OS Project #0-2 보고서

2반

20141494

강동욱

1. 사용한 library function

1-1 additional funtion

**/\*additional functions \*/**

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

두 list\_elem의 위치를 교환한다. 그 과정에서 list\_elem을 가리키는 포인터와 list\_elem이 가리키는포인터도 교환한다.

**void list\_shuffle(struct list \*list);**

list를 무작위 순서로 배열한다. 매 실행 다른 순서로 배열되며, list의 크기만큼 무작위 list\_swap이 일어난다.

**struct list\_elem \*random\_select(struct list\*);**

list\_ shuffle을 위해 list안의 임의의 list\_elem을 return하는 함수.

**unsigned hash\_int\_2(int i);**

i에 대한 hash값을 리턴한다. knuth's multiplicative method를 사용하여, 모든 값을 균등한 분포로 hash값을 할당한다.

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

현재 bitmap을 size만큼 확장한다. 현재의 bitmap+size 크기에 해당하는 크기로 realloc하고 초기화한다.

1-2 list function

void list\_init (struct list \*);

list를 초기화한다.

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

list의 시작점에대한 list\_elem을 리턴한다.

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

현재 list\_elem에 대한 다음 list\_elem을 리턴한다.

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

현재 list의 마지막 list\_elem을 리턴한다.

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

현재 list\_elem에 대한 이전 list\_elem을 리턴한다.

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

list의 head list\_elem을 리턴한다.

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

list의 tail list\_elem을 리턴한다.

void list\_insert (struct list\_elem \*, struct list\_elem \*);

첫 번째 argument의 위치에 두번쨰 argument를 삽입한다.

void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last);

첫번째 argument에 해당하는 list\_elem 뒤에 두번째 argument에 해당하는 list\_elem부터 이어진, 세번째 argument에 해당하는 list\_elem에서 끝나는 list\_elem들을 삽입한다.

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

list 앞에 list\_elem을 삽입한다.

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

list 뒤에 list\_elem을 삽입한다.

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

list에서 list\_elem을 제거한다.

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

list에서 첫번째 list\_elem을 제거하고, 그 list\_elem을 리턴한다.

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

list에서 마지막 list\_elem을 제거하고, 그 list\_elem을 리턴한다.

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

list에서 첫번째 list\_elem을 리턴한다.

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

list에서 마지막 list\_elem을 리턴한다.

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

list의 크기를 리턴한다.

bool list\_empty (struct list \*);

list가 비어있는지에 대한 bool값을 리턴한다.

void list\_reverse (struct list \*);

list를 역순으로 정렬한다.

void list\_sort (struct list \*,

list\_less\_func \*, void \*aux);

list를 크기순으로 정렬한다.

void list\_insert\_ordered (struct list \*, struct list\_elem \*,

list\_less\_func \*, void \*aux);

list에 정렬된 순서로 list\_elem을 삽입한다.

void list\_unique (struct list \*, struct list \*duplicates,

list\_less\_func \*, void \*aux);

두 리스트의 값을 중복되지 않게 출력한다.

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

list의 가장 큰 값을 가지는 list\_elem을 리턴한다.

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

list의 가장 작은 값을 가지는 list\_elem을 리턴한다.

#define list\_entry(LIST\_ELEM, STRUCT, MEMBER) \

((STRUCT \*) ((uint8\_t \*) &(LIST\_ELEM)->next \

- offsetof (STRUCT, MEMBER.next)))

현재 list\_elem가 나타내는 list\_item을 리턴한다.

1-3 hash function

#define hash\_entry(HASH\_ELEM, STRUCT, MEMBER)

((STRUCT \*) ((uint8\_t \*) &(HASH\_ELEM)->list\_elem - offsetof (STRUCT, MEMBER.list\_elem)))

현재 hash\_elem이 나타내는 hash\_item을 리턴한다.

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

hashtalble을 초기화한다

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

hashtable을 비운다.

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

hashtable을 없앤다.

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

hashtable에 hash\_elem을 bucket에 추가한다.

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

hash\_elem이 이미 존재 할경우 교체하고, 기존값을 return한다.

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

hash\_elem을 찾아 return한다.

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

hash\_elem을 찾아 hash에서 제거하고 리턴한다.

/\* Iteration. \*/

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

hashtable 값에 제곱혹은 세제곱한다.

void hash\_first (struct hash\_iterator \*, struct hash \*);

hashtable의 첫번째 값을 리턴한다.

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

hashtable의 다음 값을 리턴한다.

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

hashtable의 현재 값을 리턴한다.

/\* Information. \*/

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

hashtable의 크기를 리턴한다.

bool hash\_empty (struct hash \*);

hashtable이 비었는지에 대한 bool값을 리턴한다.

1-4 bitmap function

/\* Creation and destruction. \*/

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

bitmap을 생성한다.

void bitmap\_destroy (struct bitmap \*);

bitmap을 제거한다.

/\* Bitmap size. \*/

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

bitmap의 크기를 리턴한다.

/\* Setting and testing single bits. \*/

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

bitmap의 idx번째 값을 바꾼다.

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

bitmap의 idx번째 값을 true로 설정한다.

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

bitmap의 idx번째 값을 false로 바꾼다.

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

bitmap의 idx번째 값을 반대로 바꾼다.

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

bitmap의 idx번째 값을 리턴한다.

/\* Setting and testing multiple bits. \*/

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

bitmap의 모든 값을 해당 bool값으로 바꾼다.

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

bitmap의 start 인덱스부터 start+cnt 인덱스까지 해당 bool 값으로 바꾼다

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

비트맵의 해당 bool값의 개수를 리턴한다.

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

비트맵에 start~ star+cnt 인덱스 해당 bool값이 존재하는지에 대한 bool값을 리턴한다.

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

비트맵에 start~ star+cnt true값이 존재하는지에 대한 bool값을 리턴한다.

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

비트맵에 start~ star+cnt true값이 존재하지 않는지 대한 bool값을 리턴한다.

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

비트맵에 start~ star+cnt true값이 모두 true인지에대한 bool값을 리턴한다.

/\* Finding set or unset bits. \*/

#define BITMAP\_ERROR SIZE\_MAX

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

비트맵에 start에서 start+cnt인덱스에서 해당 bool값의 개수를 리턴한다.

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

비트맵에 start에서 start+cnt인덱스에서 해당 bool값이 있다면 그 값을 반대로 바꾸고 그 개수를 리턴한다.