Assignment#3-3

시스템 프로그래밍 실습

제출일: 5월 24일 금요일

분 반: 화요일

담당 교수: 신영주

학 번: 2015722025

학 과: 컴퓨터정보공학부

이 름: 정용훈

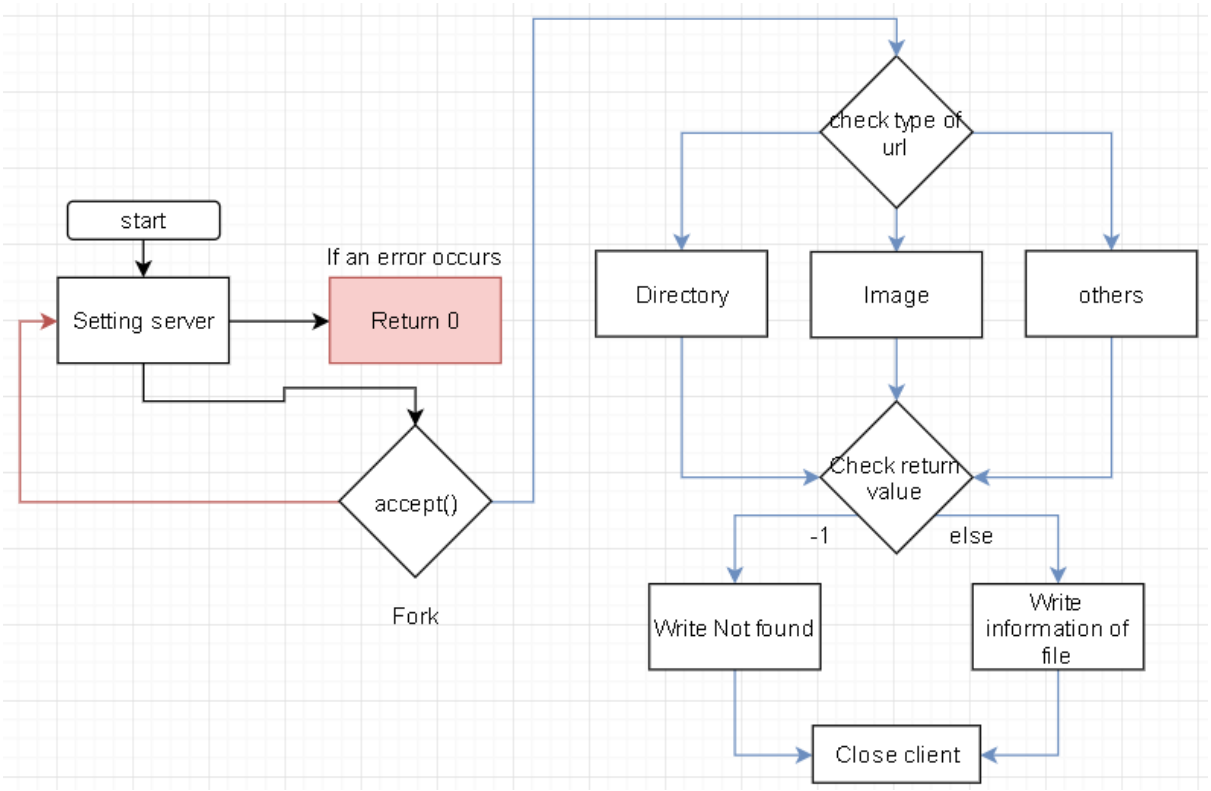
1. Introduction

3-3과제는 3-2의 출력을 따르며, fork와 signal함수를 통하여 동시에 다수의 클라이언트가 접속할 수 있도록 하며, 일정 동작을 반복적으로 수행하는 server를 구현하는 것이다. 또한 추가적으로 허용된 IP만 접속될 수 있도록 한다. 구현하는데 어려움은 크게 없지만 개념적으로 이해하는데 중요한 과제라고 생각된다.

2. Flow Chart

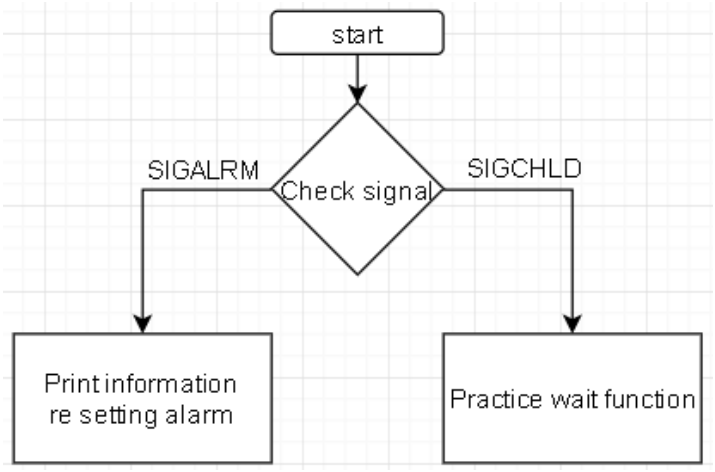
이번 3-3과제에서는 Main함수에 추가되는 fork관련함수와 signal함수만 있을 뿐 다른 점은 없으며, 추가적으로 signal과 history에 관련된 함수가 추가된다.

Main



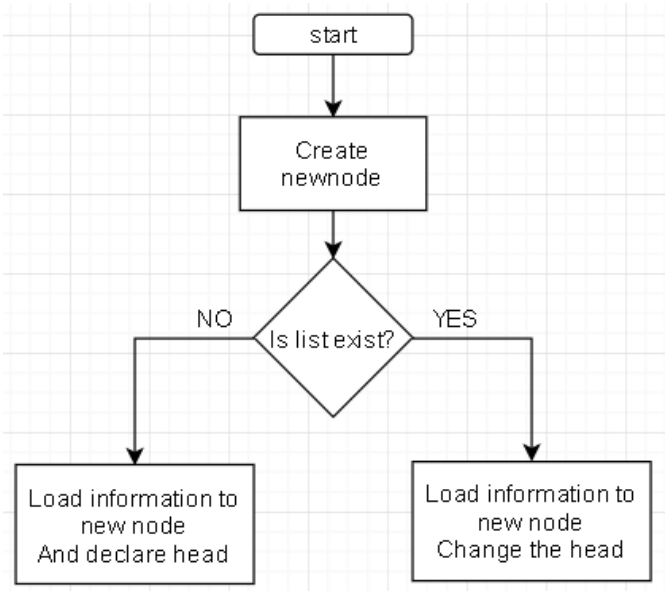
전 과제와 메인 함수의 다른 부분은 fork를 통해 process가 나뉘어 동작하게 된다. 그럼으로써 request를 동시에 처리할 수 있으며, throughput의 증가를 기대할 수 있다. 동작은 accept을 통하여 client를 받은 후 fork를 통해 parent process는 continue시켜 다른 request를 받을 수 있도록 대기 시키고, child process는 아래 동작을 계속 수행할 수 있도록 설계되어있다.

SignalHandler



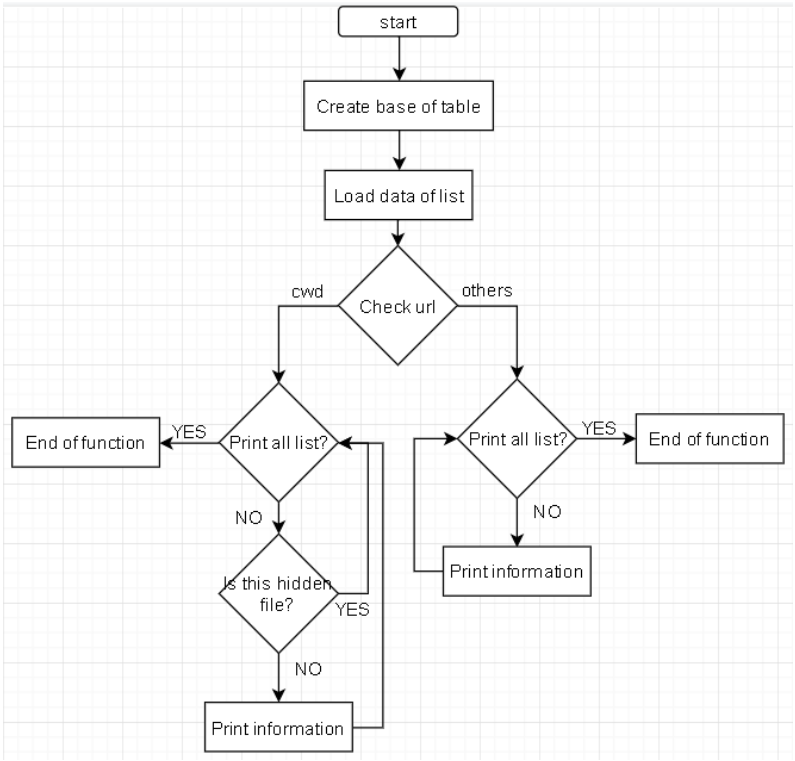
추가된 함수로 Signal을 정의해주는 함수다.

UpdateInfo



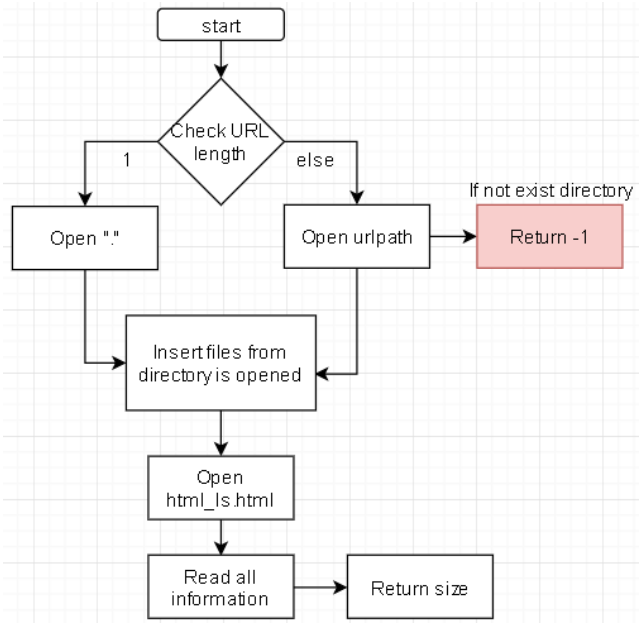
History출력을 위해 list를 생성하는 함수다. 특징으로는 가장 최신의 데이터가 먼저 보여지기 때문에 Head를 계속해서 update해주며 정보들을 link해준다. Link된 list들은 alarm signal을 통해 10초마다 출력되는데, 연결된 list들 중 최대 10개까지 출력된다. 출력 방법은 아래 print함수들과 10개의 제약이 걸려있다는 것 말고는 동일하다.

PrintHTML(create file of html_ls.html)



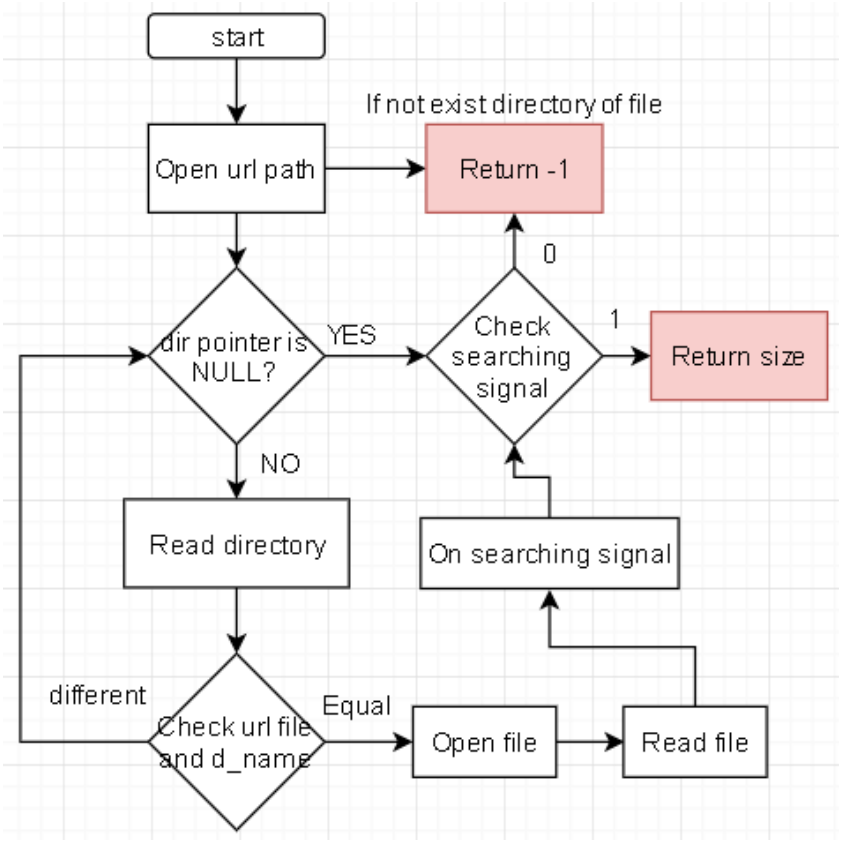
3-1 과제인 html_ls.hml을 만드는 과정으로 옵션 구현이 –a과 –al만 있기 때문에 구조만 동작 자체는 똑같지만 구조를 변경하게 되었다. 해당 함수는 Html함수에서 call하는 함수로써 아래 Html함수를 참고하면 이해하기 쉽다.

Html



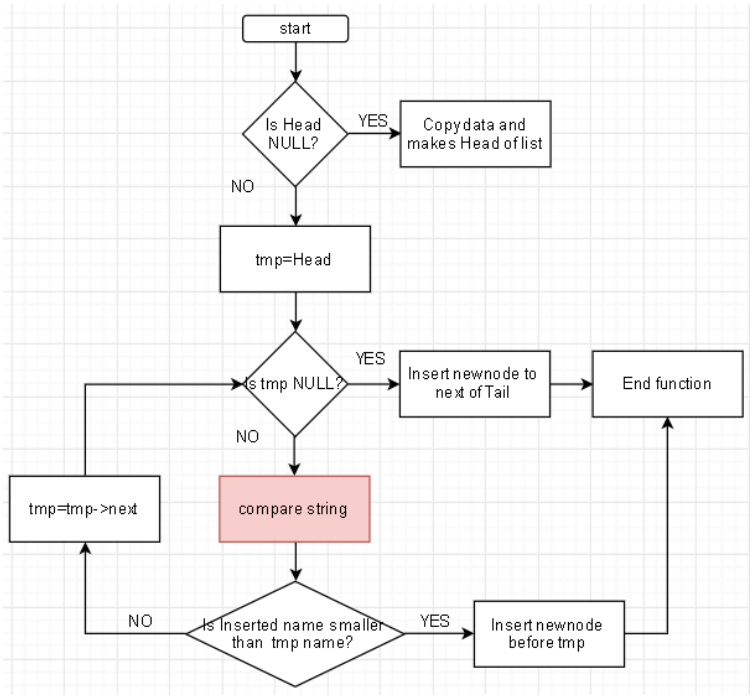
Open html을 하기 전 과정이 printHTML로써 html파일을 생성한 후 실행하게 된다.

Image & Normal (others)



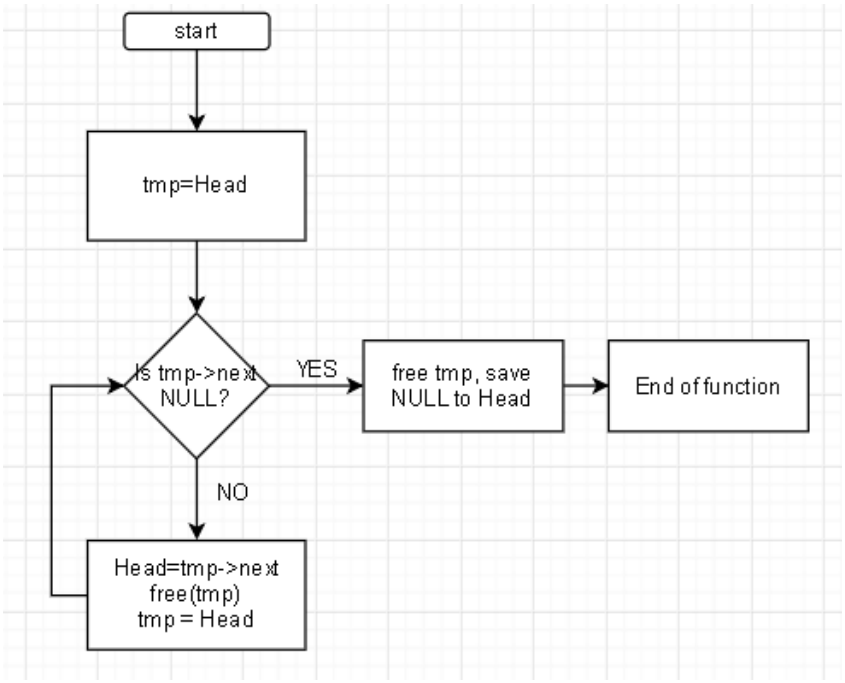
다음 함수는 Image와 나머지 다른 파일들을 처리해주는 함수로써 정보를 binary로 읽어와 write해주는 작업을 하게 된다 사실 image와 나머지 파일은 모두 binary를 통하여 읽을 수 있기 때문에 나눠줄 필요는 없다고 생각된다.

Insertnode



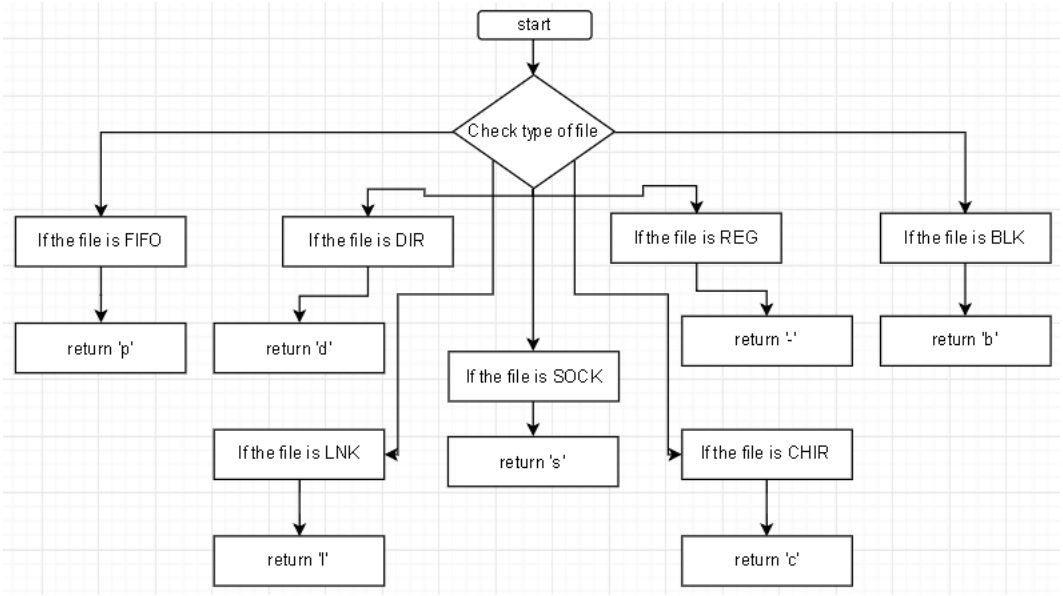
따로 sort함수를 사용하지 않고 insert를 실행 할 때부터 sort가 되며, list가 생성된다. Insert함수는 기존 함수와 동일하게 사용되므로 전체적인 변화는 없지만, S옵션을 사용면 size를 비교해야 하므로 compare를 하는 부분이 변경되게 된다. 해당 문제는 함수는 같고 조건만 바꿔주어 해결하였다.

Deletelist



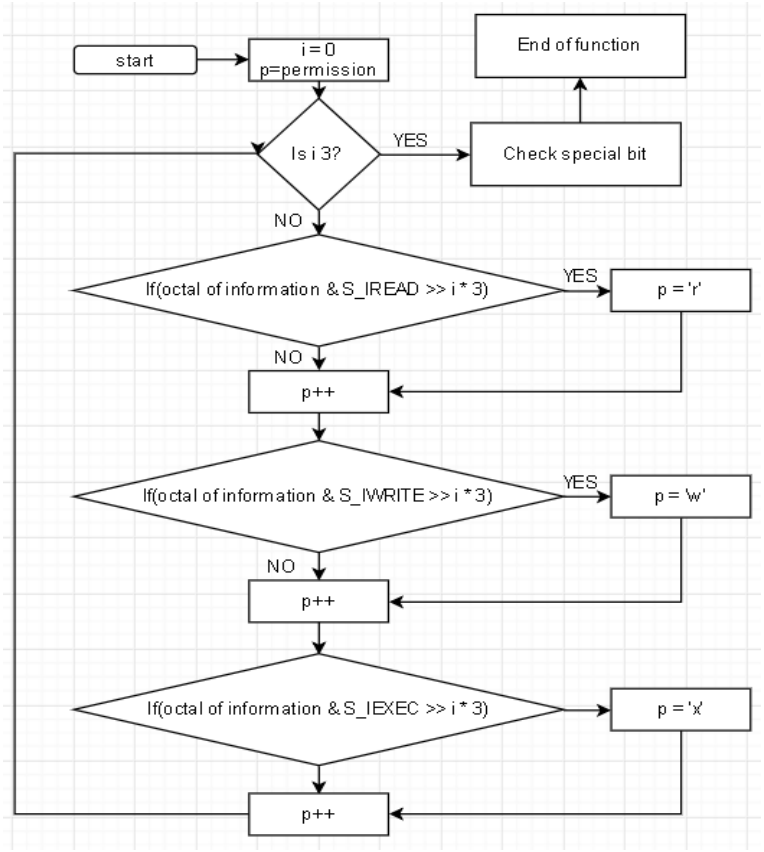
모든 정보 출력 후 linked list를 제거하는 함수다.

printType



파일의 정보를 받아와 st_mode를 통하여 파일의 type을 정의하는 함수다

printPerm



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);
        set alarm (10);
        ////////////////////////////////////////////////////////////////////
```

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

        len = sizeof(client_addr);
        client_fd = accept;

        Check accessible user;
        load url;

        >>FORK()<<

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

SignalHandler

```
void signalHandler(int sig)
{
        if(sig == SIGALRM)
        {
                print information of history;
                alarm(10);
        }

        if(sig == SIGCHLD)
                wait(&status);
}
```

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

```
  3297 pts/17   00:00:00 web_server
  3303 ?        00:00:00 Web Content
  3341 pts/17   00:00:00 web_server
  3343 pts/17   00:00:00 web_server
  3344 pts/17   00:00:00 web_server
  3345 pts/17   00:00:00 web_server
  3348 pts/4    00:00:00 ps
sp20157522025@ubuntu:~$
```

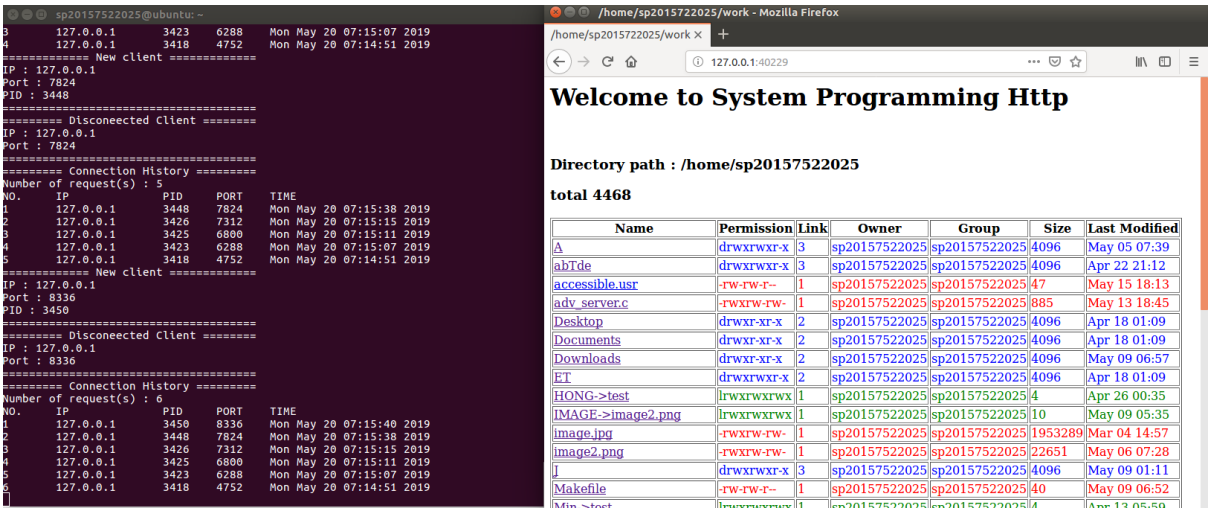
다음은 작성한 코드에 sleep을 통하여 fork가 잘 동작하는지 확인하기 위해 터미널을 두 개를 사용하여 ps -e명령을 통해 확인한 결과이다. 결과를 보면 fork가 동작하기 때문에 process가 같은 이름으로 여러 개 동작하는 것을 확인할 수 있다.

```
  3263 ?        00:00:01 Web Content
  3297 pts/17   00:00:00 web_server
  3303 ?        00:00:00 Web Content
  3356 pts/4    00:00:00 ps
sp20157522025@ubuntu:~$
```

그 후 sleep시간이 모두 지나면서 기존에 서버 하나만 남고 모두 종료된다.

```
sp20157522025@ubuntu:~$ ./web_server
========== Connection History ==========
Number of request(s) : 0
NO.     IP              PID     PORT    TIME
```

다음은 연결을 한번도 하지 않은 경우의 터미널 상태이며 다음은 페이지를 여러 번 동작시켰을 때의 터미널 결과다.



```
sp20157522025@ubuntu: ~
3       127.0.0.1       3423    6288    Mon May 20 07:15:07 2019
4       127.0.0.1       3418    4752    Mon May 20 07:14:51 2019
============= New client =============
IP : 127.0.0.1
Port : 7824
PID : 3448
=====================================
========= Disconeected Client ========
IP : 127.0.0.1
Port : 7824
=====================================
========= Connection History =========
Number of request(s) : 5
NO.     IP              PID     PORT    TIME
1       127.0.0.1       3448    7824    Mon May 20 07:15:38 2019
2       127.0.0.1       3426    7312    Mon May 20 07:15:15 2019
3       127.0.0.1       3425    6800    Mon May 20 07:15:11 2019
4       127.0.0.1       3423    6288    Mon May 20 07:15:07 2019
5       127.0.0.1       3418    4752    Mon May 20 07:14:51 2019
============= New client =============
IP : 127.0.0.1
Port : 8336
PID : 3450
=====================================
========= Disconeected Client ========
IP : 127.0.0.1
Port : 8336
=====================================
========= Connection History =========
Number of request(s) : 6
NO.     IP              PID     PORT    TIME
1       127.0.0.1       3450    8336    Mon May 20 07:15:40 2019
2       127.0.0.1       3448    7824    Mon May 20 07:15:38 2019
3       127.0.0.1       3426    7312    Mon May 20 07:15:15 2019
4       127.0.0.1       3425    6800    Mon May 20 07:15:11 2019
5       127.0.0.1       3423    6288    Mon May 20 07:15:07 2019
6       127.0.0.1       3418    4752    Mon May 20 07:14:51 2019
```

/home/sp2015722025/work - Mozilla Firefox

/home/sp2015722025/work

127.0.0.1:40229

Welcome to System Programming Http

Directory path : /home/sp20157522025

total 4468

Name	Permission	Link	Owner	Group	Size	Last Modified
A	drwxrwxr-x	3	sp20157522025	sp20157522025	4096	May 05 07:39
abTde	drwxrwxr-x	3	sp20157522025	sp20157522025	4096	Apr 22 21:12
accessible.usr	-rw-rw-r--	1	sp20157522025	sp20157522025	47	May 15 18:13
adv_server.c	-rwxrw-rw-	1	sp20157522025	sp20157522025	885	May 13 18:45
Desktop	drwxr-xr-x	2	sp20157522025	sp20157522025	4096	Apr 18 01:09
Documents	drwxr-xr-x	2	sp20157522025	sp20157522025	4096	Apr 18 01:09
Downloads	drwxr-xr-x	2	sp20157522025	sp20157522025	4096	May 09 06:57
ET	drwxrwxr-x	2	sp20157522025	sp20157522025	4096	Apr 18 01:09
HONG->test	lrwxrwxrwx	1	sp20157522025	sp20157522025	4	Apr 26 00:35
IMAGE->image2.png	lrwxrwxrwx	1	sp20157522025	sp20157522025	10	May 09 05:35
image.jpg	-rwxrw-rw-	1	sp20157522025	sp20157522025	1953289	Mar 04 14:57
image2.png	-rwxrw-rw-	1	sp20157522025	sp20157522025	22651	May 06 07:28
J	drwxrwxr-x	3	sp20157522025	sp20157522025	4096	May 09 01:11
Makefile	-rw-rw-r--	1	sp20157522025	sp20157522025	40	May 09 06:52
Min->test	lrwxrwxrwx	1	sp20157522025	sp20157522025	4	Apr 13 05:59

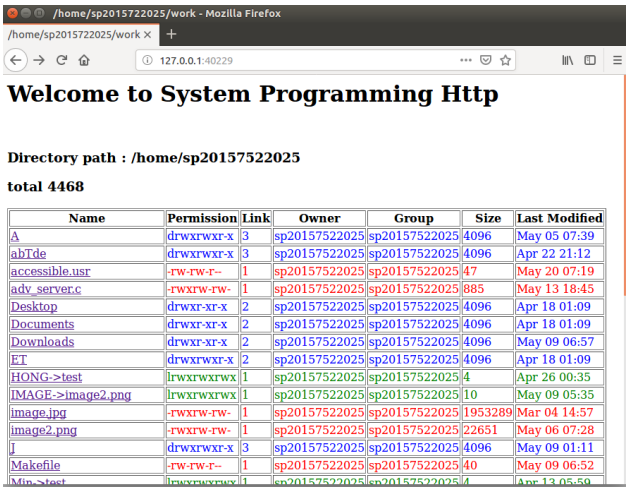
```
========= Connection History =========
Number of request(s) : 17
NO.      IP              PID     PORT    TIME
1        127.0.0.1       3475    16016   Mon May 20 07:16:55 2019
2        127.0.0.1       3472    15504   Mon May 20 07:16:53 2019
3        127.0.0.1       3469    14480   Mon May 20 07:16:47 2019
4        127.0.0.1       3468    13968   Mon May 20 07:16:46 2019
5        127.0.0.1       3467    13456   Mon May 20 07:16:46 2019
6        127.0.0.1       3466    12944   Mon May 20 07:16:45 2019
7        127.0.0.1       3465    12432   Mon May 20 07:16:44 2019
8        127.0.0.1       3464    11920   Mon May 20 07:16:43 2019
9        127.0.0.1       3463    11408   Mon May 20 07:16:42 2019
10       127.0.0.1       3462    10896   Mon May 20 07:16:42 2019
```

또한 다음과 같이 연결 횟수는 17번이지만 10개의 기록만 출력하며, 아래와 같이 지속적으로 접속을 한다고 해도 최신 history만 가져온다.

```
========= Connection History =========
Number of request(s) : 20
NO.      IP              PID     PORT    TIME
1        127.0.0.1       3485    18576   Mon May 20 07:18:28 2019
2        127.0.0.1       3484    18064   Mon May 20 07:18:27 2019
3        127.0.0.1       3482    17552   Mon May 20 07:18:25 2019
4        127.0.0.1       3475    16016   Mon May 20 07:16:55 2019
5        127.0.0.1       3472    15504   Mon May 20 07:16:53 2019
6        127.0.0.1       3469    14480   Mon May 20 07:16:47 2019
7        127.0.0.1       3468    13968   Mon May 20 07:16:46 2019
8        127.0.0.1       3467    13456   Mon May 20 07:16:46 2019
9        127.0.0.1       3466    12944   Mon May 20 07:16:45 2019
10       127.0.0.1       3465    12432   Mon May 20 07:16:44 2019
```

```
sp20157522025@ubuntu:~$ cat accessible.usr
128.134.52.61
192.168.*.1
128.*.*.62
127.0.*.*
```

다음과 같이 허용된 IP를 설정하였으며,



127.0.*.*에 해당하기 때문에 잘 동작하는 것을 확인할 수 있다. 하지만 다음과 같이 허용 IP를 변경하면

```
sp20157522025@ubuntu:~$ cat accessible.usr
128.134.52.61
192.168.*.1
128.*.*.62
111.*.*.*
```

다음과 같이 deny되는 것을 확인할 수 있다.

5. Conclusion

3-3과제는 개념적인 이해만 하면 구현하기 어렵지 않은 과제였다. 해당 과제를 하면서 헷갈렸던 fork가 반환하는 값을 정학하게 알게 되었다. Fork를 통하여 부모 프로세스에 반환되는 값은 자식프로세서의 ID이며, 자식 프로세서는 0을 반환한다. 해당 문제를 고민했던 이유는 자식 프로세서의 id정보를 부모 프로세서에서 저장하여 history내역으로 보여주어야 하는데 wait를 통하여 저장하게 되면 history출력 타이밍이 자식프로세서가 동작하는 것과 맞지 않아 이상한 ID를 출력하는 문제를 볼 수 있어서 연결하는 동시에 PID를 저장하는 방법이 필요했다. Fork를 통하여 문제를 해결하였고 프로그램이 잘 동작하는 것을 확인할 수 있었다.