시스템프로그래밍실습 보고서

Assignment 3-1

과 목 시스템프로그래밍실습

담당교수 이기훈교수님

학 과 컴퓨터정보공학부

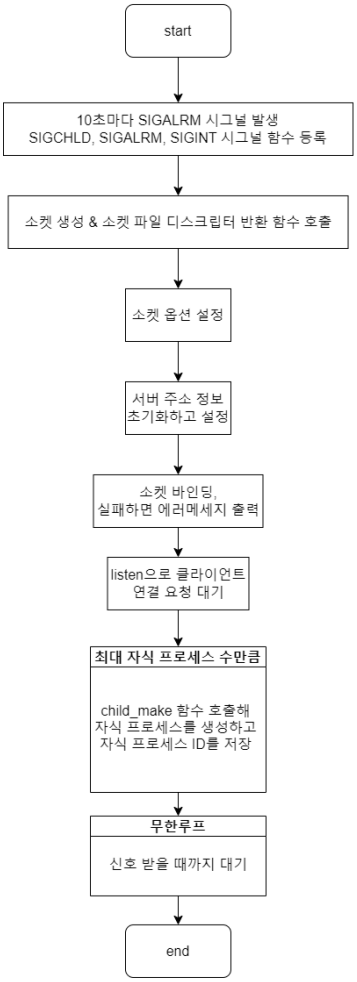
학 번 2021202058

이 름 송채영

1. Introduction

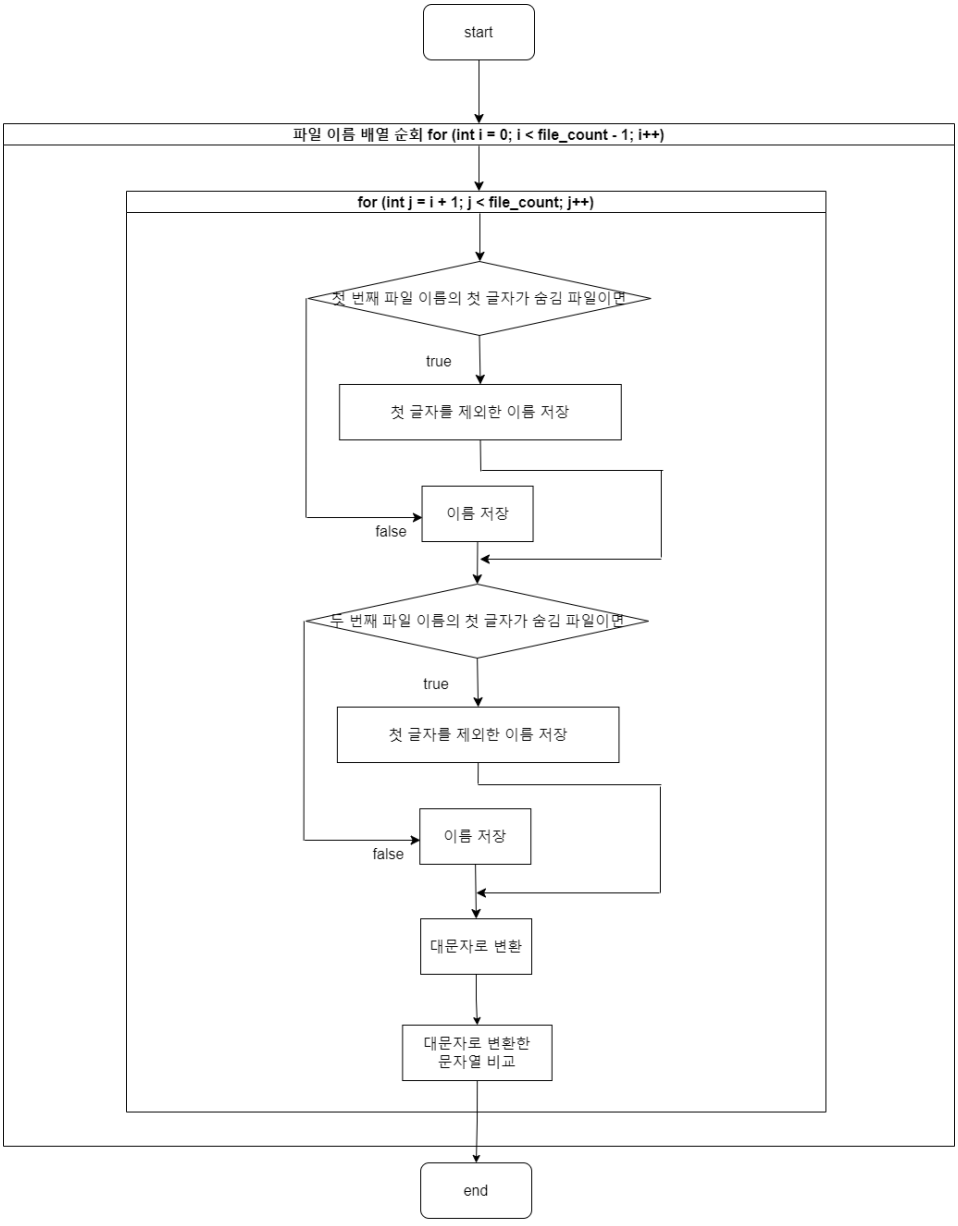
이번 과제는 Assignment 2-3 과제를 기반으로 해 Pre-forked 방식으로 서버를 구현하는 것이다. 부모 프로세스는 5 개의 자식 프로세스를 생성하고 관리하며 자식 프로세스에 연결된 클라이언트의 접속 기록을 출력해야 한다. (SIGCHLD, SIGALRM, SIGUSR1, SIGINT, SIGTERM) 시그널을 처리하고 연결 기록을 출력하는 작업을 추가해 구현한다. 또한, 프로세스 종료 시에는 모든 프로세스를 완전히 종료해야 하며, 좀비 프로세스가 생성되지 않도록 해야한다.

2. Flow chart

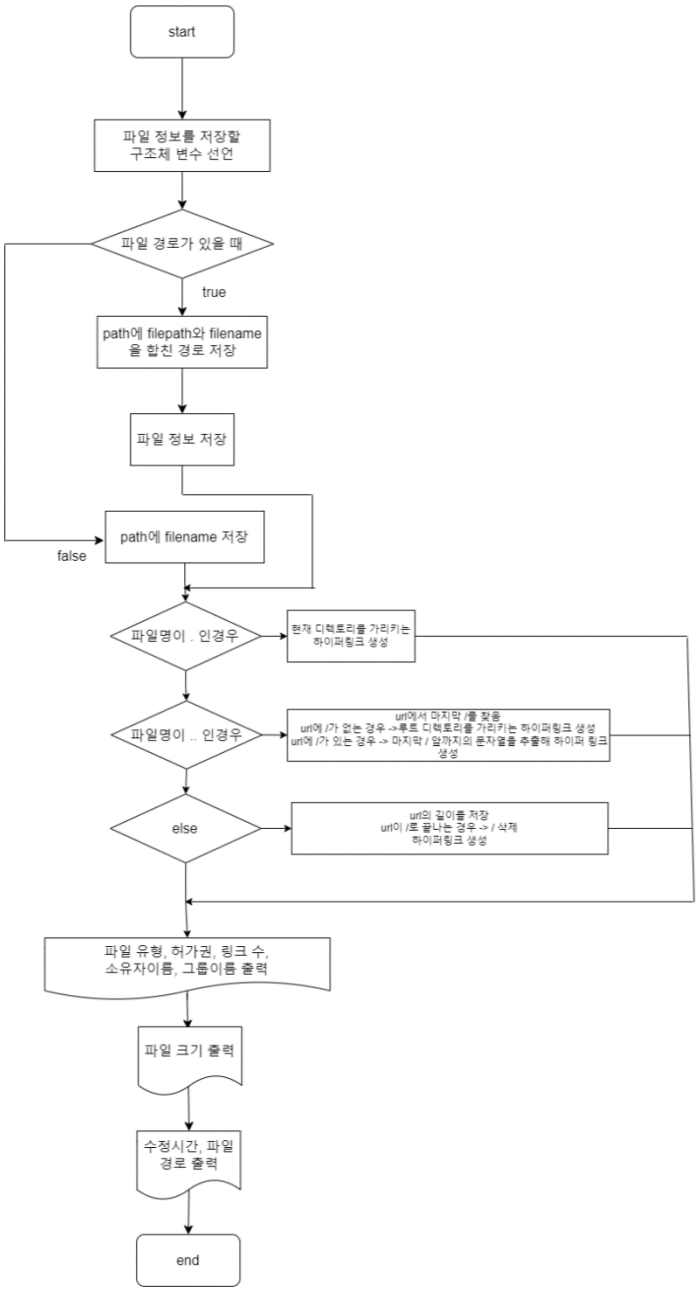


코드의 전체적인 흐름을 flow chart 로 나타내었다. 우선 main 함수에 대해 자세히 설명해보면 다음과 같다. 프로그램이 시작하면 10 초마다 SIGALRM 시그널이 발생하도록 alarm 을 설정하고 SIGCHLD, SIGALRM, SIGINT 시그널 핸들러를 handle_signal 로 등록한다. 소켓을 생성하고 소켓 파일 디스크립터 반환 함수를 호출한 후 소켓 옵션을 설정한다. memset 을 사용하여 서버 주소를 초기화 하고 설정해준다. 소켓 바인딩을 실행하는데, 이때 실패하면 에러 메세지를 출력해주고, listen() 함수를 사용해

클라이언트의 연결 요청을 대기한다. 최대 자식 프로세스를 5 로 설정하고 최대 자식 프로세스만큼 child_make 함수를 호출해 자식 프로세스를 생성하고 자식 프로세스 ID 를 저장한다. 무한루프를 돌아 신호를 받을 때까지 대기한다. 프로그램을 종료한다.

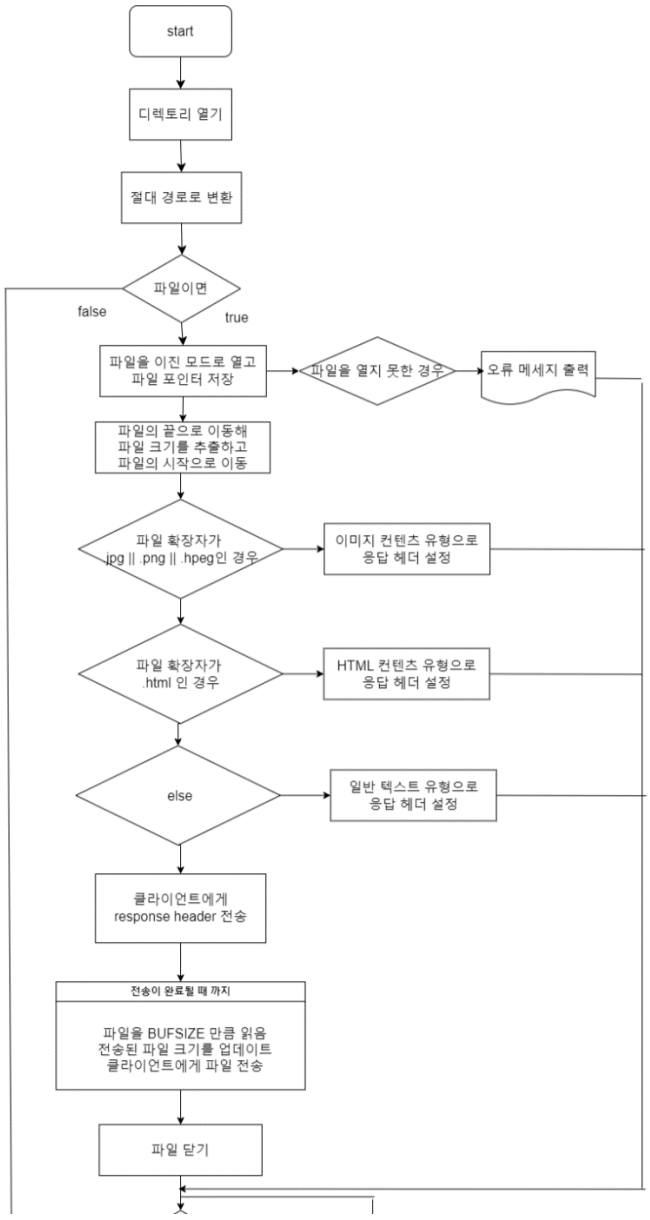


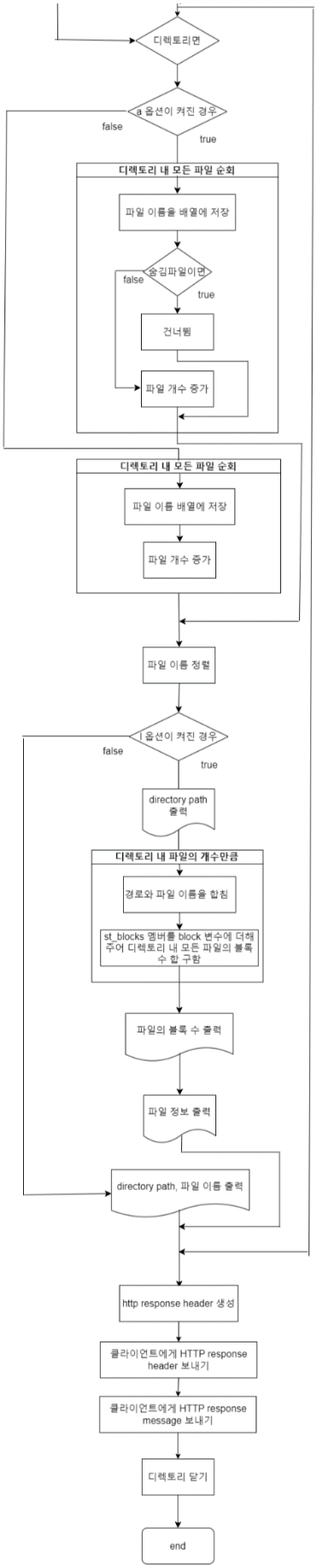
다음은 sort 함수의 flow chart 이다. 우선 파일의 개수만큼 반복한다. 첫 번째 파일이 숨김 파일일 경우, 파일의 이름을 길이만큼 반복하고 첫 글자를 제외한 이름을 저장한다. 숨김파일이 아닐 경우 이름을 저장해준다. 두 번째 파일도 동일하게 진행한다. 먼저 대문자로 변환해준 후 대문자로 변환한 문자열을 비교한다.



다음은 print_file_info, 파일 정보를 출력하는 함수의 flow chart 이다. 파일 정보를 읽어 구조체 변수인 filestat 에 저장한다. 파일 경로가 있고, filepath 와 filename 이 같지 않으면 파일 경로가 존재하는 경우이므로 경로와 파일 이름을 합치며, 파일 경로가 없는 경우 파일이름을 정보로 읽는다. 파일명이 ., .., else 인 경우로 나누어 하이퍼링크를 준다. 먼저 .인 경우 현재 디렉토리를 가리키는 하이퍼링크를 생성하며, ..일 경우 url 에서 마지막 /을 찾고 /이 있으면 루트 디렉토리를 가리키는 하이퍼링크를 생성하고, 없을 경우 마지막 / 앞까지의 문자열을 추출한 후 하이퍼링크를 생성해준다. else 인 경우 url 의 길이를 저장한 후 url 이 /로 끝나면 /을 삭제한 후 하이퍼링크를 생성하도록

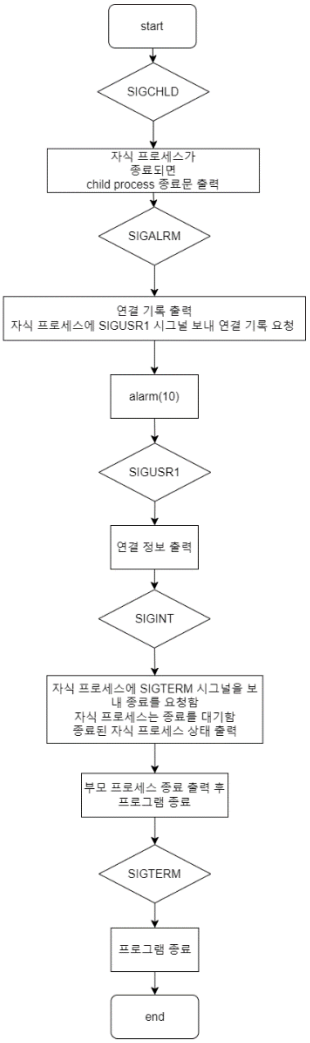
하였다. 파일 유형, 허가권, 링크 수, 소유자,호출해주며, 파일 크기, 수정 시간, 파일 경로를 출력한 후 프로그램을 종료한다.



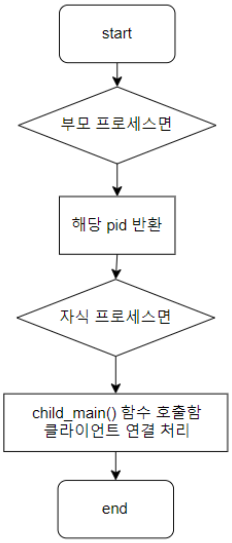

디렉토리면
a 옵션이 켜진 경우
false
true
디렉토리 내 모든 파일 순회
파일 이름을 배열에 저장
숨김파일이면
false
true
건너뜀
파일 개수 증가
디렉토리 내 모든 파일 순회
파일 이름 배열에 저장
파일 개수 증가
파일 이름 정렬
l 옵션이 켜진 경우
false
true
directory path 출력
디렉토리 내 파일의 개수만큼
경로와 파일 이름을 합침
st_blocks 멤버를 block 변수에 더해 주어 디렉토리 내 모든 파일의 블록 수 합 구함
파일의 블록 수 출력
파일 정보 출력
directory path, 파일 이름 출력
http response header 생성
클라이언트에게 HTTP response header 보내기
클라이언트에게 HTTP response message 보내기
디렉토리 닫기
end

다음은 list files 함수의 flow chart 이다. 파일을 열어준 후 절대경로로 변환한다. 연 파일이 파일일때와 디렉토리 일 때로 나눈다. 파일일 때, 파일을 이진 모드로 열고 파일 포인터를 file 변수에 저장한다. 파일을 열지 못했을 경우 오류 메세지를 출력한다. 파일의 끝으로 이동해 파일 크기를 추출한 후 파일의 시작으로 이동한다. 파일 확장자에 따라 content-type 을 설정하였는데, .jpg, .png, .jpeg 일 때 이미지 컨텐츠 유형으로 응답 헤더를 설정한다. 파일 확장명이 .html 인 경우 HTML 컨텐츠 유형으로 응답 헤더를 설정한다. else 일 경우 일반 텍스트 유형으로 응답 헤더를 설정하고 클라이언트에게 response header 를 전송하였다. 전송이 완료될 때 까지 반복하며 파일을 BUFSIZE 만큼 읽어서 전송된 파일 크기를 업데이트 하고 클라이언트에게 파일을 전송한다. 파일을 닫는다. 디렉토리일 경우 a 옵션이 꺼져 있을 때와 켜져 있을 때, l 옵션이 켜져 있을 때로 나눈다. a 옵션이 꺼져 있으면 디렉토리 내 모든 파일을 순회하며 파일이름을 배열에 저장한다. .과 ..은 건너뛰고 그 외의 파일은 파일 개수를 증가시킨다. a 옵션이 켜져 있을 경우, 디렉토리 내 모든 파일을 파일 이름을 배열에 저장하고 파일 개수를 증가시킨다. 파일 이름 정렬 함수를 호출해 정렬한다. l 옵션이 켜져 있으면, 디렉토리 경로와 total 을 출력해주고 파일 정보 출력 함수를 호출해 출력한다. HTTP response header 를 생성해 작성해 준 후, 클라이언트에게 HTTP response header 와 message 를 보낸 후 디렉토리를 닫고 프로그램을 종료한다.

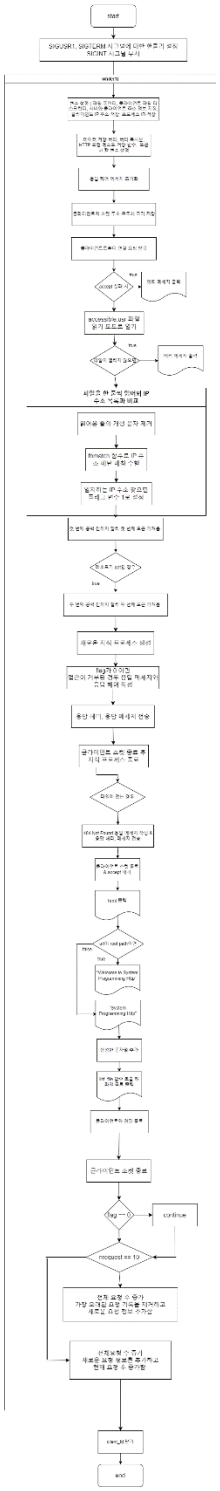
위 함수에서 html 에 directory path 와 total 그리고 filename 을 table 형태로 출력해주기 위해 html 코드로 출력하는 부분을 추가했다. 또한 html_ls.html 파일일 경우 건너뛰는 코드를 추가해 html_ls.html 파일이 출력되지 않도록 하였다. flow chart 의 흐름은 기존과 변함이 없다.



다음으로 handle_signal 함수에 대한 flow chart 이다. 우선 SIGCHLD 시그널 핸들러일 경우 자식 프로세서가 종료되면, 모든 종료된 자식 프로세스를 처리하며 Child process 가 종료되었다는 구문을 출력해준다. signal 이 SIGALRM 일 경우 연결 기록을 출력한다. 자식 프로세스의 개수만큼 자식 프로세스에 SIGUSR1 시그널을 보내 연결 기록을 요청한다. alarm 을 10 으로 설정한다. SIGUSR1 시그널의 경우 최근 연결 기록부터 출력하는 인덱스 n 과 trequest 부터 시작하는 인덱스 i 를 변수로 선언한 후 while (i > (trequest - nrequest)) while 문 안에서 연결정보를 출력한다. 이후 인덱스를 감소시키고 연결 기록의 이전 항목으로 이동시킨다. 다음으로 SIGINT 시그널인 경우 자식 프로세스의 개수만큼 자식 프로세스에 SIGTERM 시그널을 보내 종료를 요청한다. 자식 프로세스의 종료를 대기하며 종료된 자식 프로세스 상태를 출력한다. for 문 밖에서는 부모 프로세스의 종료를 출력한 후 프로그램을 종료한다. 마지막으로 SIGTERM 의 경우 프로그램을 종료한다.



다음으로 child_make 함수이다. 부모 프로세스의 경우 해당 pid 를 반환하며 자식 프로세스의 경우 child_main 함수를 호출해 클라이언트 연결을 처리한다.



마지막으로 다음으로 child_main 함수이다. SIGUSR1, SIGTERM 시그널에 대한 핸들러를 설정하고, SIGINT 시그널은 무시한다. 필요한 변수를 선언한 후 while 이 1 일 때 클라이언트 IP 주소 저장 변수와 클라이언트로부터 받은 데이터를 저장할 버퍼, 버퍼 복사본, HTTP 요청 메소드 저장 변수, HTTP 요청 메시지 토큰화 할 포인터를 선언한 후 응답 헤더와 메시지를 초기화한다. 클라이언트의 소켓 주소 구조체 크기를 저장한 후

클라이언트로부터 연결 요청을 받으며, accept 를 실패하면 에러 메세지를 출력해준다. accessible.usr 파일을 읽기 모드로 열어 파일을 열지 못했을 경우 오류 메시지를 출력한다. 파일을 한 줄씩 읽어와 IP 주소 목록과 비교한다. 읽어온 줄의 개행 문자를 제거하고 fnmatch 함수를 사용하여 IP 주소 패턴 매칭을 수행한다. 이때 일치하는 IP 주소를 찾으면 플래그 변수 1 로 설정한다. 첫 번째 공백 전까지 문자열을 잘라서 첫 번째 토큰을 가져오는데 이때 토큰이 없는 경우 무시한다. url 배열을 초기화한 후 메서드가 GET 인 경우 두 번째 공백 전까지 문자열을 잘라서 두 번째 토큰을 가져온다. flag 가 0 이면 접근이 거부된 경우 응답 메시지와 응답 헤더를 작성하고 응답 헤더와 메시지를 전송한다. 클라이언트 소켓을 종료하고 다음 클라이언트를 요청하고 자식 프로세스를 종료한다. 현재 작업 디렉토리를 cwd 에 저장하고 요청한 파일의 경로를 만들어, 경로가 없는 경우 404 Not Found 응답 메시지를 작성한다. 응답 헤더와 메시지를 전송한 후 클라이언트 소켓을 종료한다. 다시 accept 대기로 돌아간 후 html 출력을 위해 필요한 부분을 구현한다. url 이 root path 인 경우 lflag 를 1 로 설정해주고 "Welcome to System Programming Http"을 출력한다. 하위디렉토리일 경우 aflag 와 lflag 를 1 로 설정해주고 "System Programming Http"를 출력한다. cwd 의 파일 목록을 출력한 후 클라이언트와 연결과 소켓과의 연결을 종료한다. flag 가 0 이면 다음 클라이언트 요청 처리를 위해 반복문의 처음으로 이동한다. nrequest 가 10 일 경우 전체 요청 수를 증가하고 가장 오래된 요청 기록을 제거한 후 새로운 요청 정보를 추가한다. else 일 경우 전체 요청 수를 증가하고 새로운 요청 정보를 추가하고 현재 요청 수를 증가한다. client_fd 를 닫고 프로그램을 종료한다.

3. Pseudo Code

```
선택정렬() //sort
{
      for(파일 수만큼 파일 이름 배열 순회)
      {
            if(첫번째 파일 첫 글자가 숨김 파일이면)
            {
                  filename[i]의 길이만큼 반복하고 첫 글자 제외하고 저장
            }
            else(숨김파일이 아니면)
            {
                  이름 저장
            }
            if(두번째 파일 첫 글자가 숨길 파일이면)
            {
                  filename[i]의 길이만큼 반복하고 첫 글자 제외하고 저장
            }
            else(숨김파일이 아니면)
            {
                  이름 저장
            }
            hidden_file 1 대문자로 변환
            hidden_file 2 대문자로 변환
            대문자로 변환한 문자열 비교
}
```

sort 함수의 pseudo code 이다.

```
파일 정보 출력() //print_file_info
{
	파일 정보 읽어 구조체 변수 filestat에 저장
	파일 권한 정보 저장할 변수 선언
	if(파일 경로가 있으면)
	{
		파일 경로와 파일 이름을 합치고 stat함수로 파일 정보 읽음
	}
	else
	{
		파일 이름으로 정보 읽음
	}
	if(파일명이 .인 경우) 현재 디렉토리를 가리키는 하이퍼링크 생성
	else if(파일명이 ..인 경우)
	{
		url에서 마지막 /를 찾음
		url에 /가 없는 경우 -> 루트 디렉토리를 가리키는 하이퍼링크 생성
		url에 /가 있는 경우 -> 마지막 / 앞까지의 문자열을 추출 후 하이퍼링크 생성
	}
	else
	{
		url의 길이 저장
		url이 /로 끝나는 경우 -> /를 삭제
		하이퍼링크 생성
	}
	// 파일 유형, 허가권, 링크 수 , 소유자, 그룹, 출력
	if(파일 유형이 디렉토리면) permission 배열의 0번째는 d
	else if(파일 유형이 링크면) permission 배열의 0번째는 l
	else permission 배열의 0번째는 -
	파일 허가권 정보 설정

	if(디렉토리면) 파란색으로 출력
	else if(link 파일이면) 초록색으로 출력
	else 빨간색으로 출력

	파일 크기 출력
	수정시간, 파일 경로 출력
}
```

print_file_info 함수의 pseudo code 이다.

```
파일 목록 출력() //list_files
{
	디렉토리 열기
	절대경로로 변환
	if(파일이면)
	{
		파일을 이진 모드로 열고 파일 포인터 저장
		if(파일 열지 못하면) 에러메세지 출력
		파일 끝으로 이동해 파일 크기 추출한 후 파일 시작으로 이동

		파일 확장자에 따라 content-type 설정
		if (jpg, png, jpeg)
		{
			이미지 콘텐츠 유형으로 응답 헤더 설정
		}
		else if(.html)
		{
			HTML 컨텐츠 유형으로 응답 헤더 설정
		}
		else
		{
			일반 텍스트 유형으로 응답 헤더 설정
		}
		클라이언트에게 response header를 전송
		while(전송이 완료될 때까지)
		{
			파일을 BUFSIZE 만큼 읽음
			전송된 파일 크기를 업데이트
			클라이언트에게 파일을 전송
		}
		파일 닫기
	}
	else //디렉토리면
	{
		if(a 옵션이 꺼져있는 경우)
		{
			while(디렉토리 내 모든 파일 순회)
			{
				html_ls.html 파일은 건너뜀
				숨김파일이면 넘어감
				아니면 파일 개수 증가
			}
		}
		else
		{
			while(디렉토리 내 모든 파일 순회)
			{
				html_ls.html 파일은 건너뜀
				파일 이름 배열에 저장
			}
		}
		파일 이름 정렬
		if(l 옵션이 켜져있는 경우)
		{
			directory path, total 출력
			HTML 파일에 테이블 태그 출력
			HTML 파일에 테이블 헤더 삽입
			파일 정보 출력
		}
		HTTP response header 생성
		클라이언트에게 response header, message 보내기
	}
	디렉토리 닫기
}
```

list_files 함수의 pseudo code 이다.

```
handle_signal
{
        if(signal == SIGCHLD)
        {
                자식 프로세서가 종료되면, 모든 종료된 자식 프로세서 처리
        }
        if(signal == SIGALRM)
        {
                연결 기록 출력
                for(자식 개수만큼)
                {
                        자식 프로세스에 SIGUSR1 시그널을 보내 연결 기록을 요청
                }
                alarm(10)
        }
        if(signal == SIGUSR1)
        {
                while(i > (trequest - nrequest))
                {
                        연결 정보 출력
                        인덱스 감소
                        연결 기록 이전 항목으로 이동
                }
        }
        if(signal == SIGUNT)
        {
                for(자식 개수만큼)
                {
                        자식 프로세스에 SIGTERM 시그널 보내 종료 요청
                        자식 프로세스 종료 대기
                        종료된 자식 프로세스 상태 출력
                }
                부모 프로세스 종료 출력
                프로그램 종료
        }
        if(signal == SIGTERM)
        {
                프로그램 종료
        }
}
```

handle_signal 함수의 pseudo code 이다.

```
child_make
{
        자식 프로세스 생성
        if(부모 프로세스면)
        {
                해당 pid 반환
        }
        if(자식 프로세스면)
        {
                child_main() 함수 호출해 클라이언트 연결 처리
        }
}
```

child_make 함수의 pseudo code 이다.

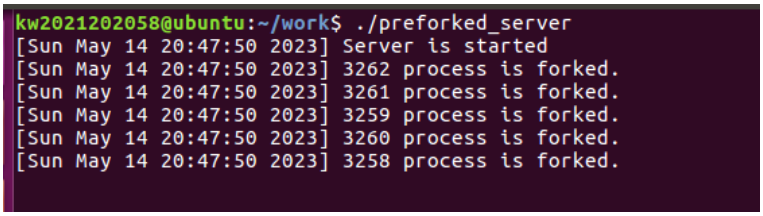
```
child_main
{
		SIGUSR1, SIGTERM 시그널 핸들러 설정
		SIGINT 시그널 무시

		while(1)
		{
			응답 헤더, 메세지 초기화
			클라이언트로부터 연결 요청 받음
			if(accept 실패 시) 에러 메세지 출력

			accessible.usr 파일을 읽기 모드로 열기
			if(파일을 열지 못하면) 오류메세지 출력

			while(파일에서 한 줄씩 읽어와 IP 주소 목록과 비교)
			{
				읽어온 줄의 개행 문자 제거
				fnmatch 함수 사용해서 IP 주소 패턴 매칭 수행
					일치하는 IP 주소 찾으면 플래그 변수 1로 설정
					break;
			}
			파일 닫기

			첫번째 공백 전까지 문자열 잘라서 첫 번째 토큰 가져옴
			if(토큰이 없는 경우) continue
			url 배열 초기화
			if(method가 GET인 경우)
			{
				두번째 공백 전까지 문자열 잘라서 두 번째 토큰 가져옴
			}
			현재 작업 디렉토리 cwd변수에 저장

			새로운 자식 프로세스 생성

			접근이 거부된 경우 응답 메세지와 헤더 작성
			응답 헤더와 메세지 전송
			continue

			if(파일이 없는 경우)
			{
				404 Not Found 응답 메세지 작성
				응답 헤더, 메세지 전송
				클라이언트 소켓 종료
				accpet대기로 돌아감
				continue
			}
			if(url이 root path)
			{
				lflag = 1
				"Welcome to System Programming Http" 태그
			}
			else //하위 디렉토리
			{
				aflag =1, lfalg = 1
				"System Programming Http"
			}
			list_files 함수 호출해 cwd의 파일 목록 출력

			if(flag == 0) continue
			if(nrequest == 10)
			{
				전체 요청 수 증가
				가장 오래된 요청 기록 제거, 새로운 요청 정보 추가
			}
			else
			{
				전체 요청 수 증가
				새로운 요청 정보를 추가
				현재 요청 수 증가
			}
		}
}
```

child_main 함수의 pseudo code 이다.

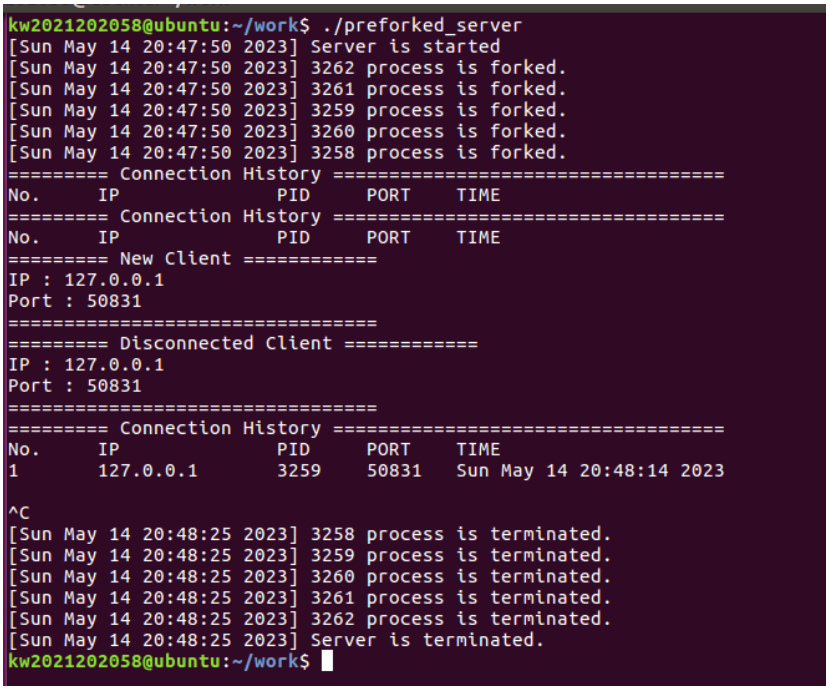
```
main
{
        10초마다 SIGALRM 시그널 발생
        SIGCHLD, SIGALRM, SIGINT 시그널 핸들러 함수로 등록

        소켓 생성 & 소켓 파일 디스크립터 반환 함수 호출
        실패 -> 에러 메세지 출력

        소켓 옵션 설정

        서버 주소 초기화 하고 설정함

        소켓 bind
        실패 _> 에러 메세지 출력

        클라이언트 연결 요청 대기

        최대 자식 프로세스 수 = 5

        for(최대 자식 프로세스 수만큼 반복)
        {
                child_make 함수 호출해 자식 프로세스 생성 하고 자식 프로세스 ID를 저장

        }
        for(무한루프)
        {
                신호 받을 때까지 대기
        }
}
```

main 함수의 pseudo code 이다.

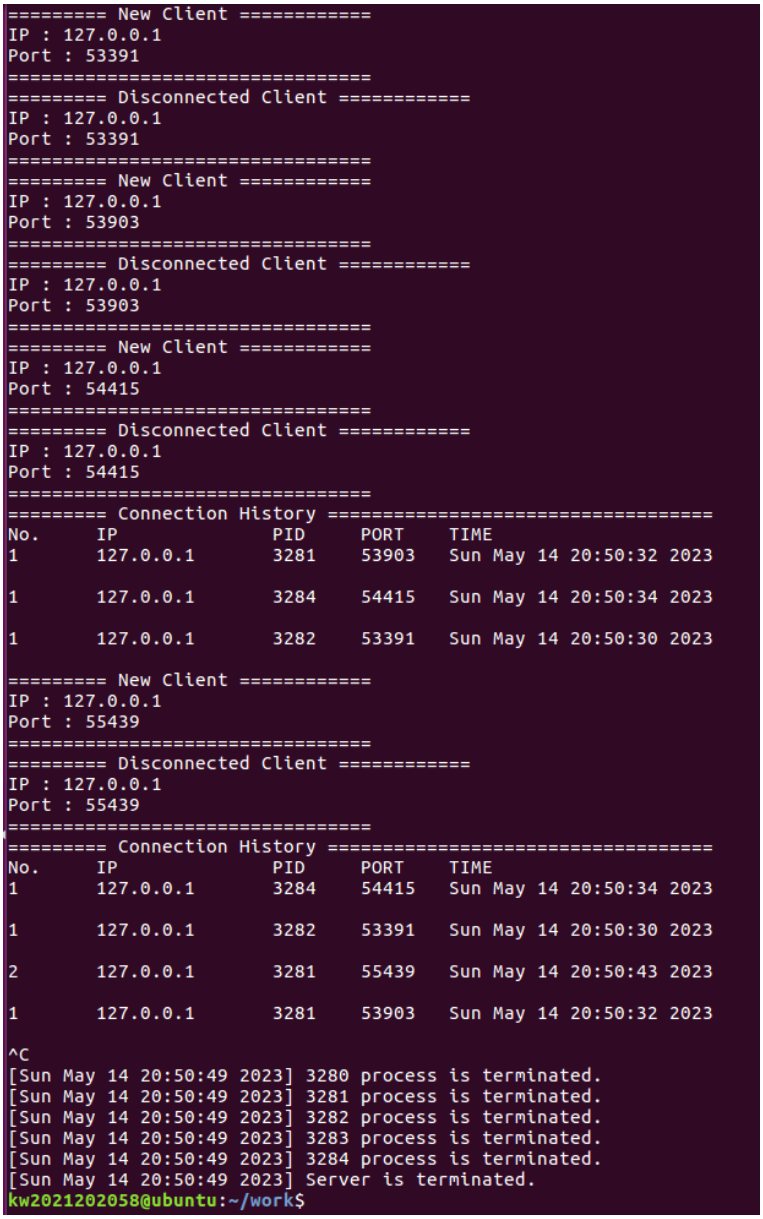
4. 결과화면



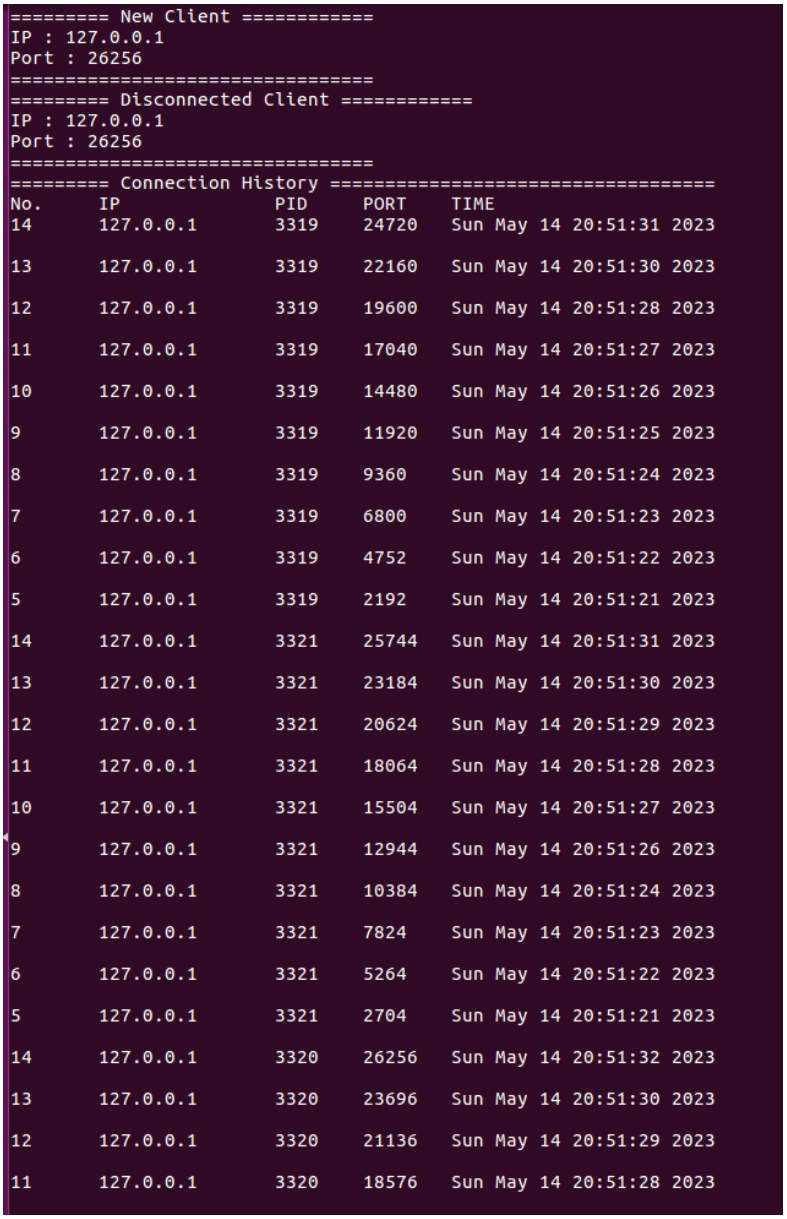
server program 이 시작되고 Child process 가 생성될 때의 출력화면으로 문제에서 요구하는 대로 잘 출력이 되는 것을 확인할 수 있다.



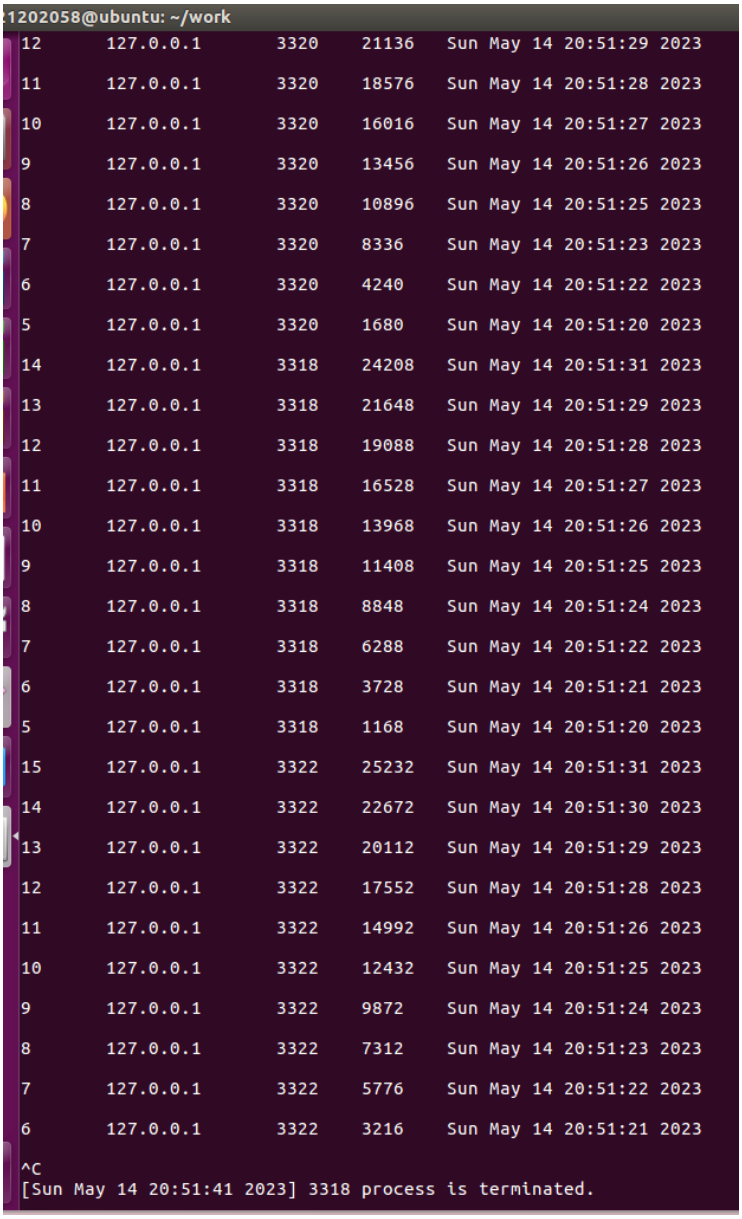
10 초 마다 자식 프로세스에 연결된 client 의 접속 기록인 connection history 가 잘 출력되는 것을 확인할 수 있다. History 제목은 부모가 출력하며 하위에 있는 내용은 자식 프로세스가 출력하는 것을 보여준다.



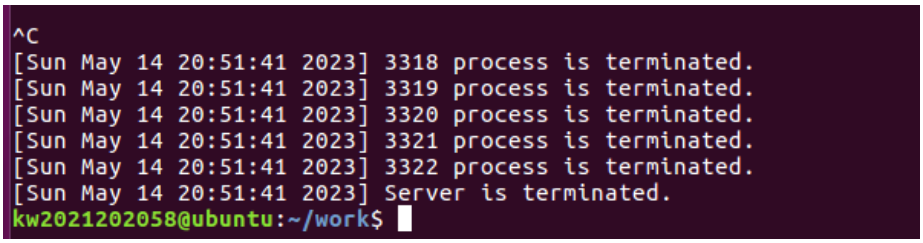
client 를 접속하고 종료했을 때 New client 와 Disconnected client 가 잘 출력이 되며 connection history 역시 잘 출력되는 것을 확인할 수 있다.



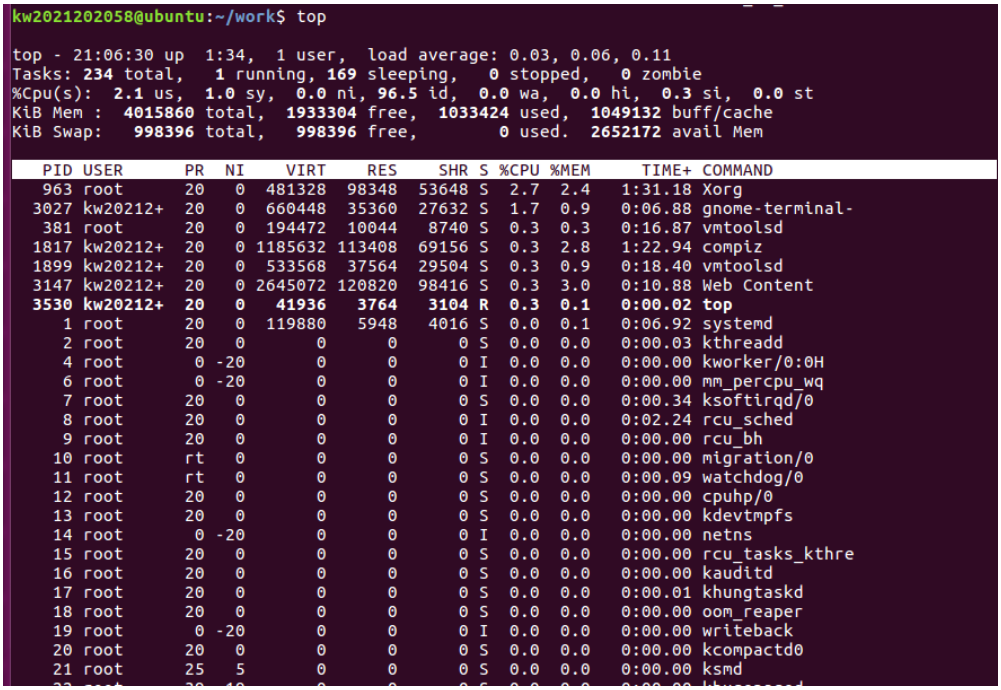
5 개의 child 프로세스는 하나 당 최대 10 개의 history 를 저장할 수 있으므로 3319 pid 는 14 부터 5 까지 10 개 저장하였고, 3321 pid 는 14 부터 5 까지 10 개 출력하는 것을 확인할 수 있다.



3320 pid 는 14 부터 5 까지 10 개 저장하였고, 3318 pid 역시 14 부터 5 까지 10 개 저장하였다. 마지막으로 3322 pid 는 15 부터 6 까지 10 개 저장하는 것을 확인할 수 있다. history 의 번호는 즉 No.는 자식 프로세스마다 개별적으로 유지하는 것 역시 확인할 수 있다.



또한 ctrl + c 로 종료 시, 즉 SIGINT signal 이 발생했을 때 child process 를 먼저 종료한 후 server 를 종료하며 각각에 알맞은 출력문이 출력 되는 것을 확인할 수 있다.



zombie 프로세스가 생성되지 않았다는 것을 확인하는 결과화면이다. 오른쪽 상단에 "0, zombie"라고 뜬 것을 통해 zombie 프로세스가 생성되지 않았음을 알 수 있다.

5. 고찰

지난 과제까지 socket_fd 와 client_fd 를 전역변수로 선언해 두었다. 이번 과제 역시 처음에는 전역 변수로 선언해 두는데 실행했을 때 Server : Can't open stream socket 에러문이 출력되었고, 이는 소켓이 정상적으로 생성되지 않음을 의미한다. 정확한 오류의 원인은 모르겠지만, 여러 프로세스가 동시에 전역 변수에 접근하거나 수정하려고 해서 오류가 발생했던 것 같다. 이를 해결하기 위해 해당 변수를 사용하는 함수에 인자나, 변수로 선언해주었고 오류를 해결할 수 있었다. 이번 과제에서 history 내용을 출력하는 것과 관련해서 동기화 문제를 고려하지 않는다고 적혀 있는데, 이것이 의미하는 바가 무엇인지 잘 모르겠다. 현재 결과는 같은 pid 끼리 묶여서 잘 출력되는데 만약 같은 pid 끼리 출력이 되지

않아도 괜찮다는 의미로 예상된다. 또한 현재 시간을 출력해 주는 부분에서 마지막 ']' 부분이 다음 줄에 출력되었는데, 이 점을 개행 문자를 찾아 '₩0'으로 바꿔주어 출력이 잘 되도록 해결하였다.

6. Reference

시스템프로그래밍실습 강의자료