**Operating Systems : Project #0-2. Pintos data structure analysis**

**-document-**

1반 20131527 김상혁

**추가한 스트럭트**

struct list\_infos

* list의 이름과 각 list의 주소 값을 저장한다.

struct hash\_infos

* hash의 이름과 각 hash의 주소 값을 저장한다.

struct bitmap\_infos

* bitmap의 이름과 각 bitmap의 주소 값을 저장한다.

struct list\_data

* list\_elem와 data를 저장한다.

struct hash\_data

* hash\_elem와 data를 저장한다.

**List part**

void list\_init (struct list \*)

* 넘겨 받은 list를 사용 할 수 있게 초기화한다.

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

* 넘겨 받은 list의 첫 번째 list\_elem를 반환한다.

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

* 넘겨 받은 list\_elem에 다음 list\_elem를 반환한다.

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

* 넘겨 받은 list의 tail list\_elem를 반환한다.

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

* 넘겨 받은 list의 마지막 list\_elem를 반환한다.

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

* 넘겨 받은 list\_elem의 이전 list\_elem를 반환한다.

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

* 넘겨 받은 list의 head list\_elem를 반환한다.

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

* 넘겨 받은 list의 head list\_elem를 반환한다.

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

* 넘겨 받은 list의 tail list\_elem를 반환한다.

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

* 첫 번째로 넘겨 받은 list\_elem 앞에 두 번째로 넘겨 받은 list\_elem를 연결한다.

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

* 넘겨 받은 first list\_elem부터 last list\_elem까지 원래 있던 list에서 분리 한 후 넘겨 받은 before list\_elem 앞에 연결한다.

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

* 넘겨 받은 list의 앞 쪽에 넘겨 받은 list\_elem를 삽입한다. 삽입 할 때 list\_insert를 사용한다.

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

* 넘겨 받은 list의 뒤쪽에 넘겨 받은 list\_elem를 삽입한다. 삽입 할 때 list\_insert를 사용한다.

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

* 넘겨 받은 list\_elem를 제거한 후 다음으로 연결 되어 있는 list\_elem를 반환한다.

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

* 넘겨 받은 list의 마지막 list\_elem를 제거한다.

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

* 넘겨 받은 list의 첫 번째 list\_elem를 제거한다.

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

* 넘겨 받은 list의 첫 번째 list\_elem를 반환한다.

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

* 넘겨 받은 list의 마지막 list\_elem를 반환한다.

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

* 넘겨 받은 list의 크기를 반복문을 통해 구한 후 list의 크기를 반환한다.

bool list\_empty (struct list \*)

* 넘겨 받은 list가 비어있는지 확인 한다. 비어있을 경우 true를 반환하고 비어있지 않을 경우 false를 반환한다. 넘겨 받은 list의 head값과 tail값을 비교하여 비어있는지 확인한다.

void list\_reverse (struct list \*)

* 넘겨 받은 list의 list\_elem 순서를 반대로 재배치 한다. 반복문을 통해 list\_elem를 탐색하면서 swap를 통해 list\_elem의 순서를 반대로 배치한다.

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

* 넘겨 받은 list의 list\_elem 순서를 list\_data에 저장되어 있는 data를 기준으로 오름차순 정렬한다.

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

* 넘겨 받은 list\_elem의 list\_data data 값 보다 큰 값을 가지고 있는 넘겨 받은 list의 list\_elem 앞에 넘겨 받은 list\_elem를 삽입한다.

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

* 넘겨 받은 첫 번째 list의 list\_elem의 앞 뒤 data 값이 같을 경우 list\_remove통해 제거 한후 넘겨 받은 두 번째 list가 NULL이 아닐 경우에는 list\_push\_back를 통해 제거 한 list\_elem를 넘겨 받은 두 번째 list에 삽입한다.

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

* 넘겨 받은 list의 list\_elem 중 가장 큰 값을 가지고 있는 list\_elem를 반환한다.

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

* 넘겨 받은 list의 list\_elem 중 가장 작은 값을 가지고 있는 list\_elem를 반환한다.

**추가 구현 함수**

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

* 넘겨 받은 두 list\_elem를 바꾸어준다.

void list\_shuffle(struct list \*list)

* 넘겨 받은 list의 list\_elem를 무작위로 바꾸어준다.

bool less\_function\_l(const struct list\_elem\*, const struct list\_elem\*, void \*aux)

* 넘겨 받은 두 list\_elem에 대하여 첫 번째 list\_elem가 작을 경우 true를 반환하고 그 외의 경우에는 false를 반환한다.

int find\_list\_struct(struct list\_infos\*, int, char\*)

* 입력 받은 list struct의 이름을 통해 list의 주소 값이 저장되어 있는 index를 반환한다.

struct list\_elem\* find\_list\_elem(struct \*list, int)

* 넘겨 받은 list의 넘겨 받은 index번째에 list\_elem를 반환한다.

**Hash part**

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

* 넘겨 받은 hash를 초기화 한다.

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

* 넘겨 받은 hash의 모든 성분들을 제거한다.

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

* 넘겨 받은 hash의 모든 성분을 제거한 후 hash의 buckets를 제거한다.

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

* 넘겨 받은 hash에 hash\_elem를 삽입한다.

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

* 넘겨 받은 hash에 넘겨 받은 hash\_elem를 삽입한다. 만약 넘겨 받은 hash에 넘겨 받은 hash\_elem가 있을 경우 제거하고 삽입한다.

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

* 넘겨 받은 hash에 넘겨 받은 hash\_elem가 존재 할 경우 hash\_elem를 반환한다.

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

* 넘겨 받은 hash에서 넘겨 받은 hash\_elem를 제거한 후 hash\_elem를 반환한다.

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

* 넘겨 받은 hash의 성분을 입력에 따라 제곱 또는 세 제곱을 한다.

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

* 넘겨 받은 hash의 성분을 hash\_iterator에 연결한다.

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

* 넘겨 받은 hash\_iterator의 다음 hash\_elem를 반환한다.

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

* 넘겨 받은 hash\_iterator의 hash\_elem를 반환한다.

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

* 넘겨 받은 hash의 성분 개수를 반환한다.

bool hash\_empty (struct hash \*)

* 넘겨 받은 hash가 비어있을 경우 true를 반환하고 비어있지 않을 경우 false를 반환한다.

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

* 버프에 있는 hash의 SIZE bytes를 반환한다.

unsigned hash\_string (const char \*)

* hash의 string S를 반환한다.

unsigned hash\_int (int)

* 정수 i에대한 hash를 반환한다.

**추가 구현**

unsigned hash\_int\_2(int i)

* 넘겨 받은 i값이 음수 일 경우 -2\*i를 반환하고 양수일 경우 에는 2\*i-1를 반환한다. 만약 i가 0일경우에는 0을 반환한다.

int find\_hash\_struct(struct hash\_infos\*, int, char\*)

* 입력 받은 hash struct의 이름을 통해 hash의 주소 값이 저장되어 있는 index를 반환한다.

bool less\_function\_h(const struct hash\_elem\*, const struct hash\_elem\*, void \*aux)

* 넘겨 받은 두 hash\_elem에 대하여 첫 번째 hash\_elem가 작을 경우 true를 반환하고 그 외의 경우에는 false를 반환한다.

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

* 넘겨 받은 hash\_elem에 대한 hash key값을 반환한다.

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

* 넘겨 받은 hash\_elem의 data 값을 제곱하여 저장한다.

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

* 넘겨 받은 hash\_elem의 data 값을 세 제곱하여 저장한다.

void hash\_action\_dex(struct hash\_elem\*, void \*aux)

* 넘겨 받은 hash\_elem를 제거한다.

**Bitmap part**

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

* 넘겨 받은 bit\_cnt 크기의 bitmap를 만들어 만든 bitmap를 반환한다.

void bitmap\_destroy (struct bitmap \*);

* 넘겨 받은 bitmap를 제거한다.

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

* 넘겨 받은 bitmap의 길이를 반환한다.

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

* 넘겨 받은 bitmap의 idx번째를 넘겨 받은 bool값으로 변환한다.

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

* 넘겨 받은 bitmap의 idx번째를 true로 변환한다.

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

* 넘겨 받은 bitmap의 idx번째를 false로 변환한다.

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

* 넘겨 받은 bitmap의 idx번째가 true일 경우 false를 변환하고 false일 경우에는 true로 변환한다.

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

* 넘겨 받은 bitmap의 idx번째를 반환한다.

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

* 넘겨 받은 bitmap 전체를 넘겨 받은 bool 값으로 변환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개를 넘겨받은 bool값으로 변환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개까지 넘겨 받은 bool값과 같은 값을 갖는 값의 개수를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개까지 넘겨 받은 bool값이 있으면 true를 반환하고 없으면 false를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개까지 true값이 있으면 true를 반환하고 없으면 false를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개까지 true값이 있으면 false를 반환하고 없으면 true를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 cnt개까지 전부 true일 경우 true를 반환하고 전부 true가 아닐 경우 false를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 bitmap를 탐색하면서 현재 탐색하고 있는 지점부터 cnt개까지가 전부 넘겨 받은 bool값과 같을 경우 현재 탐색 지점을 반환한다. 만약 조건에 해당하는 지점이 없을 경우 BITMAP\_ERROR를 반환한다.

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

* 넘겨 받은 bitmap의 start지점부터 bitmap를 탐색하면서 현재 탐색하고 있는 지점부터 cnt개까지가 전부 넘겨 받은 bool값과 같은 지점이 존재하는 경우 그 지점부터 cnt개까지를 넘겨받은 bool 반대 값으로 변환시킨다.

void bitmap\_dump (const struct bitmap \*)

* 넘겨 받은 bitmap를 16진수로 출력한다.

**추가 구현**

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

* 넘겨 받은 bitmap를 넘겨 받은 size만큼 확장 시켜 반환한다.

int find\_bitmap\_struct(struct bitmap\_infos\*, int, char\*)

* 입력 받은 bitmap struct의 이름을 통해 bitmap의 주소 값이 저장되어 있는 index를 반환한다.