로고이(가) 표시된 사진

자동 생성된 설명

시스템프로그래밍실습

Assignment 3-3 과제

수업 명 : 시스템프로그래밍실습

과제 이름 : assignment3-3

담당 교수님 : 김태석 교수님

학 번 : 2019202005

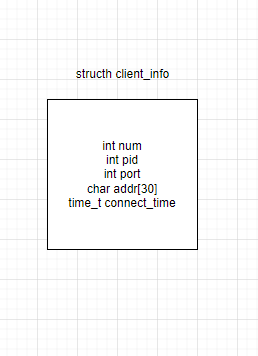
이 름 : 남종식

* **과제 소개**

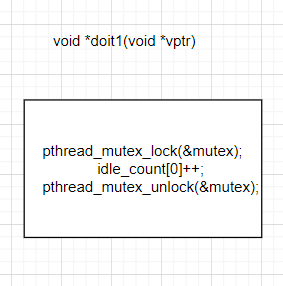
이번 과제는 저번 과제를 기반으로 하여 클라이언트의 연결 정보를 로그 파일에 기록하는 기능을 추가하는 것입니다. 서버가 시작되고 클라리언트가 접속하면 해당 정보를 터미널에 출력뿐만 아니라 로그 파일에 작성합니다. 이때 터미널에는 history를 저번과제와 동일하게 출력하지만 로그 파일 작성에 있어서는 history는 제외하고 작성합니다. 클라이언트가 연결되고 연결이 끊겼을 때 출력하는 클라이언트의 정보가 이번 과제에서 추가되었습니다.

먼저, 클라인언트의 요청한 경로를 출력해여 하며 연결이 끊겼을 때는 경로와 함께 클라이언트의 연결 지속 시간 또한 출력해야 합니다. 이때 시단의 단위는 마이크로 초입니다. 저번 과제에서 sleep(5)를 설정한 시간은 제외하고 측정하여 출력해야 합니다. 로그 파일에는 child프로세스마다 생성한 스레드를 이용해 기록하며 로그 파일에 접근하는 부분에는 semaphore를 이용해 동기화해야 합니다.

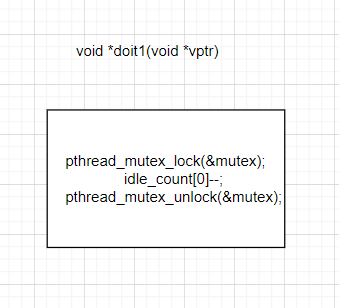
* **Flow Chart**
* **struct client\_info**

****

* **Void \*doit1(void\*vptr)**

****

* **Void \*doit1(void\*vptr)**

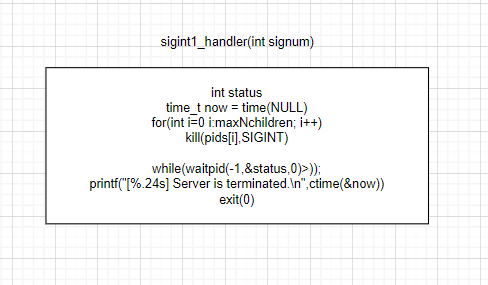
****

* **Sigint\_handler(int signum)**

**텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명**

* **Sigint1\_handler(int signum)**

****

* **Sigusr\_handler2(int signum)**

**텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명**

* **Void threadFunction(void\*arg)**

**텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

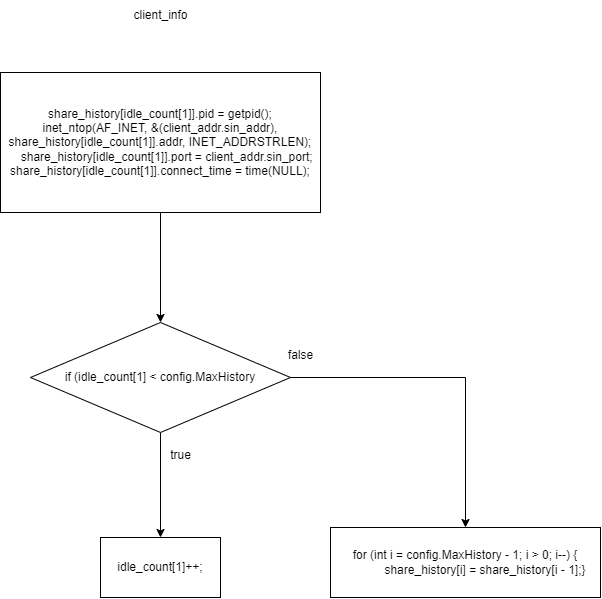
자동 생성된 설명**

* **Child\_make function**

**텍스트, 도표, 라인, 평행이(가) 표시된 사진

자동 생성된 설명**

* **Client\_info function**

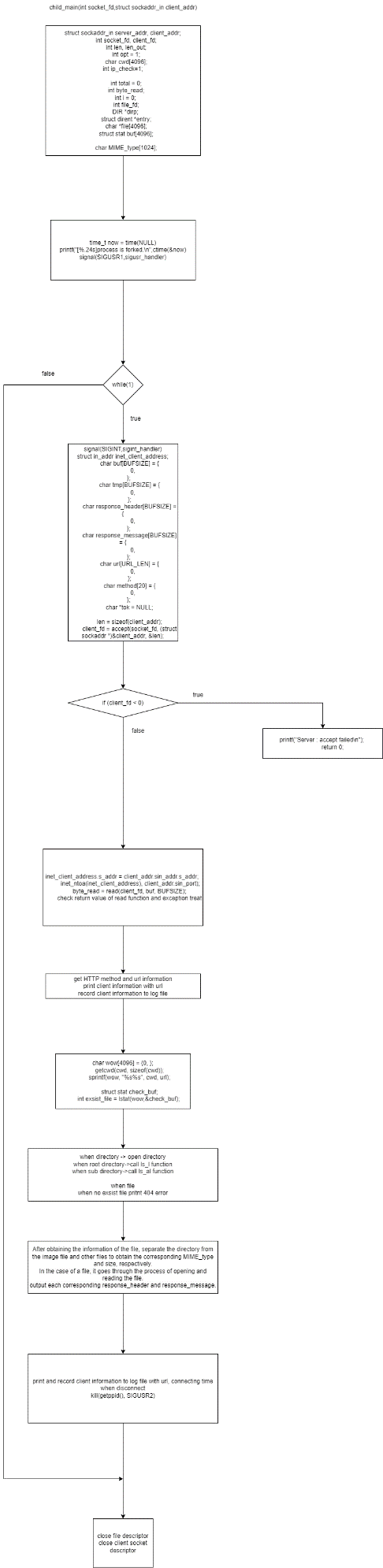


* **Main fuction** (세마포어 및 로그 파일 부분 추가)

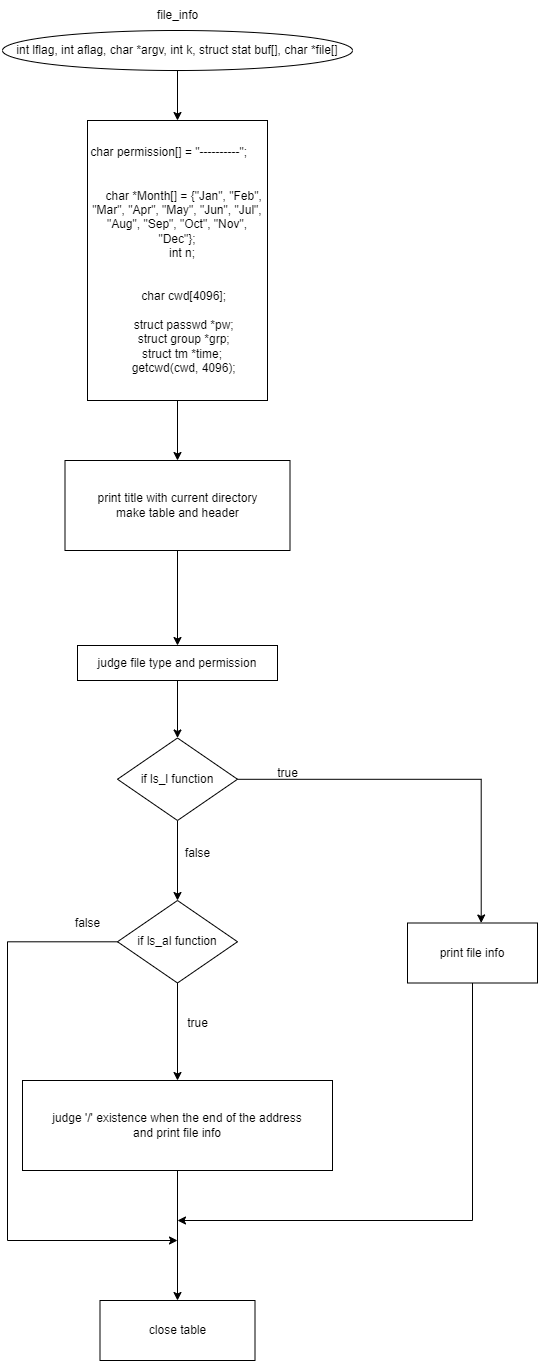
텍스트, 영수증, 흑백, 문서이(가) 표시된 사진

자동 생성된 설명

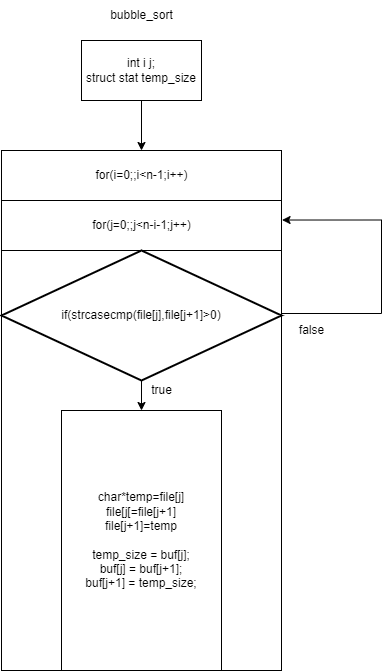
* **Child\_main function** (로그파일 부분 추가)



* **File\_info function**



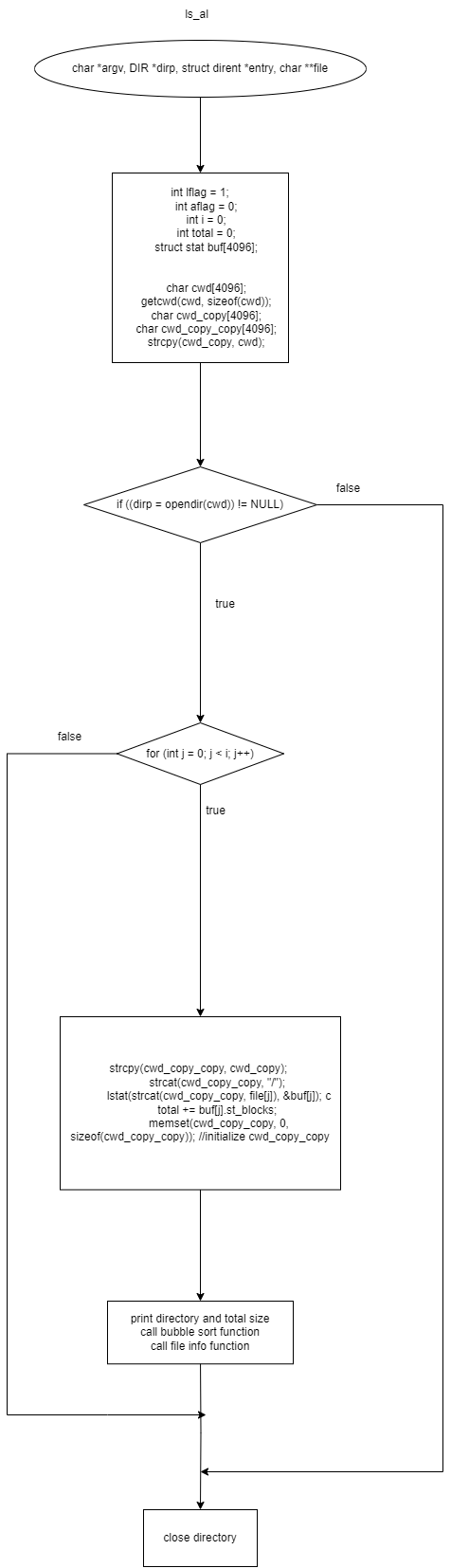
* **Bubble sort function**



* **Ls\_l function**



* **Ls\_al function**



* **Pseudo Code**

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <pwd.h>

#include <grp.h>

#include <time.h>

#include <fnmatch.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <signal.h>

#include <pthread.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define URL\_LEN 256

#define BUFSIZE 1024

#define PORTNO 40000

int shm\_id

void \*shm\_addr

int\* idle\_count

int shm\_id1

void \*shm\_addr1

struct client\_info \*share\_history

FILE\* logfile

sem\_t \*semaphore

pthread\_t tidA

mutex initialize

define file info function

define bubble sort

define ls -l function

define ls -al function

define print\_recent\_clients

define handle\_signal

static int MaxNchildren

static pid\_t \*pids

define struct client info

define struct HttpdConfig

define client\_info function to store client information

define sigint\_handler

define sigint1\_handler

define sigusr\_handler2

define \*doit1(void \*vptr)

define \*doit2(void \*vptr)

void\* threadFunction(void\* arg)

Main function

{

Make semaphore

Open httpd.conf file

When get httpd.conf

Add null string instead of \n

Get value of conf

Close httpd.conf file

Make shared memory and attach them to process

Make idle\_count shared memory and history shared memory

declare server address, client address

declare socket descriptor, opt=1

Call the socket function to create a socket file descriptor

Call setsocketopt to enable reuse of previously used port numbers

Initialize server\_addr structure

Save address information for the server socket

Bind a socket with socket\_fd to the IP address and port number specified in the server\_addr structure

Socket wait to accept incoming connections from client sockets.

signal handler

alarm in 10 seconds

call sigint1\_handler

print server is started with time

open log file and record information to log file

close log file

call sigusr\_handler2

maxNchildren=5

make pids array

call child\_make function

fork and call child\_main function

in parent process

signal(SIGINT,sigint1\_handler)

print(“process is forked”)

open log file and record information to log file

close log file

idle\_count +1

print count of idle process

return pid

while(1)

when idle process count < minidlecount

for(int i=(idle\_count[0]); i<5; i++)

call child\_make(socket\_fd,client\_addr);

when idle process count > maxidlecount

for(int i=(idle\_count[0]); i>5; i--)

kill(pids[b], SIGINT)

idle\_count[0]—

print count of idle process

open log file and record information to log file

close log file

In child main function

signal(SIGINT,sigint1\_handler);

declare pointer to directory stream and Pointer to a dirent structure

declare file list stat structure that stores information about a file or directory

Define variables required to process client requests

signal(SIGUSR2,sigusr\_handler2);

while (1)

{

Set signal handler(sigint)

define size of client address

get the socket file descriptor associated with the client

call gettimeofday function to calculate connecting time

store ip address

Make an exception when read function return 0

Call client\_info function to get client information

print the client's IP address and port number with time

open log file and record information to log file

close log file

idle\_count-1

kill(getppid(), SIGUSR2)

Copy HTTP method information and url information

get current working directory

get path of wow (wow=cwd+url)

Open the file to read all lines, compare it to the ip address, and proceed if the corresponding ip address exists, and if there is no ip address, print an access restriction message

get information of files

open directory

{

when root directory

{

call ls -l function

}

when sub directory

{

call ls -al function

}

}

get file info

when directory

{

MIME\_type=text/html

get size of response message

print response header

write response header

write response message

}

when image file

{

MIME\_type=image/\*

get file size

file open

file read

print response header

write response header

write response message

close file descriptor

}

when file

{

MIME\_type=text/plain

get file size

file open

file read

print response header

write response header

write response message

close file descriptor

}

Call gettimeofday function to calculate connecting time

Calculate connecting time with micro seconds

Sleep(5)

print the client's IP address and port number with time

open log file and record information to log file

close log file

close semaphore

kill(getppid(), SIGUSR2)

close client socket descriptor

}

}

close socket descriptor

}

define print\_recent\_clients function

{

print client information

alarm(10)

}

define file info function about directory

{

get current directory

print title current directory

make table

make header

for (n = 0; n < k; n++)

{

judge file mode and permission

get File Owner's Information

get the information of the file's owning group

get the last modified time of that file

function for Parsing Time Information

when root directory

{

when directory hyperlink with blue color :file[n]

when link file hyperlink with green color :file[n]

when file hyperlink with red color :file[n]

}

when sub directory

{

When the '/' is at the end of the address

{

when directory hyperlink with blue color :url file[n]

when link file hyperlink with green color :url file[n]

when file hyperlink with red color :url file[n]

}

When there is no '/' at the end of the address

{

when directory hyperlink with blue color :url/file[n]

when link file hyperlink with green color :url/file[n]

when file hyperlink with red color :url/file[n]

}

}

}

close table

}

define bubble sort function

{

initialize value

Bubble sort the file names alphabetically

}

define ls -l function

{

get current directory

get current directory+url

when open directory

{

Repeat read directory

{

store file names without hidden files

}

for (int j = 0; j < i; j++)

{

get current directory+url+/

get info about current directory+url+/+file

get block size

initialize current directory+url+/+file

}

print directory path & total size

call bubble\_sort function

print file info

}

close directory

}

define ls -al function

{

get current directory

get current directory+url

when open directory+url

{

Repeat read directory

{

store file list with hidden file

}

for (int j = 0; j < i; j++)

{

get current directory+url+/

get info about current directory+url+/+file

get block size

initialize current directory+url+/+file

}

print directory path & toatl size

call bubble\_sort function

print file info

}

close directory

}

* **결과화면**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

먼저 어떠한 클라이언트가 연결되지 않은 경우 출력 화면입니다. History에 아무 클라이언트도 출력되지 않고 서버가 시작되고 5개의 프로세스가 fork된 다음 차례로 프로세스가 종료되고 서버가 종료된 모습을 확인할 수 있습니다.

fork되고 terminate되는 동안 idle process의 개수도 저번 과제처럼 출력되는 모습을 확인할 수 있습니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

다음으로 server\_log.txt파일에 작성된 내용을 캡처한 화면입니다. 터미널에 출력된 내용에서 history부분만 기록되지 않은 모습을 확인할 수 있습니다. history출력 부분을 제외하고는 같은 내용이 server\_log.txt파일에 작성된 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

다음은 프로그램을 실행시키고 f5를 통해 새로 고침을 진행 후 출력 화면입니다. 왼쪽 화면을 보면 서버 쪽에서 화면을 잘 출력하고 있는 모습을 확인할 수 있으며 오른쪽 화면을 통해 터미널에서 클라이언트가 연결되고 연결이 끊길 때 클라이언트의 정보를 잘 출력하고 있는 모습을 확인할 수 있습니다. 10초가 지나면 history를 통해 연결된 클라이언트의 정보 또한 확인할 수 있습니다.

저번 과제와 달리 이번 과제에서는 클라이언트가 연결되었을 때 클라이언트가 요청한 경로를 출력해주었고 연결이 끊겼을 때는 경로와 함께 클라이언트의 연결 지속 시간을 마이크로 초로 출력했습니다. 현재 루트 위치에서 새로 고침을 하였기 때문에 경로가 /로 출력된 모습입니다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Server\_log.txt파일 또한 클라이언트가 연결되었을 때 요청한 경로가, 연결이 끊겼을 때 경로와 연결 지속 시간이 포함되어 기록된 모습을 확인할 수 있었습니다. 그리고 history부분은 출력하지 않는 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

다음은 2019202005\_semaphore\_server.c파일의 하이퍼링크를 클릭했을 떼 출력되는 화면입니다. 먼저 서버 쪽 에서는 파일의 내용을 출력하는 모습을 확인할 수 있으며 터미널에서는 url 부분에 /2019202005\_semaphore\_server.c 이렇게 클라이언트가 요청한 주소가 출력되는 모습을 확인할 수 있습니다.

루트 디렉토리 아래 2019202005\_semaphore\_server.c파일을 클릭했으니 경로가 잘 출력되는 것을 알 수 있습니다.

텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

자동 생성된 설명

history기록 없이 터미널에 출력된 내용들이 server\_log.txt파일에 잘 작성된 모습을 확인할 수 있습니다. 요청한 주소와 연결지속시간 또한 계속 잘 작성되고 있는 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

다시 먼저 2019202005\_semaphore\_server.c파일 요청 후 위에서 출력되는 내용과 똑같은 내용이 출력됩니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

server\_log.txt파일에 내용이 잘 기록되고 있습니다.

연결될 때의 클라이언트가 요청한 주소와 연결이 끊겼을 때 클라이언트가 요청한 주소가 일치하고 연결 지속 시간이 추가되어 출력된 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

다음으로 kw디렉토리의 하이퍼링크를 클릭한 화면입니다. 왼쪽 서버 측 화면에서는 kw 디렉토리로 이동하여 하위 디렉토리 및 파일과 해당하는 path와 total 크기를 출력하는 모습을 확인할 수 있습니다. 오른쪽 터미널에서는 요청한 주소 출력부분에서 /kw로 제대로 출력되고 있는 모습입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

server\_log.txt파일에 위 내용 아래에 잘 기록되고 있습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

kw디렉토리 아래 university디렉토리로 이동한 모습을 출력하는 화면이며 왼쪽 서번 측 화면에서는 university의 하위 파일들을 출력하는 모습을 확인할 수 있습니다. 오른쪽 터미널 화면에서는 요청한 주소 부분에서 /kw/university로 제대로 요청한 주소가 출력되는 모습입니다. 연결 지속 시간 또한 제대로 출력되고 있습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

server\_log.txt파일에 위 내용 아래에 잘 기록되고 있습니다.

* **고찰**

저번 과제에 있어서 시행착오가 너무 많았기 때문에 이에 비해서는 수월하게 마지막 과제를 진행할 수 있었습니다. 먼저, 과제 조건에 맞추어서 로그 파일에 기록하는 부분을 스레드와 세마포어를 이용하여 동기화했습니다. 이를 통해 여러 스레드가 동시에 로그 파일에 접근하는 것을 막을 수 있었습니다. 시스템 프로그래밍 수업시간에 배운 내용과 같이 세마포어를 사용하여 critical section을 설정하면, 한 번에 하나의 스레드만 접근할 수 있도록 보장할 수 있었습니다.   
클라이언트의 접속 정보를 로그 파일에 작성하는 과정에서, 연결 지속 시간은 실제 연결이 종료될 때까지의 시간을 측정하여 출력했으며, 이를 로그 파일에 기록했습니다. 이때는 마이크로 초로 측정하여 기록하였는데 이를 계산하기 위해서는 시스템 프로그래밍 강의자료 time and date를 참고하였습니다.  
마이크로 초로 계산하라는 제안서를 본 후 중간고사 이전 시스템 프로그래밍 수업 시간에 배운 내용이 떠올랐으며 time and date 강의자료를 통해 이 강의 자료에서 나오는 gettimeofday함수를 이용하였습니다.   
과제를 진행하면서 파일 입출력과 스레드, 세마포어 등 다양한 개념과 기능을 조합하여 프로그램을 작성하는 경험을 할 수 있었습니다. 이를 통해 다중 프로세스 및 다중 스레드 환경에서의 동기화와 자원 공유에 대한 이해도를 높일 수 있었습니다.

과제를 통해 교수님께서 수업 시간에 설명해 주셨던 동기화와 자원 공유의 중요성을 느낄 수 있었고, 프로세스 관리와 성능 향상을 조금이나마 한 것 같아 좋았으며 실제 서버 환경에 대해 생각해 볼 수 있었습니다.

**Reference**

2023년 1학기 시스템프로그래밍 & 시스템 프로그래밍 실습 강의자료

Assignment 3-3

2023년 1학기 시스템 프로그래밍 1학기 강의자료 3. Files and directories

2023년 1학기 시스템 프로그래밍 1학기 강의자료 4. Time and date

2023년 1학기 시스템 프로그래밍 1학기 강의자료 6. sockets

2023년 1학기 시스템 프로그래밍 실습 13주차 강의자료 log file semaphore