

## 1. 作業目標

延續第一個作業的程式，增加支援類似 Redis List 的操作，指令包含 LPUSH, LPOP, RPUSH, RPOP, LLEN, LRange。

## 2. Code review

Definition of Data Structures: 基本上就是定義 node 的結構(包含前後 Pointer)，並且額外定義 head 和 tail 來方便實現 doubly linked list(不一定要定義 tail，不過我覺得寫起來會更方便)，createNode()則是為了方便使用 malloc 配置出記憶體體的函數，在增加 node 時都會用到。

```

struct Node {
    char* key;
    char* value;
    struct Node* prev;
    struct Node* next;
};

struct doubly_linked_list{
    struct Node* head;
    struct Node* tail;
};

struct Node* createNode(char* key, char* value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode) {
        newNode->key = strdup(key);
        newNode->value = strdup(value);
        newNode->prev = NULL;
        newNode->next = NULL;
    }
    return newNode;
}

```

## a. LPUSH

LPUSH 基本上就是雙向鏈結串列 insert\_at\_head 的操作，這邊分成兩種情況處理，一個是當前沒有任何 node，一個是當前已經有一個或以上的 node。

```

void insert_at_head(struct doubly_linked_list* ptr_to_DL_list, char* key, char* value) {
    struct Node* newNode = createNode(key, value);
    if (ptr_to_DL_list->head == NULL) { //當前的DL list是空的
        ptr_to_DL_list->head = newNode;
        ptr_to_DL_list->tail = newNode;
    } else {
        newNode->next = ptr_to_DL_list->head;
        ptr_to_DL_list->head->prev = newNode;
        ptr_to_DL_list->head = newNode;
    }
}

```

## b. RPUSH

RPUSH 基本上就是雙向鏈結串列 insert\_at\_tail 的操作，這邊分成兩種情況處理，一個是當前沒有任何 node，一個是當前已經有一個或以上的 node。

```

void insert_at_tail(struct doubly_linked_list* ptr_to_DL_list, char* key, char* value) {
    struct Node* newNode = createNode(key, value);
    if(ptr_to_DL_list->tail==NULL){ //當前的DL list是空的
        ptr_to_DL_list->head=newNode;
        ptr_to_DL_list->tail=newNode;
    }
    else{
        newNode->prev=ptr_to_DL_list->tail;
        ptr_to_DL_list->tail->next=newNode;
        ptr_to_DL_list->tail=newNode;
    }
}

```

### c. LPOP

LPOP 基本上就是雙向鏈結串列 delete\_at\_head 的操作，這邊分成兩種情況處理，一個是當前只剩下一個 node，一個是當前還剩下兩個或以上的 node。這邊的操作是 LPOP 之後我們就直接印出該 node 的 key 和 value，就沒有額外 return node。

```
void delete_at_head(struct doubly_linked_list* ptr_to_DL_list){
    if(ptr_to_DL_list->head){//head要不是NULL，否則當前沒東西
        printf("key %s, value %s\n",ptr_to_DL_list->head->key,ptr_to_DL_list->head->value);
        if(ptr_to_DL_list->head==ptr_to_DL_list->tail){//當前Node只剩一個
            ptr_to_DL_list->head=NULL;
            ptr_to_DL_list->tail=NULL;
        }
        else{//當前node至少還有兩個
            ptr_to_DL_list->head->next->prev=NULL;
            ptr_to_DL_list->head=ptr_to_DL_list->head->next;
        }
    }
}
```

### d. RPOP

RPOP 基本上就是雙向鏈結串列 delete\_at\_tail 的操作，這邊分成兩種情況處理，一個是當前只剩下一個 node，一個是當前還剩下兩個或以上的 node。這邊的操作是 RPOP 之後我們就直接印出該 node 的 key 和 value，就沒有額外 return node。

```
void delete_at_tail(struct doubly_linked_list* ptr_to_DL_list){
    if(ptr_to_DL_list->head){//head要不是NULL，否則當前沒東西
        printf("key %s, value %s\n",ptr_to_DL_list->tail->key,ptr_to_DL_list->tail->value);
        if(ptr_to_DL_list->head==ptr_to_DL_list->tail){//當前Node只剩一個
            ptr_to_DL_list->head=NULL;
            ptr_to_DL_list->tail=NULL;
        }
        else{//當前node至少還有兩個
            ptr_to_DL_list->tail->prev->next=NULL;
            ptr_to_DL_list->tail=ptr_to_DL_list->tail->prev;
        }
    }
}
```

### e. LLEN:從左到右計算總共幾個 node。

```
int get_len(struct doubly_linked_list* ptr_to_DL_list){
    struct Node* cur=ptr_to_DL_list->head;
    int sum=0;
    while(cur){
        sum+=1;
        cur=cur->next;
    }
    return sum;
}
```

### f. LRANGE:從左到右全部印出來。

```
void print_from_left(struct doubly_linked_list* ptr_to_DL_list) {
    struct Node* cur = ptr_to_DL_list->head;
    while (cur) {
        printf("Key: %s, Value: %s\n", cur->key, cur->value);
        cur = cur->next;
    }
}
```

### 3. 編譯方式與執行結果

編譯方式:

Step1: gcc -c redis\_str.c -o redis\_str.o

Step2: gcc -o DLL Double\_linked\_list.c redis\_str.o

Step3 ./DLL

WSL 上 UBUNTU 的執行結果

```
eason@LAPTOP-Q69P3FAE:/mnt/c/Users/user/HW3$ ./DLL
Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LPUSH key1 value1 key2 value2 key3 value3

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
RPUSH key4 value4 key5 value5 key6 value6

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LRANGE
Key: key3, Value: value3
Key: key2, Value: value2
Key: key1, Value: value1
Key: key4, Value: value4
Key: key5, Value: value5
Key: key6, Value: value6
```

LPUSH 四個 pair

RPUSH 四個 pair

LRANGE 印出 8 個 pair

WSL: Ubuntu 0

```
Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LPOP
LPOP: key key3, value value3

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LPUSH key7 value7

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
RPOP
RPOP: key key6, value value6

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LLEN
```

LPOP

LPUSH 1 個 pair

RPOP

WSL: Ubuntu 0

```
Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LLEN
LLEN: 5
Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LRANGE
Key: key7, Value: value7
Key: key2, Value: value2
Key: key1, Value: value1
Key: key4, Value: value4
Key: key5, Value: value5

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
LPUSH key9 value9

Enter the operation
```

WSL: Ubuntu 0

合併 HW1 的功能: SET, GET, DELETE, SHOW

```
Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
SET key9 value9

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
SET key10 value10

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
SET key11 value11

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
GET key11
The value to key11 is value11
```

```
SET value12 key12

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
SET value13 key 13

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
SHOW
All the key-pairs
key11 value11
key9 value9
value12 key12
value13 key

Enter the operation
Redis list Operation(LPUSH,LPOP,RPUSH,RPOP,LLEN,LRANGE)
Redis String Operation(SET,GET,SHOW,DEL)
```

#### 4. 分析

基本上這個作業沒有太大的問題，就是雙向鏈結串列寫出來之後就還行，倒是使用者介面的部分，在讀取不定個數的 input 時其實沒那麼簡單，而且我們輸入的格式是 command(例如 SET 或 LPUSH 等等) + 不定個數 input，後來是在 chatgpt 的建議之下用 fget 讀取整行後，先計算 strlen(command)，再用 while 搭配 sscanf，才能在讀取完 command 之後一次讀取多個 key-value pairs。

感謝助教批改!!!