Connection Server

```
void *connect_sockets(void* socketNum)
{
    int *num =(int *) socketNum;
    int valread, curSocket = *num;
    char buffer[1024] = {0};
    while(1)
    {
```

Wenn der Schlüssel bereits vorhanden ist, wird der Wert überschrieben.

```
402             if(strstr(command, "PUT") != NULL ||strstr(command, "DEL") != NULL || strstr(command, "SUB") != NULL ||
403                 strstr(command, "OP") != NULL)  //Write-mode
404             {                                   // 5x Mutex solve Leser
405                                                 // Writing Problem
406                 pthread_mutex_lock(&mutex2);
407                 wait2Write ++;
408                 pthread_mutex_unlock(&mutex2);
409
410                 while(ReaderMode > 0)
411                 {
412                     sleep(1);
413                 }
414                 pthread_mutex_lock(&mutex1);        //locking to prevent xase conditions.for Zombie Problem?
415                 if (strstr(command, "PUT") != NULL)
416                 {
417                 ExtractKey(buffer,key);
418                     ExtractValue(buffer, value);
419                     put(key,value,curSocket);
420                     callSubs(key);
421             }
422             if (strstr(command, "DEL") != NULL) {
423                 ExtractKey(buffer,key);
424                 del(key, curSocket);
425                 callSubs(key);
426             }
427             if (strstr(command, "SUB") != NULL) {
428                 ExtractKey(buffer,key);
429                 sub(key, curSocket);
430             }
431             if (strstr(command, "OP") != NULL) {
432                 ExtractKey(buffer,key);
433                 ExtractValue(buffer, value);
```

Inhaltsverzeichnis

1. PUT Befehl
2. GET Befehl
3. DELETE Befehl
4.BEG Befehl
5.END Befehl
6.SUB Befehl
7.OP Befehl
8.QUIT Befehl

PUT Befehl

```
186 int put(char* key, char* value, int curSocket)
187 {
188     if(isError(key,curSocket)== -1)return -1;
189     if(isAlphanumeric(value) == 0)
190     {
191         send(curSocket , "Value is not Alphanumeric!\n", strlen("Value is not Alphanumeric!\n") , 0 );
192         return -1;
193     }
194
195     char fileName[1024]={'\0'}; //key + ".txt" ending
196     getFileName(key,fileName);
197
198     //printf("adding to filename => %s\n",fileName);
199         FILE * file;
200
201     /* open the file for writing*/
202     file = fopen (fileName,"w+");
203     //can be writen w instead of w+
204         //Write
205         fprintf (file, "%s\n",value);
206
207     /* close the file*/
208     fclose (file);
209     strcat(value,"\n");
210
211         //printf("Key added\n");
212         sendResponse("PUT", key, value,curSocket);
213     return 1;
214 }
```

Die put() Funktion hinterlegt einen Wert (value) mit dem Schlüsselwert (key).

Wenn der Schlüssel bereits vorhanden ist, wird der Wert überschrieben.

```
37 int isError(char* key,int curSocket){
38     if(isAlphanumeric(key) == 0)
39     {
40         send(curSocket , "Key is not Alphanumeric!\n", strlen("Key is not Alphanumeric!\n") , 0 );
41
42         return -1;
43     }
44     else return 0;
45 }
```

GET Befehl

```
214  }
215  int get(char* key, int curSocket){
216      if(isError(key,curSocket)== -1)return -1;
217
218      char fileName[1024]={'\0'}; //key + .txt ending
219      getFileName(key,fileName);
220
221
222      FILE *file;
223      char readed[1024];
224
225      file = fopen (fileName, "r+");
226      //can be writen r instead of r+
227       if (file != NULL)
228       {
229              while(fscanf(file, "%s", readed)!=EOF) ///EOF=End Of File
230              {
231              printf("%s (From GET)\n", readed );
232          }
233          fclose (file);
234          strcat(readed,"\n");
235          sendResponse("GET", key, readed,curSocket);
236          }
237          else
238          {
239              sendResponse("GET", key, "key_nonexistent\n",curSocket);
240              return 1;
241          }
242          return -1;
243  }
```

Sucht einen Schlüsselwert (key) in der Datenhaltung und gibt den hinterlegten Wert (value) zurück. GET

Wenn der Schlüsselwert nicht vorhanden ist, gibt uns key_nonexistent.

```
void getFileName(char* key, char* into)
{
    strcpy(into, key);
    strcat(into,".txt");
}
```

DELETE Befehl

Sucht einen Schlüsselwert und entfernt ihn zusammen mit dem Wert aus der Datenhaltung.

Wenn die Datei locked ist, scheitern die Befehle und gibt eine Fehlermeldung zurück.

```
244  int del(char* key, int curSocket)
245  {
246      if(isError(key,curSocket)== -1)return -1;
247
248      char fileName[1024]={'\0'}; //key + .txt
249      getFileName(key,fileName);
250
251      if (0==remove(fileName))
252          sendResponse("DEL", key, "key_deleted\n",curSocket);
253              ///printf("The file is Deleted \n");
254      else
255          sendResponse("DEL", key, "key_nonexistent\n",curSocket);
256              ///printf("the file is NOT Deleted\n");
257  }
```

```
37  int isError(char* key,int curSocket){
38      if(isAlphanumeric(key) == 0)
39      {
40          send(curSocket , "Key is not Alphanumeric!\n", strlen("Key is not Alphanumeric!\n") , 0 );
41
42          return -1;
43      }
44      else return 0;
45  }
```

BEG Befehl

BEG beginnt einen exklusiven Zugriff auf den Key-Value-Store für Clients

```
385             if(strstr(command, "BEG") != NULL && strlen(command) == 4)
386             {
387                 send(curSocket , "Sole access granted!", strlen("Sole access granted!") , 0 );
388                 soleSock = curSocket;
389                 printf( "Socket:%i has sole access!\n",curSocket);
390             }
```

END Befehl

END beendet den exklusiven Zugriff wieder

```
391         if(strstr(command, "END") != NULL && strlen(command) == 4 && curSocket == soleSock)
392         {
393             send(curSocket , "Sole access removed!", strlen("Sole access removed!") , 0 );
394             soleSock = -1;// you may write number whatever you want.but give the same number later in the code
395             printf( "Socket:%i released sole access!\n",curSocket);
396         }
```

SUB Befehl

Ein Client kann einen Schlüssel “subscribe”

die Änderungen vom key verfolgen

```
286  void sub(char* key, int curSocket)
287  {
288      if(isError(key,curSocket)== -1)return ;
289
290      char fileName[1024]={'\0'}; //key + .txt
291      getFileName(key,fileName);
292
293      FILE *file;
294      file = fopen (fileName, "r");
295       if (file == NULL)          //Test the first key
296       {
297          sendResponse("SUB", key, "key_nonexistent",curSocket);
298          return;
299       }
300          fclose (file);
301
302      char content[1024]={'\0'};
303      getContentOfFile("SUB.txt",content);
304      //printf("Content File :%s#\n",content);
305
306      char number[1024]={'\0'};
307      sprintf(number, "%d", curSocket); ///sends formatted output to a string pointed to, by str.
308      strcat(content,number);
309      strcat(content," ");
310      strcat(content,key);
311      strcat(content,"#");//printf("Content with added:%s#\n",content);
312
313      writeString2File("SUB.txt",content);
314      char value[1024]={'\0'};
315      getContentOfFile(fileName,value);
316      sendResponse("SUB", key, value,curSocket);
317  }
```

```
318  void callSubs(char* key)
319  {
320      char content[1024]={'\0'};
321      char subbedtoKey[1024]={'\0'};
322      getContentOfFile("SUB.txt",content);
323      int curStart=0, subLength=0;
324      while(content[curStart] != '\0')
325      {
326          subLength=0;
327          while(content[curStart+subLength] != '#')
328          {
329              subLength++;
330          }
331          char sub[1024]={'\0'};
332          memcpy( sub, &content[curStart], subLength);
333          ///printf(" sub: %s|\n",sub);
334          int curSocket = extractNumber(sub);
335          ExtractKey(sub,subbedtoKey);
336
337
338          if(strcmp(subbedtoKey, key)==0 && latestSender != curSocket) ///  subbedtoKey = key
339          {
340              printf("sended %i.Socket\n",curSocket);
341              send(curSocket , latestAction, strlen(latestAction) , 0 );
342
343          }
344          curStart += subLength + 1;
345      }
346  }
```

OP Befehl?

Sie können erfahren wer der Benutzer, das Datum oder die Betriebszeit ist.(In Linux)

```
256  void op(char* key, char* value) //THERE's A Problem in this function :(
257  {
258      /**
259      int fd[2];
260
261  //  fd[0]=read
262  //  fd[1]=write
263
264      */
265      //Father process continue to run server.
266      int id = fork();
267      if(id==0 &&  strcmp(value,"who")==0){ //child process  ayrıca date ve who için
268      //printf("yiiiii");
269          execlp("who","who",NULL);//but they're overriting the child process so I cannot make consistent :(
270      }
271      if(id==0 && strcmp(value,"uptime")==0){ //child process  ayrıca date ve who için
272      //printf("yiiiii");
273          execlp("uptime","uptime",NULL);//but they're overriting the child process so I cannot make consistent :(
274      }
275      if(id==0 &&strcmp(value,"date")==0){ //child process  ayrıca date ve who için
276      //printf("yiiiii");
277          execlp("date","date",NULL);//but they're overriting the child process so I cannot make consistent :(
278      }
279      /**
280      I cannot use execlp for writing to file because it's overriting process and don't continue.
281      Same problem with Pipes too.Unfortunately We could'nt solve Problem
282      */
283
284  }
```

Aber .... :(

QUIT Befehl

Sie können den Server beenden.

```
if (strstr(command, "QUIT") != NULL) {
    printf("Socket %i disconnected\n", curSocket);
    send(curSocket , "CONNECTION LOST!", strlen("CONNECTION LOST!") , 0 );
    break;
}
```

Vielen Dank für Ihre Aufmerksamkeit.

QUELLE:

https://www.youtube.com/watch?v=cex9XrZCU14&list=PLfqABt5AS4FkW5mOn2Tn9ZZLLDwA3kZUY // Channel: CodeVault
https://www.youtube.com/watch?v=bdIiTxtMaKA&t=1s //Channel: Jacob Sorber
https://www.geeksforgeeks.org/socket-programming-cc/
https://aticleworld.com/socket-programming-in-c-using-tcpip/
https://stackoverflow.com/questions/2884230/zeroing-out-memory
https://github.com/PacktPublishing/Hands-On-Network-Programming-with-C
https://www.cplusplus.com/
https://stackoverflow.com/questions/7064314/what-is-0-in-c
https://stackoverflow.com/questions/14461695/what-does-0-stand-for/14461711#:~:text=To%20the%20C%20language%2C%20'%5C,particular%20zero%20as%20a%20character.&text=%5C0%20is%20zero%20character.