**Pintos Project 0-2 : Pintos Data Structure**

|  |  |
| --- | --- |
| 과목 명 : | 운영체제 |
| 담당 교수 : | 김영재교수님 |
| 학번 : | 20151600 |
| 이름 : | 장승민 |

**프로젝트 제목 : pintos project 0-2 Pintos Data Structure**

**제출일 : 2019-10-06**

1. **List**
2. void list\_init (struct list \*);

* list를 초기화 하는 함수로써 head.prev는 NULL로 head.next는 list tail로 tail.prev는 list.head로 tail.next는 NULL로 각각 초기화를 해준다.

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

* list의 시작 element를 반환해준다.

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

* 현재 element의 다음 element를 반환해준다.

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

* list의 마지막 element를 반환해준다.
* list구조를 이용한 반복문에 주로 사용

1. struct list\_elem \*list\_rbegin (struct list \*);

* list의 tail전에 element를 찾아서 반환해준다.

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

* 전달받은 parameter의 element의 이전 element를 반환해준다.

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

* 전달받은 parameter의 element의 head element를 반환해준다.

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

* 전달받은 parameter의 element의 head element를 반환해준다.

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

* 해당 리스트의 tail을 반환해준다.

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

* 첫번째 parameter가 가리키고 있는 element 뒤에 두번째 parameter의 element를 껴준다.

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

* first가 가르키는 element에서 last가 가르키는 element사이에 있는 element들을 잘하서 before element가 가르키는 자리에 삽입해준다.

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

* list의 처음 위치에 입력 받은 element를 추가한다.

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

* list의 맨뒤 위치에 입력 받은 element를 추가한다.

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

* 입력 받은 element의 원소를 해당 list의 배열에서 지운다.

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

* 입력 받은 list의 맨 앞 element를 제거한다.

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

* 입력 받은 list의 맨 뒤 element를 제거한다.

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

* 입력 받은 list의 맨 앞 element의 주소를 반환해준다.

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

* 입력 받은 list의 맨 뒤 element의 주소를 반환해준다.

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

* 해당 list 원소의 개수를 반환해준다.

1. bool list\_empty (struct list \*);

* 해당 list가 빈 list인지(원소의 개수가 0)를 알려준다.

1. void list\_reverse (struct list \*);

* list의 element들의 순서를 지금에서 element들을 역순으로 바꾼다.

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

* list의 element의 순서를 정할 때 법칙을 함수 포인터로 정해주기 위해서 함수 포인터를 정의 해논것이다.

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

* list의 두번째 parameter list\_less\_func의 함수 포인터를 기준으로 하여 첫번째 입력 받은 list를 Sorting한다.

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

* 2번째 입력 받은 원소를 1번째 입력받은 list에 입력하는데 list\_less\_func의 기준으로 순서대로 입력한다.

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

* list에 중복받은 원소가 있으면 이 원소들을 삭제해준다.

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

* 해당 list에서 가장 큰 값을 갖는 element를 찾아서 반환해준다.

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

* 해당 list에서 가장 작은 값을 갖는 element를 찾아서 반환해준다.

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

* 첫번째 argument로 넘긴 list\_elem으로부터 해당 list\_elem을 포함하고 있는 2번째 argument struct형태를 반환라는 define문
* List로부터 값을 얻어와 처리하는데 주로 사용

1. **Hash Table**
2. bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux);

* argument로 넘어가는 hash를 초기화한다.
* Hash에 놓기 위해 필수적으로 수행해야한다.
* Hash에 넣을 때 hash\_hash\_func와 hash\_less\_func 함수 포인터를 이용해 법칙을 생성해서 hash를 구성한다.

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

* hash함수를 clear하는 함수이다.

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

* Hash 함수를 제거하는 함수이다.

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

* hash함수에 hash\_elem을 집어넣는 함수 이다.
* Insert 하는 방식은 hash\_함수 법칙에 따라 넣는다.

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

* hash table에다 추가 하는 함수, 그와 같은 원소가 이미 있으면 대체해서 넣어준다.

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

* 2번째 paramter를 받아 hash에서 그 해당 원소를 찾는다.

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

* 2번째 paramter를 받아 hash에서 그 해당 원소를 제거한다.

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

* 해당 해쉬에 대해 함수 파라미터로 받은 기준으로 연산을 수행한다.

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

* 해당 해쉬의 첫번째 원소를 반환한다.

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

* 해당 Hash의 parameter로 받은 iterator 다음 element의 원소를 반환한다.

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

* prarameter로 받은 iterator의 현재 element를 반환한다.

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

* 현재의 Hash사이즈를 반환한다.

1. bool hash\_empty (struct hash \*);

* 해당 Hash가 비웠는지 안 비웠는지를 검사하는 함수이다.

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

* 해쉬의 size-byte를 반환한다.

1. unsigned hash\_string (const char \*);

* 문자열의 Hash를 반환한다.

1. unsigned hash\_int (int);

* Integer I의 해쉬를 반환한다.

1. **Bitmap**
2. struct bitmap \*bitmap\_create (size\_t bit\_cnt);

* size 크기가 bit\_cnt인 bitmap을 만들어 그 주소를 반환한다.

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

* size 크기가 bit\_cnt인 bitmap을 만들어 그 주소를 반환한다.
* buf라는 미리 활당된 메모리 사용

1. size\_t bitmap\_buf\_size (size\_t bit\_cnt);

* 입력 받은 bit\_size에 요구되는 바이트의 크기를 반환한다.

1. void bitmap\_destroy (struct bitmap \*);

* 입력 받은 bit\_map을 free 시킨다.

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

* 입력받은 bitmap의 사이즈를 반환한다.

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

* 첫번째 paramter로 받은 bitmap을 두번째 파라미터 값으로 초기화 시킨다.

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

* 입력 받은 bitmap를 size\_t idx만큼 true로 자동으로 세팅한다.

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

* 입력 받은 bitmap를 size\_t idx만큼 false로 자동으로 세팅한다.

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

* 입력 받은 bitmap을 size\_t 만큼 flip 연산을 해준다.
* True-> false ; false->true 로 바꿔준다.

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

* size\_t만큼 bitmap의 값을 리턴해준다.

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

* bitmap 전체를 입력 받은 값으로 모두 초기화 해준다.

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

* 2번째 파라미터부터 3번째 파라미터의 개수만큼 4번째 파라미터의 값으로 bitmap를 초기화 해준다.

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

* 2번째 파라미터부터 3번째 파라미터의 개수만큼 4번째 파라미터의 값이 몇개 있는 지를 반환한다.

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

* start부터 cnt 만큼 4번째 모두 4번째 파라미터인지(true or false) 조사해서 true,false를 반환한다.

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

* start부터 cnt 만큼 4번째 모두 4번째 파라미터인지(true) 조사해서 true,false를 반환한다.

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

* start부터 cnt 만큼 4번째 모두 4번째 파라미터인지(true) 조사해서 true,false의 역을 반환한다.
* !bitmap\_any을 반환한다.

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

* start부터 cnt 만큼 4번째 모두 4번째 파라미터인지(false) 조사해서 true,false를 반환한다.

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

* start부터 시작해서 cnt만큼을 4번째 parameter와 비교하면서 연속된 bit의 시작 자리를 반환한다.
* 존재하지 않으면 BITMAP\_ERROR를 반환한다.

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

* start부터 시작해서 cnt만큼 4번째 Parameter와 비교하면서 연속된 bit의 시작 자리를 찾고 시작자리부터 cnt만큼 연속된 bit 값을 그 해당 값의 반대값으로 flip 연산을 한다.

1. size\_t bitmap\_file\_size (const struct bitmap \*);

* 입력 받은 bitmap의 사이즈만큼을 byte크기를 반환한다.

1. bool bitmap\_read (struct bitmap \*, struct file \*);

* 2번째 파라미터가 가르 키는 file로부터 bitmap을 읽어온다.

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

* 2번째 파라미터가 가르 키는 file에 bitmap을 쓴다.

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

* 현재의 bitmap의 정보를 16진수로 출력한다.

1. **프로젝트 함수**
2. void list\_swap(struct list\_elem \*a , struct list\_elem \*b)

* swap을 하려는 두 list elements를 a,b를 받아 2개의 element를 swap한다.

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

* shuffle하려는 list를 입력 받고 그 Parameter로 넘어온 list의 element를 무작위로 섞는다.

1. unsigned hash\_int\_2(int i)

* 기존의 hash해주는 함수가 있지만 새로 만들었다.
* Hash값을 구하려는 Integer을 받아 Integer i의 hash 값을 돌려준다.

1. Struct bitmap \* bitmap\_expand(struct bitmap \* bitmap,int size)

* 확장하고 싶은 bitmap과 size를 받아 해당 bitmap의 길이를 size만큼 늘린다.(뒤로 확장)
* 성공시 확장된 bitmap, 실패시 NULL을 반환한다.

1. **내가 만든 함수(포인터 함수)**
2. bool compare(const struct list\_elem \*a , const struct list\_elem \* b, void \*aux)

* a,b가 parameter로 넘어오면 compare함수로 크기를 비교해서 리턴해준다.

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

* Hash의 해당 데이터 값을 반환해주는 함수
* 반환된 값은 hash\_int로 hash\_byte가 반환된다.

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

* a,b가 parameter로 넘어오면 해당 값을 비교해서 반환해주는 함수이다.

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

* 넘어온 hash의 값을 3제곱해주는 함수이다.

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

* 넘어온 hash의 값을 제곱해주는 함수이다.

1. void hash\_action\_destroy(struct hash\_elem \* e,void \*aux)

* hash함수를 제거 할 때 free 해주는 역할을 하는 함수이다.