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

|  |  |
| --- | --- |
| 과목 명 : | [CSE4070] 운영체제 1반 |
| 담당 교수 : | 박성용 교수님 |
| 학번 : | 20151521 |
| 이름 : | 김기범 |

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

**제출일 : 2019년 10월 6일**

목표 : pintos kernel에서 사용하는 data structure인 list, hash table, bitmap을 이해하고 이의 기능을 구현하는 함수를 작성하여 실행가능한 프로그램을 만든다.

1. **List**

|  |  |
| --- | --- |
| Prototype | void list\_init (struct list \*list) |
| Parameter | 초기화 시킬 list |
| Return | 없음 |
| Function | 리스트를 비어있는 리스트로 초기화해준다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_begin (struct list \*list) |
| Parameter | 첫 노드의 주소를 찾을 list |
| Return | Head에 연결된 첫 노드의 주소를 반환 |
| Function | 입력 받은 elem의 다음 elem를 찾아준다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_next (struct list\_elem \*elem) |
| Parameter | 다음 elem의 주소를 찾을 현재 elem |
| Return | 입력 받은 elem의 다음 elem 또는 입력이 마지막 elem 이면 tail을 반환 |
| Function | 연결리스트의 다음 주소를 찾아준다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_head (struct list \*list) |
| Parameter | Head를 찾을 list |
| Return | List의 head |
| Function | 입력 받은 list의 head를 반환 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_end (struct list \*list) |
| Parameter | Tail을 알고자 하는 list |
| Return | List->tail 을 반환 |
| Function | List의 tail을 반환해준다. |

|  |  |
| --- | --- |
| Prototype | bool list\_empty (struct list \*list) |
| Parameter | 비어있는지 확인할 list |
| Return | List가 비어있으면 true, 아니면 false를 반환 |
| Function | List가 비어있는지 확인한다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_remove (struct list\_elem \*elem) |
| Parameter | 원소를 지울 노드의 주소 |
| Return | 제거한 원소의 다음 원소를 반환. |
| Function | 원하는 원소를 리스트에서 제거한다. |

|  |  |
| --- | --- |
| Prototype | void list\_push\_back (struct list \*list, struct list\_elem \*elem) |
| Parameter | 새로운 원소의 주소와 삽입할 list |
| Return | 없음 |
| Function | 새로운 원소를 list의 뒤부터 push한다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_back (struct list \*list) |
| Parameter | 마지막 원소를 찾을 list |
| Return | List의 마지막 원소의 주소를 반환, 비어있는 경우는 정의 안함 |
| Function | 입력 받은 list의 마지막 원소의 주소를 반환 |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_pop\_back (struct list \*list) |
| Parameter | 맨 뒤 원소를 지울 리스트 |
| Return | 리스트에서 지운 맨 뒤 원소 |
| Function | 리스트의 원소를 뒤부터 pop한다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_front (struct list \*list) |
| Parameter | 맨 앞 원소를 찾을 list |
| Return | 맨 앞 원소의 주소 |
| Function | 입력 받은 list의 맨 앞 원소를 찾는다. |

|  |  |
| --- | --- |
| Prototype | void list\_insert (struct list\_elem \*before, struct list\_elem \*elem) |
| Parameter | 삽입할 원소와 삽입할 원소의 위치의 원소 |
| Return | 없음. |
| Function | 리스트에 원하는 원소 elem를 before 앞에 삽입한다. |

|  |  |
| --- | --- |
| Prototype | void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux) |
| Parameter | 원소를 삽입할 list와 값을 비교할 less function, 보조테이터 aux |
| Return | 없음 |
| Function | List에서 원소를 sorting을 유지한채로 삽입한다. |

|  |  |
| --- | --- |
| Prototype | size\_t list\_size (struct list \*list) |
| Parameter | 사이즈를 알고 싶은 list |
| Return | 입력 받은 리스트의 크기를 반환한다. |
| Function | 리스트의 사이즈를 구한다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux) |
| Parameter | 가장 큰 원소를 찾을 list와 작은 값을 찾아 bool을 반환하는 less, 보조 데이터 aux |
| Return | 가장 큰 원소 또는 list가 비어있으면 tail |
| Function | List의 원소 중 가장 큰 원소를 찾는다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux) |
| Parameter | 가장 작은 원소를 찾을 list와 작은 값을 찾아 bool을 반환하는 less, 보조 데이터 aux |
| Return | 가장 작은 원소 또는 list가 비어있으면 tail |
| Function | List의 원소 중 가장 작은 원소를 찾는다. |

|  |  |
| --- | --- |
| Prototype | struct list\_elem \*list\_pop\_front (struct list \*list) |
| Parameter | 맨 앞의 원소를 삭제할 list |
| Return | 삭제한 elem의 주소 |
| Function | 리스트의 앞부터 원소를 pop한다. |

|  |  |
| --- | --- |
| Prototype | void list\_push\_front (struct list \*list, struct list\_elem \*elem) |
| Parameter | 삽입할 list와 새로 넣을 원소를 저장한 elem |
| Return | 없음 |
| Function | List의 앞에 새로운 원소를 삽입한다. |

|  |  |
| --- | --- |
| Prototype | void list\_reverse (struct list \*list) |
| Parameter | 순서를 바꿀 list |
| Return | 없음 |
| Function | List의 연결된 순서를 바꿔준다 |

|  |  |
| --- | --- |
| Prototype | void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux) |
| Parameter | 정렬할 list, 작은 값을 찾는 less function, 보조데이터 |
| Return | 없음 |
| Function | 입력 받은 list를 정렬해준다. |

|  |  |
| --- | --- |
| Prototype | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last) |
| Parameter | 기준이 되는 before의 주소와 삭제할 범위의 first와 last의 주소 |
| Return | 없음 |
| Function | list에서 first부터 last까지 삭제하고 before앞에 삽입한다. |

|  |  |
| --- | --- |
| Prototype | void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux) |
| Parameter | 같은 값이 있는지 확인할 list, 그 값들을 append할 duplicates |
| Return | 없음 |
| Function | list에서 less function에 의해 같은 값들을 list에서 삭제하고 duplicates로 append한다. |

|  |  |
| --- | --- |
| Prototype | bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void\* aux) |
| Parameter | 크기를 비교할 2개 원소의 주소와 보조데이터 aux |
| Return | 값을 비교하고 결과에 따라 true/ false를 반환 |
| Function | 두 개 원소의 값을 비교해서 bool 값을 반환한다. |

* **-추가 구현함수**

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

struct list\_item\* item1 = list\_entry(a, struct list\_item, elem);

struct list\_item\* item2 = list\_entry(b, struct list\_item, elem);

int temp;

temp = item1->data;

item1->data = item2->data;

item2->data = temp;

}

Parameter : 값을 swap할 리스트의 주소 a와 b

Return : 없음

Function : 입력 받은 두 노드의 값을 swap 해준다.

* 입력 받은 두 elem\* 를 list\_entry를 이용해 list\_item구조체로 접근하고 list\_item 속 data를 임시 변수 temp를 이용해 바꾼다.

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

struct list\_elem \*e1, \*e2;

int i = list\_size(list)ㄴ, j;

srand(time(NULL));

for(e1 = list\_begin(list); e1 != list\_end(list); e1 = list\_next(e1)){

j = rand() % (i + 1);

e2 = list\_begin(list);

while(j-- > 0) e2 = list\_next(e2);

list\_swap(e1, e2);

i--;

}

}

Parameter : 원소들을 섞을 리스트

Return : 없음

Function : 리스트의 원소를 랜덤하게 섞어준다.

* Srand()를 이용해 난수를 생성하고 맨 앞의 노드부터 랜덤하게 선택된 인덱스의 노드와 swap하여 셔플을 완성한다.

1. **Hash Table**

|  |  |
| --- | --- |
| Prototype | bool hash\_less(const struct hash\_elem \*a, const struct hash\_elem \*b, void\* aux) |
| Parameter | 값을 비교할 해시의 원소 2개와 보조데이터 aux |
| Return | A의 원소가 더 크면 fasle, |
| Function | 해시의 두 원소의 값을 비교해 true / false를 반환한다. |

|  |  |
| --- | --- |
| Prototype | void hash\_square(struct hash\_elem\* a, void\* aux) |
| Parameter | 값을 변경하고 싶은 원소의 주소 |
| Return | 없음 |
| Function | 해시의 값을 두배로 만든다. |

|  |  |
| --- | --- |
| Prototype | void hash\_triple(struct hash\_elem\* a, void\* aux) |
| Parameter | 값을 변경하고 싶은 원소의 주소 |
| Return | 없음 |
| Function | 해시의 값을 세배로 만든다. |

|  |  |
| --- | --- |
| Prototype | void hash\_destructor(struct hash\_elem\* a, void\* aux) |
| Parameter | 지우고 싶은 해시 element와 보조데이터 aux |
| Return | 없음 |
| Function | 입력 받은 element를 메모리 해제해준다. |

|  |  |
| --- | --- |
| Prototype | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux) |
| Parameter | 새로 만들 해시테이블과 사용할 hash\_hash\_func과 hash\_less func |
| Return | 초기화 성공하면 true, 실패하면 false를 반환 |
| Function | 해시테이블을 초기화한다. |

|  |  |
| --- | --- |
| Prototype | unsigned hash\_func(const struct hash\_elem\* a, void\* aux) |
| Parameter | key값을 얻기 위한 data를 저장하고 있는 a와 보조데이터 aux |
| Return | Data의 key값 |
| Function | 입력 받은 data를 key으로 변환한다. |

|  |  |
| --- | --- |
| Prototype | void hash\_first (struct hash\_iterator \*, struct hash \*) |
| Parameter | 해시테이블과 첫번째 원소를 가리킬 iterator |
| Return | 없음 |
| Function | 해시테이블의 첫번째 원소를 찾는다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_next (struct hash\_iterator \*) |
| Parameter | 현재 해시 element의 Hash\_iterator |
| Return | 해시테이블에서 다음 원소의 주소를 반환 |
| Function | 해시테이블에서 다음 원소를 찾아준다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_cur (struct hash\_iterator \*) |
| Parameter | 현재 hash\_iterator의 주소 |
| Return | 해시테이블에서 현재 element를 반환하거나 테이블의 마지막이면 NULL을 반환 |
| Function | 해시테이블에서 현재 element를 반환한다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*) |
| Parameter | 해시테이블과 삽입할 새로운 원소 |
| Return | 새로운 원소와 같은 값이 없으면 NULL, 있으면 그 element의 주소 |
| Function | 새로운 원소를 해시테이블에 삽입한다. |

|  |  |
| --- | --- |
| Prototype | void hash\_apply (struct hash \*, hash\_action\_func \*) |
| Parameter | Action function을 적용할 해시와 action function |
| Return | 없음 |
| Function | 해시에 action function을 적용한다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*) |
| Parameter | 찾을 값을 가진 elem과 찾을 hash table |
| Return | 값을 찾으면 그 값을, 없으면 NULL을 반환. |
| Function | Hash table안에서 원하는 값을 찾는다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*) |
| Parameter | 해시 테이블과 삭제할 원소 |
| Return | 삭제한 원소의 주소를 반환. |
| Function | 원하는 원소를 찾아서 해시테이블에서 지운다. |

|  |  |
| --- | --- |
| Prototype | bool hash\_empty (struct hash \*) |
| Parameter | 비어있는지 확인할 해시테이블 |
| Return | 해시테이블이 비어있으면 true, 아니면 false 반환 |
| Function | 해시테이블이 비어있는지 확인하고 그 값을 반환한다. |

|  |  |
| --- | --- |
| Prototype | size\_t hash\_size (struct hash \*) |
| Parameter | 크기를 확인할 해시테이블 |
| Return | 해시테이블의 사이즈 |
| Function | 해시테이블의 크기를 반환한다. |

|  |  |
| --- | --- |
| Prototype | void hash\_clear (struct hash \*, hash\_action\_func \*) |
| Parameter | 삭제할 해시테이블과 삭제할 때 사용할 destructor 함수 |
| Return | 없음 |
| Function | 해시테이블의 element와 bucket을 삭제한다. |

|  |  |
| --- | --- |
| Prototype | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*) |
| Parameter | 해시테이블과 새로운 원소 |
| Return | 이미 값이 있으면 그 element의 주소, 없다면 NULL |
| Function | 해시테이블에 새로운 원소를 추가하고 기존의 것이 있으면 대체한다. |

**-추가 구현함수**

Unsigned hash\_int\_2(int i){

i = (i \* 256) % 13;

return hash\_bytes(&i, sizeof i);

}

Parameter : 원소의 data 값

Return : 원소의 data에 대한 key 값

Function : hash\_int와 동일하게 data를 해시 테이블에 저장하기 위해 key값으로 변환한다.

* 입력 받은 data의 값을 임의로 변환하고 이를 다시 key 값으로 만들기 위해 hash\_bytes로 전달한다.

1. **Bitmap**

|  |  |
| --- | --- |
| Prototype | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| Parameter | 원하는 bitmap의 사이즈 |
| Return | 할당성공하면 bitmap의 주소를, 실패하면 NULL을 반환 |
| Function | 입력 받은 사이즈의 크기로 bitmap을 할당한다. |

|  |  |
| --- | --- |
| Prototype | size\_t bitmap\_size (const struct bitmap \*b) |
| Parameter | 사이즈를 알고싶은 bitmap |
| Return | Bitmap의 사이즈 |
| Function | Bitmap의 사이즈를 반환한다. |

|  |  |
| --- | --- |
| Prototype | bool bitmap\_test (const struct bitmap \*b, size\_t idx) |
| Parameter | Bitmap과 확인할 index |
| Return | Bitmapd의 idx 값에서의 bool값(true / false) |
| Function | Bitmap의 idx의 값이 true인지 false인지 확인한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_destroy (struct bitmap \*) |
| Parameter | 삭제할 bitmap |
| Return | 없음 |
| Function | 입력 받은 Bitmap을 삭제해준다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_set (struct bitmap \*, size\_t idx, bool) |
| Parameter | 값을 설정할 bitmap과 설정할 bool 값 |
| Return | 없음 |
| Function | 입력 받은 범위의 Bitmap값을 입력 받은 bool값으로 설정한다. |

|  |  |
| --- | --- |
| Prototype | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt) |
| Parameter | 지정된 범위의 값이 모두 true인지 확인할 bitmap과 확인해볼 시작과 끝 인덱스 start, cnt |
| Return | Start~start+cnt의 bitmap 값이 모두 true이면 true, 아니면 false를 반환. |
| Function | Bitmap의 지정된 범위의 값이 모두 true인지 확인한다. |

|  |  |
| --- | --- |
| Prototype | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt) |
| Parameter | 지정된 범위 안의 값 중 true가 있는지 확인할 bitmap과 확인해볼 시작과 끝 인덱스 start, cnt |
| Return | Start~start+cnt의 bitmap 값이 하나라도 true이면 true, 아니면 false를 반환. |
| Function | Bitmap의 지정된 범위에서 bitmap 값이 하나라도 true인지 확인한다. |

|  |  |
| --- | --- |
| Prototype | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool) |
| Parameter | 탐색할 bitmap과 탐색 시작 인덱스 start, 범위 인덱스 cnt, 탐색하는 bool 값 |
| Return | Bitmap의 start~start+cnt에서 입력 받은 bool값이 하나라도 있으면 true, 없으면 false 반환 |
| Function | Bitmap의 start~start+cnt에서 입력 받은 bool값이 존재하는지 확인한다. |

|  |  |
| --- | --- |
| Prototype | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool) |
| Parameter | 입력 받은 bool값의 개수를 찾을 bitmap과 시작인덱스 start와 범위인덱스 cnt와 그 bool값 |
| Return | Bitmap의 start~start+cnt에서 입력 받은 bool값의 개수를 반환 |
| Function | Bitmap의 지정된 범위에서 입력 받은 bool값의 개수를 반환한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_dump (const struct bitmap \*) |
| Parameter | 16진수로 출력할 bitmap |
| Return | 없음 |
| Function | Bitmap의 값을 16진수로 출력해준다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_set\_all (struct bitmap \*, bool) |
| Parameter | 값을 바꿀 bitmap 과 바꿀 bool값 |
| Return | 없음 |
| Function | 입력 받은 bool값으로 bitmap을 set 한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool) |
| Parameter | 값을 바꿀 bitamp과 시작과 끝 index, 바꾸려는 bool값 |
| Return | 없음 |
| Function | Start와 cnt를 입력받아 start~start+cnt까지 bitmap 값을 입력 받은 bool값으로 변경한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_flip (struct bitmap \*, size\_t idx) |
| Parameter | 값을 반대로 저장할 bitmap과 그 인덱스 idx |
| Return | 없음 |
| Function | Bitmap의 idx가 true 이면 false로, false이면 true로 set |

|  |  |
| --- | --- |
| Prototype | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt) |
| Parameter | 확인하려는 bitmap과 start 인덱스와 확인할 개수 cnt |
| Return | Bitmap의 start~start+cnt가 모두 false이면 true, 아니면 false를 반환. |
| Function | Bitmap의 start~start+cnt의 값에 따라 true/false를 반환한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_reset (struct bitmap \*, size\_t idx) |
| Parameter | 값을 바꿀 비트맵과 인덱스 idx |
| Return | 없음 |
| Function | 입력 받은 bitmap의 idx 값을 false로 set한다. |

|  |  |
| --- | --- |
| Prototype | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool) |
| Parameter | 탐색할 bitmap, 시작인덱스 start, 범위인덱스 cnt, 찾으려는 bool값 |
| Return | Bitmap의 start~start+cnt에서 원하는 bool값이 연속으로 존재하면 첫 인덱스를 반환하고, 없으면 BITMAP\_ERROR를 반환 |
| Function | Bitmap의 지정한 범위만큼 원하는 bool값이 존재하는지 확인하고 있으면 bool값을 flip시킨다. 없으면 BITMAP\_ERROR를 반환한다. |

|  |  |
| --- | --- |
| Prototype | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool) |
| Parameter | 탐색해볼 bitmap과 시작인덱스 start, 탐색 범위인덱스 cnt, 찾아볼 bool 값 |
| Return | Bitmap의 start~start+cnt에서 원하는 bool값이 연속으로 존재하면 첫 인덱스를 반환하고, 없으면 BITMAP\_ERROR를 반환 |
| Function | Bitmap의 지정한 범위만큼 원하는 bool 값이 존재하는지 확인한다. |

|  |  |
| --- | --- |
| Prototype | void bitmap\_mark (struct bitmap \*, size\_t idx) |
| Parameter | True로 set할 bitmap과 그 인덱스idx |
| Return | 없음 |
| Function | 입력 받은 bitmap의 idx값을 true로 set한다. |

* **추가 구현 함수**

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

size\_t temp\_size, i;

struct bitmap\* temp;

temp = bitmap\_create(bitmap\_size(bitmap) + (size\_t)size);

if(size < 0 || size > 65536) return NULL;

for(i = 0; i < (int)bitmap\_size(bitmap); i++){

if(bitmap\_test(bitmap, i) == true) bitmap\_mark(temp, i);

}

return temp;

}

Parameter: 사이즈를 변경할 bitmap과 변경할 size

Return : 확장에 성공하면 확장한 bitmap을, 실패하면 NULL을 반환.

Function : bitmap의 크기를 입력 받은 size만큼 뒤부터 확장한다.

* 원래의 bitmap의 사이즈와 입력 받은 사이즈를 더하여 새로운 bitmap을 생성하고 원래 bitmap의 크기만큼 for문을 돌면서 원래 bitmap의 값이 true이면 새로운 bitmap도 bitmap\_mark를 사용해 true로 만들어준다.