Bivas Maiti
bmaiti@iu.edu

Assignment 4: Memory Allocation

Data Structures-/include/malloc.h

buffpoolitem

BuffpoolItem is defined as an item in the buffpoolsnaptab table. Below is the implementation of the buffpoolitem struct

```
typedef struct{
        bpid32 pool_id;
        int total_buffers;
        int allocated_buffers;
        int allocated_bytes;
        int fragmented_bytes;
        buffitem* buffhead;

} buffpoolitem;
```

Pool_id is the id of the buffer pool
Total_buffers is the total number of buffers in the buffer pool
Allocated_buffers is the total number of buffers allocated currently to different processes.
allocated_bytes is the total number of bytes allocated currently to different processes.
fragmented_bytes is the total number of fragmentd bytes of the buffer pool.
Buffhead is a pointer to the head of a linked list which keeps track of the buffers that are currently allocated in the bufferpool, with their addresses and actual sizes of the requests. Below is the struct for buffitem

buffitem

```
typedef struct{
        void* bufaddr;
        int size;
        struct buffitem* bnext;
        struct buffitem* bprev;
} buffitem;
```

Buffaddr is the address of the buffer returned to the process that calls malloc
Size is the size requested in malloc.
Bnext points to the next node in the linked list.
Bprev points to the previous node in the linked list.

bufpoolsnaptab

This is the external table which is initialized with NBPOOLS number of rows, i.e. the maximum number of buffers pools. For each bufferpool in the system there is an entry in the system, with the bufferpool id as the id of the row.

Memory Allocation

xmalloc

xmalloc function takes an arguement 'size' of type int, allocates a suitable buffer and returns the address of the allocated memory to the caller function. Below are some of the key decisions made or functionalities implemented by xmalloc while allocating memory from buffer pools

Bivas Maiti
bmaiti@iu.edu

Is the size requested less than the maximum size of the buffer allowed by xinu? If not, return SYSERR.

```
Do If number of existing bufferpools=0?
    Create a new buffer pool of the size requested by the user (or BP_MINB if it is less than
    BP_MINB)
If there are buffer pools present in the system?
    Find the “best-fitting” buffer pool for the request, i.e. the buffer pool with minimum size which
    is more than the requested size
    Is the best fitting buffer size more than twice of the requested size?
        If no, allocate buffer in the “best-fitting” buffer pool
        If yes, create a new buffer pool and allocate buffer to the requested process
    If there are no “best fitting” buffers, i.e. the maximum buffer size is less than the requested
    size, create a new bufferpool with twice the size of the maximum buffer size and allocate a
    new buffer to the process.
```

During allocation of each request, update the bufpoolsnaptab table with values from the request, i.e. number of bytes allocated, number of buffers allocated, fragmented bytes, etc.
When a new buffer pool is created, BP_MAXN buffers are created in the buffer pool. This is the maximum value of the number of buffers in a pool.

Xfree

Xfree takes a pointer and frees that allocation which was previously allocated by xmalloc. The functionality is as follows:

Get the buffpool id from the address by decrementing the value of the pointer.

Find out the original size of the request for which the allocation was done from the linked list in buffpoolitem. Update the buffpoolsnaptab table with the values, decrementing the allocated buffers, allocated bytes, etc.

Free the buffer by calling XINU freebuf call.

xheap_snapshot

xheap_snapshot doesn’t take any argument and returns a character pointer to a string, which gives us the current snapshot of the buffpoolsnaptab table. The results from this will change with time due to different xmalloc/xfree requests.

Relevant Files

Relevant files for this implementation are:

- System/malloc.c
- Include/malloc.h
- shell/xsh_xmalloc_test.c

Testing

Code for testing is inside shell/xsh_xmalloc_test.c. This can be changed to test the implementation. Below command can be used for testing in XINU:

```
xsh $ xmalloc_test
```