

시스템프로그래밍실습

assignment 3-2

학 과: 컴퓨터정보공학부

담당교수: 이기훈 교수님

학 번: 2019202050

성 명: 이강현

1. Introduction

이번 과제는 pre-forked child process를 구현하면서 자원을 낭비하지 않고 효율적으로 사용하기 위해 상황에 맞게 프로세스를 관리하는 법을 배운다.

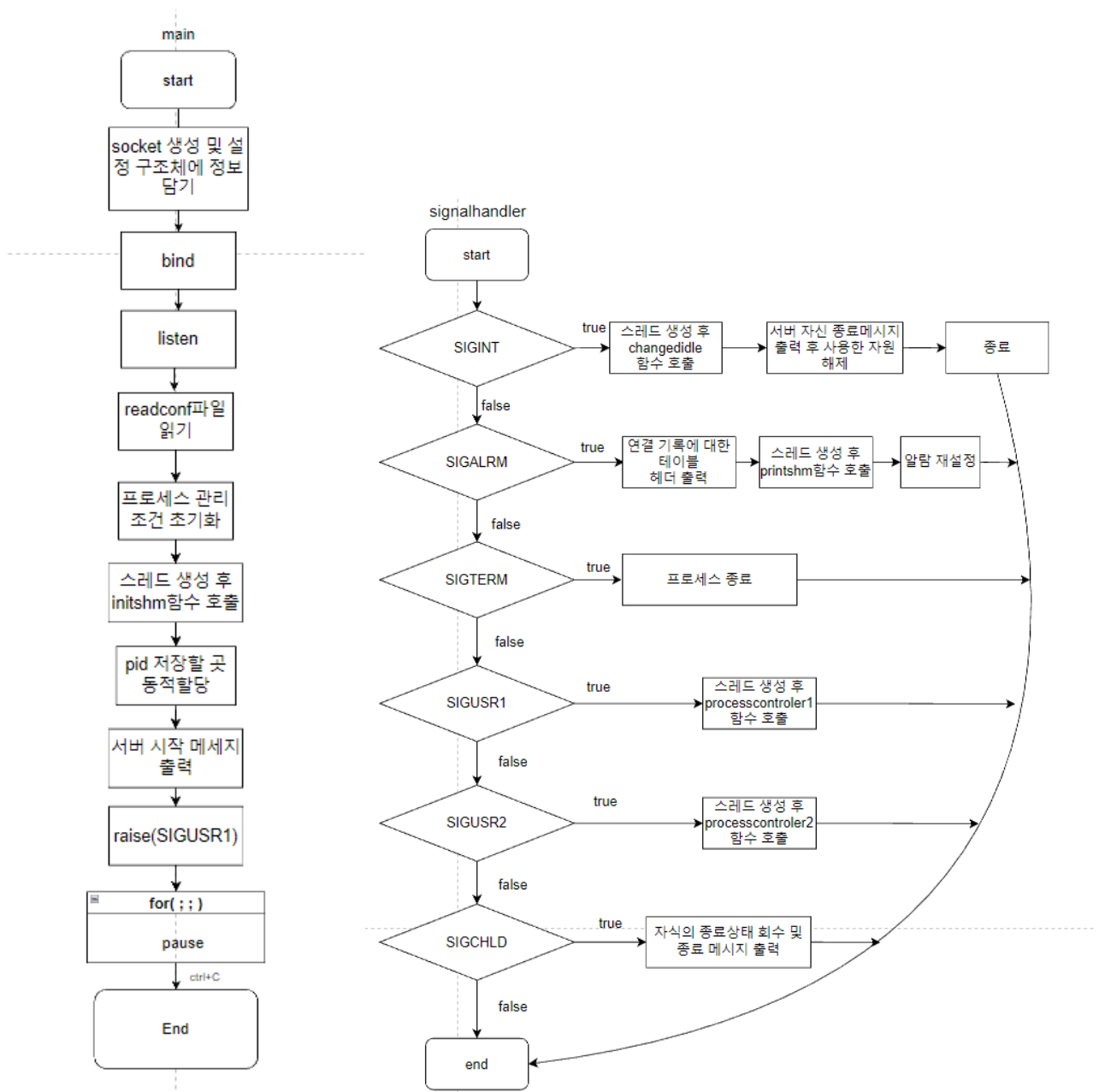
IPC의 개념을 알고 이 중 shared memory에 대해 배우고 이를 활용해본다.

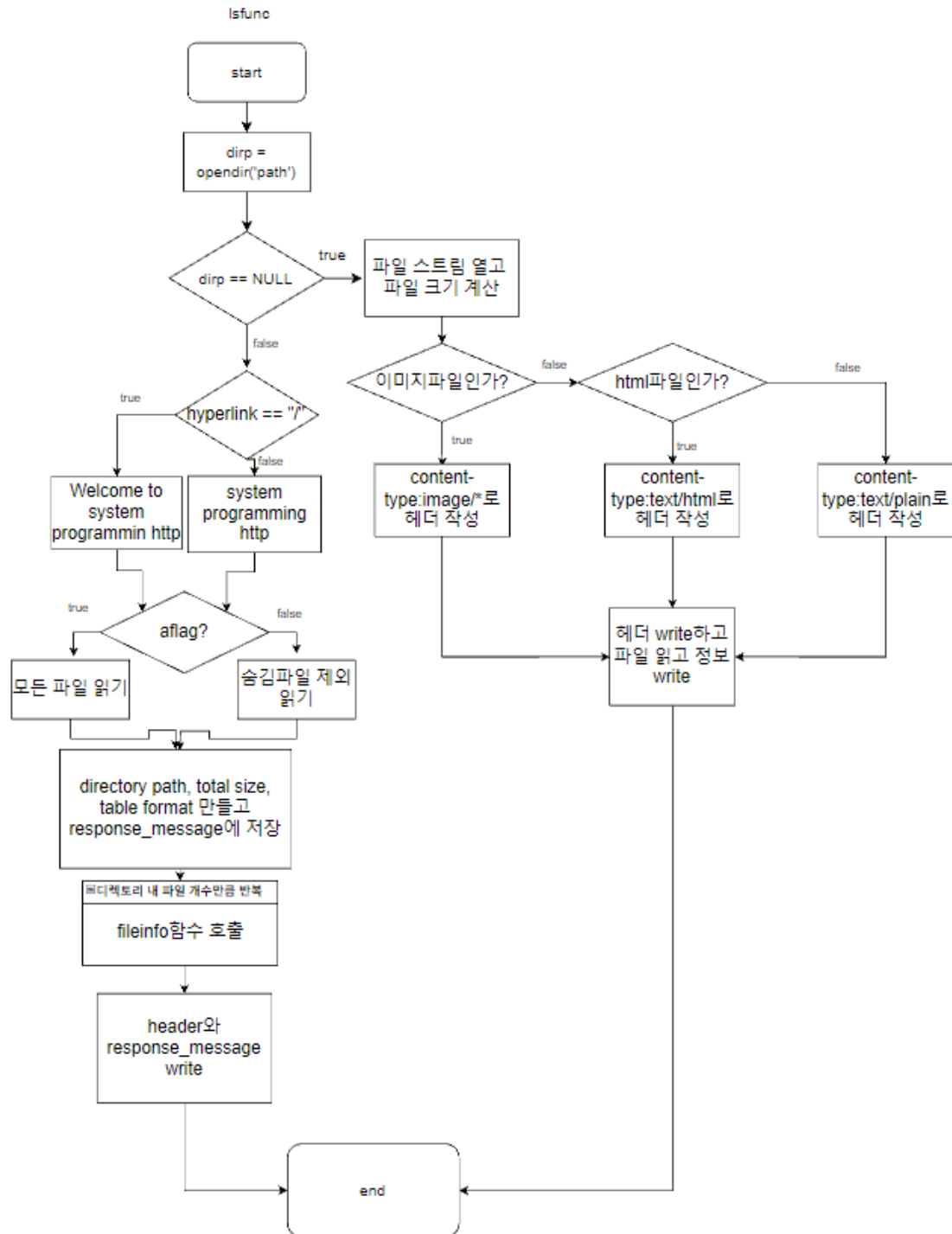
thread의 개념을 알고 프로그램에서 thread를 적절히 활용하여 함수를 실행해본다.

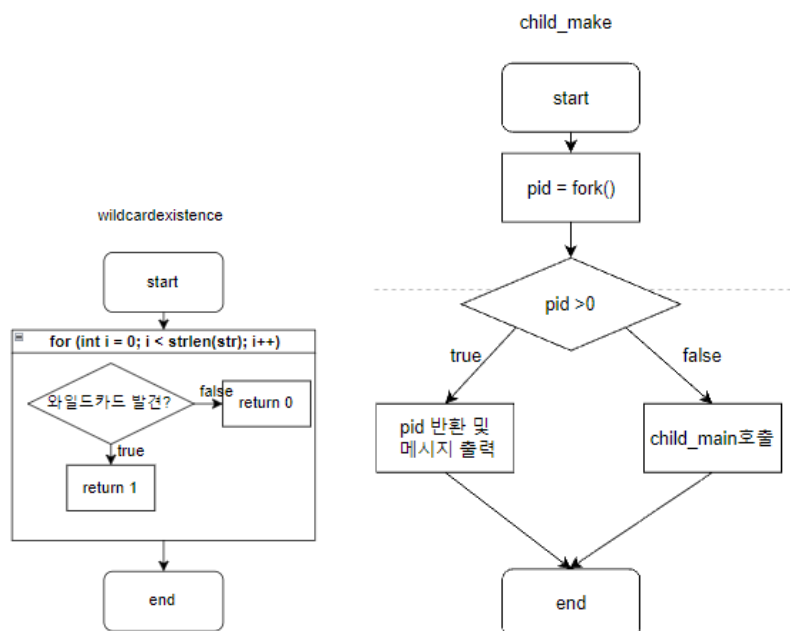
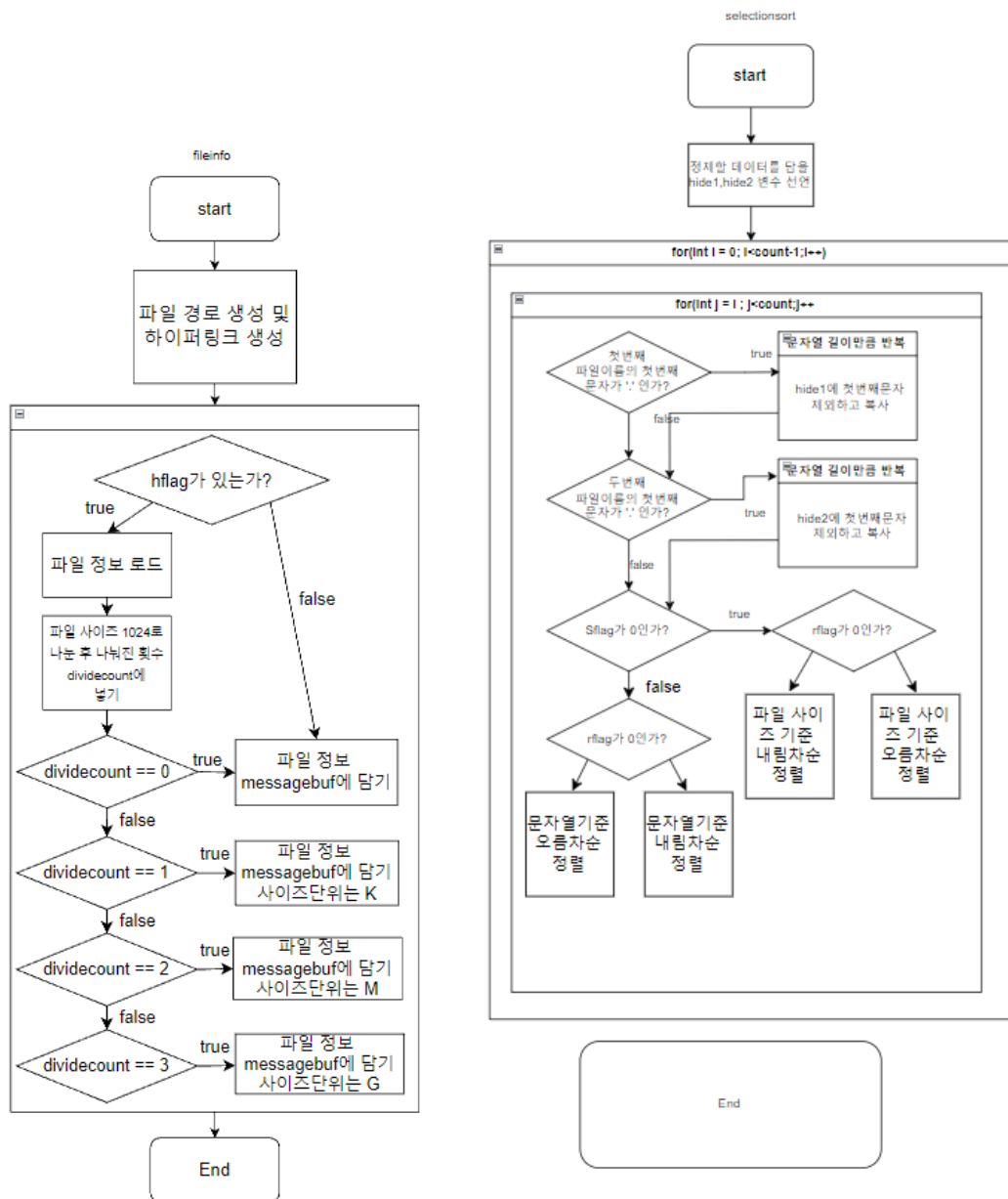
multithread의 개념을 알고 해당 개념의 동작 방식과 장점을 안다.

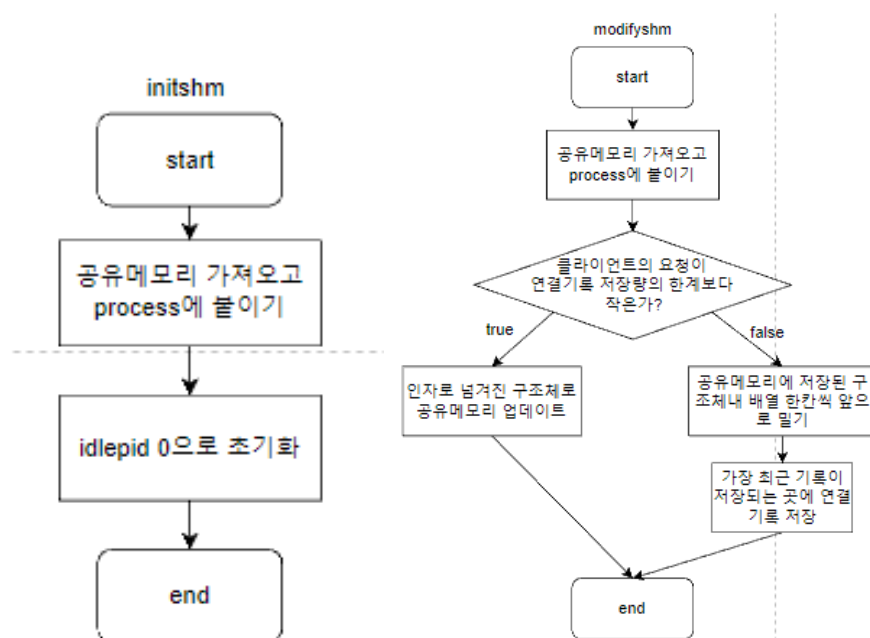
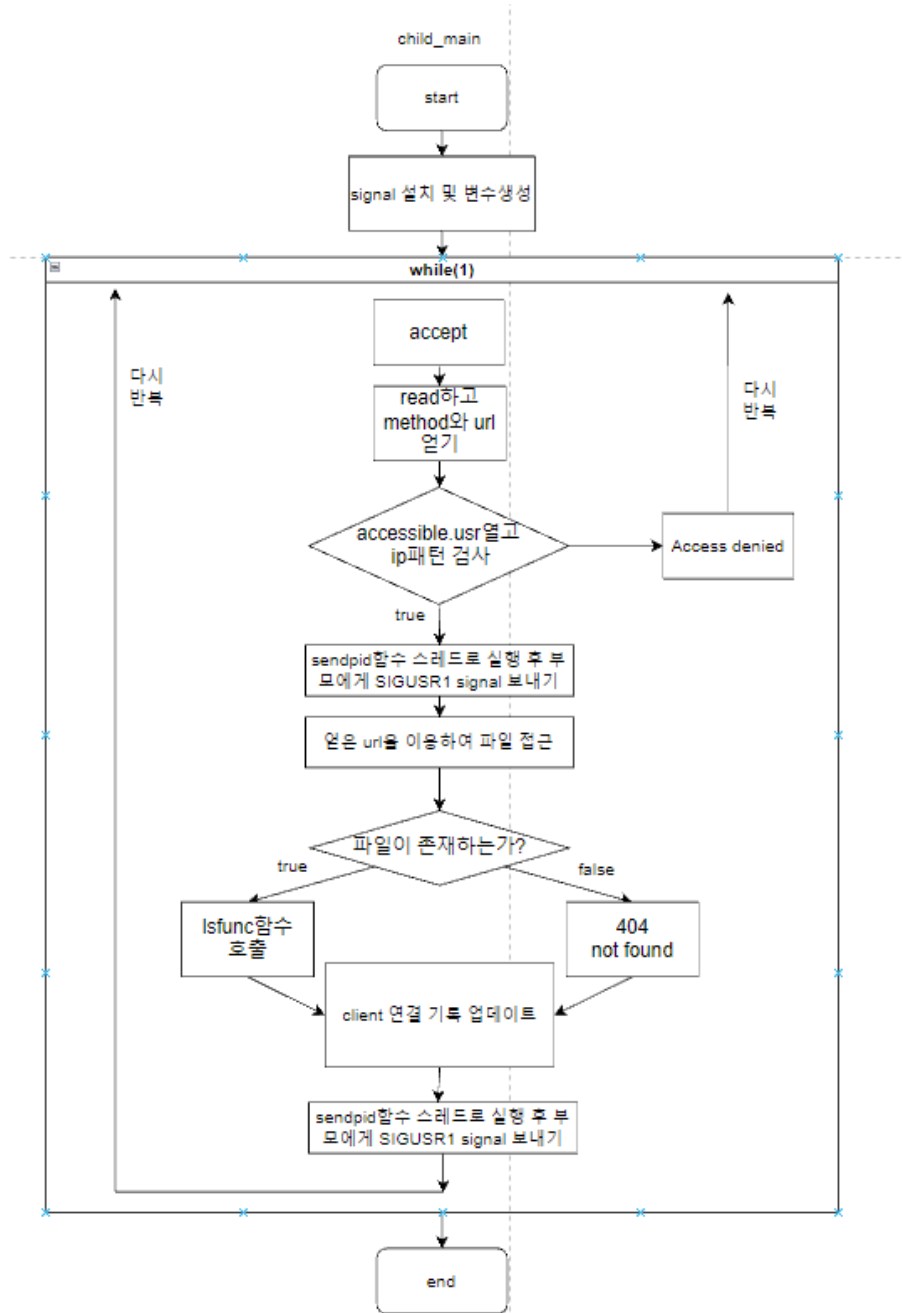
mutex를 이용하여 thread의 동기화 문제를 해결하는 법을 배운다.

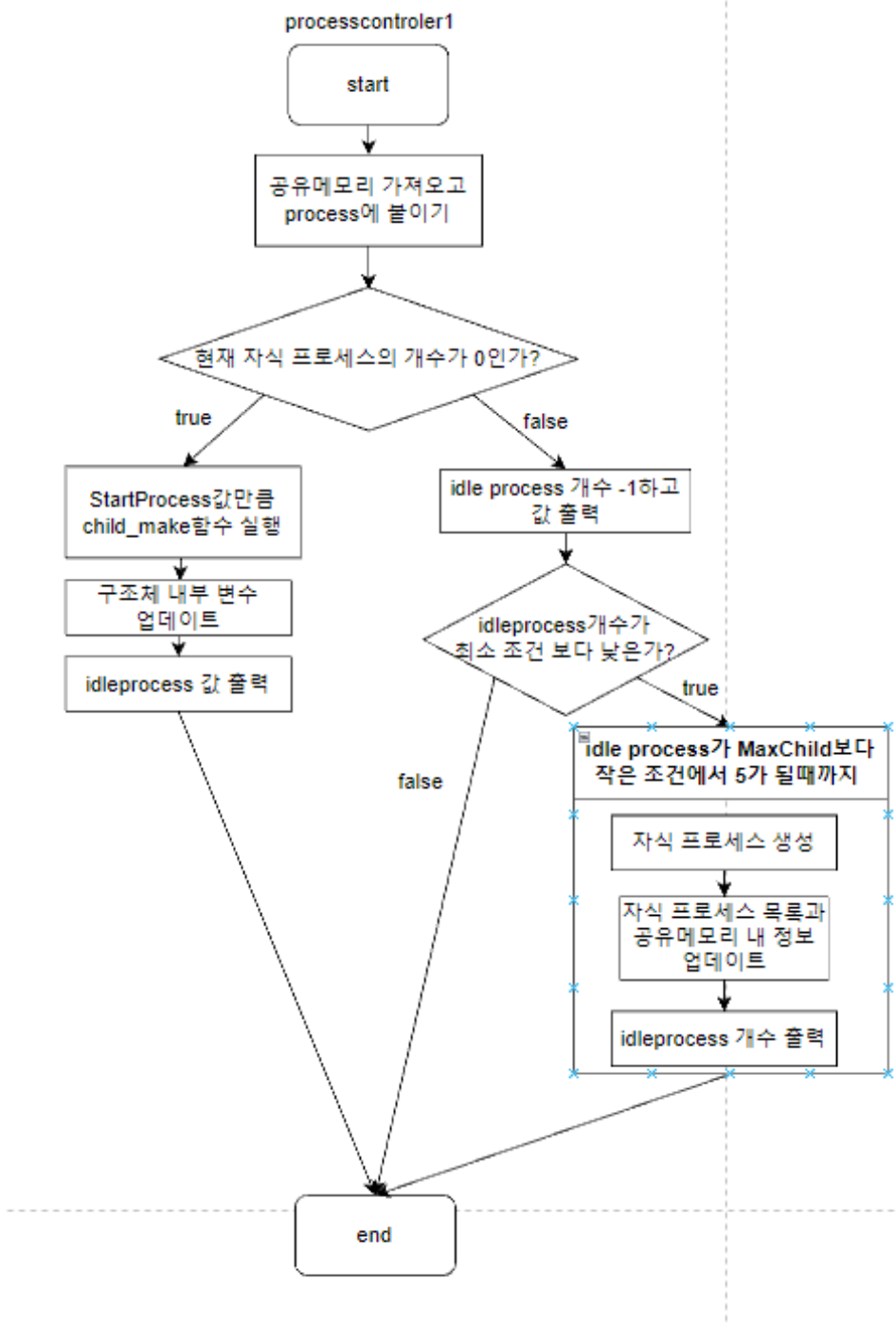
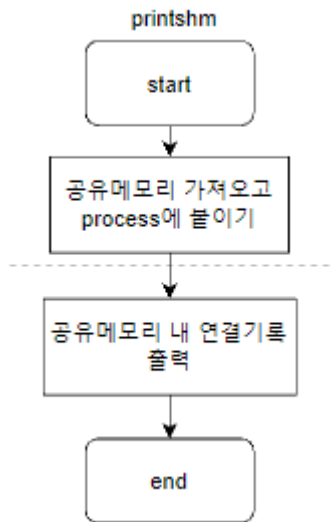
2. Flow chart

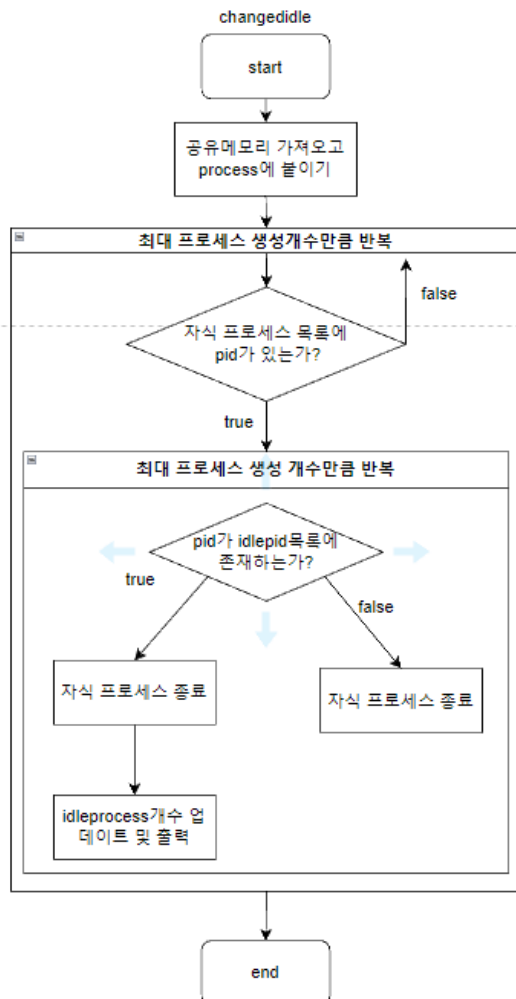
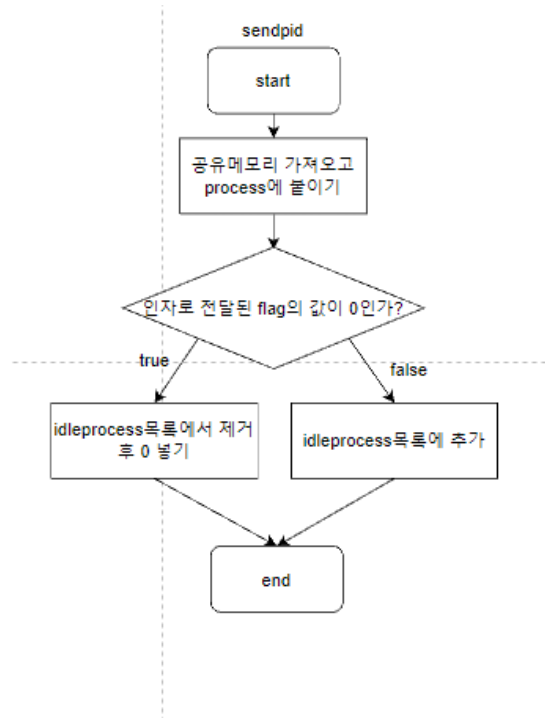
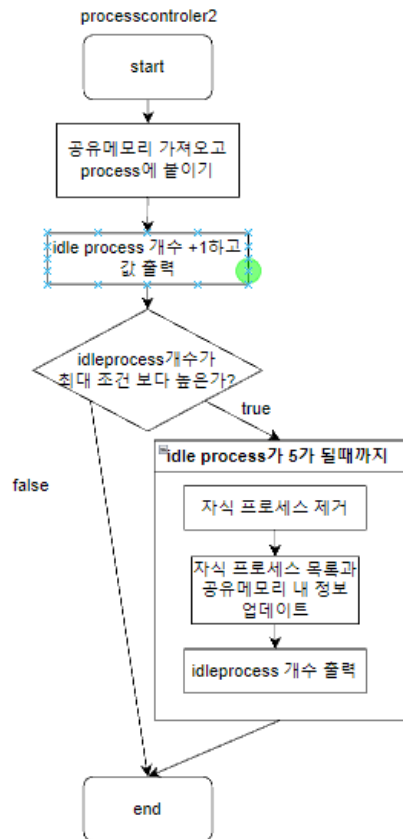












모든 함수들의 흐름도이다. main 함수는 소켓을 생성하여 소켓을 설정을 마치고 bind,listen을 마무리한 후 httpd.conf 파일을 읽은 후 해당 정보로 프로세스를 관리하기위한 조건을 초기화한다. 서버를 시작하기 전 필요한 변수들을 모두 선언 및 초기화 하고 시작되었음을 알리는 메시지를 출력한다. raise함수를 통해 자기자신에게 SIGUSR1신호를 보낸 후 pause함수로 무한루프에 빠진다. 이 후 signal만을 처리할 뿐 아무일도 하지 않는다.

child_make함수는 Maxchilds 값만큼 fork함수를 호출 한 후 부모는 pid를 출력하고 pid반환 자식은 child_main함수를 진행하도록 section을 나눈다.

child_main함수는 자식 프로세스의 일로 인자로 받은 서버 소켓을 이용하여 client와 accept를 진행하고 client의 요청을 수행하고 유지된다. client와 연결되었을 때 공유메모리에 자신이 일하고 있음을 나타내기 위해 sendpid 함수를 실행하고 부모가 변경된 idle process수를 출력하고 child process를 관리하도록 SIGUSR1 신호를 보낸다. client와 연결이 끊기면 자신이 idle process임을 나타내기 위해 sendpid함수를 실행하고 부모에게 SIGUSR2 신호를 보내 child process를 관리하게 한다.

lsfunc 함수는 main 함수에서 넘겨받은 인자에 따라 파일인지 폴더인지를 체크한 후 분기를 나누고 각각 옵션에 따라 요구받은 행동을 취한다.

파일일 경우에는 파일포인터를 선언 후 파일 스트림을 통해 파일을 불러오고 이를 client에게 전송한다.

폴더일 경우에 root directory일 경우에는 l옵션만 적용하여 파일들의 정보를 html형식으로 만들어 보내준다.

그렇지 않을 경우에는 al 옵션은 적용하여 숨김파일에 대한 정보 또한 보낸다.

lsfunc 함수내에서 실행되는 fileaccess함수와 filetype함수는 l옵션에서 제일 먼저 출력되는 파일 타입과 접근권한을 숫자에서 문자데이터로 변환해주는 함수이다.

fileinfo 함수는 lsfunc에서 받은 경로를 통해 하이퍼링크를 생성하고 파일 정보와 함께 버퍼에 담는다.

selectionsort함수는 선택정렬을 구현한 함수로 숨김파일을 구분하지 않고 정렬하기 위해 임시변수를 활용하여 구현하였다.

signalhandler함수는 SIGALRM, SIGINT, SIGTERM, SIGUSR1, SIGUSR2, SIGCHLD 6가지의 signal을 처리한다. SIGALRM은 10초마다 연결기록들을 출력하기 위해 사용되고 신호가 오면 스레드를 생성하여 printshm 함수를 실행하여 공유메모리내에 저장된 연결기록들을 출력한다. SIGINT는 ctrl+c를 입력하였을 때 발생하며 종료하기전에 스레드를 생성하여 changedidle함수를 실행하고 부모스스로 종료메세지를 출력한 후 사용한 공유메모리, mutex,동적할당된 저장소등을 모두 해제한 후 종료한다. SIGTERM신호는 자식이 받을 신호로 신호를 받게되면 프로세스를 종료한다. SIGUSR1신호는 스레드를 생성하여 processcontroler1함수를 실행한다. SIGUSR2신호는 스레드를 생성하여 processcontroler2함수를 실행한다. SIGCHLD신호는 회수할 자식이 존재하면 자식의 pid와 회수한 시간을 출력한다. 이는 회수할 자식이 더 이상 없을때까지 반복된다.

initshm 함수는 서버 시작시 다수의 프로세스가 함께 사용할 공유메모리를 초기화하는 함수이다. 공유메모리를 생성하고 메모리내에 구조체를 저장한다.

modifyshm 함수는 자식 프로세스가 클라이언트와의 연결 이후 연결되었던 기록을 공유메모리에 반영하기 위한 함수이다.

printshm 함수는 공유메모리내의 연결기록들을 출력하기 위한 함수이다.

processcontroler1,processcontroler2함수는 공유메모리내에 저장된 idle process의 개수가 변경되는 상황이거나 변경되어 프로세스를 종료하거나 생성해야 하는 조건을 검사하는 함수이다. 조건에 따라 프로세스를 생성 및 종료하고 변경된 idle process의 개수를 출력한다.

sendpid함수는 자식 프로세스가 부모 프로세스에게 자신이 idle process 혹은 working process라는 것을 알리기 위해 이러한 정보를 공유메모리내에 기록하는 함수이다.

changedidle함수는 프로그램의 종료시 사용되는 함수로 프로그램의 종료시점을 알 수 없기에 클라이언트와 연결중인 process의 종료 출력과 idle process의 종료 출력을 다르게 하기 위해 정의된 함수이다.

3. Pseudo Code

```
selectionsort
파일 사이즈 변수선언
파일 경로 만들기 위한 변수선언
문자열 정제 후 저장할 변수 선언

만약 입력 경로가 NULL이 아니라면
    for(파일 개수만큼)
        경로 생성하여 파일 사이즈를 얻어내고 변수에 저장
for(파일개수 -1만큼)
    for(파일개수 -1만큼)
        만약 파일의 앞 문자가 '.'이라면
            앞문자 제외하여 변수에 저장
        아니라면
            그대로 저장

    소문자 모두 대문자로 변경

    if(!Sflag)
        if(!rflag)
            문자열 기준 오름차순 정렬
        else
            문자열 기준 내림차순 정렬
    else
        if(!rflag)
            파일사이즈 기준 내림차순 정렬
            사이즈가 같다면 문자열 기준 정렬
        else
            문자열 기준 오름차순 정렬
            사이즈가 같다면 문자열 기준 정렬
```

```
filetype
입력받은 파일의 mode가 디렉토리라면
    입력받은 문자열의 맨 앞 글자를 d로 변경
입력받은 파일의 mode가 심볼릭 링크라면
    입력받은 문자열의 맨 앞 글자를 l로 변경
|
fileaccess
입력받은 파일의 mode의 user access가 read가 가능하다면
    입력받은 문자열의 2번째를 r로 변경
입력받은 파일의 mode의 user access가 write가 가능하다면
    입력받은 문자열의 3번째를 w로 변경
입력받은 파일의 mode의 user access가 execution이 가능하다면
    입력받은 문자열의 4번째를 x로 변경

입력받은 파일의 mode의 group access가 read가 가능하다면
    입력받은 문자열의 5번째를 r로 변경
입력받은 파일의 mode의 group access가 write가 가능하다면
    입력받은 문자열의 6번째를 w로 변경
입력받은 파일의 mode의 group access가 execution이 가능하다면
    입력받은 문자열의 7번째를 x로 변경

입력받은 파일의 mode의 other access가 read가 가능하다면
    입력받은 문자열의 8번째를 r로 변경
입력받은 파일의 mode의 other access가 write가 가능하다면
    입력받은 문자열의 9번째를 w로 변경
입력받은 파일의 mode의 other access가 execution이 가능하다면
    입력받은 문자열의 10번째를 x로 변경
```

```
lsfunc
입력받은 경로로 디렉토리 열기
디렉토리가 null이라면
    파일 스트림 열고 파일 크기 저장
    if(이미지파일)
        이미지파일 전용 헤더 생성
    elseif(html파일)
        html파일 전용 헤더 생성
    else
        일반파일 전용 헤더 생성
헤더 파일 write
파일 바이트단위로 읽고 client에게 보내기
종료
```

```
if(!aflag)
    숨김파일 제외 파일 저장
else
    모두 읽기

if(!lflag)
    정렬 하고 파일들 이름만 출력
else
    정렬하고 파일들 정보를 담은 테이블 생성
    fileinfo 함수를 파일 개수만큼 호출
    헤더파일 write하고 정제된 파일 정보 html형식으로 client에게 보내기
종료
```

```
,
signalhandler
signal이 발생시 실행
if(ctrl+c 신호 발생시 SIGINT)
    스레드를 생성하여 changedidle함수를 실행
    부모 스스로 종료메시지를 출력
    공유메모리 제거, mutex destroy, 동적할당 free
    부모 종료
else if(signal이 지정한 알람시간이 다 되었음을 알림 SIGALRM)
    연결 기록 제목을 출력
    스레드를 생성하여 printshm함수 실행
    알람 재설정
else if(SIGUSR1신호 일때)
    스레드를 생성하여 processcontroler1함수 실행
else if(SIGUSR2신호 일때)
    스레드를 생성하여 processcontroler2함수 실행
else if(SIGTERM신호 일때)
    프로세스 종료
else if(SIGCHLD신호 일때)
    회수할 자식의 종료상태가 존재하면 회수한 자식 프로세스 pid와 회수한 시간 출력 없을때까지 반복
```

child_main

필요한 구조체, 변수 선언 및 초기화

while(1)

client accept

method와 url 토큰으로 얻기

접근가능한 ip주소가 저장된 파일 열고 현재 클라이언트 ip와 비교

일치하면 accessflag = 1, 아니면 0

accessflag == 0일때

접근 불가 메시지를 담은 페이지를 보내고 continue

accessflag == 1일때

sendpid함수의 인자로 보낼 자신의 pid와 flag를 구조체에 넣기

스레드를 생성하여 sendpid함수를 실행

부모에게 SIGUSR1신호 보내기

현재디렉토리를 루트 디렉토리로 간주하고 url이어붙여 경로 생성

해당 경로에 파일이 존재하지 않는다면 에러전용 헤더와 메시지를 보내주기

존재한다면 lsfunc함수 호출 후 client disconnect

sendpid함수의 인자로 보낼 자신의 pid와 flag를 구조체에 넣기

스레드를 생성하여 sendpid함수를 실행

부모에게 SIGUSR2신호 보내기

child_make

fork실행

if(부모라면)

자식 프로세스의 pid와 fork된 시간 출력

if(자식이라면)

child_main함수 실행

initshm

공유메모리 생성 및 process에 attach

구조체를 공유메모리에 저장 후 변수 초기화

modifyshm

존재하는 공유메모리 가져오기 및 process에 attach

if(저장된 history수가 MaxHistory보다 작을때)

인자로 전달된 구조체의 정보를 기반으로 공유메모리 정보 업데이트

else

공유메모리내 구조체 속 배열들 한칸씩 앞으로 밀기

배열의 맨 마지막 인덱스에 인자로 전달된 정보 넣어 공유메모리 업데이트

printshm

존재하는 공유메모리 가져오기 및 process에 attach

공유메모리내의 연결기록 정보 출력

```

processcontroller1
존재하는 공유메모리 가져오기 및 process에 attach
if(현재 자식프로세스가 0개라면)
    StartProcess값 만큼 자식 프로세스 생성
    구조체 내부 정보 업데이트
    idle process수 출력
else
    idle process--
    idle process수 출력
    if(idle process개수가 MinIdleNum값 보다 작다면)
        while(idle process 개수가 5가 되거나 전체 자식 프로세스 개수가 Maxchilds값보다 작을때까지)
            자식 프로세스 생성 후 전체 pid저장 배열 업데이트 및 공유메모리내 idle process 목록 업데이트
            변경된 idle process 수 출력

processcontroller2
존재하는 공유메모리 가져오기 및 process에 attach
idle process++
idle process수 출력
if(idle process개수가 MaxIdleNum값 보다 크다면)
    while(idle process 개수가 5가 될때까지)
        자식 프로세스 제거 후 전체 pid저장 배열 업데이트 및 공유메모리내 idle process 목록 업데이트
        변경된 idle process 수 출력

```

```

sendpid
존재하는 공유메모리 가져오기 및 process에 attach
if(인자로 전달된 flag가 0일때)
    공유메모리 내 idle process 목록에서 인자로 전달된 pid를 찾아 값을 0으로 바꿈
else
    공유메모리 내 idle process 목록에서 0을 찾아 값을 인자로 전달된 pid로 바꿈

changedidle
존재하는 공유메모리 가져오기 및 process에 attach
for(Maxchilds값 만큼 반복)
    if(전체 자식 프로세스 목록에서 값이 0이 아닌 pid를 찾으면)
        for(Maxchild값 만큼 반복)
            if(전체 자식 프로세스 목록에서 찾은 값과 같은 공유메모리내의 idle process값을 찾으면)
                자식을 종료시키고 idle process 개수를 업데이트 및 출력
            else
                자식을 종료시킴

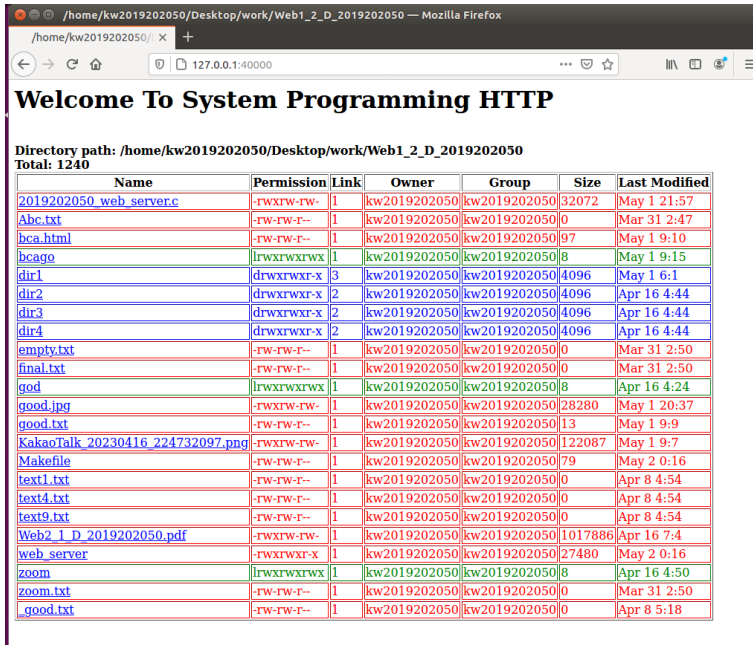
```

```

main
소켓 생성
소켓 설정
bind
listen
httpd.conf파일을 열고 읽기
읽는 내용으로 프로세스 관리 조건 변수 초기화
스레드 생성 후 initshm함수 실행
전체 자식프로세스 목록을 담기 위해 동적할당 및 초기화
서버 시작 메세지 출력
raise함수를 통해 SIGUSR1 신호를 자신에게 보내기
무한 루프에 빠져 pause함수호출

```

4. 결과화면



Directory path: /home/kw2019202050/Desktop/work/Web1_2_D_2019202050
Total: 1240

Name	Permission	Link	Owner	Group	Size	Last Modified
2019202050_web_server.c	-rwxrwxrwx	1	kw2019202050	kw2019202050	32072	May 1 21:57
Abc.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:47
bca.html	-rw-rw-r--	1	kw2019202050	kw2019202050	97	May 1 9:10
bcago	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	May 1 9:15
dir1	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1
dir2	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
dir3	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
dir4	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
empty.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
final.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
god	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	Apr 16 4:24
good.jpg	-rwxrwxrwx	1	kw2019202050	kw2019202050	28280	May 1 20:37
good.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	13	May 1 9:9
KakaoTalk_20230416_224732097.png	-rwxrwxrwx	1	kw2019202050	kw2019202050	122087	May 1 9:7
Makefile	-rw-rw-r--	1	kw2019202050	kw2019202050	79	May 2 0:16
text1.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
text4.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
text9.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
Web2_1_D_2019202050.pdf	-rwxrwxrwx	1	kw2019202050	kw2019202050	1017886	Apr 16 7:4
web_server	-rwxrwxr-x	1	kw2019202050	kw2019202050	27480	May 2 0:16
zoom	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	Apr 16 4:50
zoom.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
good.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 5:18

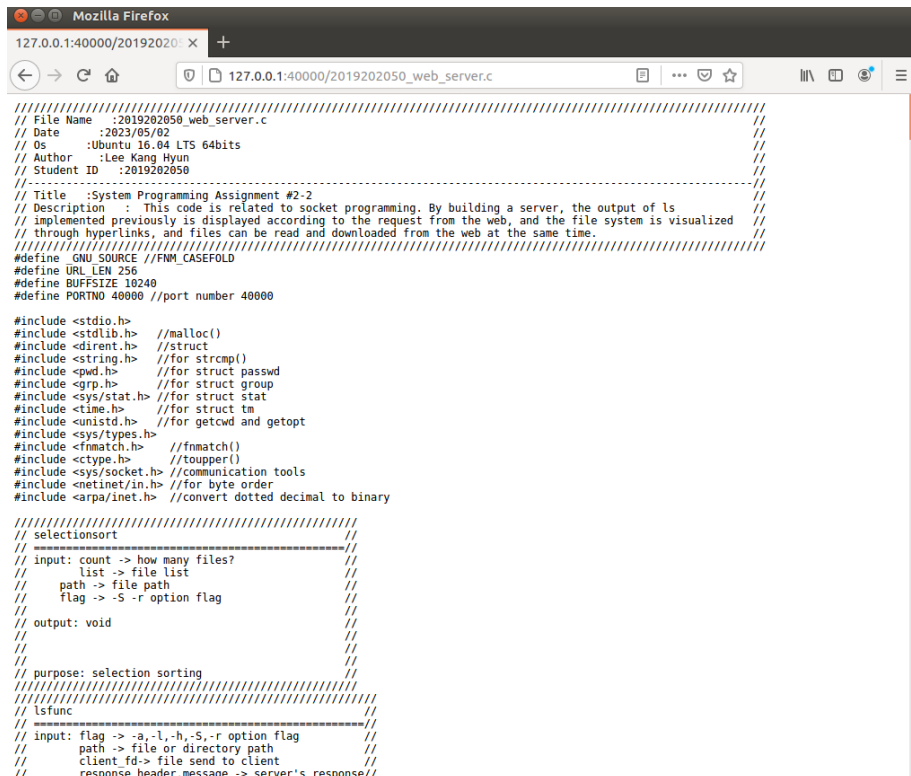
루트 디렉토리의 모습이다. 상위 디렉토리로 접근하지 못하도록 -i 옵션만 적용된 상태이다.



Directory path: /home/kw2019202050/Desktop/work/Web1_2_D_2019202050/dir1
Total: 12

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1
..	drwxrwxr-x	6	kw2019202050	kw2019202050	4096	May 2 0:16
hi	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1

하위 폴더로 하이퍼링크를 통해 들어간 모습이다. 하위 디렉토리이므로 상위 디렉토리로의 접근이 가능하게끔 -al 옵션이 적용되었다.

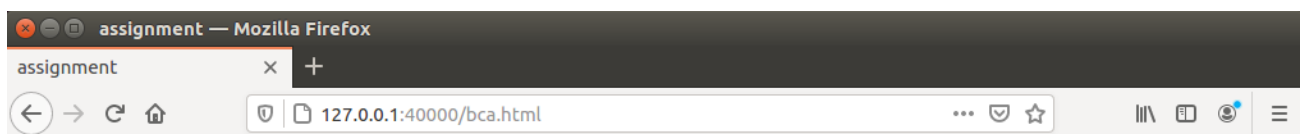


```
// File Name : 2019202050_web_server.c
// Date : 2023/05/02
// Os : Ubuntu 16.04 LTS 64bits
// Author : Lee Kang Hyun
// Student ID : 2019202050
// Title : System Programming Assignment #2-2
// Description : This code is related to socket programming. By building a server, the output of ls
// implemented previously is displayed according to the request from the web, and the file system is visualized
// through hyperlinks, and files can be read and downloaded from the web at the same time.
//
// =====
#define GNU_SOURCE //FNM_CASEFOLD
#define URL_LEN 256
#define BUFSIZE 10240
#define PORTNO 40000 //port number 40000

#include <stdio.h>
#include <stdlib.h> //malloc()
#include <dirent.h> //struct
#include <string.h> //for strcmp()
#include <pwd.h> //for struct passwd
#include <grp.h> //for struct group
#include <sys/stat.h> //for struct stat
#include <time.h> //for struct tm
#include <unistd.h> //for getcwd and getopt
#include <sys/types.h>
#include <fnmatch.h> //fnmatch()
#include <ctype.h> //toupper()
#include <sys/socket.h> //communication tools
#include <netinet/in.h> //for byte order
#include <arpa/inet.h> //convert dotted decimal to binary

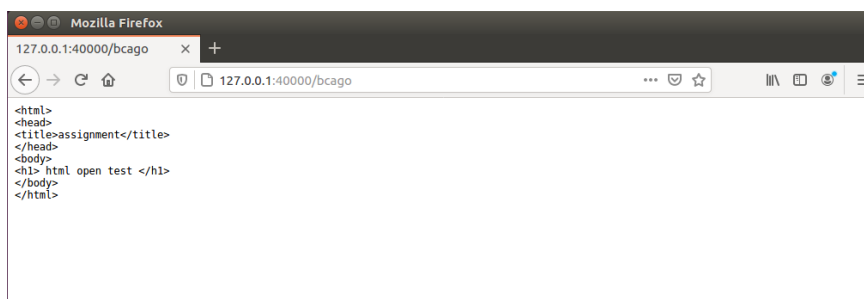
// =====
// selection sort
//
// input: count -> how many files?
// list -> file list
// path -> file path
// flag -> -S -r option flag
//
// output: void
//
// purpose: selection sorting
// =====
// lsfunc
//
// input: flag -> -a,-l,-h,-S,-r option flag
// path -> file or directory path
// client_fd -> file send to client
// response_header,message -> server's response//
```

source code를 하이퍼링크를 통해 열어본 모습이다. 서버에서 보낸 파일의 정보를 읽을 수 있다.

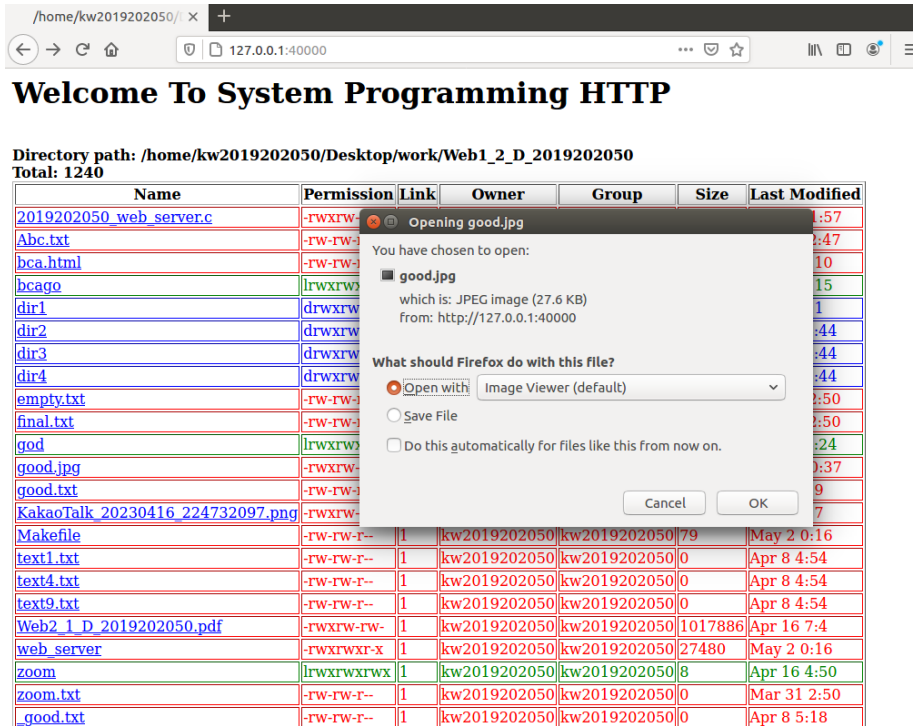


html open test

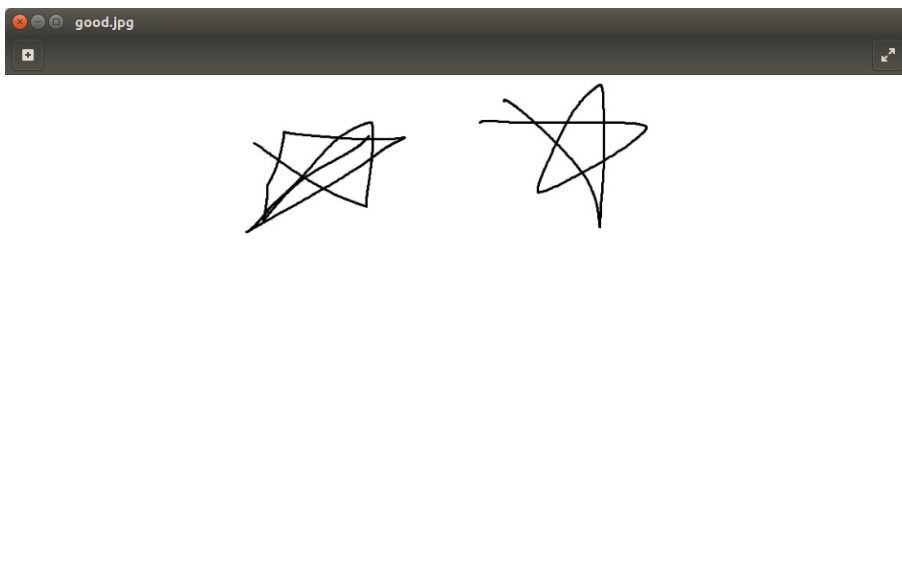
이번엔 html파일을 열어본 모습이다.



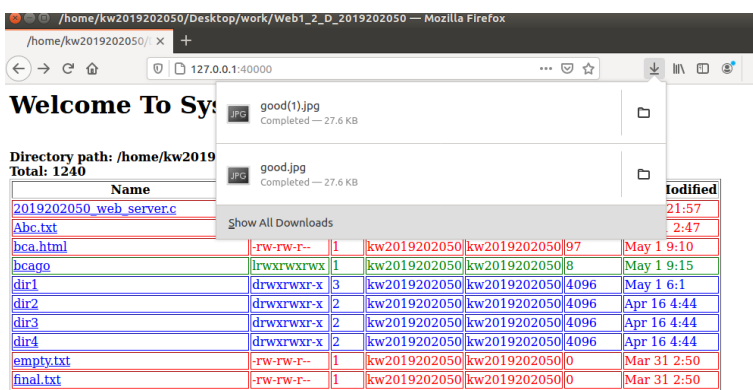
위의 html파일에 심볼릭 링크로 연결된 파일을 열었을 때 파일 자체에 적힌 코드를 확인할 수 있다.



이미지 파일을 클릭했을 때 확인할 수 있는 결과이다. open with를 클릭하면



위와 같이 사진을 볼 수 있고



save를 하게 되면 파일이 다운된 것을 알 수 있다.

Welcome To System Programming HTTP

Directory path: /home/kw2019202050/Desktop/work/Web1_2_D_2019202050
Total: 1240

Name	Permission	Link	Owner	Group	Size	Last Modified
2019202050_web_server.c	-rwxrwxrwx					Mar 31 2:57
Abc.txt	-rwxrwxrwx					Mar 31 2:47
bca.html	-rwxrwxrwx					Mar 31 2:10
bcago	lrwxrwxrwx					Mar 31 2:15
dir1	drwxrwxrwx					Mar 31 2:1
dir2	drwxrwxrwx					Mar 31 2:44
dir3	drwxrwxrwx					Mar 31 2:44
dir4	drwxrwxrwx					Mar 31 2:44
empty.txt	-rwxrwxrwx					Mar 31 2:50
final.txt	-rwxrwxrwx					Mar 31 2:50
god	lrwxrwxrwx					Apr 16 4:24
good.jpg	-rwxrwxrwx				28280	May 1 20:37
good.txt	-rwxrwxrwx				13	May 1 9:9
KakaoTalk_20230416_224732097.png	-rwxrwxrwx				122087	May 1 9:7
Makefile	-rwxrwxrwx				79	May 2 0:16
text1.txt	-rwxrwxrwx				0	Apr 8 4:54
text4.txt	-rwxrwxrwx				0	Apr 8 4:54

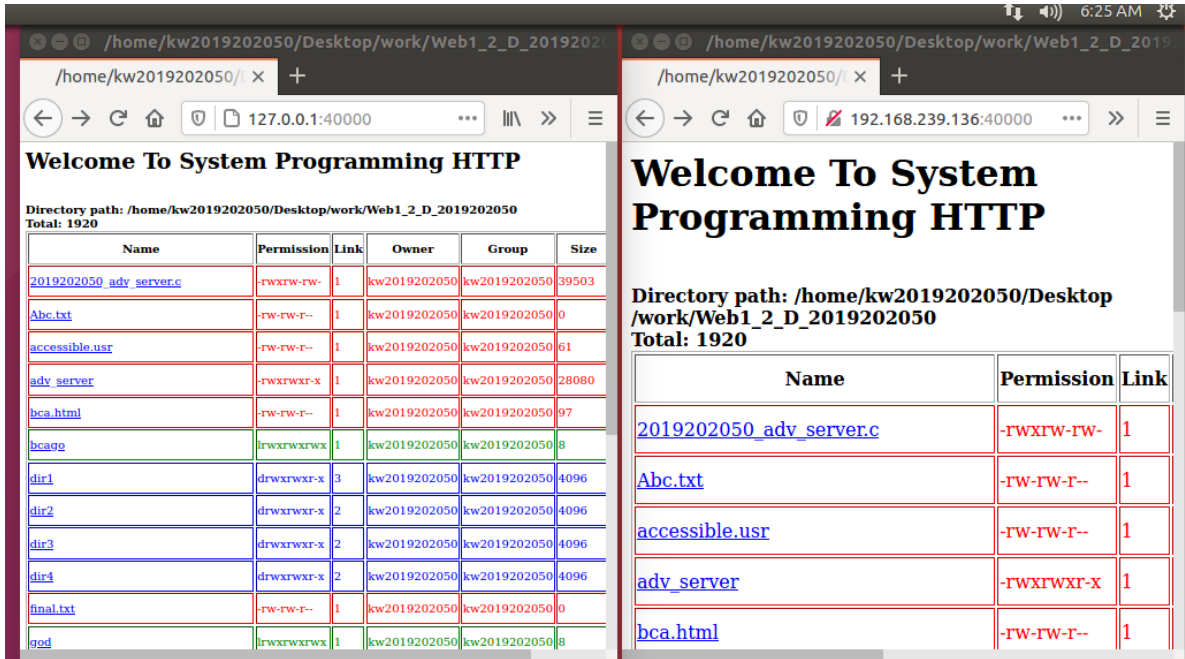
실행파일 또한 파일을 다운할 수 있다.

```
kw2019202050@ubuntu: ~
kw2019202050@ubuntu:~$ ifconfig
ens33      Link encap:Ethernet  HWaddr 00:0c:29:b4:82:98
            inet addr:192.168.239.136  Bcast:192.168.239.255  Mask:255.255.255.0
            inet6 addr: fe80::4438:d69f:cd3e:702a/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:9057 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6524 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:6508179 (6.5 MB)  TX bytes:1757660 (1.7 MB)

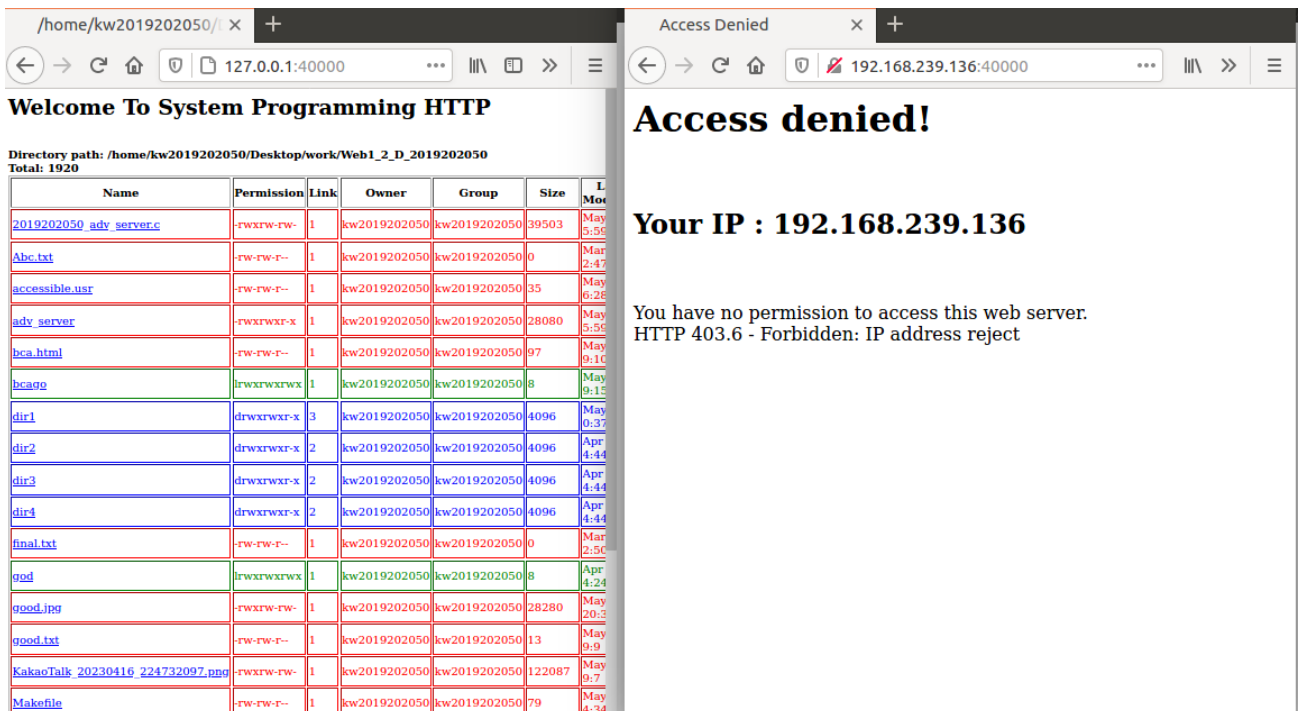
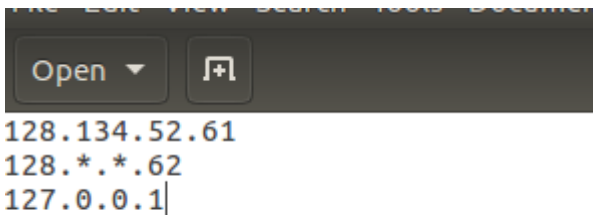
lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:3208 errors:0 dropped:0 overruns:0 frame:0
            TX packets:3208 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:839415 (839.4 KB)  TX bytes:839415 (839.4 KB)
```

위 사진을 보면 local과 vmware에서 직접 이더넷과 연결된 두 가지의 ip를 확인할 수 있다. 따라서 accessible.user를 통한 접근가능 ip에 대한 검증을 두 가지 ip를 통해 진행한다.

```
accessible.user (~/Desktop/work/Web1_2_D_2019202050) - gedit
128.134.52.61
192.168.*.*
128.*.*.62
127.0.0.1
192.168.239.*
```



먼저 두 가지 ip를 모두 파일에 입력한 경우 두 경우 다 결과페이지를 보여준다.



192.168.239.136 ip는 접근가능한 ip목록에서 사라졌으므로 접근이 불가능하다는 것을 알린다.

```
[httpd.conf]
MaxChilds: 10
MaxIdleNum: 6
MinIdleNum: 4
StartProcess: 5
MaxHistory: 10
```

<- 서버의 프로세스와 연결기록을 관리하는 조건이 되는 변수들이다.

```
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ ./ipc_server
[Mon May 22 10:15:22 2023] Server is started.
[Mon May 22 10:15:22 2023] 4278 process is forked.
[Mon May 22 10:15:22 2023] IdleProcessCount : 1
[Mon May 22 10:15:22 2023] 4279 process is forked.
[Mon May 22 10:15:22 2023] IdleProcessCount : 2
[Mon May 22 10:15:22 2023] 4280 process is forked.
[Mon May 22 10:15:22 2023] IdleProcessCount : 3
[Mon May 22 10:15:22 2023] 4281 process is forked.
[Mon May 22 10:15:22 2023] IdleProcessCount : 4
[Mon May 22 10:15:22 2023] 4282 process is forked.
[Mon May 22 10:15:22 2023] IdleProcessCount : 5
```

실행파일을 실행시키면 서버를 시작했다는 문구와 함께 StartProcess에 해당하는 값만큼 프로세스가 생성된다.

```
===== New Client =====
IP : 127.0.0.1
Port : 43663
=====

[Mon May 22 10:15:40 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 44175
=====

[Mon May 22 10:15:40 2023] IdleProcessCount : 3
[Mon May 22 10:15:40 2023] 4292 process is forked.
[Mon May 22 10:15:40 2023] IdleProcessCount : 4
[Mon May 22 10:15:40 2023] 4293 process is forked.
[Mon May 22 10:15:40 2023] IdleProcessCount : 5
===== Connection History =====
No.   IP       PID    PORT    TIME
1     127.0.0.1  4279   44175   Mon May 22 10:15:40 2023
2     127.0.0.1  4278   43663   Mon May 22 10:15:40 2023
===== New Client =====
IP : 127.0.0.1
Port : 44687
=====

[Mon May 22 10:15:48 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 45199
=====

[Mon May 22 10:15:48 2023] IdleProcessCount : 3
[Mon May 22 10:15:48 2023] 4301 process is forked.
[Mon May 22 10:15:48 2023] IdleProcessCount : 4
[Mon May 22 10:15:48 2023] 4302 process is forked.
[Mon May 22 10:15:48 2023] IdleProcessCount : 5
===== Connection History =====
No.   IP       PID    PORT    TIME
1     127.0.0.1  4281   45199   Mon May 22 10:15:48 2023
2     127.0.0.1  4280   44687   Mon May 22 10:15:48 2023
3     127.0.0.1  4279   44175   Mon May 22 10:15:40 2023
4     127.0.0.1  4278   43663   Mon May 22 10:15:40 2023
```

클라이언트와 연결을 하게 되면 10초에 한번씩 출력되는 연결기록에는 가장 최근 기록 순으로 출력이 된다.

```

===== Connection History =====
No.      IP          PID      PORT      TIME
1        127.0.0.1      4334     48783     Mon May 22 10:17:21 2023
2        127.0.0.1      4335     48271     Mon May 22 10:17:20 2023
3        127.0.0.1      4279     47759     Mon May 22 10:17:19 2023
4        127.0.0.1      4278     47247     Mon May 22 10:17:18 2023
5        127.0.0.1      4302     46735     Mon May 22 10:17:16 2023
6        127.0.0.1      4301     46223     Mon May 22 10:17:14 2023
7        127.0.0.1      4282     45711     Mon May 22 10:17:14 2023
8        127.0.0.1      4281     45199     Mon May 22 10:15:48 2023
9        127.0.0.1      4280     44687     Mon May 22 10:15:48 2023
10       127.0.0.1      4279     44175     Mon May 22 10:15:40 2023
===== New Client =====
IP : 127.0.0.1
Port : 49295
=====

[Mon May 22 10:17:45 2023] IdleProcessCount : 5
===== New Client =====
IP : 127.0.0.1
Port : 49807
=====

[Mon May 22 10:17:45 2023] IdleProcessCount : 4
===== Connection History =====
No.      IP          PID      PORT      TIME
1        127.0.0.1      4344     49807     Mon May 22 10:17:45 2023
2        127.0.0.1      4343     49295     Mon May 22 10:17:45 2023
3        127.0.0.1      4334     48783     Mon May 22 10:17:21 2023
4        127.0.0.1      4335     48271     Mon May 22 10:17:20 2023
5        127.0.0.1      4279     47759     Mon May 22 10:17:19 2023
6        127.0.0.1      4278     47247     Mon May 22 10:17:18 2023
7        127.0.0.1      4302     46735     Mon May 22 10:17:16 2023
8        127.0.0.1      4301     46223     Mon May 22 10:17:14 2023
9        127.0.0.1      4282     45711     Mon May 22 10:17:14 2023
10       127.0.0.1      4281     45199     Mon May 22 10:15:48 2023

```

또한 최대 10개까지만 저장하기 때문에 그 이후로는 가장 오래된 연결기록을 삭제하고 최근기록을 저장한다.

```

kw201920205@ubuntu:~/Desktop/work/Webl_2_0_201920205$ ./ipc_server
[Mon May 22 10:11:07 2023] Server is started.
[Mon May 22 10:11:07 2023] 4126 process is forked.
[Mon May 22 10:11:07 2023] IdleProcessCount : 1
[Mon May 22 10:11:07 2023] 4127 process is forked.
[Mon May 22 10:11:07 2023] IdleProcessCount : 2
[Mon May 22 10:11:07 2023] 4128 process is forked.
[Mon May 22 10:11:07 2023] IdleProcessCount : 3
[Mon May 22 10:11:07 2023] 4129 process is forked.
[Mon May 22 10:11:07 2023] IdleProcessCount : 4
[Mon May 22 10:11:07 2023] 4130 process is forked.
[Mon May 22 10:11:07 2023] IdleProcessCount : 5
===== New Client =====
IP : 127.0.0.1
Port : 32911
=====

[Mon May 22 10:11:10 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 33423
=====

[Mon May 22 10:11:11 2023] IdleProcessCount : 3
[Mon May 22 10:11:11 2023] 4137 process is forked.
[Mon May 22 10:11:11 2023] IdleProcessCount : 4
[Mon May 22 10:11:11 2023] 4138 process is forked.
[Mon May 22 10:11:11 2023] IdleProcessCount : 5
===== New Client =====
IP : 127.0.0.1
Port : 33935
=====

[Mon May 22 10:11:14 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 34447
=====

[Mon May 22 10:11:15 2023] IdleProcessCount : 3
[Mon May 22 10:11:15 2023] 4145 process is forked.
[Mon May 22 10:11:15 2023] IdleProcessCount : 4
[Mon May 22 10:11:15 2023] 4146 process is forked.
[Mon May 22 10:11:15 2023] IdleProcessCount : 5
===== New Client =====
IP : 127.0.0.1
Port : 34959
=====

[Mon May 22 10:11:15 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 35471
=====

[Mon May 22 10:11:16 2023] IdleProcessCount : 3
[Mon May 22 10:11:16 2023] 4154 process is forked.
[Mon May 22 10:11:16 2023] IdleProcessCount : 4
===== New Client =====

```

```

===== New Client =====
IP : 127.0.0.1
Port : 35983
=====

[Mon May 22 10:11:16 2023] IdleProcessCount : 3
===== New Client =====
IP : 127.0.0.1
Port : 36495
=====

[Mon May 22 10:11:16 2023] IdleProcessCount : 2
===== New Client =====
IP : 127.0.0.1
Port : 37007
=====

[Mon May 22 10:11:17 2023] IdleProcessCount : 1
===== Connection History =====
No.      IP          PID      PORT      TIME
1        127.0.0.1      4145     37007     Mon May 22 10:11:17 2023
2        127.0.0.1      4146     36495     Mon May 22 10:11:16 2023
3        127.0.0.1      4138     35983     Mon May 22 10:11:16 2023
4        127.0.0.1      4137     35471     Mon May 22 10:11:16 2023
5        127.0.0.1      4130     34959     Mon May 22 10:11:15 2023
6        127.0.0.1      4129     34447     Mon May 22 10:11:15 2023
7        127.0.0.1      4128     33935     Mon May 22 10:11:14 2023
8        127.0.0.1      4127     33423     Mon May 22 10:11:11 2023
9        127.0.0.1      4126     32911     Mon May 22 10:11:10 2023
===== New Client =====
IP : 127.0.0.1
Port : 37519
=====

[Mon May 22 10:11:17 2023] IdleProcessCount : 0

```

최대 process 개수 내에서 idleprocess의 개수에 따라 프로세스를 조건에 맞게 생성하는 모습이다.

```
===== Disconnected Client =====
IP : 127.0.0.1
Port : 40591
=====

[Mon May 22 10:13:40 2023] IdleProcessCount : 5
===== Disconnected Client =====
IP : 127.0.0.1
Port : 41103
=====

[Mon May 22 10:13:40 2023] IdleProcessCount : 6
===== Disconnected Client =====
IP : 127.0.0.1
Port : 41615
=====

[Mon May 22 10:13:40 2023] IdleProcessCount : 7
[Mon May 22 10:13:40 2023] IdleProcessCount : 6
[Mon May 22 10:13:40 2023] IdleProcessCount : 5
[Mon May 22 10:13:40 2023] 4180 process is terminated.
[Mon May 22 10:13:40 2023] 4182 process is terminated.
===== Disconnected Client =====
IP : 127.0.0.1
Port : 42127
=====

[Mon May 22 10:13:41 2023] IdleProcessCount : 6
===== Disconnected Client =====
IP : 127.0.0.1
Port : 42639
=====

[Mon May 22 10:13:41 2023] IdleProcessCount : 7
[Mon May 22 10:13:41 2023] IdleProcessCount : 6
[Mon May 22 10:13:41 2023] 4205 process is terminated.
[Mon May 22 10:13:41 2023] IdleProcessCount : 5
[Mon May 22 10:13:41 2023] 4199 process is terminated.
```

반대로 idle process수가 너무 많이 남을 때 조건에 맞게 종료시키는 모습이다.

```
[Mon May 22 10:08:38 2023] IdleProcessCount : 1
===== New Client =====
IP : 127.0.0.1
Port : 31887
=====

[Mon May 22 10:08:38 2023] IdleProcessCount : 0
===== Connection History =====
No.      IP          PID      PORT      TIME
1        127.0.0.1     4051     31887     Mon May 22 10:08:38 2023
2        127.0.0.1     4043     31375     Mon May 22 10:08:38 2023
3        127.0.0.1     4044     30863     Mon May 22 10:08:37 2023
4        127.0.0.1     4035     30351     Mon May 22 10:08:37 2023
5        127.0.0.1     4036     29839     Mon May 22 10:08:37 2023
6        127.0.0.1     4029     29327     Mon May 22 10:08:36 2023
7        127.0.0.1     4028     28815     Mon May 22 10:08:36 2023
8        127.0.0.1     4027     28303     Mon May 22 10:08:36 2023
9        127.0.0.1     4026     27791     Mon May 22 10:08:36 2023
10       127.0.0.1     4025     27279     Mon May 22 10:08:35 2023
===== Disconnected Client =====
IP : 127.0.0.1
Port : 27279
=====

===== New Client =====
IP : 127.0.0.1
Port : 32399
=====

[Mon May 22 10:08:50 2023] IdleProcessCount : 1
[Mon May 22 10:08:50 2023] IdleProcessCount : 0
```

최대 10까지 연결을 받는지 확인하기 위해 10번 이상 요청을 보낸 결과 idle process의 수가 0이라면 연결을 받지 못하고 웹페이지가 블록되어 있는 것을 확인하였고 자식 프로세스중 하나가 이전 클라이언트와의 연결을 끝나치면 새로운 클라이언트와 바로 연결되었다.


```

kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ make
make: 'ipc_server' is up to date.
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ ./ipc_server
[Mon May 22 14:49:09 2023] Server is started.
[Mon May 22 14:49:09 2023] 5508 process is forked.
[Mon May 22 14:49:09 2023] IdleProcessCount : 1
[Mon May 22 14:49:09 2023] 5509 process is forked.
[Mon May 22 14:49:09 2023] IdleProcessCount : 2
[Mon May 22 14:49:09 2023] 5510 process is forked.
[Mon May 22 14:49:09 2023] IdleProcessCount : 3
[Mon May 22 14:49:09 2023] 5511 process is forked.
[Mon May 22 14:49:09 2023] IdleProcessCount : 4
[Mon May 22 14:49:09 2023] 5512 process is forked.
[Mon May 22 14:49:09 2023] IdleProcessCount : 5
===== Connection History =====
No.      IP          PID      PORT      TIME
===== New Client =====
IP : 127.0.0.1
Port : 28383
=====

[Mon May 22 14:49:28 2023] IdleProcessCount : 4
===== Connection History =====
No.      IP          PID      PORT      TIME
1        127.0.0.1      5508      28383     Mon May 22 14:49:28 2023
===== New Client =====
IP : 127.0.0.1
Port : 28895
=====

[Mon May 22 14:49:29 2023] IdleProcessCount : 3
[Mon May 22 14:49:29 2023] 5709 process is forked.
[Mon May 22 14:49:29 2023] IdleProcessCount : 4
[Mon May 22 14:49:29 2023] 5710 process is forked.
[Mon May 22 14:49:29 2023] IdleProcessCount : 5
===== New Client =====
IP : 127.0.0.1
Port : 29407
=====

[Mon May 22 14:49:30 2023] IdleProcessCount : 4
===== New Client =====
IP : 127.0.0.1
Port : 29919
=====

[Mon May 22 14:49:30 2023] IdleProcessCount : 3
[Mon May 22 14:49:30 2023] 5734 process is forked.
[Mon May 22 14:49:30 2023] IdleProcessCount : 4
[Mon May 22 14:49:30 2023] 5735 process is forked.
[Mon May 22 14:49:30 2023] IdleProcessCount : 5
^C[Mon May 22 14:49:37 2023] 5509 process is terminated.
[Mon May 22 14:49:37 2023] 5511 process is terminated.
[Mon May 22 14:49:37 2023] 5512 process is terminated.
[Mon May 22 14:49:37 2023] 5508 process is terminated.
[Mon May 22 14:49:37 2023] 5510 process is terminated.
[Mon May 22 14:49:38 2023] IdleProcessCount : 4
[Mon May 22 14:49:43 2023] 5709 process is terminated.
[Mon May 22 14:49:44 2023] IdleProcessCount : 3
[Mon May 22 14:49:49 2023] 5710 process is terminated.
[Mon May 22 14:49:50 2023] IdleProcessCount : 2
[Mon May 22 14:49:55 2023] 5734 process is terminated.
[Mon May 22 14:49:56 2023] IdleProcessCount : 1
[Mon May 22 14:50:01 2023] 5735 process is terminated.
[Mon May 22 14:50:02 2023] IdleProcessCount : 0
[Mon May 22 14:50:02 2023] Server is terminated.
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ █

```

위의 결과를 보면 프로세스 종료 전 총 프로세스의 개수는 9개이고 이중 4개가 실행중이며 5개는 idle process이다. 따라서 종료 시에 클라이언트와 연결중인 4개는 단순히 종료를 하게 되고 idle process 5개는 idle process의 수가 감소하는 것을 보이며 종료되어야 한다.

5. 고찰

이번 과제는 지난 과제와 동일하게 preforked-process server를 구현하는 것이었다. 하지만 단순히 fork를 해두는 것과는 다르게 클라이언트와 연결 유무에 따라 preforked process를 조절하기 때문에 자원을 아낄 수 있었다. 공유메모리를 활용하여 각각 프로세스에서 연결기록을 관리했었으나 이를 함께 저장하여 공간 효율성을 늘릴 수 있었다. 다만 다수가 함께 사용하는 메모리이기 때문에 동기화문제를 해결하는 것이 중요했는데 mutex를 이용하여 임계구역에 대한 접근을 조절할 수 있어 다중 스레드 환경에서의 동기화문제를 해결할 수 있었다. 또한 스레드를 이용하여 함수를 실행해보고 스레드와 프로세스의 차이와 스레드를 사용하는 방법등을 익힐 수 있었다. 과제를 진행하면서 어려운 점이 많았는데 먼저 mutex를 이용하면서 deadlock을 경험하였다. 분명히 함수내에서 lock을 사용하여 메모리에 값을 쓰고 unlock으로 풀고 나왔지만 자식 프로세스에서 공유메모리에 대한 접근이 아예 불가하였다. 결과적으로는 부모프로세스에서 자식프로세스를 생성할 때 mutex lock을 건 상태에서 생성하게 되면 자식프로세스는 lock이 걸린채로 복제가 되고 이는 부모가 mutex unlock을 하더라도 풀리지 않는다는 것을 알았다. 또 어려웠던 점은 idle process관리에 대한 것이었는데 idle process의 수는 공유메모리에 접근하여 수정하고 사용하고 하면 되었지만 어떤 process가 idle process인지 알고 관리하는 게 어려웠다. idle process가 무엇인지 알아야 idle process의 수가 너무 많아져 process를 종료해야 할 때 전체 process중 idle process를 종료해야 하기 때문이다. 그렇지 않으면 클라이언트와 서로 연결중인 process를 종료하게 될 지 모르고 이는 서버로서 제 역할을 하지 못한다고 볼 수 있다.

서버를 구현해보면서 멀티 스레드라는 개념 자체가 유용하고 좋은점도 많지만 잘못 구현하면 서로 충돌이 일어나거나 복잡해지고 오히려 더 구현하기 어려워진다는 단점도 있다고 생각했다.

6. Reference

시스템프로그래밍실습 강의자료 참조