**운영체제**

**Project 0-2**

**보고서**

**1반 20151590**

**이정원**

**사용한 함수 목록**

**1. 직접 구현한 함수** (main.c에 작성하였습니다.)

**int command(char \*cmd);**  
명령어 cmd를 구분하여 각 명령어에 해당하는 번호(integer)를 반환하는 함수

**void create(char \*arg1, char \*arg2, char \*arg3);**  
자료구조의 종류와 이름이 parameter로 들어오고 각각의 자료구조에 맞게 생성하는 함수

**void delete(char \*arg1);**  
arg1라는 이름의 자료구조를 삭제하는 함수

**void dumpdata(char \*arg1);**  
arg1라는 이름의 자료구조에 저장된 데이터를 출력하는 함수

**int getindex(int \*type, char \*name);**  
type으로 자료구조의 종류를 구분하고 name이라는 이름의 자료구조가 구조체 배열의 몇 번째 idex에 있는지 반환하는 함수 (type=0인 경우는 type을 모르는 경우로, 이 함수를 통해 어떤 자료구조인지 찾아서 그에 해당하는 번호를 type에 저장함)

**void listcmd(char \*cmd, char \*arg1, char \*arg2, char \*arg3, char \*arg4, char \*arg5);**  
list와 관련된 명령어들을 처리하는 함수. parameter로는 list의 이름 및 인덱스, 데이터 등이 들어온다.

**void hashcmd(char \*cmd, char \*arg1, char \*arg2, char \*arg3, char \*arg4, char \*arg5);**  
hash와 관련된 명령어들을 처리하는 함수. parameter로는 hash의 이름 및 데이터 등이 들어온다.

**void bitmapcmd(char \*cmd, char \*arg1, char \*arg2, char \*arg3, char \*arg4, char \*arg5);**  
bitmap과 관련된 명령어들을 처리하는 함수. parameter로는 bitmap의 이름 및 데이터 등이 들어온다.

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

**bool llf(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 hhf(const struct hash\_elem \*e, void \*aux);**  
e노드의 데이터를 입력 받아 양의 정수로 매핑하는 해시함수

**bool hlf(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 true를 리턴, 클 경우 false를 리턴한다.

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

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

**void 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);**  
리스트 노드 a와 b의 위치를 바꿔주는 함수 (list.c에 작성)

**void list\_shuffle(struct list \*list);**  
리스트 안의 노드 순서를 무작위로 바꾸는(shuffle)하는 함수  
(<time.h>를 사용해야 해서 main.c에 작성하였습니다.)

**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

**4. 그 밖에 test에서 사용한 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