GEBZE TECHNICAL UNIVERSITY

CSE344
SYSTEMS PROGRAMMING COURSE

HOMEWORK1 REPORT

Baran Solmaz
1801042601

Solution Method:

Firstly, I divided tasks in 3 types;starts with ‘^’,ends with ‘$’ and other tasks.
Secondly, I divided these tasks by that has ‘[’-‘]’ .

Starts With ‘^’ Tasks :

Tasks without ‘[’-‘]’:

I compared char by char if the characters are same,compared next one. If not, copied into a new buffer.While comparing,if next character is ‘*’,I compared current character with next character of word.

Tasks with ‘[’-‘]’:

While comparing, if the index equals to index of ‘[’ ,I compared with the characters that between ‘[’-‘]’ if one of them is exist,compared next character.If the next char of ‘]’ is ‘*’,then I continued to compare chars between ‘[’-‘]’.

Ends With ‘$’ Tasks :

Tasks without ‘[’-‘]’:

I started to compare from the end of the word,if the current character is ‘*’,I compared next character with next character of word until the chars are not the same.

Tasks with ‘[’-‘]’:

While reverse comparing, if the index equals to index of ‘]’ ,I compared with the characters that between ‘[’-‘]’ if one of them is exist,compared next character.If the previous char of ‘]’ is ‘*’,then I continued to compare chars between ‘[’-‘]’.

Other Tasks :

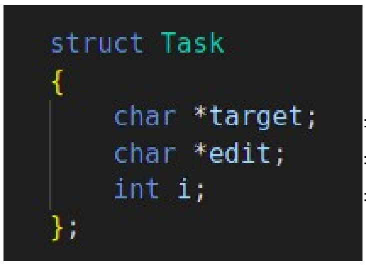
Tasks without ‘[’-‘]’:

I compared char by char if the characters are same,compared next one. If not copied into a new buffer and reseted the index of task.While comparing,if next character is ‘*’,I compared current character with next character of word.

Tasks with ‘[’-‘]’:

While comparing, if the index equals to index of ‘[’ ,I compared with the characters that between ‘[’-‘]’ if one of them is exist,compared next character.If the next char of ‘]’ is ‘*’,then I continued to compare chars between ‘[’-‘]’.While comparing , if chars are not same, I reseted the index of task and continued.

Design Decisions:

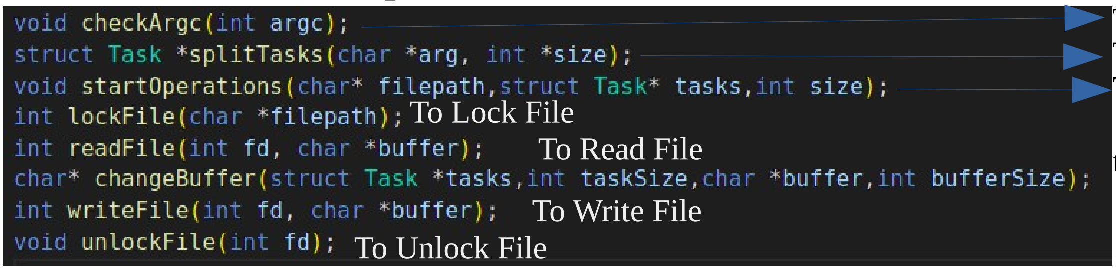


```c
struct Task
{
    char *target;
    char *edit;
    int i;
};
```

/str1/str2/i

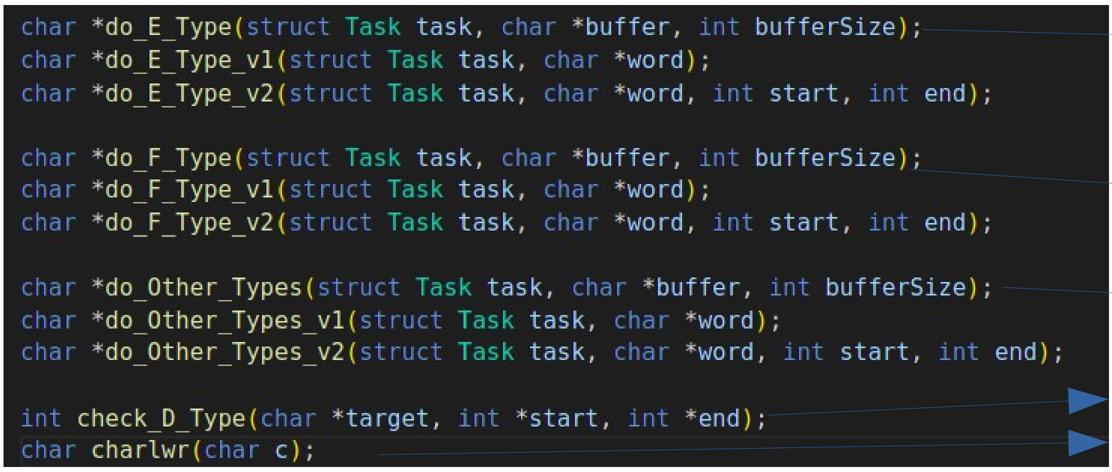
=str1
=str2
=1 if ‘i’ exist 1,then 0;

Function Explanation:



```c
void checkArgc(int argc);
struct Task *splitTasks(char *arg, int *size);
void startOperations(char* filepath,struct Task* tasks,int size);
int lockFile(char *filepath);
int readFile(int fd, char *buffer);
char* changeBuffer(struct Task *tasks,int taskSize,char *buffer,int bufferSize);
int writeFile(int fd, char *buffer);
void unlockFile(int fd);
```

To check arguments,
To split tasks by ‘;’ and ‘/’
To start from locking
file to unlock
to replacing strings
To Lock File
To Read File
To Write File
To Unlock File



```c
char *do_E_Type(struct Task task, char *buffer, int bufferSize);
char *do_E_Type_v1(struct Task task, char *word);
char *do_E_Type_v2(struct Task task, char *word, int start, int end);

char *do_F_Type(struct Task task, char *buffer, int bufferSize);
char *do_F_Type_v1(struct Task task, char *word);
char *do_F_Type_v2(struct Task task, char *word, int start, int end);

char *do_Other_Types(struct Task task, char *buffer, int bufferSize);
char *do_Other_Types_v1(struct Task task, char *word);
char *do_Other_Types_v2(struct Task task, char *word, int start, int end);

int check_D_Type(char *target, int *start, int *end);
char charlwr(char c);
```

tasks with ‘^’

tasks with ‘$’

Other Tasks

To check “[“ “]”
To lower char

_v1 : tasks that doesn’t have “[“ “]”
_v2 : tasks that has“[“ “]”