OS Project #0-2

Pintos data structure analysis

Due data : 2017.9.28 23:59

20121635 1반 장종석

목차

1. 리스트(List)관련 function
2. 해시 테이블 (Hash table)관련 function
3. 비트맵 (Bitmap)관련 function
4. 추가 구현 function

리스트 List

리스트 (List)관련 function.

* void list\_init (struct list \*)

struct list \* : 초기화할 list  
list 구조체 변수를 초기화 한다.

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

struct list \* : 찾으려 하는 domain  
return: 해당 domain의 시작 주소.  
list의 시작 원소를 반환한다. List의 head->next가 지시하는 값을 반환한다.

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

struct list \* elem : 다음원소를 구하기 위한 이전 원소를 가르키는 주소값  
return : 다음원소의 주소값.  
전달한 elem의 다음 elem 을 반환한다.

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

struct list\* : 찾으려 하는 domain  
return : 마지막원소의 주소값.  
list의 마지막 elem을반환한다. List의 tail->next 가 지시하는 값을 반환한다.

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

struct list\_elem\* : 이전원소를 구하기 위한 다음 원소를 가르키는 주소값  
return : 이전원소의 주소값.  
전달된 elem의 이전 elem의 주소를 반환한다

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

struct list \* : 찾으려 하는 domain  
return : 시작 원소의 주소값  
list의 head의 주소를 반환한다.

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

struct list\* : 찾으려 하는 domain  
return : 마지막 원소의 주소값  
list의 tail의 주소를 반환한다.

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

struct list\_elem \*before : 새로운 원소를 입력하기 위한 기준 원소의 주소값.  
struct list\_elem \*elem : 새로운 원소를 가르키는 주소값.  
전달된 list\_elem before 앞에 list\_elem elem 의 값을 연결시킨다.

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

struct list\_elem \*before : splice 한 값을 저장하려는 원소  
struct list\_elem \*first : splice 하려하는 domain 의 맨 앞 원소   
struct list\_elem \*last : splice 하려는 domaind의 마지막 뒤 원소.  
전달된 first부터 last 전까지의 원소들을 list\_elem before로 이동한다.

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

struct list\* : 삽입하려하는 domain  
struct list\_elem \*elem : 삽입하려는 elem  
list의 앞에 elem 값을 삽입한다.

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

struct list\* : 삽입하려하는 domain  
struct list\_elem \*elem : 삽입하려는 elem  
list의 마지막에 elem 값을 삽입한다.

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

struct list\_eme \*elem : 삭제하려는 원소의 주소값  
return : 삭제된 다음 원소의 주소값  
전달된 elem을 list에서 제거한 후, elem->next 값을 반환한다.

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

struct list\* : pop 하려하는 list 의 domain.  
return : pop 한 원소의 주소값  
list의 맨 앞에 있는 elem를 list에서 삭제한 후, 해당 elem 의 주소값을 반환한다.

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

struct list\* : pop 하려하는 list 의 domain.  
return : pop 한 원소의 주소값  
list의 맨 뒤에 있는 elem를 list에서 삭제한 후, 해당 elem 의 주소값을 반환한다

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

struct list \* : domain  
return : domain의 가장 앞 원소를 가르키는 주소값.  
list의 맨 앞에 있는 elem 의 주소값을 반환한다.

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

struct list \* : domain  
return : domain의 가장 마지막 원소를 가르키는 주소값.  
list의 맨 뒤에 있는 elem 의 주소값을 반환한다.

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

struct list \* : domain  
return : domain 의 길이  
list의 길이를 반환한다.

* bool list\_empty (struct list \*)

struct list \* : domain  
return : true/ false   
list가 비어있는지 확인하는 함수로, 비어있다면 true, elem이 들어있으면 false를 반환한다.

* void list\_reverse (struct list \*)

struct list \* : domain  
list를 뒤집는다.

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

struct list \* list : domain list  
list\_less\_func : 원소값을 비교하는 함수  
void \*aux : 보조 변수  
list\_less\_func에 의해 list를 정렬한다. List\_less\_func는 list의 에 있는 두개의 elem 값을 비교하여, swap을 한다. Aux의 경우 보조 변수로, NULL처리를 하였다.

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

struct list \* : domain list  
struct list\_elem \* : insert 하녀는 원소의 주소값  
list\_less\_func : 원소값을 비교하는 함수  
void \*aux : 보조변수  
list\_less\_func 에 따라 새로들어온 elem 값을 비교하여 정렬된 상태에 유지되게 값을 넣어준다.

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

struct list \* : domain list  
struct list \* duplicates : 중복된 값을 저장하기 위한 list  
list\_less\_func : 원소값을 비교하는 함수  
void \*aux : 보조변수  
list\_less\_func에 따라, list에 중복되지 않은 데이터를 남긴다. Duplicates의 경우, 중복되어 제거된 elem 값들이 저장된다.

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

struct list \* : domain list  
list\_less\_func : 원소값을 비교하는 함수  
void \*aux : 보조변수  
return : 최대값을 갖는 원소의 주소값.  
list\_less\_func를 이용하여 list 중 최대값을 갖는 elem의 주소를 반환한다.

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

struct list \* : domain list  
list\_less\_func : 원소값을 비교하는 함수  
void \*aux : 보조변수  
return : 최대값을 갖는 원소의 주소값  
list\_less\_func를 이용하여 list 중 최소값을 갖는 elem의 주소를 반환한다.

* #define list\_entry(LIST\_ELEM, STRUCT, MEMBER) ((STRUCT \*) ((uint8\_t \*) &(LIST\_ELEM)->next - offsetof (STRUCT, MEMBER.next)))

어떤 list\_elem 구조체 변수가 있을 때 그 구조체 변수를 포함하는 구조체 변수의 주소를 구하는 매크로 함수이다.

해쉬테이블(hash table)관련 function.

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

struct hash \* : 초기화 할 hash.  
hash\_hash\_func : 해싱하기 위한 함수이다.  
hash\_less\_func : 해시값을 비교하기 위한 함수이다.  
void \*aux : 보조 변수로, 단순히 정수를 저장하는 구조에서는 필요하지 않아 NULL값을 넣어줌.  
return : hash가 초기화 되면 true 아닐경우 false 를 반환한다.  
hash 를 초기화하는 함수.

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

해쉬의 버킷을 비운다.

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

해시의 버킷의 메모리를 제거한다..

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

해시에 새로운 데이터를 추가한다..

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

해시에 새로운 데이터를 추가한다. 만약 동일한 데이터가 있는 경우 전 데이터를 제거하고 새롭게 데이터를 추가하며, 제거된 데이터의 주소를 반환한다. 새로운 데이터의 경우 NULL를 반환한다.

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

해시에서 전달받은 e와 같은 값을 가지는 데이터를 찾아 해당 원소의 주소를 반환한다.

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

해시에서 전달받은 e와 같은 값을 가지는 데이터를 찾아 해당 원소를 제거한 후, 삭제된 원소의 주소를 반환한다.

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

hash를action funcion해당하는 역할을 하도록 한다.

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

iterator가 탐색하고자 하는 해시의 첫 버킷을 가르키도록 한다.

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

iterator 가 가르키고 있는 데이터의 다음 데이터를 가르키도록 한다.

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

iterator가 가르키고있는 현재위치를 반환한다.

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

해시에 존재하는 데이터의 개수를 반환한다.

* bool hash\_empty (struct hash \*)

해시에 데이터가 있으면 true, 없을경우 false.를 반환한다.

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

hash key를 만들어준다.

* unsigned hash\_int (int)

파라미터로 넘겨받은 정수형 변수를 hash\_bytes 함수를 이용해 해싱한 결과 값을 반환한다.

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

리스트에서의 list\_entry와 동일한 역할을 하는 매크로 함수이다. hash\_elem 구조체를 이용해 그 구조체를 포함하는 구조체의 주소를 구한다.

비트맵(bitmap)관련 function.

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

size\_t bit\_cnt : 생성하려 하는 비트맵 크기  
struct bitmap : 생성된 비트맵 주소값.  
bit\_cnt 크기의 비트맵을 생성하고 그 주소를 반환한다.

* void bitmap\_destroy (struct bitmap \*)

struct bitmap \* 메모리 해제를 하려는 bitmap  
파라미터로 넘겨받은 비트맵을 메모리에서 해제한다.

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

const struct bitmap \* : bitmap domain  
return : bitmap 길이  
파리미터로 넘겨받은 비트맵의 길이를 반환한다.

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

struct bitmap\* : domain bitmap  
size\_t idx : bitmap의 index  
bool value : setting value  
비트맵의 idx 번째 비트를 value(1 또는 0)로 설정한다.

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

struct bitmap \* :domain bitmap  
size\_t idx : bitmap’s index  
비트맵의 idx 번째 비트를 참(1)으로 설정한다.

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

struct bitmap\* : domain bitmap  
size\_t idx : bitmap’s index  
비트맵의 idx 번째 비트를 거짓(0)으로 설정한다.

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

struct bitmap \* : domain bitmap  
size\_t idx : bitmap’s index  
비트맵의 idx 번째 비트를 toggle 한다

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

const struct bitmap \* : domain bitmap  
size\_t idx : bitmap’s index  
return 1/0  
비트맵의 idx 번째 비트를 반환한다.

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

struct bitmap \* :domain bitmap  
bool value : setting value  
비트맵의 모든 비트를 valuie로 설정한다.

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

struct bit map \* : bitmap domain  
size\_t start : start index  
size\_t cnt : count of index  
bool value : setting value  
비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들을 value로 설정한다.

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

const struct bitmap \* : bitmap domain  
size\_t start : start index  
size\_t cnt : count of index  
bool value : set value  
비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들 중 값이 value인 비트들의 수를 반환한다.

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

비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들 중 값이 value인 비트가 있다면 참을 없다면 거짓을 반환한다.

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

비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들 중 값이 참(1)인 비트가 있다면 참을 없다면 거짓을 반환한다.

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

비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들 중 값이 참(1)인 비트가 없다면 참을 있다면 거짓을 반환한다.

* bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt)
* 비트맵의 start 번째 비트부터 start 비트를 포함한 cnt 길이만큼의 비트들이 모두 참(1)이면 참을 아니면 거짓을 반환한다.
* size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool value)

비트맵의 start 번째 비트부터 연속된 cnt개 만큼의 비트가 모두 value로 설정된 가장 처음의 인덱스를 반환한다.

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

비트맵의 start 번째 비트부터 연속된 cnt개 만큼의 비트가 모두 value로 설정된 가장 처음의 인덱스를 반환하며, 해당하는 비트들을 모두 토글한다.

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

비트맵을 1byte씩 끊어서 한 줄에 16byte씩 16진수로 출력한다. 비트맵의 시작 주소를 0x00000000이라 했을 때 각 줄의 처음에는 그 줄의 시작 주소가 출력된다

추가구현 function.

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

struct list\_elem \*a : swap 하려는 리스트 원소1  
struct list\_elem \*b : swap 하려는 리스트 원소2  
리스트의 원소 a, b를 서로 교체한다.

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

struct list \*list : list domain  
리스트에 존재하는 원소들을 임의의 순서로 섞는다.

* unsigned hash\_int\_2(int i)

int i : 해시 원소 값.  
return : 해싱 키값.  
파라미터로 넘겨받은 정수형 변수를 일련의 수식을 통하여 해싱한 결과 값을 반환한다.

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

struct bitmap\* : size를 키우려는 bitmap  
int size : 확장시킬 size  
return : bitmap 주소값.  
파라미터로 넘겨받은 비트맵과 같은 내용을 갖지만 길이가 뒤로 size만큼 늘어난 비트맵을 생성하고 그 비트맵의 주소를 반환한다.

* int SystemPrompt(struct list \*data\_set)

struct list \*data\_set : data\_struct를 연결하는 domain  
프로그램이 실행되면 main이 되는 함수로, input을 받는 역할을 한다.

Return 0 / -1(error)

* int MakeTokenTable(char\* inputCmd, char\*\*\*tokenTable)

char \*inputCmd : 프로그램 실행 중 입력받는 parameter  
char \*\*\*tokenTable : 입력받은 인자를 잘라서 저장하는 table

return : token count  
프로그램중 입력되는 인자들을 인자에 맞게 tokenize 하여 배열에 저장해주는 역할을 한다.

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

const struct list\_elem \*a : 비교 대상이 되는 원소1  
const struct list\_elem \*b : 비교 대상이 되는 원소2  
void \*aux : 보조변수

return 1/0  
전달된 리스트 원소들의 데이터 값을 비교하는 함수.

* unsigned hash\_hash(const struct hash\_elem \*e, void \*aux)

const struct hash\_elem \*e : 해싱키를 찾으려는 원소  
void \*aux : 보조변수

return : hashing key

전달된 e가 가르키는 원소값을 찾아 해싱키를 계산하는 함수를 호출하여 그 값을 전달하는 함수

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

const struct hash\_elem \*a : 비교 대상이 되는 해시 원소1

const struct hash\_elem \*b : 비교 대상이 되는 해시 원소2

void \*aux : 보조변수

return true / false

전달된 해시 원소들의 데이터 값을 비교하는 함수.

* void hash\_square(struct hash\_elem \*e, void \*aux)

struct hash\_elem \*e : 제곱하려는 해시 원소

void \*aux : 보조변수

전달된 e가 가르키는 값을 제곱하는 함수.

* void hash\_triple(struct hash\_elem \*e, void \*aux)

struct hash\_elem \*e : 세제곱하려는 해시의 원소

void \*aux : 보조변수

전달된 e 가 가르키는 값을 세제곱하는 함수.

* void hash\_clear\_func(struct hash\_elem \*e, void \*aux)

struct hash\_elem \*e : 제거하려는 해시의 원소

void \*aux : 보조변수

전달된 e가 가르키는 값을 제거해주는 함수.