2019. 6. 4.

Assignment # 4-2

금요일 - 이성원 교수님

2015722087 컴퓨터정보공학부

김민철

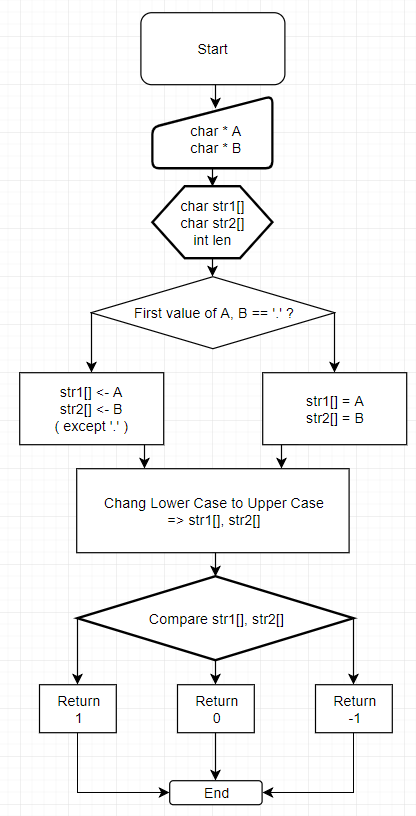
Process\_pool\_management web-server구현

**Introduction**

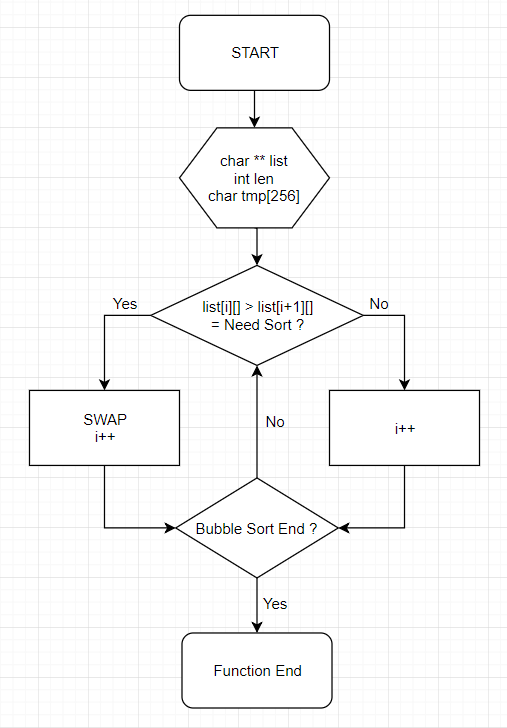
이번 과제에서는 4-1 과제에서 구현하였던 Pre forked web-server에서, 모든 자식 프로세스에서의 연결 기록을 공유 메모리에 있는 하나의 데이터로 모아, 가장 최신의 기록 10개 까지 출력하게하며, 공유 메모리에 접근 할때는 스레드를 이용하고, 동기화를 위하여 mutex\_lock, unlock을 이용하고, Idle 프로세스의 개수를 조절하는 기능을 추가한다.

**Flow Chart**

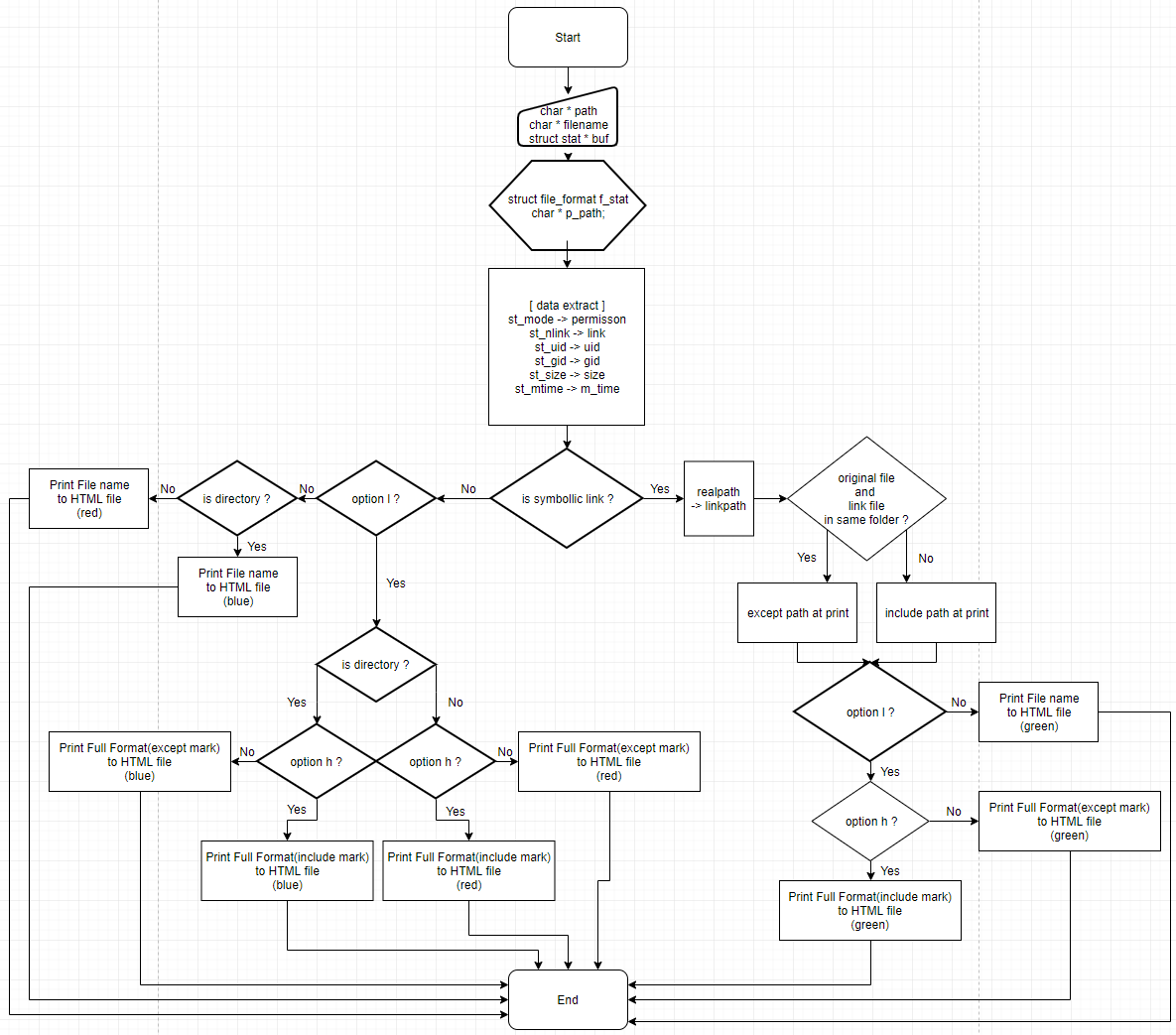
-int strcmp\_i



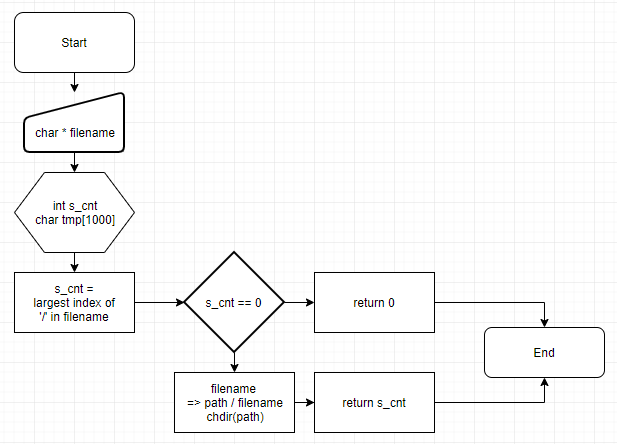
-void sort\_list



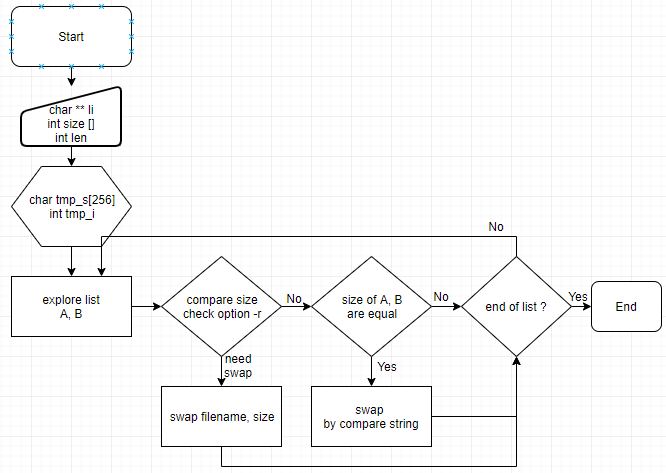
-void view\_advanced\_list



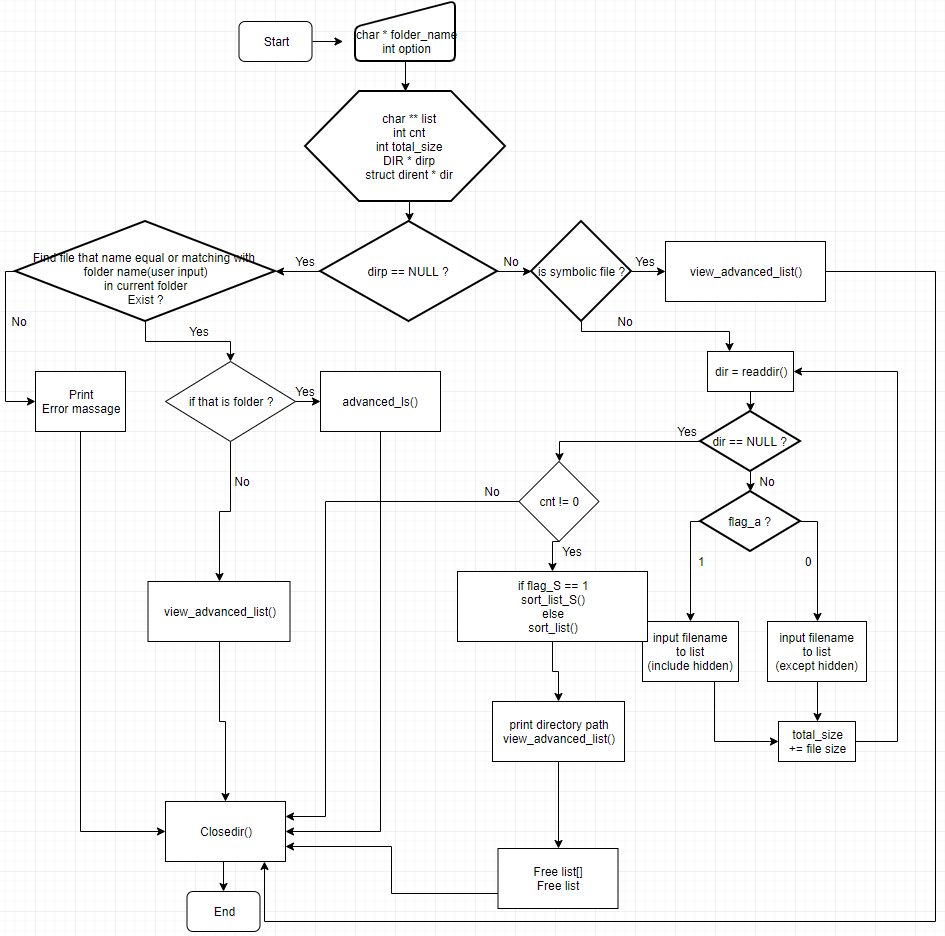
-int check\_real\_path



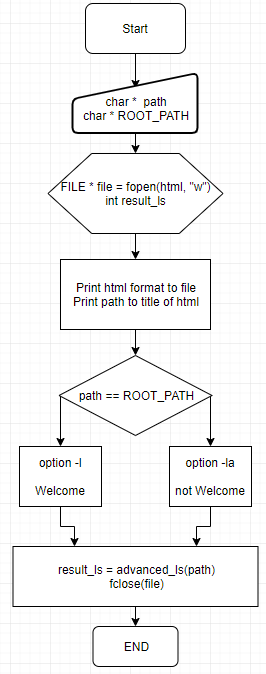
-void sort\_list\_S



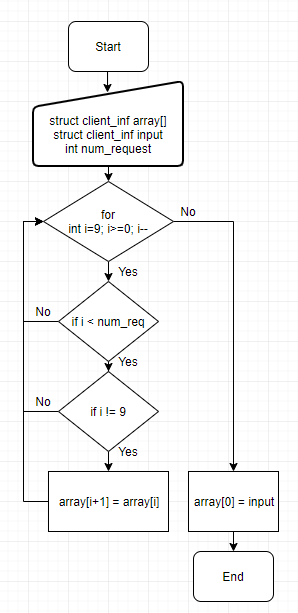
-int advanced\_ls



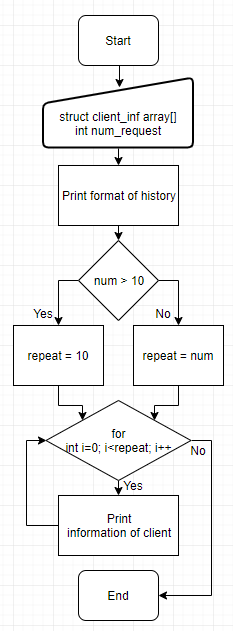
- int html\_ls



- add\_client

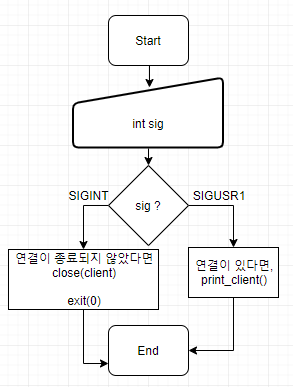


- print\_client

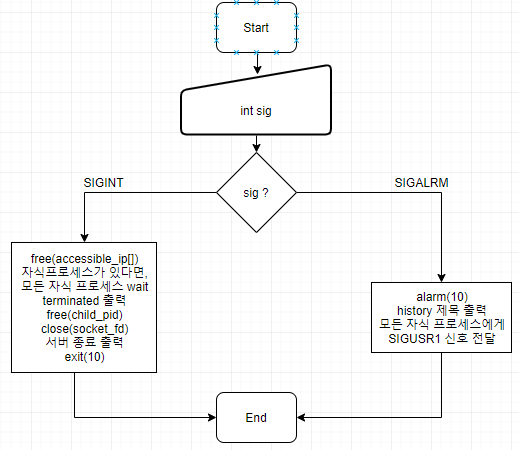


- signal\_handler\_c

SIGINT -> 무시, 기존의 SIGINT -> SIGUSR2

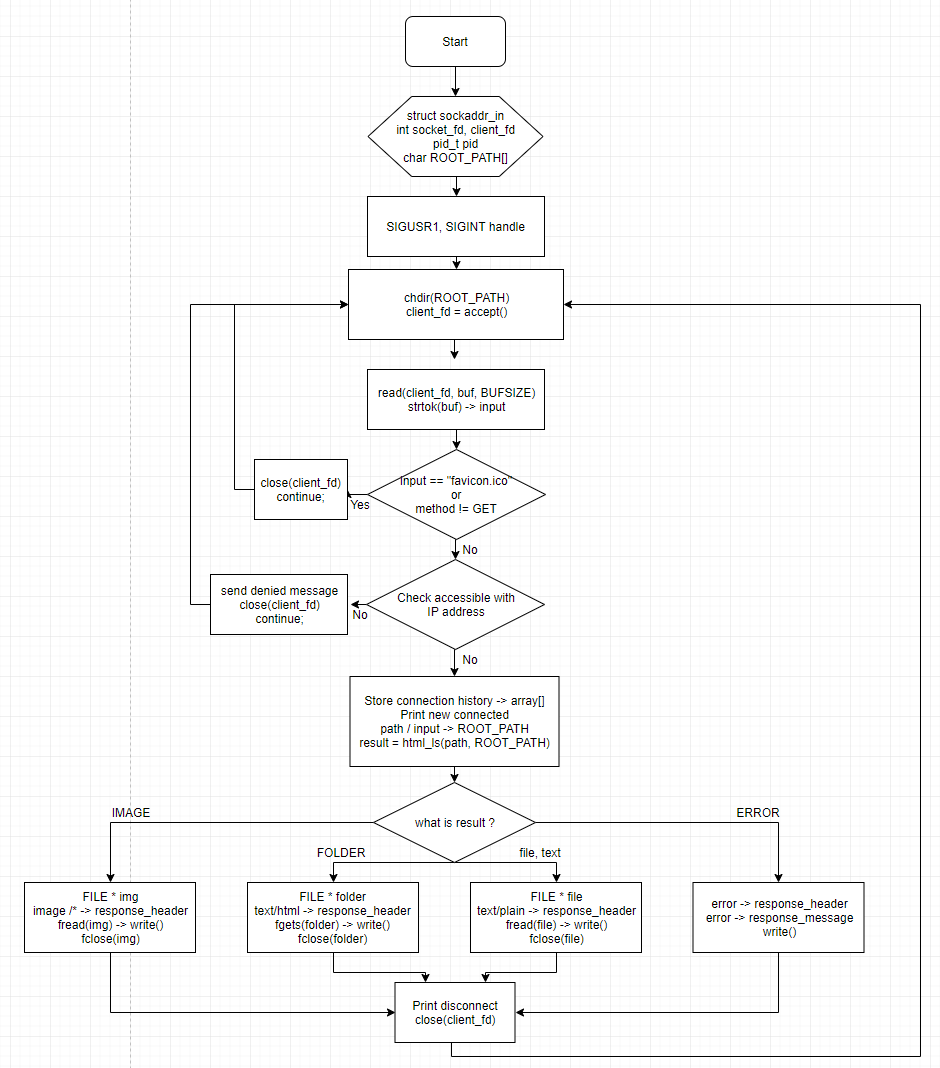


- signal\_handler\_p

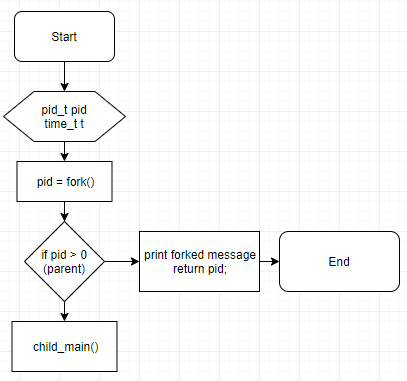


SIGINT 일 경우 wait 이전에 kill(모든 자식, SIGUSR2) 추가

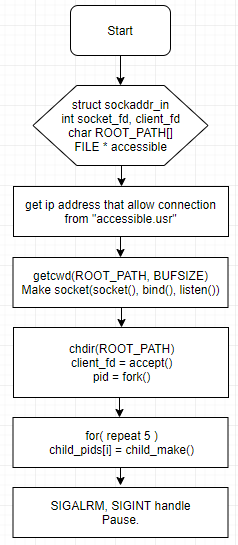
-child\_main



- child\_make



- int main



**Pseudo code**

int check\_real\_path(char \* filename)

{

for (int i = 0; i < strlen(filename); i++) // check directory.

{

입력된 파일 이름에 '/' 가 있을 경우 해당 index => s\_cnt

}

if 파일 이름에 '/' 가 있다면 -> real path

{

devide < path > / < file >

<path> 로 작업중인 디렉토리 변경

s\_cnt 값 반환

}

else real path 가 아니라면

0 반환

}

void swap(char \* str1, char \* str2)

{

str1과 str2 swap

}

int strcmp\_i ( char \* 비교할 첫 번째 문자열 A, char \* 비교할 두 번째 문자열 B )

{

<예외 처리> : r 옵션이면 거꾸로.

A랑 B가 “.” , “..” 이면 “..” 이 뒤로,

둘 중 하나만 “.” 이면 “.”이 앞으로

둘 중 하나만 “..” 이면 “..”이 앞으로

if( A 가 B 보다 길면 )

len = B의 길이;

else

len = A의 길이

for(len 만큼 반복)

{

if(소문자라면)

대문자로 바꾼다

}

If 옵션이 –r 이라면

if( A > B ) return = -1;

else if( B > A ) return = 1;

else return = 0;

else

if( A > B ) return = 1;

else if( B > A ) return = -1;

else return = 0;

}

void sort\_list(char \*\* li, int len)

{

for (저장된 파일이름 개수-1 만큼 반복)

{

if (더 이상 내용이 없다)

break;

for (저장된 파일이름 개수-1 만큼 반복)

{

if ( 앞에 저장된 파일이름 > 뒤에 저장된 파일 이름)

두 파일 이름을 swap

}

};

}

void sort\_list\_S(char \*\* li, int size[], int len)

{

// sort like bublle sort

for (int i = 0; i < len - 1; i++) // total cycle

{

if 리스트가 끝났다면

break;

for (int j = 0; j < len - 1; j++) // one cycle

{

if 파일 사이즈가 다르다면, 정렬 필요 -r 옵션에 따라서.

{

swap file name

swap file size

}

else if 파일 사이즈가 같다면, -> 문자열로 정렬

{

if 정렬이 필요하다면

{

swap file name

}

}

}

}

return;

}

void view\_advanced\_list(char \* 파일 경로, char \* 파일 이름, struct stat \* 파일 정보)

{

St\_mode에 저장되어 있는 파일 타입에 따라 permission[0] 결정

St\_mode에 저장되어 있는 user 권한에 따라 permission[1~3] 결정

St\_mode에 저장되어 있는 group 권한에 따라 permission[4~6] 결정

St\_mode에 저장되어 있는 other 권한에 따라 permission[7~9] 결정

* Permission = “- --- --- ---”

st\_nlink -> f\_stat->link

st\_uid -> f\_stat->uid

st\_gid -> f\_stat->gid

st\_size -> f\_stat->size

st\_mtime -> f\_stat->m\_time

if 옵션이 –h 라면

size = size / 1024.0, check\_h++(단위가 몇번 상승했는지 체크)

check\_h 값에 따라서 단위(K, M, G) 결정

// Full Format 출력

if 파일 type 이 symbolic link 라면

{

if l옵션이 아니라면

파일 이름만 HTML 파일로 출력 (green)

else Full Format 출력

If 옵션이 –h 라면, 사이즈 출력시 단위와 함께 출력

Symbolic link 파일과 가리키는 경로 파일이 같은 폴더에 있다면 -> 가리키는 파일명만 HTML 파일로 출력(green)

Symbolic link 파일과 가리키는 경로 파일이 다른 폴더에 있다면 -> 절대 경로 모두 HTML 파일로 출력(green)

}

else symbolic link 가 아니라면

{

if ㅣ옵션이 아니라면

directory일 경우 파일 이름만 HTML 파일로 출력 (blue)

아닐 경우 파일 이름만 HTML 파일로 출력 (red)

else l 옵션 이라면

if directory라면 (blue)

If 옵션이 –h 라면, FullFormat 출력시 단위도 HTML 파일로 출력

else FullFormat HTML 파일로 출력

else directory가 아니라면 (red)

If 옵션이 –h 라면, FullFormat 출력시 단위도 HTML 파일로 출력

else FullFormat HTML 파일로 출력

}

int advanced\_ls(char \* 입력한 폴더)

{

if 폴더가 아니라면

{

혹시 절대경로로 입력된 폴더인지 확인 -> 맞으면 해당 디렉토리로 변경

-> 아니면 현재 디렉토리

HTML 파일이면 return

if table이 만들어져 있지 않고, flag\_p가 -1이 아니라면,

옵션 l 에 따라 테이블 생성 -> flag\_table=0;

while 디렉토리 탐색

{

if 파일을 찾았다면

view\_advanced\_list()

If fnmatch 로 매칭이 된다면,

폴더면 다시 advanced\_ls()

파일이면 view\_advanced\_list()

}

If 출력한 파일이 없다면

에러 메시지 출력 -> 반환 -1

else 1 반환

}

else // is folder

{

p 플래그가 표시되어있다면 return 3 => 폴더

flag\_table == 0 이라면 테이블이 열려있으므로 close

flag\_table = 1 set -> 테이블이 안만들어져있다.

심볼릭 링크로 연결된 폴더인지 확인

-> 심볼릭 링크 폴더이고 옵션이 l 이면 심볼릭 링크만 Full Format 출력

아니면 탐색 시작

while 입력된 폴더 탐색

{

HTML 파일은 처리하지 않음.

if 파일이 없다면

반복문 탈출

if 히든 파일이고 옵션이 –l 이나 default 라면

{

List에 파일 이름 입력

Total size += 파일 size

}

else if 옵션이 –a 이나 –la 라면 모든 파일 입력

{

List에 파일 이름 입력

Total size += 파일 size

}

}

Directory path 출력

if 파일이 존재한다면

{

List 정렬.

Total size 출력

if l 옵션이라면 -> Full Format Table 생성

else -> 파일 이름 Table 생성

for 파일 개수 만큼 반복

view\_advanced\_list()

테이블 close

flag\_table=1

}

}

폴더 닫기

}

int html(char \* 찾을 경로, char \* 루트 경로)

{

origin\_wd에 루트 경로 복사

HTML 파일 기본 포맷 입력, 타이틀에 현재 디렉토리 입력.

HTML\_FILE 에 쓰기위한 fopen

title에 찾을 경로 입력

찾을 경로가 루트 경로이면 -l옵션하고 Welcome, 아니면 -la 옵션하고 Welcome 생략

result = advanced\_ls(path);

테이블 close

html 파일 포맷 close

open한 파일 close

result 반환.

}

- int main

{

"accessible.usr" 파일 오픈

접근 허용된 IP주소 읽어와서 accessible\_ip 배열에 저장

"httpd.conf" 파일 오픈

각 요소의 데이터를 읽어와서 각 요소에 맞는 데이터 저장

소켓 생성 -> 연결 준비(socket(), bind(), listen())

공유 메모리 생성 후 현재 프로세스에 연결, SH\_DATA 구조체 형태로 사용

공유 메모리의 데이터값 0으로 초기화

10초 뒤 알람 설정

SIGPIPE 무시 신호 처리

자식 프로세스 5개 fork() 하고 자식 프로세스의 pid를 child\_pids 배열에 저장

SIGALRM, SIGINT, SIGUSR1, SIGUSR2, 30 시그널 처리

Pause;

}

void child\_main()

{

29, SIGUSR1, SIGUSR2 신호 처리 선언

while(1)

{

ROOT PATH로 디렉토리 변경

클라이언트로 부터의 연결을 기다림

연결이 되면 예외처리 확인 (EXIT, favicon.ico)

접근 허용된 IP주소 확인(strcmp, fnmatch)

공유 메모리 get하고 현재 프로세스에 연결 -> SH\_DATA 구조체 형태로 사용

현재 클라이언트의 연결 정보를 공유 메모리에 입력하는 스레드 생성

-> 스레드는 connect\_client() 함수를 사용

연결 상태 출력

루트 경로가 아닌 경우 루트 경로 / input -> path

루트 경로인 경우 루트 경로 -> path

result 에 html\_ls(path) 의 결과 출력 (폴더:3, 파일:1, 오류:-1)

if 이미지 파일인 경우

이미지 파일 open.

헤더 메세지 : image/\*

응답 메세지에 이미지 파일 내용 전송

else if 폴더인 경우

html\_ls의 결과인 html 파일 open

헤더 메세지 : text/html

응답 메세지에 html 파일 내용 전송

else if 파일인 경우

파일 open

헤더 메세지 : text/plain

응답 메세지에 파일 내용 전송

else

에러 예외처리 메세지 전송

5초간 대기

현재 클라이언트의 연결 끊김을 입력하는 스레드 생성

-> 스레드는 disconnect\_client() 함수를 사용

연결 종료 메세지 출력

연결 종료.

}

}

pid\_t child\_make()

{

pid = fork() // 자식 프로세스 생성

부모 프로세스라면

forked가 되었다는 메세지 출력

return pid;

자식 프로세스라면

child\_main() 함수 실행

}

void connect\_client()

{

mutex\_lock

현재 idle --

현재 busy ++

요청개수 ++

히스토리에 연결 기록 저장

mutex\_unlock

부모에게 30 신호를 전달

}

void disconnect\_client()

{

mutex\_lock

현재idle ++

현재 busy --

mutex\_unlock

부모에게 30 신호를 전달

}

void check\_idle()

{

mutex\_lock

현재자식개수 > 최대자식 이라면, 첫번째 자식에게 SIGUSR2 전달

현재 idle > 최대 idle 이라면, 첫번째 자식에게 SIGUSR2 전달

현재 idle < 최대 idle 이라면, child\_make()

mutex\_unlock

}

void reduce\_idle()

{

mutex\_lock

tmp\_pid = 아무 자식이든 죽을때까지 wait후 pid 반환

mutex\_unlock

}

void print\_client()

{

if (num\_request > MaxHistory)

repeat = MaxHistory

else

repeat = num\_request

히스토리 제목 출력

repeat 만큼 반복

array에 들어있는 연결 정보 출력

}

void signalHandler\_c(int sig)

{

if sig == 29

client와의 연결이 종료되지 않았다면 -> 종료

exit

if sig == SIGUSR2

IDLE 이라면 부모에게 USR1 신호를 보내고 exit.

BUSY 라면 부모에게 USR2 신호를 보냄

}

void signalHandler\_p(int sig)

{

if sig == SIGINT

free accessible\_ip

kill(모든 자식, SIGUSR2)

waitpid (모든 자식)

printf(terminated)

공유 메모리 제거

close(소켓)

exit

if sig == SIGALRM

10초뒤 알람 설정

스레드를 통해 print\_client() 함수 실행

if sig == SIGUSR1

스레드를 통해 reduce\_idle() 함수 실행

if sig == SIGUSR2

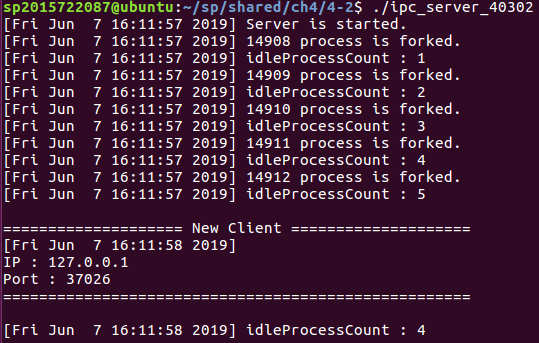
flag 값을 1 증가시키고 flag 번째 자식에게 USR2 신호 전달

if sig == 30

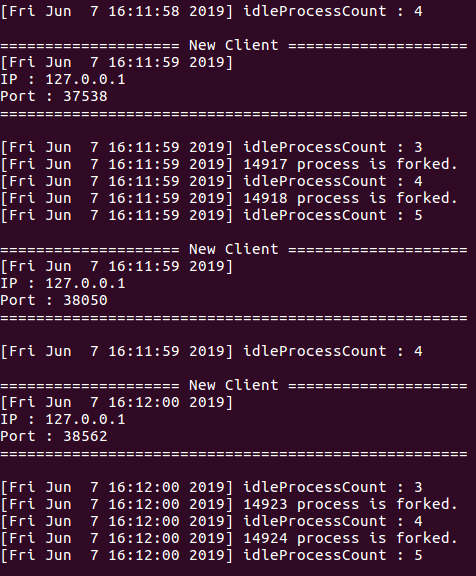
스레드를 통해 check\_idle() 함수 실행

}

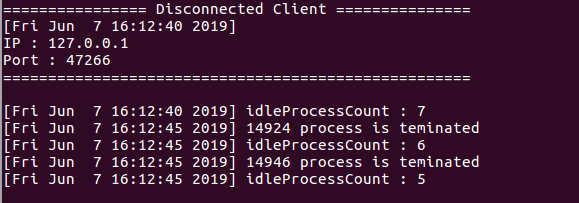
**Result**



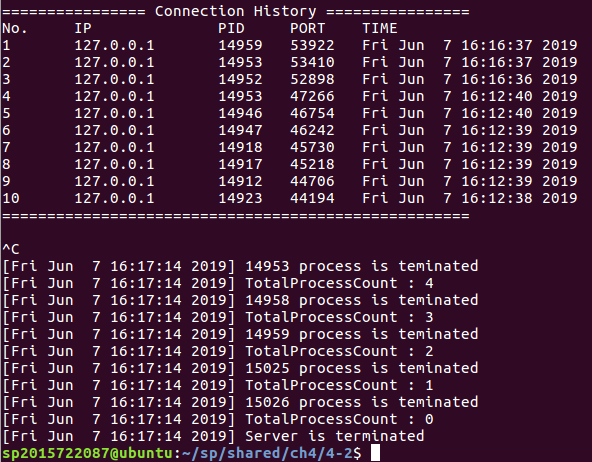
- 서버를 실행하고 started 메세지와 forked 메세지의 출력. 새로운 연결이 들어오면 idle프로세스의 개수가 하나 줄어든다.



- idle 프로세스의 개수가 MinIdleNum 보다 작을 경우 프로세스를 추가로 fork하여 개수를 유지한다.



- 여러번의 연결 종료로 인해 idle프로세스의 개수가 maxidlenum 보다 커지면서 일부 프로세스를 종료시켜 개수를 유지하는 모습이다.



- 서버를 종료시키는 모습이다 현재 전체 프로세스의 개수가 6개 이므로 6개를 종료시키는 모습을 출력하고, 공유메모리를 제거한뒤에 서버를 종료시켰다.

**Conclusion**

이번 과제는 지난번의 과제를 기반으로 기능을 추가하는 과제이지만, 공유 메모리와 스레드를 처음으로 다뤄보게 되어 걱정이 앞서는 과제였다. 공유 메모리를 어떤 식으로 사용해야 하는지, 개념은 무엇인지 잘 몰라서 강의 자료를 많이보고, 구글에서도 많이 검색해서 개념을 익힌 후 여러 가지 방법으로 사용을 해본 뒤에야 이해를 하고 사용하게 되었다.

스레드를 사용하는 것도 매우 어려웠다. 무엇인지도 모르고, 함수는 되게 많아서 이것 또한 공부를 많이하고 나서야 조금 이해하고 사용하게 되었다.

결론적으로는 한번의 과제에서 여러가지를 처음 접하고, 배우고, 사용하면서 간단하게 작성된 코드이지만 개념과 사용법을 익히게 된 것 같아 유익한 과제였다.