port : 28292
PID : 2978
======================================
[Thu Jun 13 17:40:45 2019] IdleProcessCount : 4
============= New client ============
TIME : [Thu Jun 13 17:40:47 2019]
URL : /
IP : 127.0.0.1
Port : 28804
PID : 2982
======================================
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 3
[Thu Jun 13 17:40:47 2019] 3022 process is forked.
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 4
[Thu Jun 13 17:40:47 2019] 3026 process is forked.
[Thu Jun 13 17:40:47 2019] IdleProcessCount : 5
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
2       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:40:50 2019]
URL : /
IP : 127.0.0.1
Port : 28292
PID : 2978
CONNECTING TIME: 35798(us)
```



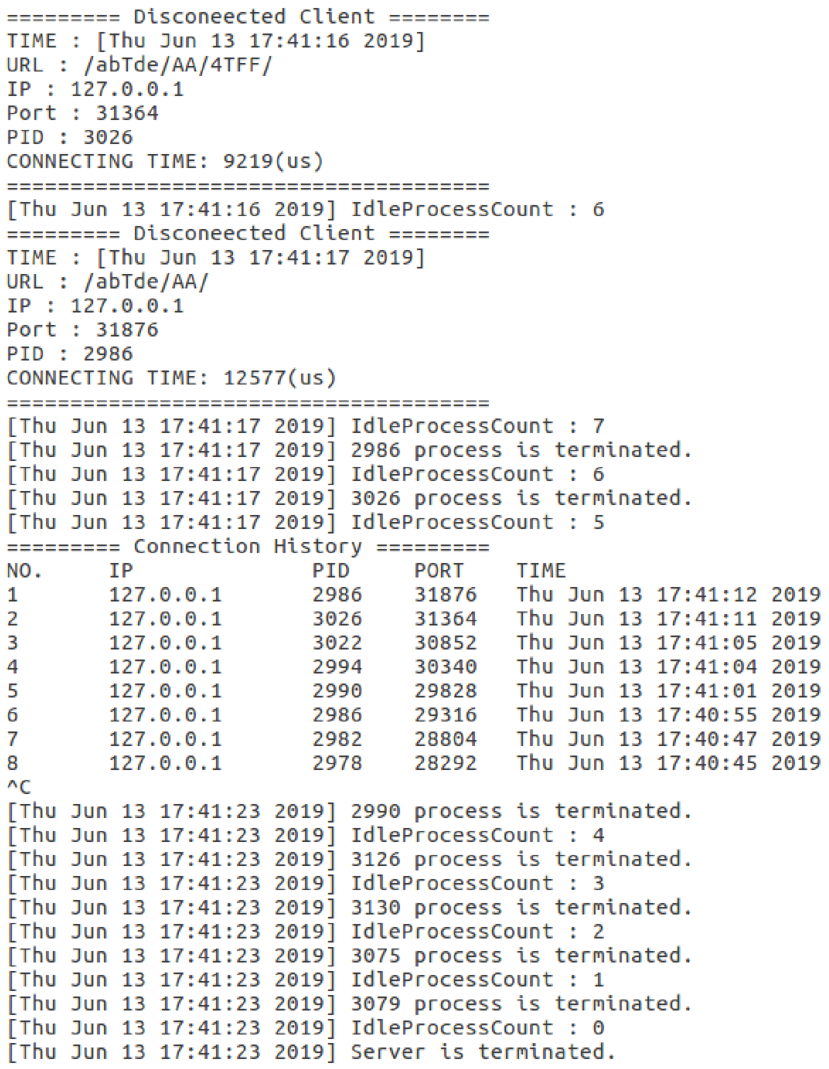
```
[Thu Jun 13 17:40:50 2019] IdleProcessCount : 6
========= Disconeected Client ========
TIME : [Thu Jun 13 17:40:52 2019]
URL : /
IP : 127.0.0.1
Port : 28804
PID : 2982
CONNECTING TIME: 34079(us)
======================================
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 7
[Thu Jun 13 17:40:52 2019] 2982 process is terminated.
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 6
[Thu Jun 13 17:40:52 2019] 2978 process is terminated.
[Thu Jun 13 17:40:52 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:40:55 2019]
URL : /abTde
IP : 127.0.0.1
Port : 29316
PID : 2986
======================================
[Thu Jun 13 17:40:55 2019] IdleProcessCount : 4
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
2       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
3       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:00 2019]
URL : /abTde
IP : 127.0.0.1
Port : 29316
PID : 2986
CONNECTING TIME: 19301(us)
======================================
[Thu Jun 13 17:41:00 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:41:01 2019]
URL : /abTde/AA
IP : 127.0.0.1
Port : 29828
PID : 2990
======================================
[Thu Jun 13 17:41:01 2019] IdleProcessCount : 4
```



```
============= New client =============
TIME : [Thu Jun 13 17:41:04 2019]
URL : /abTde/AA/4TFF
IP : 127.0.0.1
Port : 30340
PID : 2994
======================================
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:04 2019] 3075 process is forked.
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:04 2019] 3079 process is forked.
[Thu Jun 13 17:41:04 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:41:05 2019]
URL : /abTde/AA/4TFF/TE
IP : 127.0.0.1
Port : 30852
PID : 3022
======================================
[Thu Jun 13 17:41:05 2019] IdleProcessCount : 4
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:06 2019]
URL : /abTde/AA
IP : 127.0.0.1
Port : 29828
PID : 2990
CONNECTING TIME: 25363(us)
======================================
[Thu Jun 13 17:41:06 2019] IdleProcessCount : 5
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:09 2019]
URL : /abTde/AA/4TFF
IP : 127.0.0.1
Port : 30340
PID : 2994
CONNECTING TIME: 20922(us)
======================================
[Thu Jun 13 17:41:09 2019] IdleProcessCount : 6
```



```
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       3022    30852   Thu Jun 13 17:41:05 2019
2       127.0.0.1       2994    30340   Thu Jun 13 17:41:04 2019
3       127.0.0.1       2990    29828   Thu Jun 13 17:41:01 2019
4       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
5       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
6       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:10 2019]
URL : /abTde/AA/4TFF/TE
IP : 127.0.0.1
Port : 30852
PID : 3022
CONNECTING TIME: 16664(us)
======================================
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 7
[Thu Jun 13 17:41:10 2019] 3022 process is terminated.
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 6
[Thu Jun 13 17:41:10 2019] 2994 process is terminated.
[Thu Jun 13 17:41:10 2019] IdleProcessCount : 5
============= New client =============
TIME : [Thu Jun 13 17:41:11 2019]
URL : /abTde/AA/4TFF/
IP : 127.0.0.1
Port : 31364
PID : 3026
======================================
[Thu Jun 13 17:41:11 2019] IdleProcessCount : 4
============= New client =============
TIME : [Thu Jun 13 17:41:12 2019]
URL : /abTde/AA/
IP : 127.0.0.1
Port : 31876
PID : 2986
======================================
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:12 2019] 3126 process is forked.
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:12 2019] 3130 process is forked.
[Thu Jun 13 17:41:12 2019] IdleProcessCount : 5
```



```
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:16 2019]
URL : /abTde/AA/4TFF/
IP : 127.0.0.1
Port : 31364
PID : 3026
CONNECTING TIME: 9219(us)
======================================
[Thu Jun 13 17:41:16 2019] IdleProcessCount : 6
========= Disconeected Client ========
TIME : [Thu Jun 13 17:41:17 2019]
URL : /abTde/AA/
IP : 127.0.0.1
Port : 31876
PID : 2986
CONNECTING TIME: 12577(us)
======================================
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 7
[Thu Jun 13 17:41:17 2019] 2986 process is terminated.
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 6
[Thu Jun 13 17:41:17 2019] 3026 process is terminated.
[Thu Jun 13 17:41:17 2019] IdleProcessCount : 5
========= Connection History =========
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2986    31876   Thu Jun 13 17:41:12 2019
2       127.0.0.1       3026    31364   Thu Jun 13 17:41:11 2019
3       127.0.0.1       3022    30852   Thu Jun 13 17:41:05 2019
4       127.0.0.1       2994    30340   Thu Jun 13 17:41:04 2019
5       127.0.0.1       2990    29828   Thu Jun 13 17:41:01 2019
6       127.0.0.1       2986    29316   Thu Jun 13 17:40:55 2019
7       127.0.0.1       2982    28804   Thu Jun 13 17:40:47 2019
8       127.0.0.1       2978    28292   Thu Jun 13 17:40:45 2019
^C
[Thu Jun 13 17:41:23 2019] 2990 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 4
[Thu Jun 13 17:41:23 2019] 3126 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 3
[Thu Jun 13 17:41:23 2019] 3130 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 2
[Thu Jun 13 17:41:23 2019] 3075 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 1
[Thu Jun 13 17:41:23 2019] 3079 process is terminated.
[Thu Jun 13 17:41:23 2019] IdleProcessCount : 0
[Thu Jun 13 17:41:23 2019] Server is terminated.
```

SIG_INT신호를 받는 경우 ^C문구를 출력하고, Connecting time과 URL을 추가하고 나머지는 모두 동일하며, 터미널에 출력한 것과 마찬가지로 txt파일에도 잘 출력되는 것을 확인할 수 있다.

5. Conclusion

이번 4-3과제의 고찰은 전 과제와 비교적 많은 느낀 점은 없다. Semaphore의 사용은 이론을 통해서 배운 개념을 기반으로 실습에서는 그렇게 어렵지 않게 실험을 하였고, 기존 동작을 바꾸는 것이 아니라 출력을 txt에 옮기는 것이 전부이기 때문에 기존의 과제를 잘 수행해왔다면, 크게 어려움이 없는 과제다. 간단하게 semaphore를 설명하면, 전에 썼던 mutex_lock과 비슷하기 쓰이는 공유자원에 대한 접근을 제어해주는 기술로써 log.txt에 write하는 동작이 여러 부분에서 일어날 수 있기 때문에 해당 문제를 막기 위해 사용되었다. Thread의 mutex와의 차이점은 mutex는 접근을 막는다기 보다는 동기화를 목적으로 사용되지만, semaphore는 critical section자체의 접근을 아예 막아버리는 것으로 이해했다.