PA#4

2016310932 배현웅

1. 전체적인 구조
   1. 서버와 클라이언트의 통신은 과제에서 요구하는 방식으로 구성하였다. “\n” 을 기준으로 데이터를 받아 리다이렉션이 가능하게 하였고 PUT GET 명령어를 통해서 서버에서 요청한 명령을 실행한다.
   2. 클라이언트 하나당 쓰레드 하나를 연결하여 사용하였고. Entry\_lock, file\_lock 이라는 변수를 이용하여 접근을 제어했다.
   3. Entry\_lock을 이용해 각 엔트리, 해쉬테이블의 한 엔트리(인덱스)의 접근할 수 있는 쓰레드를 하나로 제한을 하였다. File\_lock을 이용하여 테이블에 있는 데이터를 파일로 내릴 때, 파일에 있는 데이터를 읽어올 때 간섭이 생기지 않도록 파일에 접근할 수 있는 쓰레드를 하나로 제어하였다.
2. 함수 설명

* Client.c



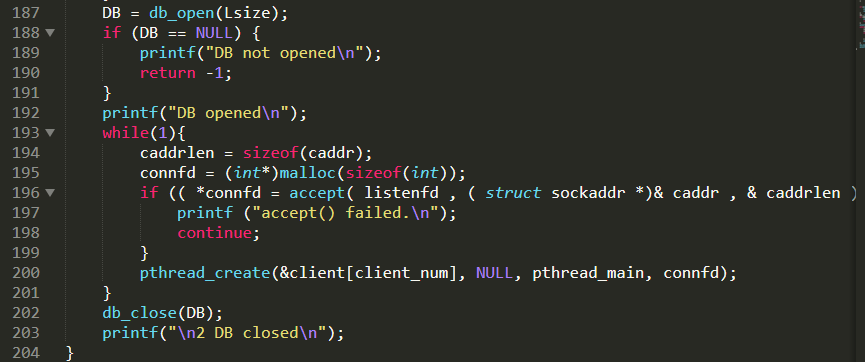
40-42 : stdin 에서 하나씩 받아와서 ‘\n’을 받을 때까지 받고 다음을 진행한다.

44-45 : stdin에서 받은 데이터를 cfd(서버와 연결되어있는 fd)로 보내주고 buf을 fflush 한다.

47-52 : 서버에서 보내준 데이터를 ‘\n’ 받을 때까지 받고 클라이언트에 출력해준다.(->명령에 대한 응답)

53-60 : 서버에서 받아온 데이터가 BYE라면 DISCONNECT를 보낸 후의 서버 응답이므로 종료하고. 56줄과 같이 받으면 서버에서 클라이언트를 더 이상 받을 수 없다는 의미이므로 클라이언트를 종료한다. 그후 buf변수를 초기화해주고 다시 받는다.

* Server.c



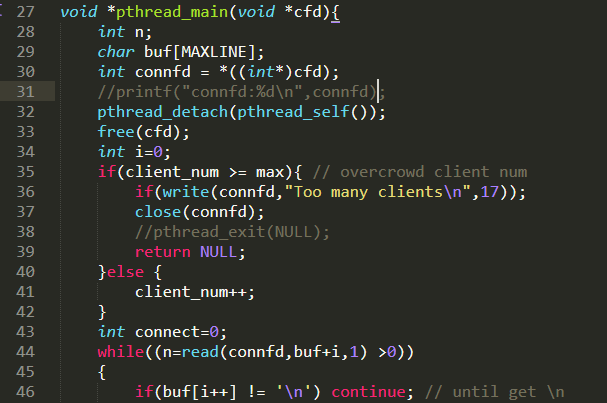
128-192 : DB오픈후 성공하면 192줄 출력

193-199 : accept 에 성공하지 못하면 다시 193줄로 간다.

200 : 연결에 성공했다면 연결된 connfd 와 같이 pthread를 만든다.

202-203 SIG\_INT, SIGTSTP 시그널을 통해 DB종료를 하기에 문제가 없지만 안정성을 위해 기입.

-pthread\_main

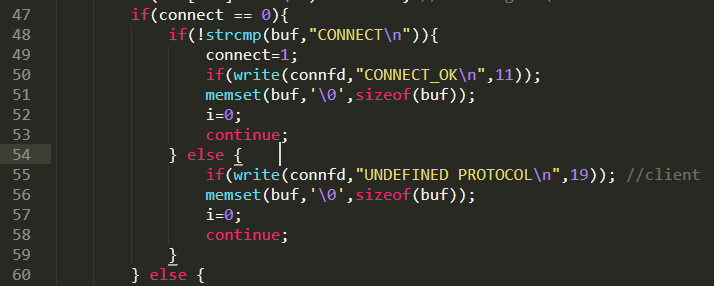


32 : detach를 통해 메인에서 쓰레드가 끝나기를 기다리지 않게 한다.

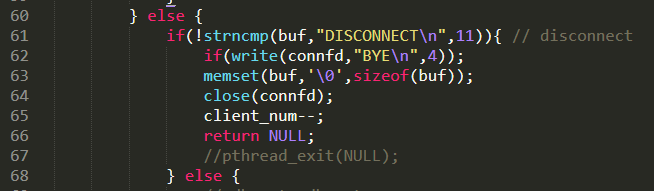
35-42 : max(최대 클라이언트 수)보다 client\_num(현재 클라이언트 수)이 넘기면 “Too many clients” 를 클라이언트로 보내주어 클라이언트 종료. 아니라면 클라이언트 수 증가

43 : CONNECT 연결 되었는지 확인하는 변수

44~ 클라이언트에서 받은 데이터를 ‘\n’ 기준으로 받아 명령 처리한다.



47-59 : CONNECT 라는 데이터를 받으면 CONNECT\_OK를 클라이언트에 보내주고 초기화, 그 후 명령어를 처리한다. 아니라면 UNDEFINED PROTOCOL을 보내주고 다시 대기한다.

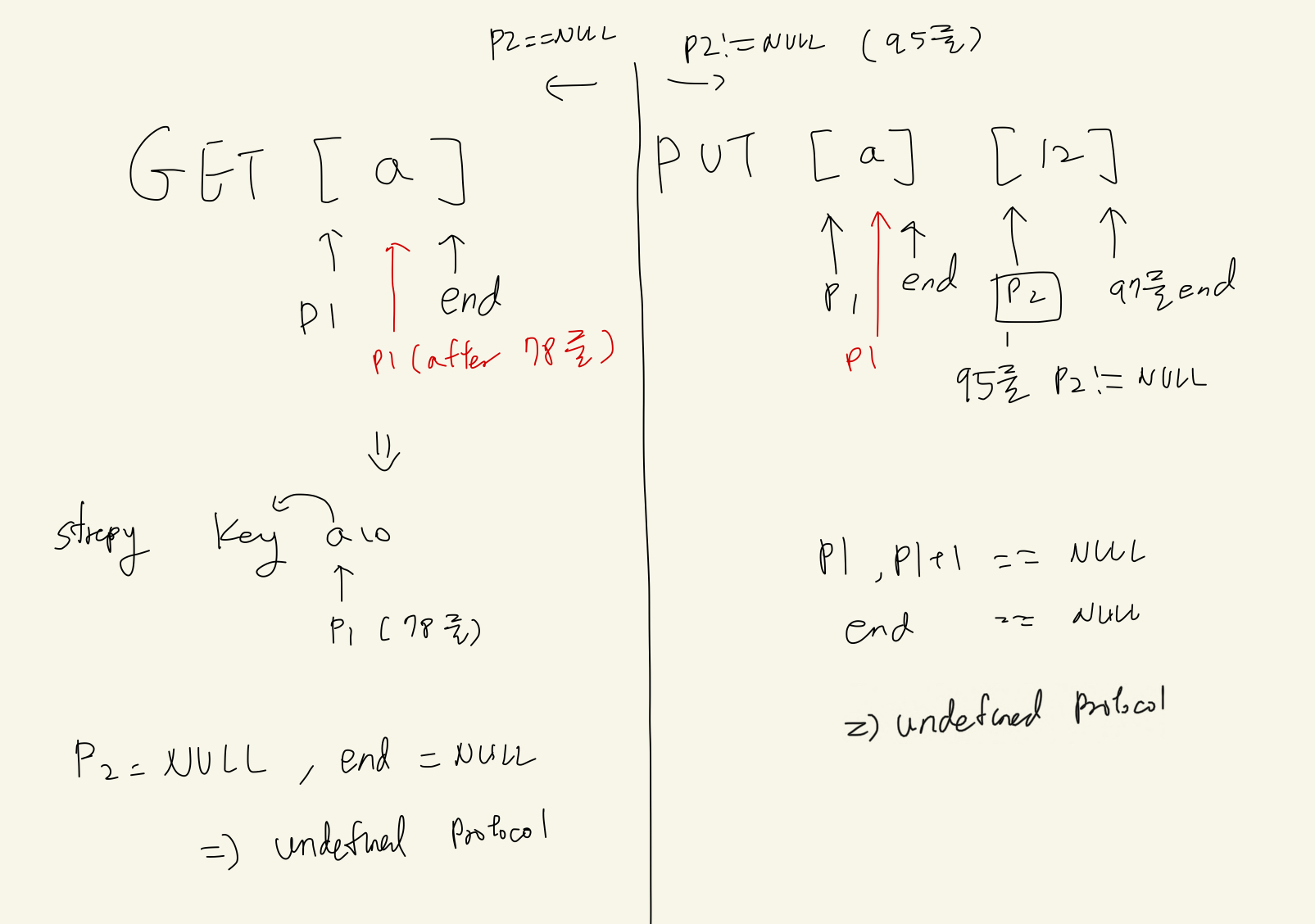


60~ CONNECT 명령어를 받은 이후(연결이 수립) 처리하는 부분

61-66 : DISCONNECT를 받으면 BYE를 클라이언트에 보내준 후, client\_num 감소, 쓰레드 종료



71-74 : val 변수에 value값을 저장하고, key 변수에 key값을 저장하기 위해 선언.



받은 데이터 파싱의 원리는 위와 같다. Strchr함수를 통해서 데이터를 받고 [ 와 ] 를 포인터로 주소를 정하여 ‘[‘ 하나 뒤부터 ‘]’을 널로 바꾸어 주어 저장해준다.

GET은 p2(두번째 ‘[‘ 찾는 포인터] 가 널이므로 104줄의 if문 의 과정을 넘어가고, PUT같은 경우는 if문을 실행한다. 올바르지 않은 데이터가 들어오면 NULL이 만들어지므로(‘[‘ 와 ‘]’ 을 찾을 수 없거나 ‘[‘ 이후에 문자가 없는경우) – UNDEFINED PROTOCOL을 출력하고 다시 입력을 받는다.



111- 113 앞에 GET이라는 문자를 받으면 113번 함수를 실행할 준비를 한다.

114-121 데이터가 없으면 GETINV 출력, 값이 존재하면 GETOK 출력



122~ PUT 명령어 실행준비

123-128 : p2가 널이라면 두번째 ‘[‘을 찾지못했다는 것이므로 잘못된 명령어 처리한다.

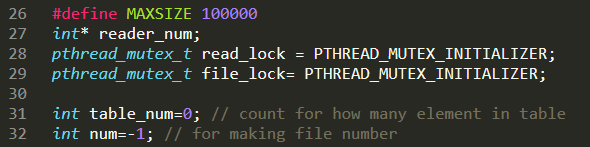
129-134 : PUT명령어 실행하고 클라이언트로 결과 보내준다.

134-139 : GET,PUT 받지 않고 다른 명령어 받으면 다시 입력받을 준비를 한다.

141-142 : 다시쓰기 위한 널 초기화

148-150: 쓰레드가 끝날려고 하면 close하고 클라이언트 수를 감소시킨 후(149줄) 종료한다.

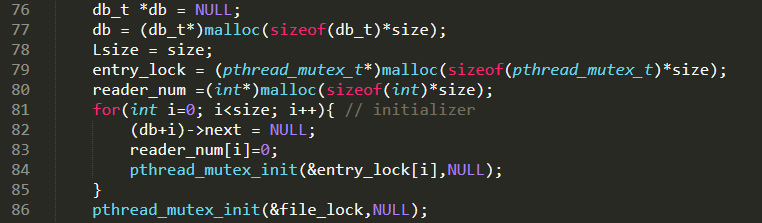
여러 쓰레드에서 명령어 처리하기 위한 쓰레드 락 설정 (IN DB.C)



28-29 : 쓰레드 락을 위한 변수 설정 및 초기화

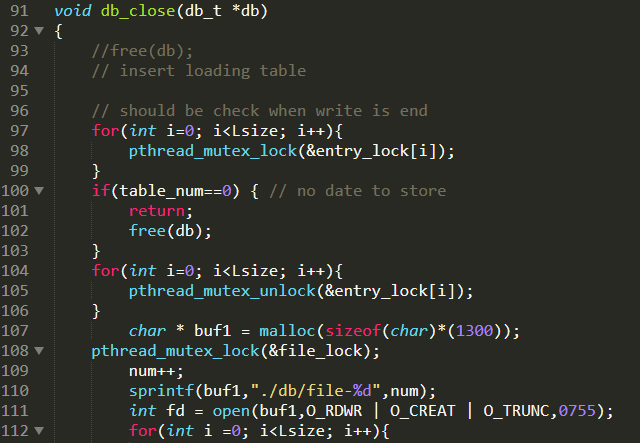
Entry\_lock 변수는 db.h 파일에 설정되어있다.

DB\_OPEN함수



79-86 : 변수 초기화 및 데이터 초기화

DB\_CLOSE 함수



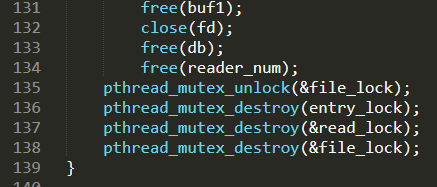
97-99 : DB를 파일로 내리기 전에 HASH\_TABLE 전체 엔트리에 락을 걸어 해쉬테이블에 접근하는 함수가 없을 때 까지 대기한다.

100-103 : 해쉬테이블에 아무런 값이 없다면 그냥 종료한다.

104-106 : 제출 함수에는 없앴습니다.

108 : 파일 락을 걸어 파일에 접근하는 함수가 존재한다면 기다리고 진행합니다.

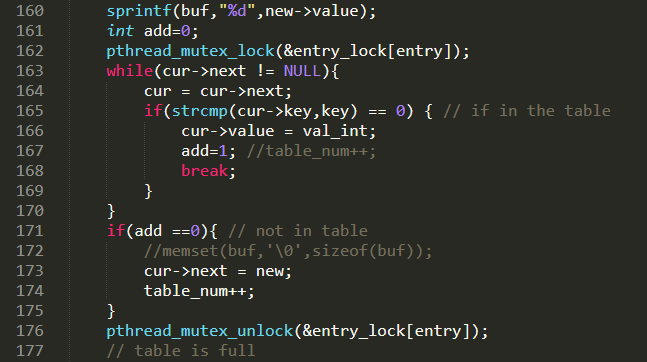
109~ 그후의 함수는 동일합니다.



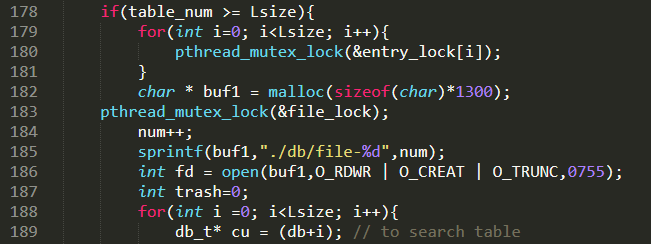
할당한 데이터 해제, 135-136 사이에 entry\_lock 해제를 해놓았습니다.

DB\_PUT

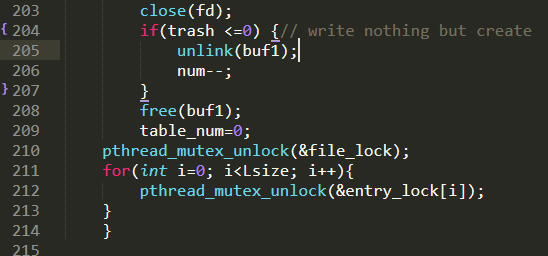
\



162 – 176 : 해쉬 함수를 통해 얻은 entry 에 락을 걸어 다른 함수가 접근하지 못하게 합니다.

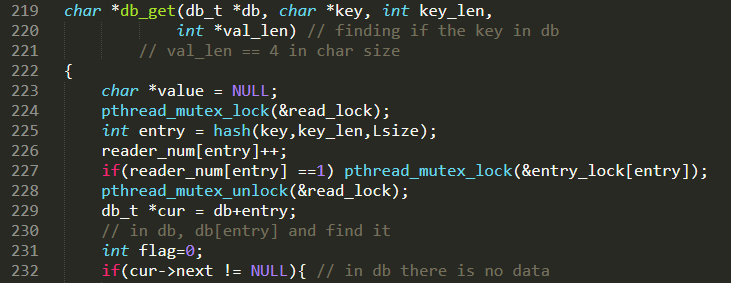


178~ : table 이 꽉 찼을 때 내리는 것. 그래서 179-181줄에서 해쉬테이블 전체에 락을 걸고, 183줄과 같이 file에 접근해야하므로 file\_lock 설정

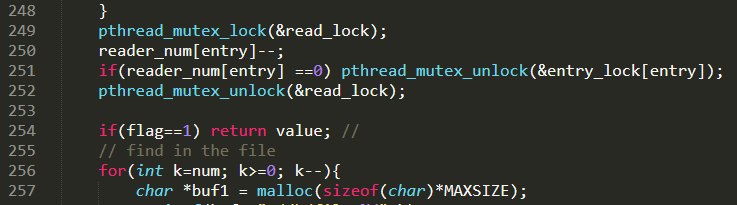
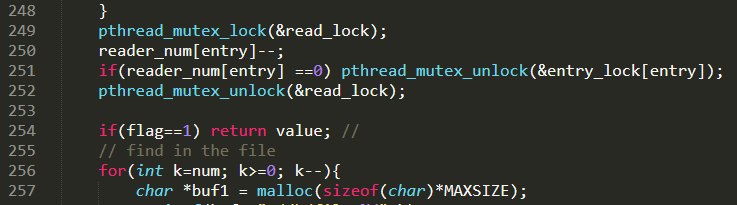


걸어 주었던 락 해제한다.

DB\_GET



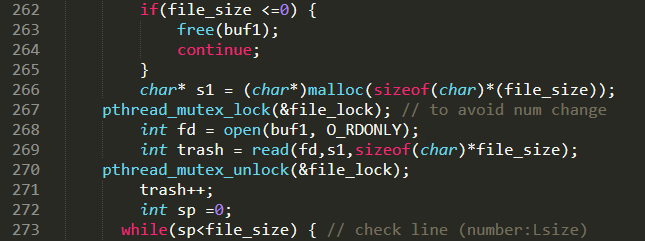
224-228 : 넣으려는 변수의 해쉬값 = entry 이므로 그 엔트리에 접근할 수 없게 락을 걸어준다.



249-252 : 접근한 이후 락 해제

254 : 테이블에서 찾았다면 (flag==1) value 출력해주고 함수 종료

256~ : 파일에서 데이터 찾는다.



267-270 : 파일에 접근하여 열 때 file\_lock을 통해 파일에 접근하는 함수가 없도록 한다. 파일의 정보를 읽은 후에는 락 해제