**OS project 0-2 document**

20141544 안시현

**사용한 function**

따로 정의한 함수

void list\_swap(struct list\_elem \*a, struct list\_elem \*b) (list.c)

* list\_elem인 a와 b의 위치를 서로 바꾸어줌

struct bitmap \*bitmap\_expand (struct bitmap \*bitmap, int size); (bitmap.c)

* Bitmap의 크기를 size만큼 확장하고 0으로 채움

void list\_shuffle(struct list \*list) (list.c)

* 인자로 전달된 list를 무작위로 재배치

unsigned hash\_int\_2 (int i) (main.c)

* hash값을 계산하여 return해주는 hash함수

void list\_delete (struct list \*);

* 사용 후 List의 node들을 free해주는 함수

bool list\_less\_func (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux)

* List\_elem a의 data가 b보다 작을 시 true 값 반환

void list\_create()

* list를 만들어 주는 함수. List\_init()를 호출하여 초기화를 진행해줌

struct list\_elem\* list\_peek (struct list \*plist,int n)

* List 속 n번쨰 list\_elem를 반환함

void about\_list();

* list와 관련된 명령어를 처리함수

void about\_hash();

* hash와 관련된 명령어 처리함수

void hash\_create();

* Create 관련 명령어 처리함수

void about\_bitmap();

* bitmap관련 명령어 처리함수

void about\_delete();

* delete관련 명령어 처리함수

void about\_dumpdata(char table\_name[]);

* dumpdata 관련 명령어 처리함수. Table\_name에 list, hash , bitmap 이름을 전달받아 처리함

int find\_idx(char table\_name[]);

* list, hash, bitmap의 각각의 index를 찾아 반환하는 함수

int make\_number(char number[]);

* 문자열을 정수로 만들어줌. Atoi()함수 이용

**List 관련**

void list\_init (struct list \*)

* List를 초기화 해주는 함수.

struct list\_elem \*list\_begin (struct list \*)

* List의 첫 element를 반환

struct list\_elem \*list\_next (struct list\_elem \*)

* Parameter로 전달된 list\_elem의 다음 원소를 반환

struct list\_elem \*list\_end (struct list \*)

* 전달된 list의 tail을 반환

struct list\_elem \*list\_rbegin (struct list \*)

* 전달된 list의 마지막 element를 반환

struct list\_elem \*list\_prev (struct list\_elem \*)

* 전달된 list의 prev element를 반환

struct list\_elem \*list\_rend (struct list \*);

* 전달된 list의 head를 반환

struct list\_elem \*list\_head (struct list \*

* 전달된 list의 head를 반환;

struct list\_elem \*list\_tail (struct list \*);

* 전달된 list의 tail을 반환

void list\_insert (struct list\_elem \*before, struct list\_elem new\_elem\*)

* List\_elem before에 전에 두번째 인자로 전달된 new element를 추가함.

void list\_splice (struct list\_elem \*before,

struct list\_elem \*first, struct list\_elem \*last)

* 첫번째 인자인 List\_elem before 전에 first~last 까지의 list elem를 추가함

void list\_push\_front (struct list \*, struct list\_elem \*)

* 전달된 list의 제일 앞에 list\_elem를 추가함

void list\_push\_back (struct list \*, struct list\_elem \*);

* 전달된 list의 제일 뒤에 list\_elem를 추가함

struct list\_elem \*list\_remove (struct list\_elem \*)

* 전달된 list의 elem를 제거함

struct list\_elem \*list\_pop\_front (struct list \*)

* 전달된 list의 가장 앞에 있는 list\_elem를 제거

struct list\_elem \*list\_pop\_back (struct list \*);

* 전달된 list의 가장 뒤에 있는 list\_elem을 제거

struct list\_elem \*list\_front (struct list \*)

* 전달된 list의 가장 앞(head 바로 뒤) 원소를 반환

struct list\_elem \*list\_back (struct list \*)

* 전달된 list의 가장 뒤(tail 바로 전) 원소를 반환

size\_t list\_size (struct list \*)

* List의 element의 개수를 반환

bool list\_empty (struct list \*);

* List가 비었으면 true , 아니면 false를 반환

void list\_reverse (struct list \*)

* 인자로 전달된 list의 list\_elem 순서를 모두 뒤집어줌

bool list\_less\_func (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux)

* List\_elem a의 data가 b보다 작을 시 true 값 반환

void list\_sort (struct list \*, list\_less\_func \*, void \*aux)

* 인자로 전달된 list를 list\_less\_func로 전달된 함수의 기준에 따라 정렬

void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux)

* 전달된 element를 list 안에 적절한 위치에 삽입함. 그 전에 list는 sorting 되어 있음이 보장되어야 사용가능

void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux)

* List 안의 겹치는 원소를 모두 제거 하여 하나씩만 남김. 예를 들어, 1 2 2 3 3 4 라는 data가 저장 되어 있을 시 함수를 수행 후 list에 남아있는 데이터는 1 2 3 4가 된다.

struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux)

* list에서 가장 큰 원소를 반환

struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux)

* list에서 가장 작은 원소를 반환

**Bitmap 관련**

struct bitmap \*bitmap\_create (size\_t bit\_cnt)

* 인자로 전달된 bit\_cnt만큼의 크기를 가지는 bitmap을 만들어서 반환. 모든 bit들은 0으로 initialize된다.

struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*, size\_t byte\_cnt)

* Block에 미리 할당된 block\_size byte의 저장 공간에 bit\_cnt만큼의 크기를 가지는 bit맵을 만들어 반환. 역시 모든 bit는 0으로 초기화

size\_t bitmap\_buf\_size (size\_t bit\_cnt)

* Bit\_cnt만큼의 bit맵을 만들기 위해 필요한 byte수를 반환

void bitmap\_destroy (struct bitmap \*)

* 전달된 bitmap을 free해줌

size\_t bitmap\_size (const struct bitmap \*)

* 전달된 bitmap의 size를 반환

void bitmap\_set (struct bitmap \*, size\_t idx, bool)

* Idx의 index를 전달된 값으로 초기화함 ( true or false)

void bitmap\_mark (struct bitmap \*, size\_t idx)

* 전달된 bitmap의 idx번쨰의 bit를 true로 set

void bitmap\_reset (struct bitmap \*, size\_t idx)

* 모든 bit를 0으로 초기화

void bitmap\_flip (struct bitmap \*, size\_t idx)

* 전달된 bitmap의 idx번 쨰 비트를 뒤집음

bool bitmap\_test (const struct bitmap \*, size\_t idx)

* idx번째의 bit가 true인지 false인지를 반환

void bitmap\_set\_all (struct bitmap \*, bool);

* 전달된 bitmap을 모두 true or false로 set함

void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool)

* 전달된 bitmap에 start부터 cnt개 만큼을 true or false로 set함

size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool)

* Bitmap의 start index부터 cnt개수 까지 bits들 중에 true 혹은 false ( 인자로 전달된 bool형)과 같은 값을 가진 bit들을 반환

bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool)

* Bitmap의 start index부터 cnt개 만큼의 bits들 중에 true 혹은 false(인자로 전달된 bool형)과 같은 값이 하나라도 있으면 true 반환

bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt)

* Bitmap의 start index부터 cnt개의 bit들이 모두 true일시 true 반환

size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool);

* Start index이후 연속적으로 두개의 연속된 true or false(전달된 bool type 변수) 값이 있으면 시작 index를 반환, 없을시 BITMAP\_ERROR값 반환

size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool)

* Start index 이후 연속된 true or false(전달된 bool type 변수)가 있으면 그 bit들을 뒤집고 나서 시작 index를 반환

size\_t bitmap\_file\_size (const struct bitmap \*)

* Bitmap을 저장하기 위해 필요한 byte수를 계산해서 반환

bool bitmap\_read (struct bitmap \*, struct file \*)

* file로부터 bitmap을 읽어들임. 성공시 true 반환

bool bitmap\_write (const struct bitmap \*, struct file \*);

* 비트맵을 file에 씀. 성공시 true반환

void bitmap\_dump (const struct bitmap \*);

* 비트맵의 내용을 16진수로 console에 출력

**Hash 관련**

bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux)

* Hash를 초기화 하고 hash function과 less function을 등록해줌

void hash\_clear (struct hash \*, hash\_action\_func \*)

* Hash의 모든 element들을 제거함

void hash\_destroy (struct hash \*, hash\_action\_func \*)

* 전달된 Hash의 Hash table을 destroy해주는 함수

struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*);

* 전달된 hash에 hash\_elem을 insert해줌

struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*);

* 전달된 hash에 hash\_elem을 삽입 하는데 기존에 같은 data가 있을 시 그 data를 지우고 다시 삽입함

struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*);

* 전달된 hash\_elem와 같은 데이터가 있는지 찾고 없을 시 NULL 반환

struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*);

* 해시 테이블에 전달된 hash\_elem와 같은 원소가 있을 시 delete하고 없을 시에 NULL 반환

void hash\_apply (struct hash \*, hash\_action\_func \*);

* 함수 포인터로 전달된 함수를 hash의 모든 element들에 적용해줌

struct hash\_elem \*hash\_next (struct hash\_iterator \*);

* 다음 hash\_elem을 return 해줌

struct hash\_elem \*hash\_cur (struct hash\_iterator \*);

* 현재 hash\_elem를 return 해줌

size\_t hash\_size (struct hash \*);

* Hash의 size를 return 해줌

bool hash\_empty (struct hash \*)

* Hash가 비었을 시 true, 아니면 false return

unsigned hash\_bytes (const void \*, size\_t);

* hash값을 return해주는 hash함수

unsigned hash\_string (const char \*);

* hash값을 return해주는 hash함수

unsigned hash\_int (int);

* hash값을 return해주는 hash함수