**운영체제**

**Project 0-2**

**보고서**

**20151619**

**최준수**

**사용한 함수 목록**

**1. 직접 구현한 함수** (main.c에 작성하였습니다.)

**int find\_type\_index(my\_list\* lists, int list\_cnt, my\_hash\* hashs, int hash\_cnt, my\_bitmap\* bitmaps, int bitmap\_cnt, char\* name, int\* type, int\* index);**  
type을 모르는 경우, name으로 list와 hashtable, bitmap을 모두 검색한 뒤, type과 index를 저장하여 준다.

Lists는 list들의 모음, list\_cnt는 list들의 모음의 길이, hashs는 hashtable의 모음, hash\_cnt는 hashtable들의 모음의 길이, bitmaps는 bitmap들의 모음, bitmap\_cnt는 bitmap의 모음의 길이이다. Char\* name는 내가 찾고자하는 이름이고, int\* type과 int\* index는 3가지 자료구조 모음을 모두 검색하고 결과값이 저장되는 파라미터이다.

**int find\_index(void\* arr, int len, int type, char\* text);**  
Find\_type\_index()와 비슷하지만, find\_index는 type을 아는경우에 사용된다. type을 아는경우, type에따라 arr를 형변환한 후, text로 검색하여 해당 text를 이름으로 갖는 자료구조의 index를 반환한다.

Void\* arr는 한가지 자료구조의 모음이고, len은 그것의 길이, type은 arr를 형변환하기위한 type이다. arr에서 text라는 이름을 가진 자료구조를 찾아 index를 반환한다.

**void free\_array(void\* arr, int len, int type, int index);**  
생성된 hashtable, list, bitmap중 delete가 호출되었다면 관리하고 있던 배열에서 삭제해준다. Void\*로 들어오는 arr를 다른 인자인 type에 맞게 형변환하여주고, index번째 hashtable/list/bitmap을 삭제하고, 한칸씩 앞으로 당겨준다.

**struct list\_elem\* list\_get\_element\_at(struct list\_elem\* first, int index);**  
list의 head의 다음인 첫번째 list\_elem\* first를 인자로 받아서 index번째에 있는 node를 반환한다.

**struct list\_elem\* list\_find\_element(struct list\_elem\* first, int data);**  
list의 head의 다음인 첫번째 list\_elem\* first를 인자를 받아서 tail이 나올때까지 순회를 하며 인자로 들어온 data와 같은 값을 가지고 있는 node가 나오면 반환한다.

**void bool\_to\_str(int bit);**  
정수 1이 bit으로 들어오면 “true”를 출력해주고, 정수 0이 bit으로 들어오면 “false”를 출력

**int str\_to\_bool(char\* str);**  
str에 “true”가 들어오면 1 (enum TRUE)를 반환하고, str에 “false”가 들어오면 0 (enum FALSE)를 반환한다.

**2. 자료구조 관련 함수를 사용할 때 추가로 구현했던 함수**  
(main.c에 작성하였습니다.)

**bool custom\_list\_less\_func(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux);**  
리스트의 a 노드에 저장된 데이터와 b 노드에 저장된 데이터 값을 비교하여 a의 데이터가 b의 데이터보다 작을 경우 true를 리턴, 클 경우 false를 리턴한다.

**unsigned custom\_hash\_hash\_func(const struct hash\_elem \*e, void \*aux);**  
e노드의 데이터를 입력 받아 라이브러리의 hash\_int()를 사용하여 양의 정수로 매핑하는 해시함수

**bool custom\_hash\_less\_func(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux);**  
해시의 a 노드에 저장된 데이터와 b 노드에 저장된 데이터 값을 비교하여 a의 데이터가 b의 데이터보다 작을 경우 양의정수(True역할)를 리턴, 클 경우 음의정수(False역할)를 리턴한다.

**void destructor(struct hash\_elem \*e, void \*aux);**  
e라는 노드의 데이터를 삭제(free)하는 함수

**void custom\_square(struct hash\_elem \*e, void \*aux);**  
노드 e의 데이터를 제곱하여 저장하는 함수

**void custom\_triple(struct hash\_elem\*e, void \*aux);**  
노드 e의 데이터를 세제곱하여 저장하는 함수

**3. 과제에서 요구한 추가구현 함수**

**void list\_swap(struct list\_elem \*a, struct list\_elem \*b); (list.c에 작성)**  
리스트 노드 a와 b의 위치를 바꿔주는 함수

pintos의 list는 double linked list이므로 각각 a와 b에서 의 prev와 next를 바꾸어주었고, 또 이전과 이후의 node에서 a와 b로 오는 포인터 주소도 바꾸어주었음. 다만, a와 b가 연속되어있는 경우는 예외처리를 통해 따로 바꾸어주었음.

**void list\_shuffle(struct list \*list); (list.c에 작성)**  
리스트 안의 노드 순서를 무작위로 바꾸는(shuffle)하는 함수  
리스트의 0~리스트사이즈 범위 내의 숫자 2개를 중복되지 않게 rand()로 뽑아서 그 2개의 element들을 swap해주었음. 리스트안에 있는 원소의 개수만큼 swap을 실행하였음

**unsigned hash\_int\_2(int i);** (hash.c에 작성)  
정수 값의 데이터가 들어왔을 때 그걸 양의 정수로 매핑하는 또 다른 해시함수. 출력 값은 각 정수를 이진수로 변환하였을 때 존재하는 1의 개수를 4(bucket size)로 나눈 나머지

**struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size);** (bitmap.c에 작성)  
해당 bitmap의 길이를 size만큼 늘려 반환하는 함수

realloc을 통해 사이즈를 늘려주었음.

**4. 라이브러리 제공 함수**

**<list관련 함수>**

**list\_init (struct list \*list);**list를 초기화하는 함수

**struct list\_elem \*list\_begin (struct list \*list);**list의 가장 앞 노드를 return하는 함수

**struct list\_elem \*list\_next (struct list\_elem \*elem);**elem노드의 다음 노드를 return하는 함수

**struct list\_elem \*list\_end (struct list \*list);**list의 마지막 노드를 return하는 함수

**struct list\_elem \*list\_prev (struct list\_elem \*elem);**elem노드의 이전 노드를 return하는 **함수**

**struct list\_elem \*list\_head (struct list \*list);**list의 head를 return하는 함수

**struct list\_elem \*list\_tail (struct list \*list);**list의 tail을 return하는 함수

**void list\_insert (struct list\_elem \*before, struct list\_elem \*elem);**before노드 앞에 elem노드를 삽입하는 함수

**void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last);**first노드부터 last노드까지 자른 다음 before노드 앞에 끼워넣는 함수

**void list\_push\_front (struct list \*list, struct list\_elem \*elem);**list의 맨 앞부분에 elem노드를 push하는 함수

**void list\_push\_back (struct list \*list, struct list\_elem \*elem);**list의 맨 마지막 부분에 elem노드를 push하는 함수

**struct list\_elem \*list\_remove (struct list\_elem \*elem);**elem노드를 제거하여 return하는 함수

**struct list\_elem \*list\_pop\_front (struct list \*list);**list의 맨 앞 노드를 pop하여 return하는 함수

**struct list\_elem \*list\_pop\_back (struct list \*list);**list의 맨 마지막 노드를 pop하여 return하는 함수

**struct list\_elem \*list\_front (struct list \*list);**list의 맨 앞 노드를 return하는 함수

**struct list\_elem \*list\_back (struct list \*list);**list의 맨 마지막 노드를 return하는 함수

**size\_t list\_size (struct list \*list);**list의 노드의 개수를 return하는 함수

**bool list\_empty (struct list \*);**list가 비어있으면 true를, 비어있지 않다면 false를 return하는 함수

**void list\_reverse (struct list \*);**list의 순서를 뒤집는 함수

**void list\_sort (struct list \*, list\_less\_func \*, void \*aux);**list를 정렬하는 함수

**void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux);**list를 정렬한 상태를 유지하면서 노드를 삽입하는 함수

**void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux);**중복된 data가 있는 노드를 삭제하여 data당 하나의 노드만 있게 만드는 함수

**struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux);**data가 최댓값인 노드를 return하는 함수

**struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux);**data가 최솟값인 노드를 return하는 함수

**<hash관련 함수>**

**bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux);**처음 hashtable을 초기화하는 함수

**void hash\_clear (struct hash \*, hash\_action\_func \*);**hashtable을 비울 때 사용하는 함수

**void hash\_destroy (struct hash \*, hash\_action\_func \*);**hashtable을 삭제할 때 사용하는 함수

**struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*);**hashtable에 노드를 삽입하는 함수

**struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*);**같은 element가 존재하면 교체하고 아니면 삽입하는 함수

**struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*);**특정한 element를 찾아 return하는 함수

**struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*);**특정한 element를 찾아 삭제한 후 return하는 함수

**void hash\_apply (struct hash \*, hash\_action\_func \*);**노드의 data들에 대해 제곱 또는 세제곱을 계산하는 함수

**struct hash\_elem \*hash\_next (struct hash\_iterator \*);**hashtable안에서 한 노드의 다음 노드를 return하는 함수

**struct hash\_elem \*hash\_cur (struct hash\_iterator \*);**hashtable안에서 한 노드의 이전 노드를 return하는 함수

**size\_t hash\_size (struct hash \*);**hashtable의 노드의 개수를 return하는 함수

**bool hash\_empty (struct hash \*);**hashtable이 비어있으면 true를, 아니면 false를 return하는 함수

**unsigned hash\_int (int);**특정 정수 값을 입력받아 hash한 결과를 return하는 함수

**<bitmap관련 함수>**

**struct bitmap \*bitmap\_create (size\_t bit\_cnt);**비트 개수가 bit\_cnt개인 bitmap을 생성하는 함수

**size\_t bitmap\_size (const struct bitmap \*);**bitmap의 size를 return하는 함수

**void bitmap\_set (struct bitmap \*, size\_t idx, bool);**입력받은 index의 비트를 bool값으로 저장하는 함수

**void bitmap\_mark (struct bitmap \*, size\_t idx);**입력받은 index의 비트를 true로 저장하는 함수

**void bitmap\_reset (struct bitmap \*, size\_t idx);**입력받은 index의 비트를 false로 저장하는 함수

**void bitmap\_flip (struct bitmap \*, size\_t idx);**입력받은 index의 비트값을 반대로 바꾸어 저장하는 함수

**bool bitmap\_test (const struct bitmap \*, size\_t idx);**입력받은 index의 비트값을 return하는 함수

**void bitmap\_set\_all (struct bitmap \*, bool);**bitmap전체의 비트를 bool값으로 저장하는 함수

**void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool);**start부터 cnt개 만큼의 비트를 bool값으로 저장하는 함수

**size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool);**start부터 cnt개 만큼의 비트중 값이 bool인 비트의 개수를 return하는 함수

**bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool);**start부터 cnt개 만큼의 비트중 어느 한 개라도 그 값이 bool이면 true를, 아니면 false를 return하는 함수

**bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt);**start부터 cnt개 만큼의 비트중 어느 한 개라도 그 값이 true이면 true를, 아니면 false를 return하는 함수

**bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt);**start부터 cnt개 만큼의 비트 모두가 값이 false이면 true를, 아니면 false를 return하는 함수

**bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt);**tart부터 cnt개 만큼의 비트 모두가 값이 true이면 true를, 아니면 false를 return하는 함수

**size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool);**start부터 cnt개 만큼의 비트중에서 연속적으로 값이 bool인 비트의 개수를 return하는 함수

**size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool);**start부터 cnt개 만큼의 비트중에서 연속적으로 값이 bool인 비트를 뒤집고 그것의 개수를 return하는 함수

**void bitmap\_dump (const struct bitmap \*);**bitmap의 bit값들을 hexadecimal로 출력하는 함수