시스템프로그래밍실습

Assignment 3-1 과제

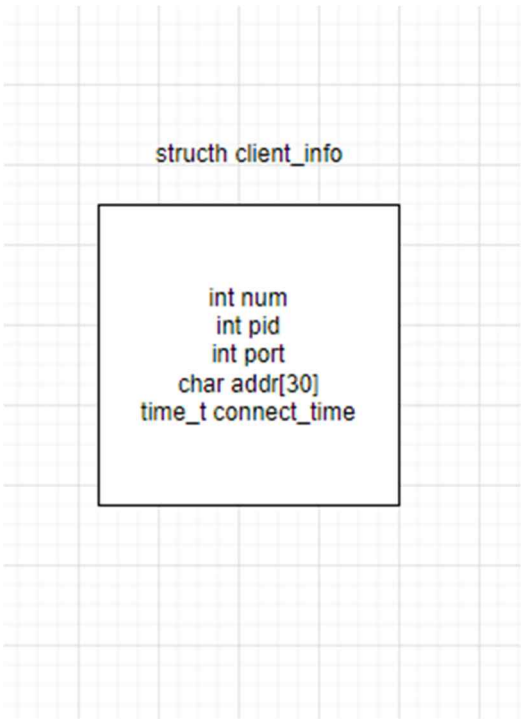
수업 명 : 시스템프로그래밍실습
과제 이름 : assignment3-1
담당 교수님 : 김태석 교수님
학 번 : 2019202005
이 름 : 남종식

과제 소개

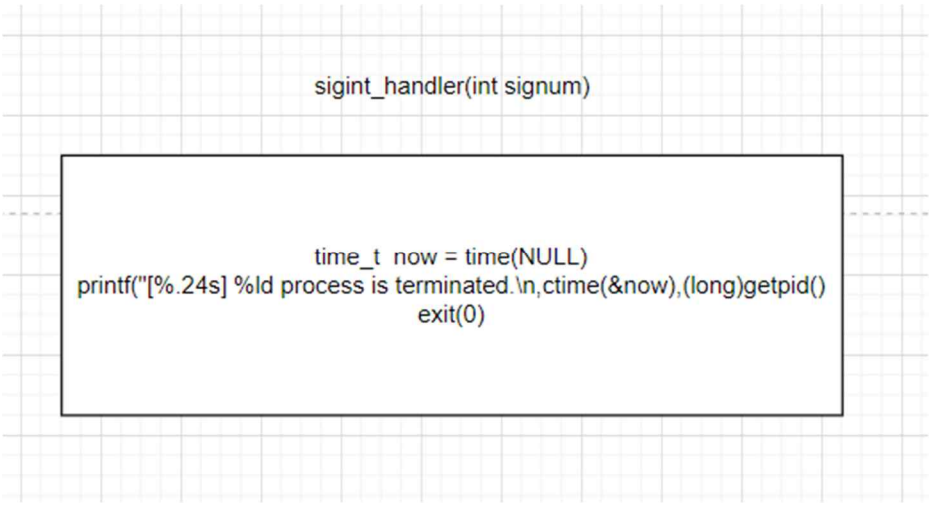
이번 과제는 저번 과제인 다중 접속과 접근 제어를 지원하는 웹 서버 프로그램을 작성하는 과제에 이어서 pre-forked 방식으로 server를 구현하는 과제입니다.
미리 5번의 fork를 통해 5개의 child 프로세스를 생성하고 이를 유지합니다.
저번 과제에서 출력했던 총 request의 수는 출력하지 않으며 connection history 및 NO IP PORT PID TIME부분은 모두 parent 프로세스에서 출력해야 하며 연결된 클라이언트의 내용들은 모두 child 프로세스에서 출력해야 합니다. 서버가 연결되고 종료되는 간단한 로그를 출력해야 하며 child 프로세스 생성 및 종료 시에는 시간 정보와 PID를 출력해야 합니다. 이번 과제에서 연결된 클라이언트의 내용 출력 부분에 있어서 동기화 문제는 고려하지 않습니다.

Flow Chart

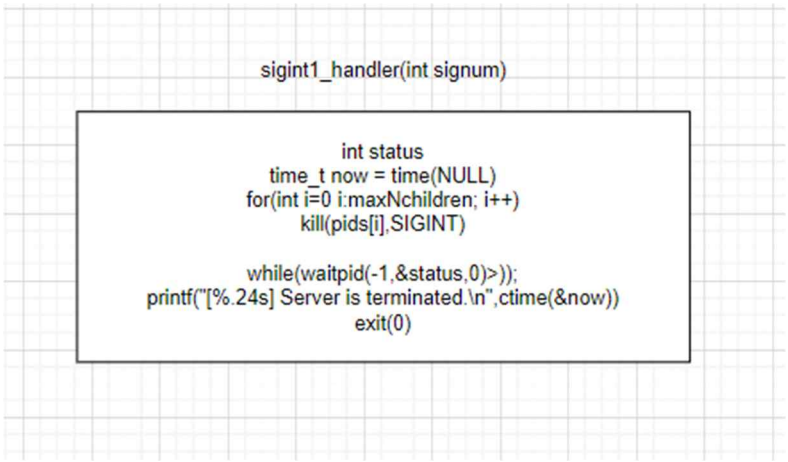
struct client_info



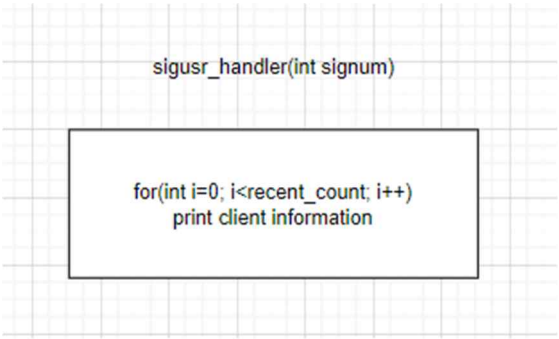
Sigint_handler(int signum)



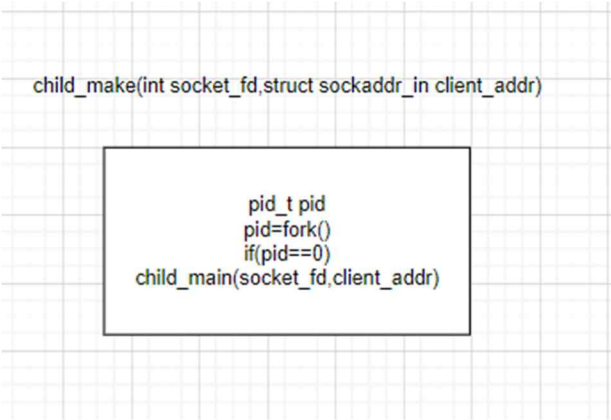
Sigint1_handler(int signum)



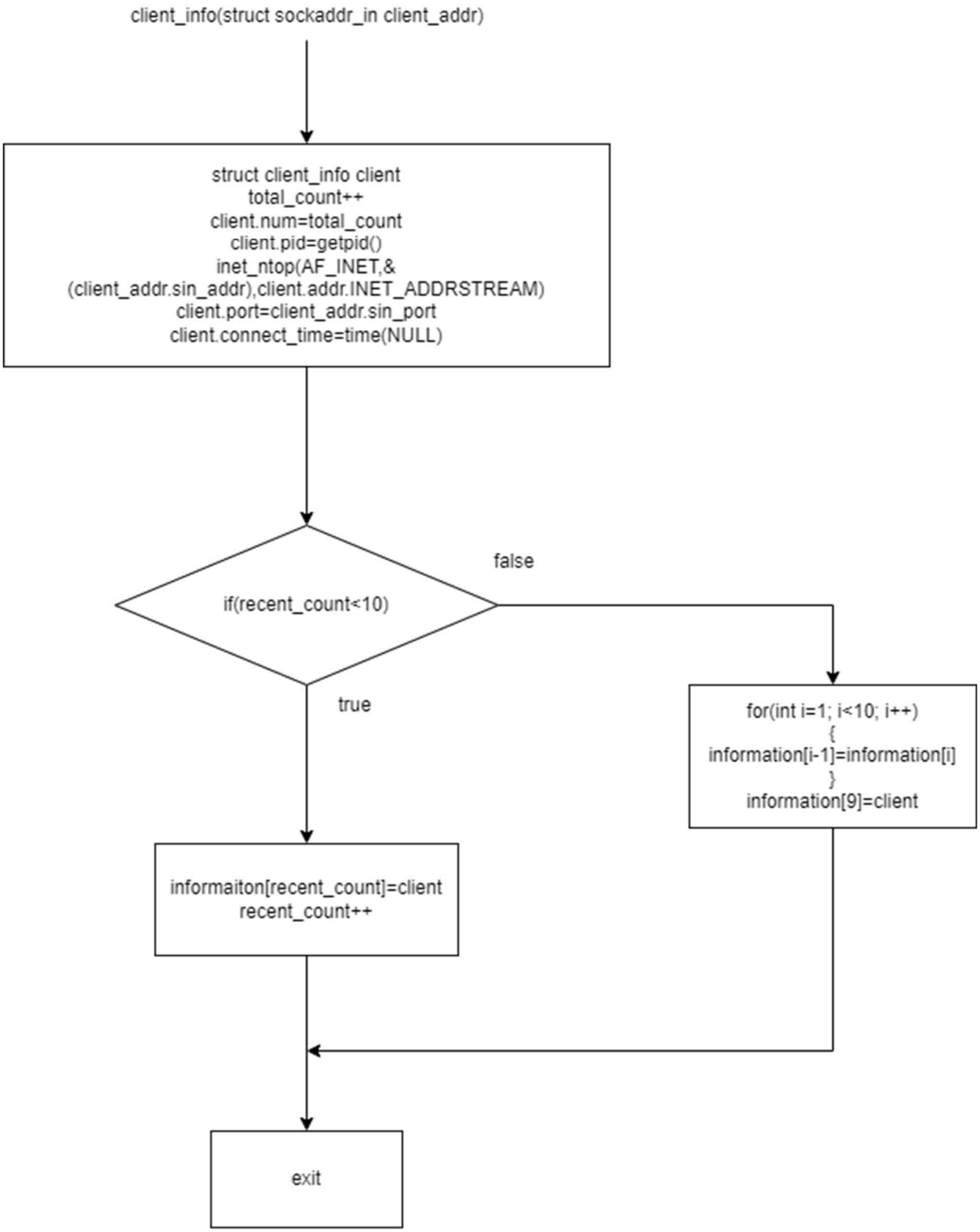
Sigusr_handler(int signum)



Child_make function

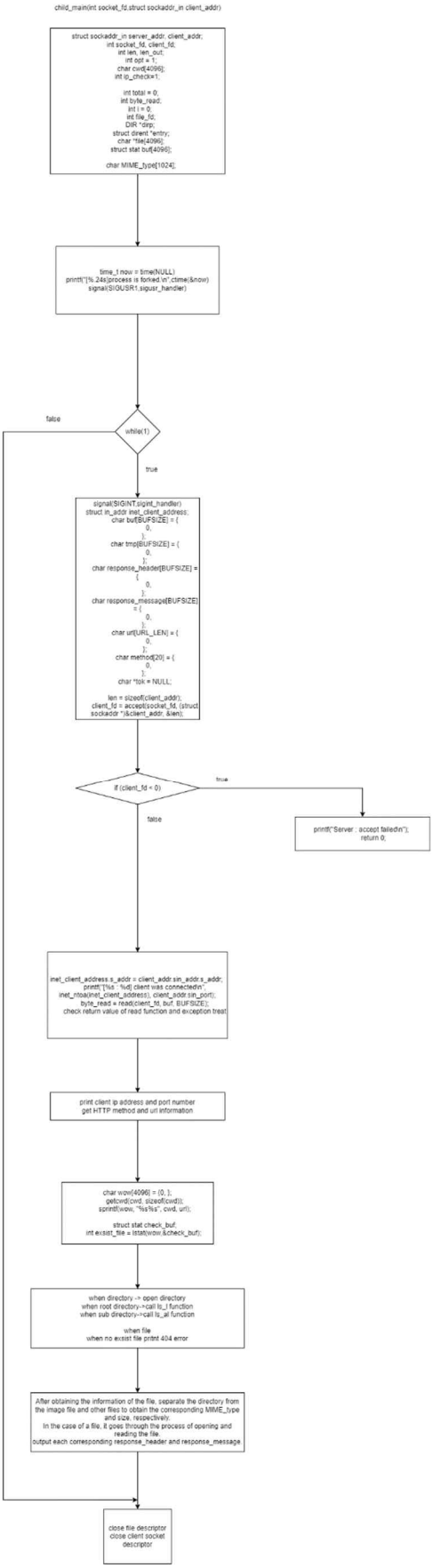


Client_info function

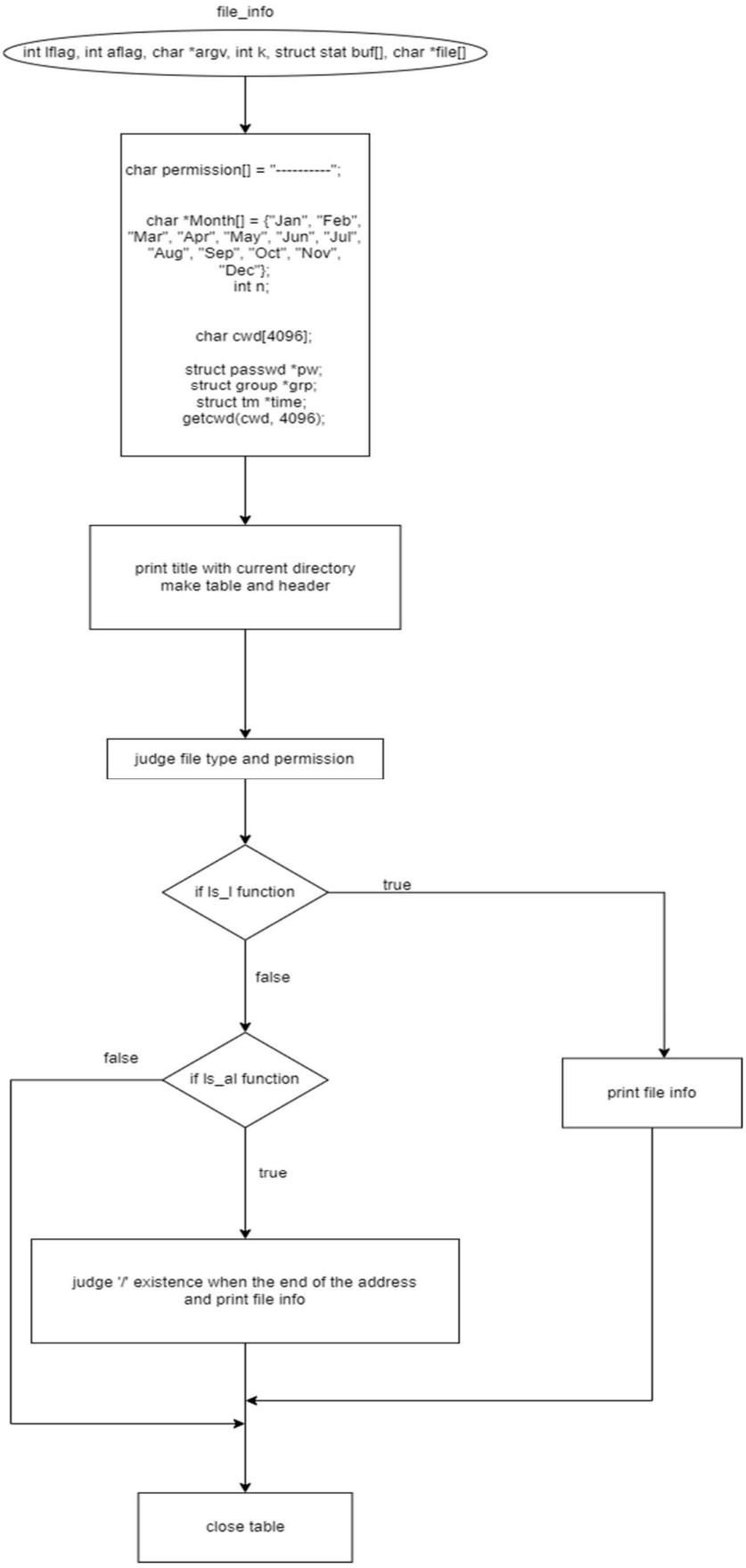


Main fuction

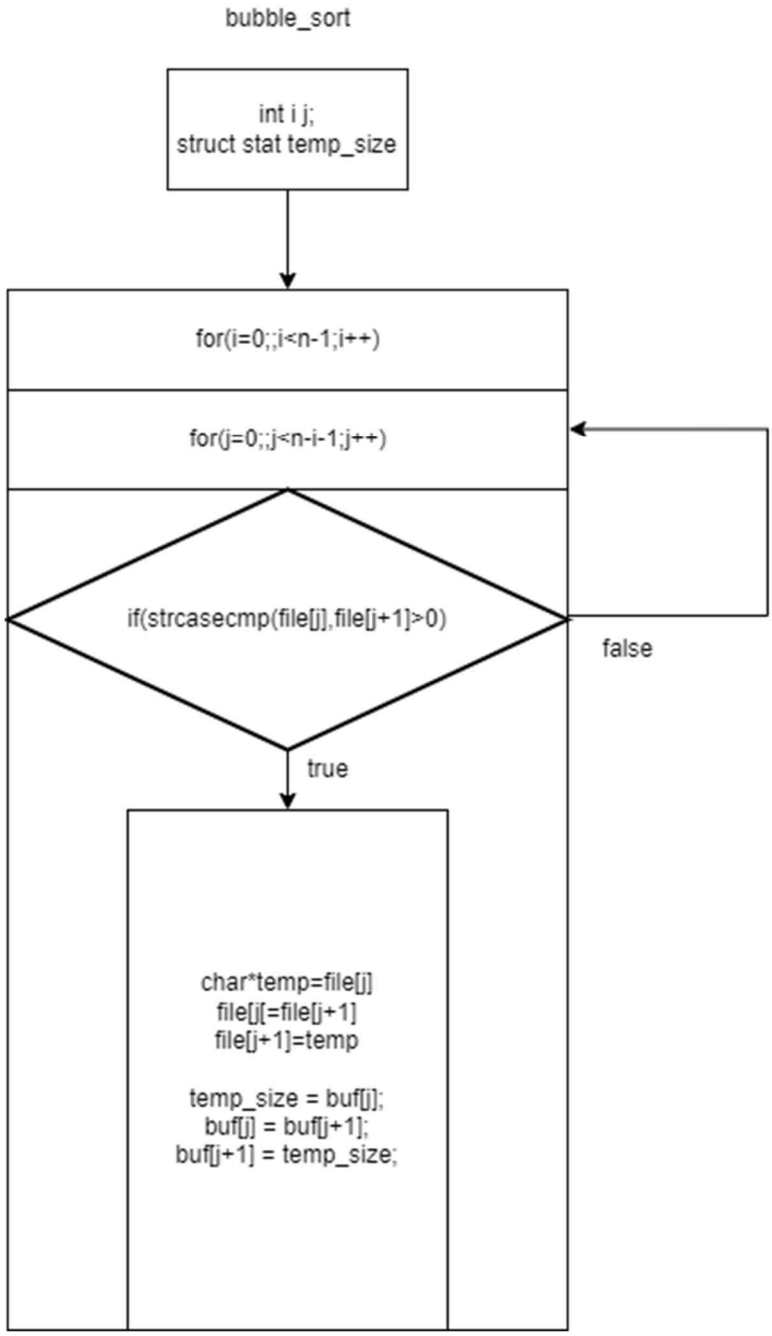
Client_main function



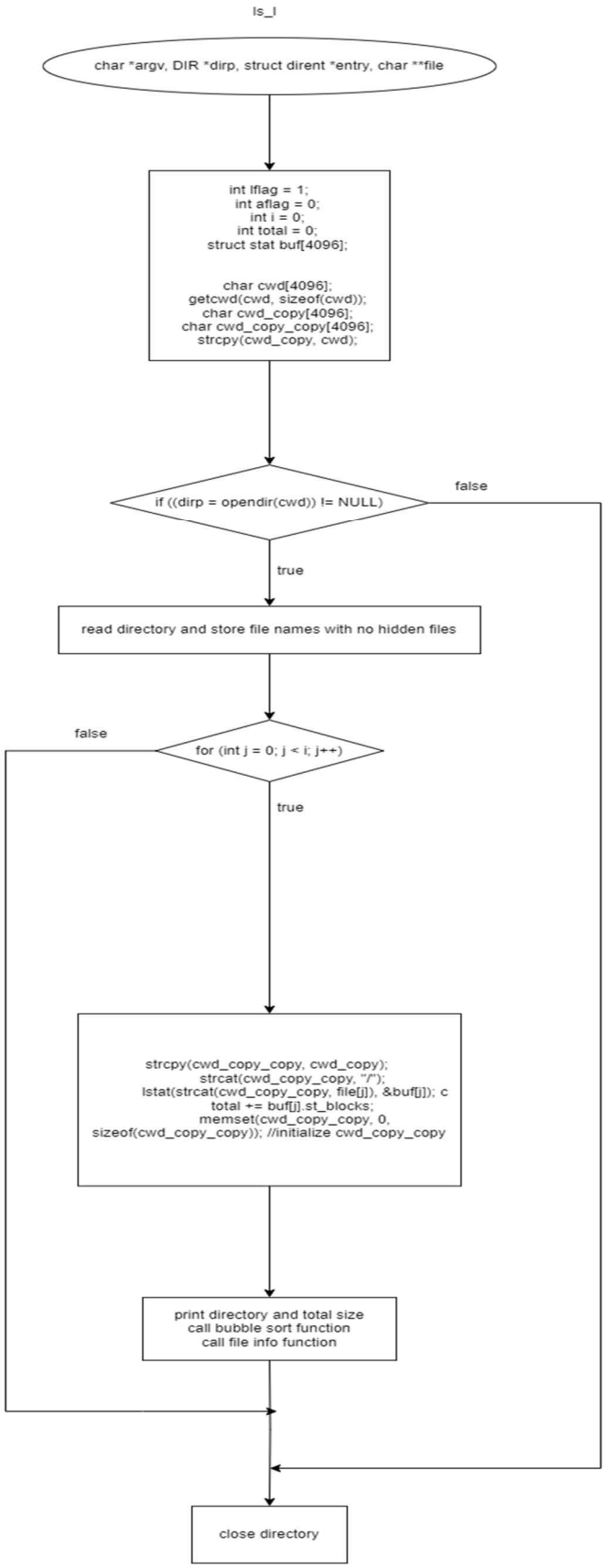
File_info function



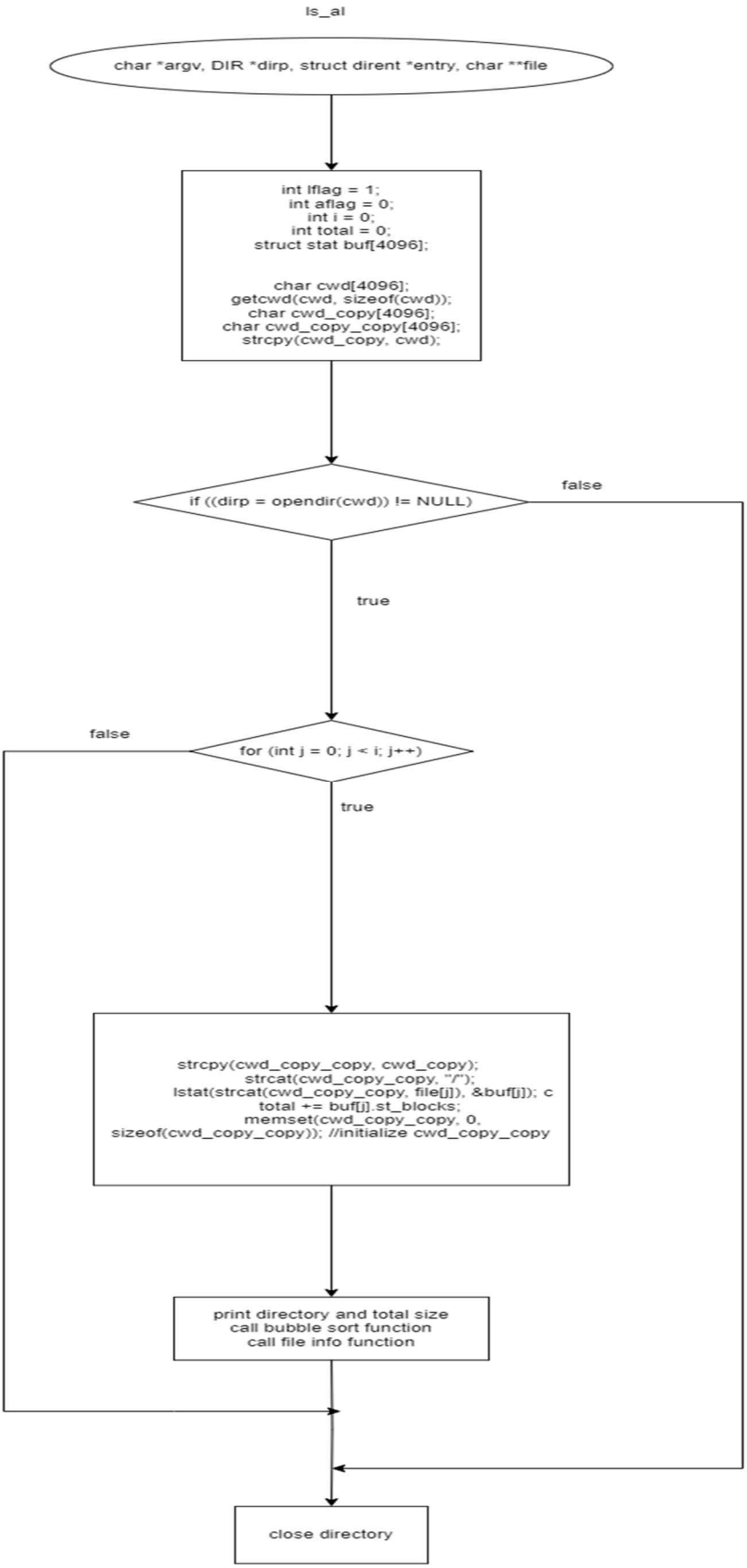
Bubble sort function



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>

#define URL_LEN 256
#define BUFSIZE 1024
#define PORTNO 40000

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

static char *buf

define struct client info
define recent_count=0, total_count=0

define client_info function to store client information
define sigint_handler
define sigint1_handler
define sigusr_handler

Main function
{
    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

maxNchildre=5
make pids array

call child_make function
fork and call child_main function

In child main function

while (1)
{
Set signal handler(sigint)
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

define size of client address
get the socket file descriptor associated with the client

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

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
            }
          print the client's IP address and port number with time
          close client socket descriptor
      }
}
      close socket descriptor
}
define print_recent_clients function
{
      print client information
      alarm(10)
}
define handle_signal
```

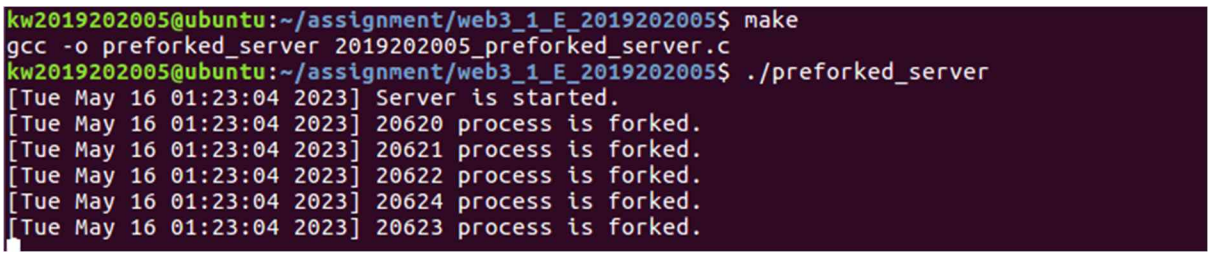
```
{
    call print_recent_clients function
}
define file info function about directory
{
    get current directory

    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+/
```

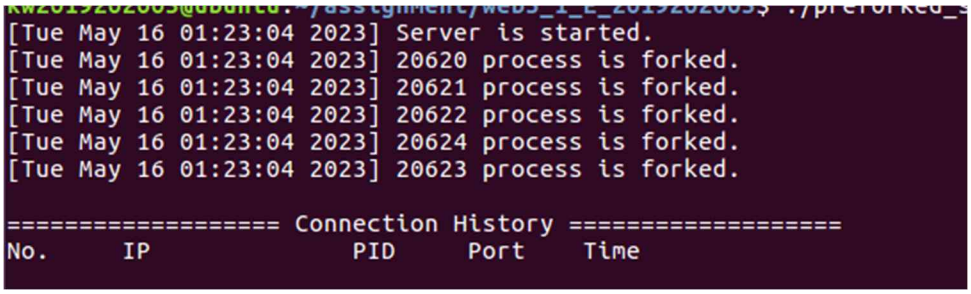
```
            get info about current directory+url+/+file
            get block size
            initialize current directory+url+/+file
        }
        print directory path & total size
        call bubble_sort function
        print file info
    }

    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
}
```

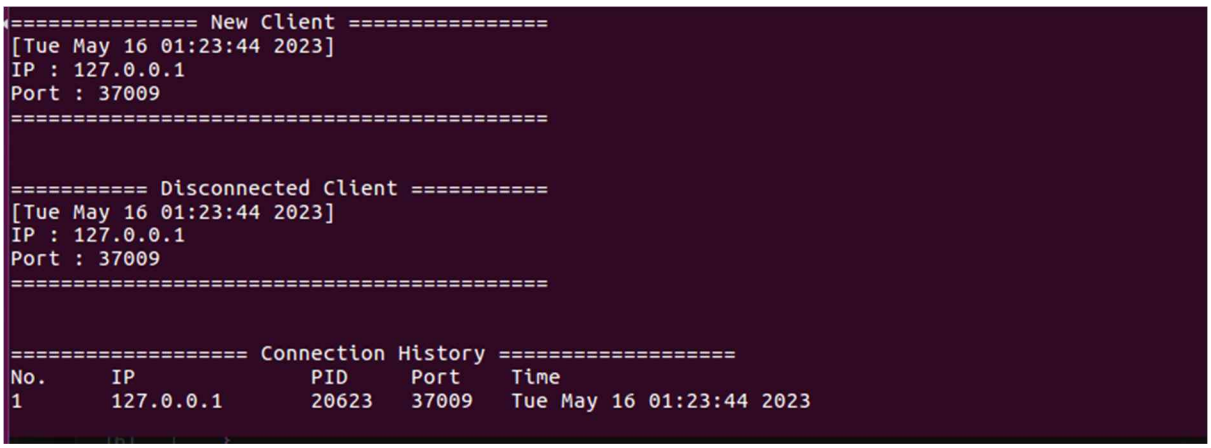
결과화면



다음은 처음에 server program 시작 시 출력되는 화면입니다. 먼저 서버가 시작되었다는 간단한 로그가 시간과 함께 출력되고 child process가 생성되어 PID와 시간과 함께 fork되었다는 문구가 출력됩니다. 총 5번의 fork가 진행되었고 5개의 child process가 생성된 모습을 확인할 수 있습니다.

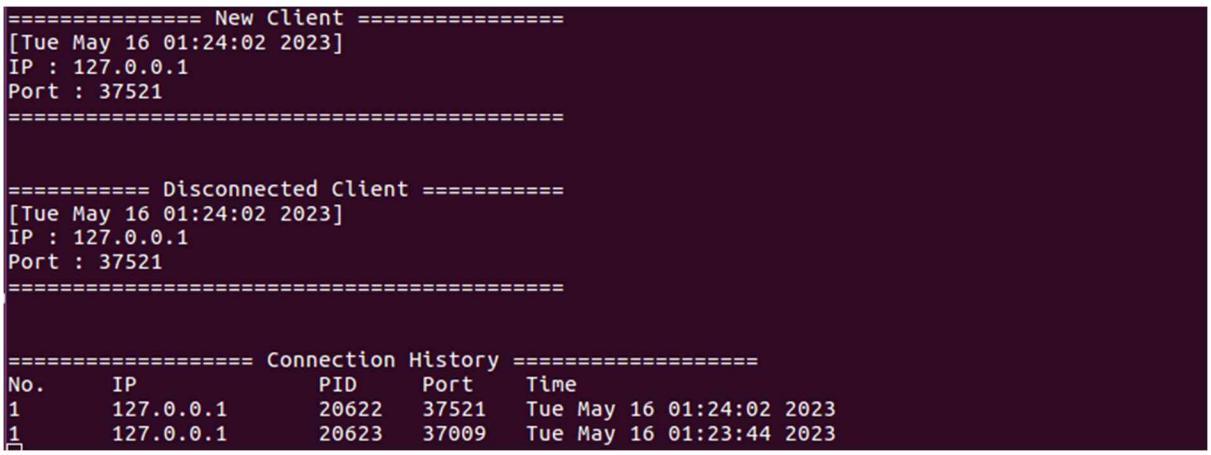


다음은 서버 프로그램을 시작하고 아무것도 하지 않고 10초후 출력되는 화면입니다. 위 문장은 부모 프로세스에서 출력하도록 처리했습니다.

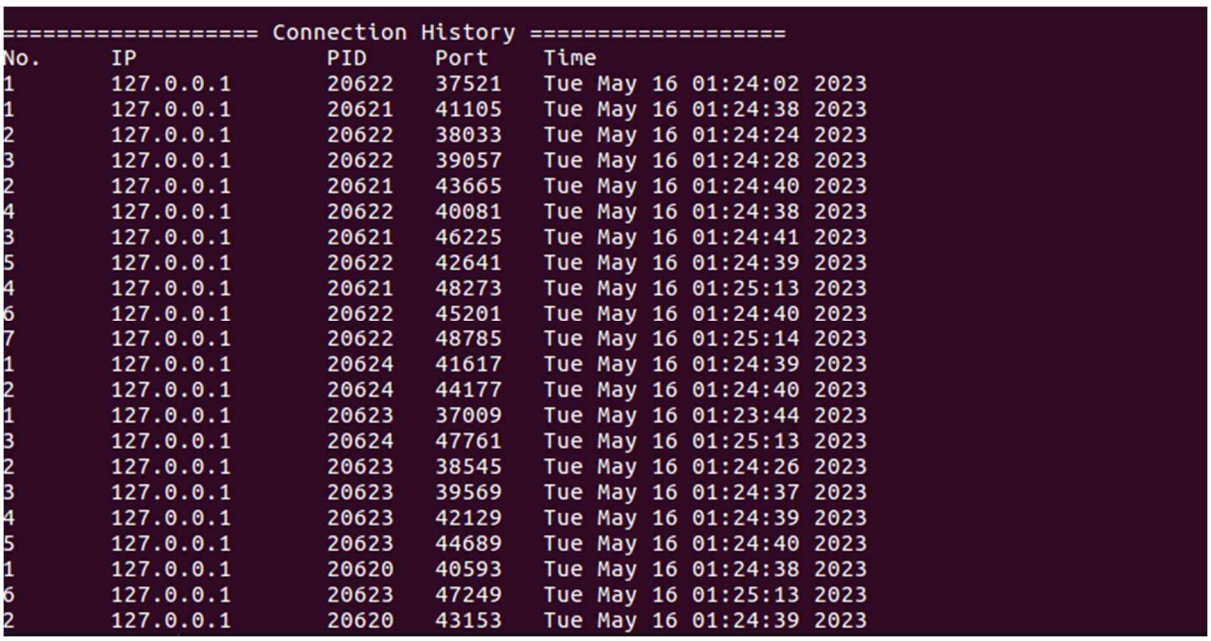


다음은 하나의 클라이언트가 연결되었을 때 화면입니다. 먼저 클라이언트가 연결되고 연결이 끊겼을 때 시간, ip 주소, port번호등 간단한 정보들을

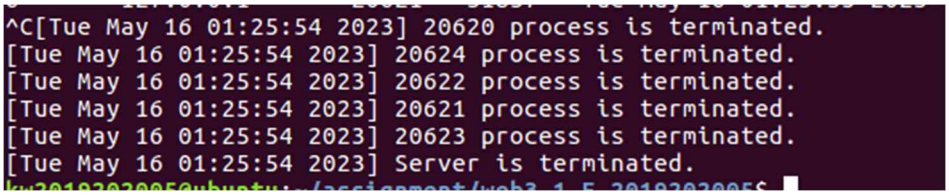
출력했습니다. 그리고 10초에 한 번씩 연결된 클라이언트의 기록을 출력할 때 클라이언트의 정보를 출력하는 부분은 자식 프로세스에서 출력하도록 처리하였습니다.



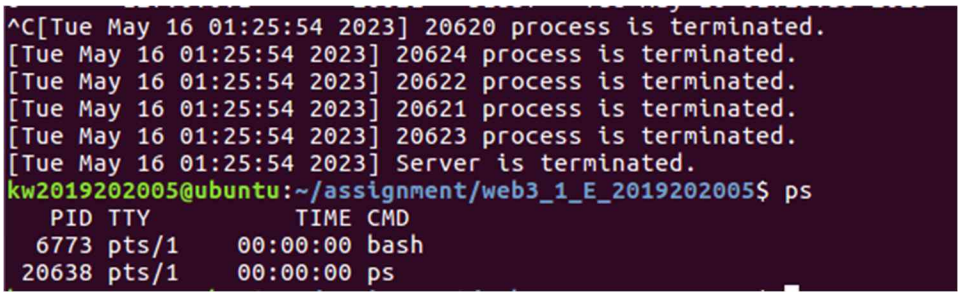
다음은 두개의 클라이언트가 연결되었을 때 출력되는 화면입니다. PID가 서로 다르기 때문에 두 클라이언트는 서로 다른 child 프로세스와 연결된 모습을 확인할 수 있습니다.



다음은 수많은 클라이언트가 연결되었을 때 출력되는 화면입니다.



다음은 child 프로세스가 종료되었을 때 출력되는 화면입니다. SIGINT signal이 발생했을 때 child 프로세스가 모두 종료되는 모습을 확인할 수 있습니다.
Child 프로세스의 PID와 종료된 시간을 출력하도록 처리했습니다.
마지막 줄에는 server종료 시 출력되는 문장입니다.



마지막으로 모든 프로세스 종료 후 ps 명령어를 통해 아직 종료되지 않은 프로세스가 있는 지 확인하였습니다.
모두 정상적으로 종료된 모습을 확인할 수 있습니다.

고찰

먼저 이번 과제는 저번 과제와 거의 비슷했지만 미리 fork를 5번 진행해 child 프로세스를 5개 만들어 놓고 진행한다는 점에서 차이가 있었습니다. 처음에는 굳이 미리 fork하는 이유가 있을까라고 생각이 들었지만 과제를 통해 Child 프로세스를 미리 생성해 둠으로써 request가 도착했을 때 미리 만들어 놓은 프로세스로 작업을 처리할 수 있고 이때 프로세스를 fork하는 비용을 줄일 수 있다는 점을 알았습니다. 그리고 저번 과제에서는 클라이언트의 연결 정보를 출력하는 부분에서 저는 모든 부분을 parent 프로세스에서 처리해주었는데 이번 과제에서는 connection history 및 NO IP PORT PID TIME부분 등 제목 부분을 모두 parent 프로세스에서 출력해야 하고 클라이언트의 자세한 정보 출력은 child 프로세스에서 처리해야 하기 때문에 둘을 출력하는 함수를 따로 만들어 이를 부모, 자식 프로세스에서 따로 호출해 사용했습니다. 그리고 동기화 문제는 이번 과제에서 고려하지 않아 서로 다른 프로세스에서 출력되는 내용이 정리되지 않고

출력되는 모습을 확인할 수 있었습니다.
SIGINT 신호가 발생하면, 모든 프로세스를 완전하게 종료해야 했기 때문에 이를 위해서 signal() 함수를 사용하여 SIGINT 신호를 처리할 핸들러 함수를 등록하고, 그 함수에서 모든 child 프로세스를 종료한 후, 부모 프로세스도 종료했습니다. 이때 실수로 exit()함수를 사용하지 않아 계속 자식 프로세스가 완전히 종료되지 않는 상황이 발생했었습니다. 그래서 계속 다시 실행시킬 때 마다 kill -9를 통해 아직 종료되지 않은 자식 프로세스 5개를 일일이 종료하고 프로그램을 실행시켰는데 그래도 원인을 금방 찾아서 exit()함수 호출을 통해 해결할 수 있었습니다.

Reference

2023년 1학기 시스템프로그래밍 & 시스템 프로그래밍 실습 강의자료
Assignment 3-1
2023년 1학기 시스템 프로그래밍 1학기 강의자료 3. Files and directories
2023년 1학기 시스템 프로그래밍 1학기 강의자료 6. sockets
2023년 1학기 시스템 프로그래밍 실습 11주차 강의자료 Pre-forked Web Server