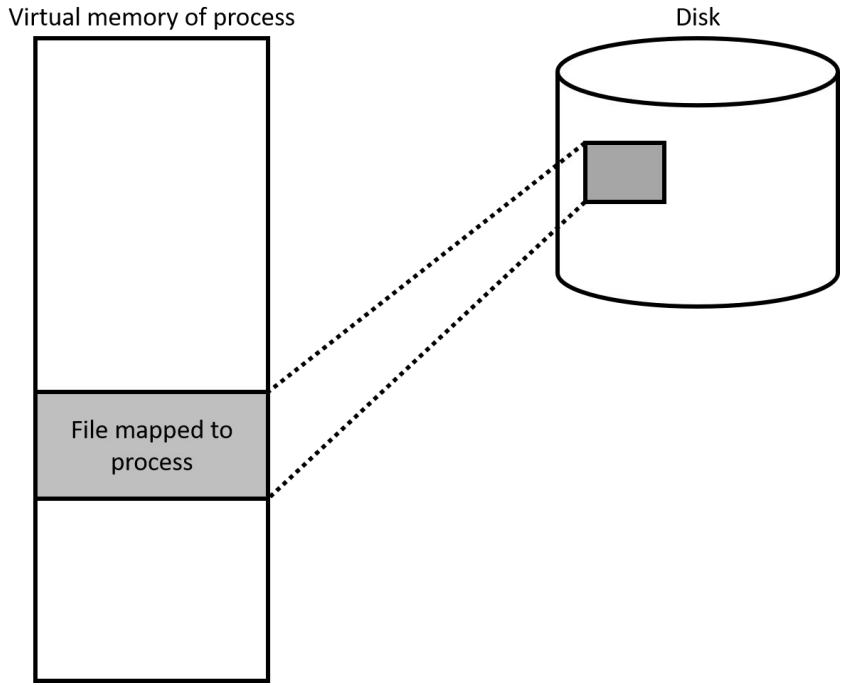
CS2106 Lab 4

Bernard Teo Zhi Yi (bernardteo@u.nus.edu)

Lab 4 Objectives

- Implementing a heap in shared memory using the first-fit algorithm

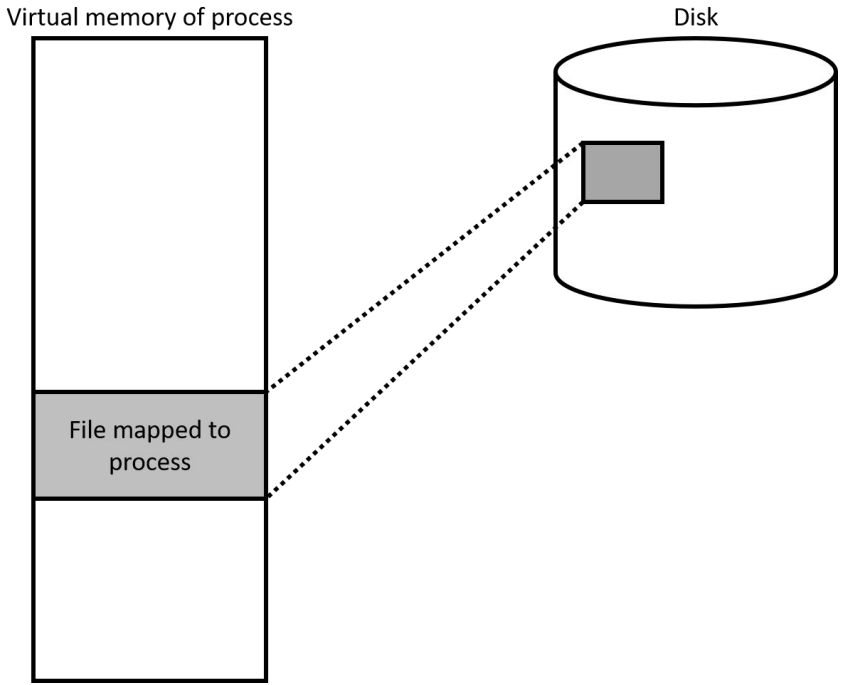
Exercise 0 – Memory Mapped Files



Exercise 0 – Memory Mapped Files

If the file does not already exist on disk:

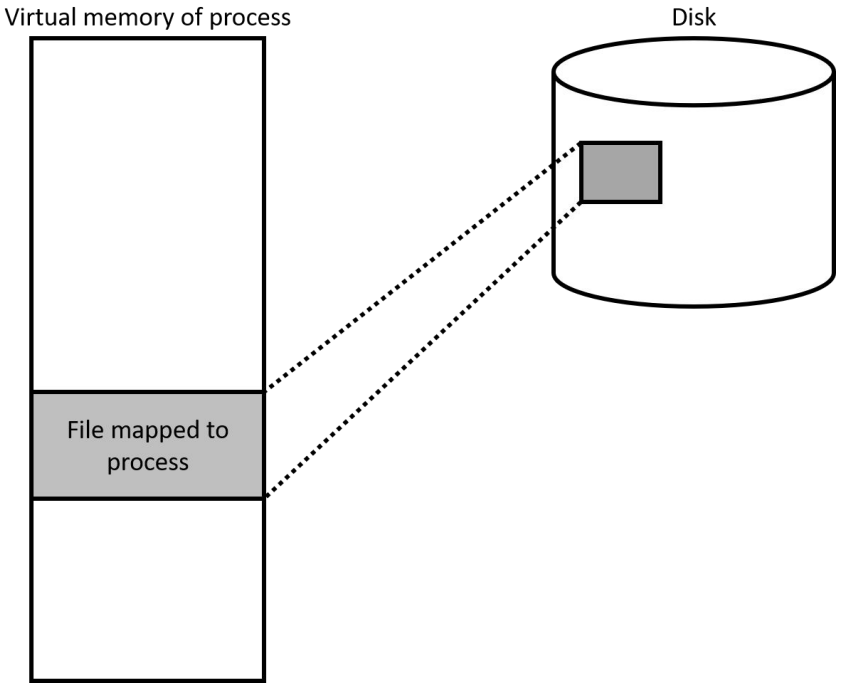
1. Create the file
2. Resize the file to the desired size
3. Map the file into memory
4. Close the file



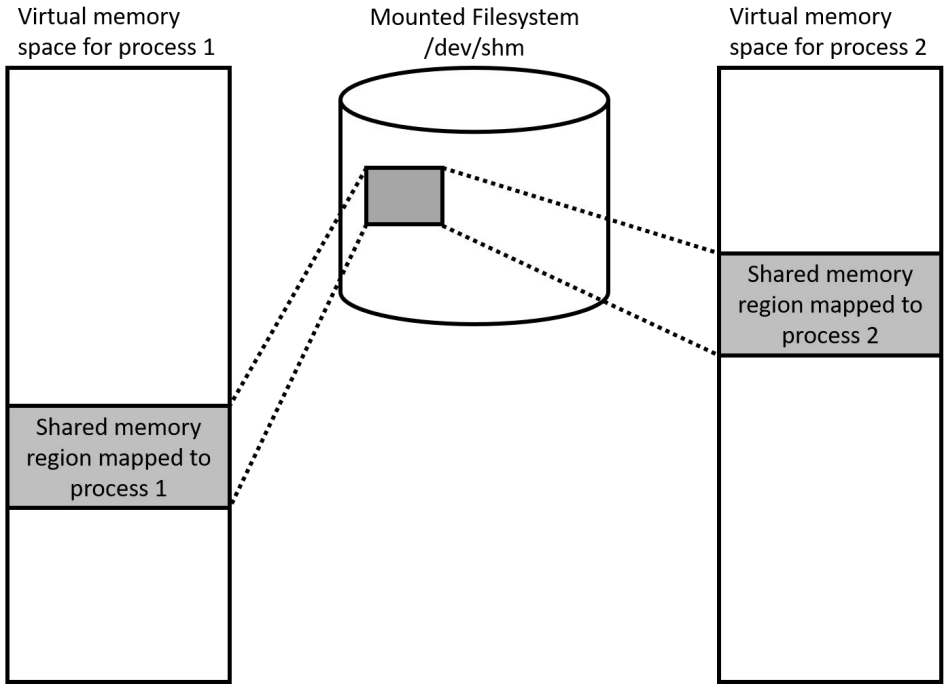
Exercise 0 – Memory Mapped Files

If the file already exists on disk:

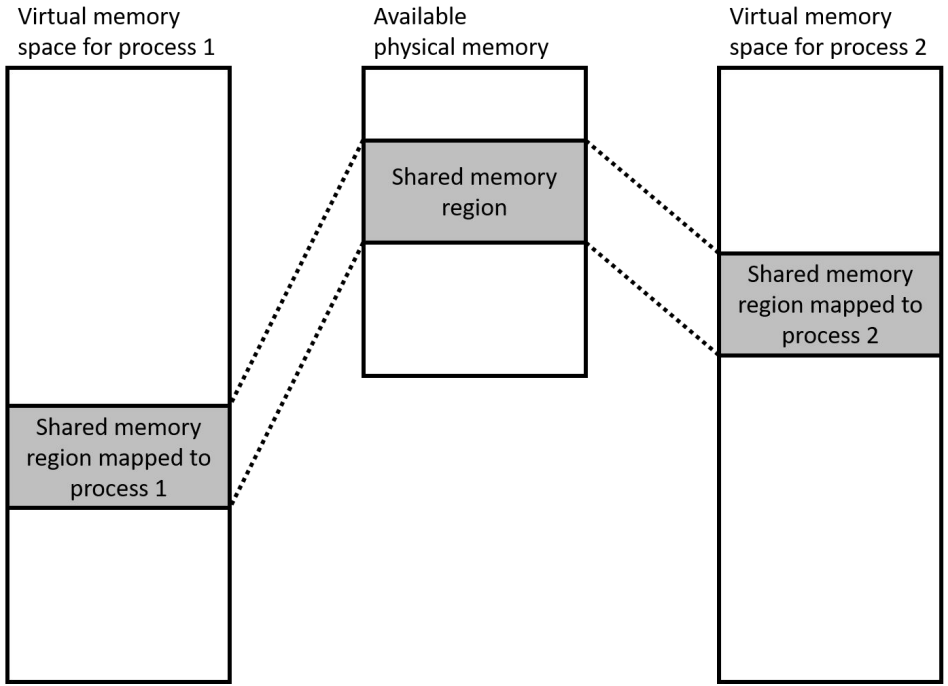
1. Open the file
2. Map the file into memory
3. Close the file



Exercise 1-4


Virtual memory space for process 1
Mounted Filesystem /dev/shm
Virtual memory space for process 2
Shared memory region mapped to process 1
Shared memory region mapped to process 2

Exercise 1-4


Virtual memory space for process 1
Available physical memory
Virtual memory space for process 2
Shared memory region
Shared memory region mapped to process 1
Shared memory region mapped to process 2

Exercise 1-4

```
// Main Process
// create new shared heap
mem = shmheap_create("test.bin", mem_length);
```

```
// Other Processes
// connect to existing shared heap
mem = shmheap_connect("test.bin");
```

```
// allocate and free many objects from any threads...
// allocate an object
ptr = shmheap_alloc(mem, sz);
// free an object
shmheap_free(mem, ptr);
```

```
// disconnect from shared heap
shmheap_disconnect(mem);
```

```
// delete shared heap
shmheap_destroy("test.bin", mem);
```

Exercise 1a

```
// Main Process
// create new shared heap
mem = shmheap_create("test.bin", mem_length);
```

```
// Other Processes
// connect to existing shared heap
mem = shmheap_connect("test.bin");
```

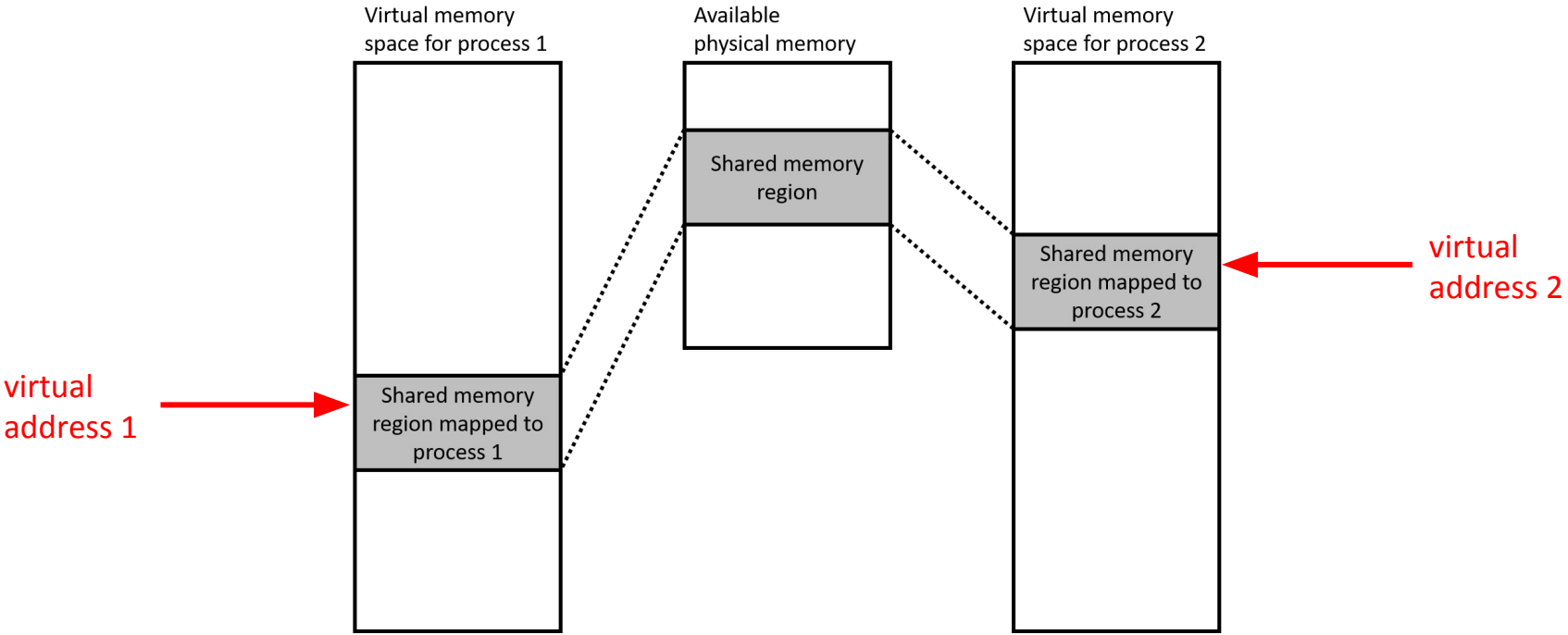
```
// allocate and free many objects from any threads...
// allocate an object
ptr = shmheap_alloc(mem, sz);
// free an object
shmheap_free(mem, ptr);
```

You only need to allocate one object for ex1, and don't need to free any objects

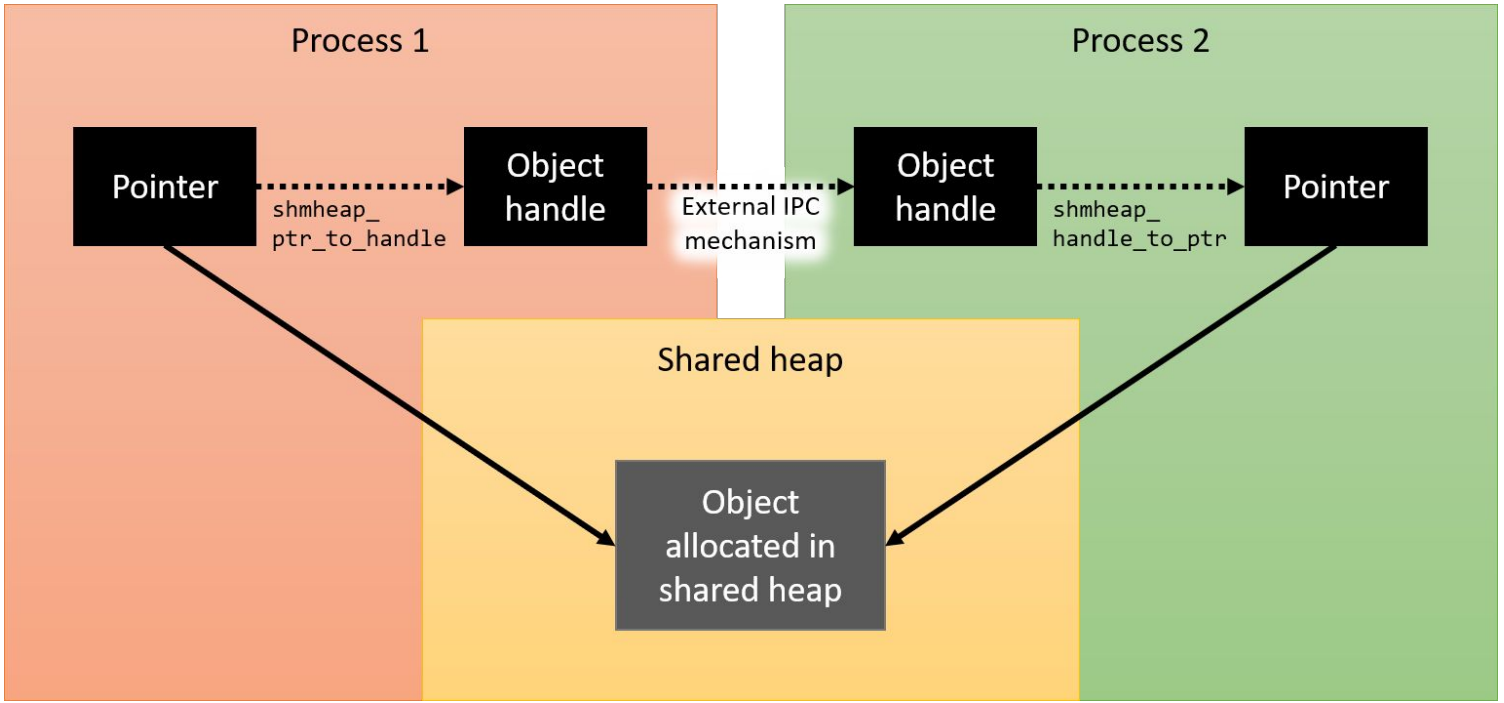
```
// disconnect from shared heap
shmheap_disconnect(mem);
```

```
// delete shared heap
shmheap_destroy("test.bin", mem);
```

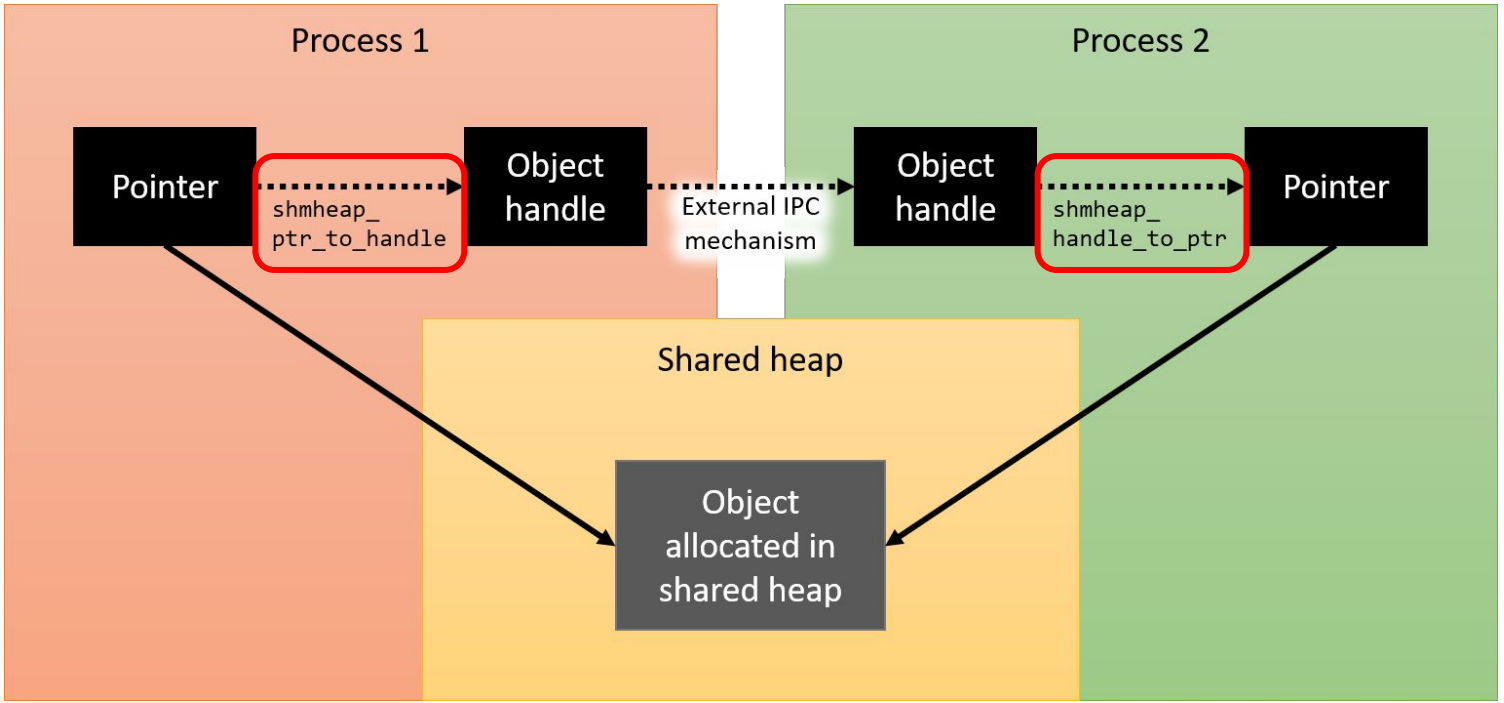
Exercise 1b


Virtual memory space for process 1
Available physical memory
Virtual memory space for process 2
Shared memory region
Shared memory region mapped to process 1
Shared memory region mapped to process 2
virtual address 1
virtual address 2

Exercise 1b


Process 1
Process 2
Pointer
shmheap_
ptr_to_handle
Object
handle
External IPC
mechanism
Object
handle
shmheap_
handle_to_ptr
Pointer
Shared heap
Object
allocated in
shared heap

Exercise 1b


Process 1
Process 2
Pointer
shmheap_
ptr_to_handle
Object
handle
External IPC
mechanism
Object
handle
shmheap_
handle_to_ptr
Pointer
Shared heap
Object
allocated in
shared heap

Exercise 1 – things you need to implement

- Structs:
  - `shmheap_memory_handle`
  - `shmheap_object_handle`

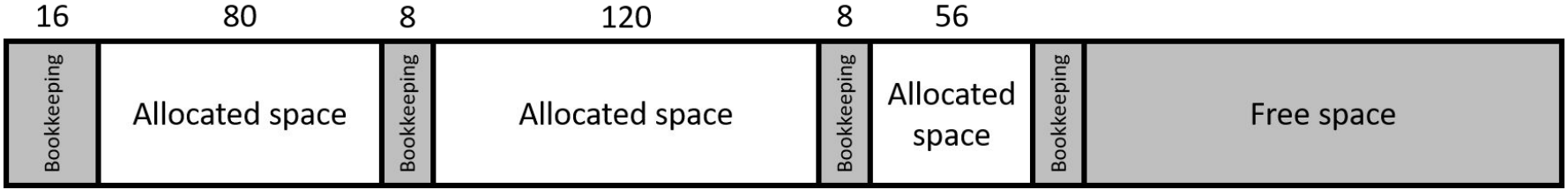
Exercise 1 – things you need to implement

- Functions:
  - `shmheap_memory_handle shmheap_create(const char *name, size_t len);`
  - `shmheap_memory_handle shmheap_connect(const char *name);`
  - `void shmheap_disconnect(shmheap_memory_handle mem);`
  - `void shmheap_destroy(const char *name, shmheap_memory_handle mem);`
  - `void *shmheap_alloc(shmheap_memory_handle mem, size_t sz);`
  - `shmheap_object_handle shmheap_ptr_to_handle(shmheap_memory_handle mem, void *ptr);`
  - `void *shmheap_handle_to_ptr(shmheap_memory_handle mem, shmheap_object_handle hdl);`

Exercise 2

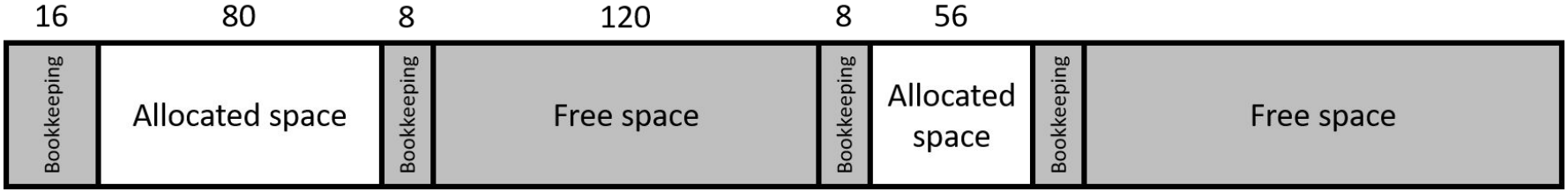
- First-fit algorithm for shmheap_alloc / shmheap_free
  - Objects should be 8-byte aligned
  - No need for synchronisation

Exercise 2



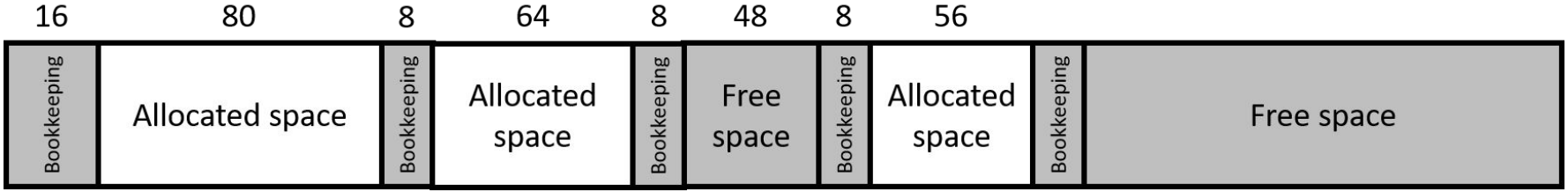
Next: Want to free 120-byte object...

Exercise 2



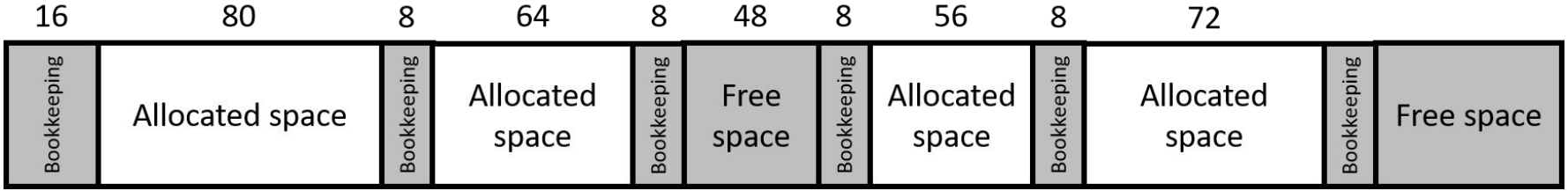
Next: Want to allocate new 64-byte object...

Exercise 2



Next: Want to allocate new 72-byte object...

Exercise 2



Next: Want to free the 56-byte object...

Exercise 2


16
80
8
64
8
112
8
72
Bookkeeping
Allocated space
Bookkeeping
Allocated space
Bookkeeping
Free space
Bookkeeping
Allocated space
Bookkeeping
Free space

Exercise 2

- Alignment requirements:
  - The objects you allocate must be 8-byte aligned, i.e. the memory address allocated must be a multiple of 8 bytes
    - If the caller requests for an allocation that is not a multiple of 8 bytes, you should round it up to the nearest multiple of 8 bytes
  - (Memory allocators typically provide some alignment requirement so that the caller can emit more efficient code)

Exercise 2 – things you need to implement

- Functions:
  - `void *shmheap_alloc(shmheap_memory_handle mem, size_t sz);`
  - `void shmheap_free(shmheap_memory_handle mem, void *ptr);`
  - `void *shmheap_underlying(shmheap_memory_handle mem);`

Exercise 3

```
// Main Process
// create new shared heap
mem = shmheap_create("test.bin", mem_length);
```

```
// Other Processes
// connect to existing shared heap
mem = shmheap_connect("test.bin");
```

In ex3, these calls may not be synchronised; you need to implement your own synchronisation mechanism

```
// allocate and free many objects from any threads...
// allocate an object
ptr = shmheap_alloc(mem, sz);
// free an object
shmheap_free(mem, ptr);
```

```
// disconnect from shared heap
shmheap_disconnect(mem);
```

```
// delete shared heap
shmheap_destroy("test.bin", mem);
```

Exercise 3

- Consider using a semaphore or mutex, but you should set it up to work with multiple processes

Exercise 4 (Bonus)

- In ex1-3, it is guaranteed that your shared heap will never run out of space
- For ex4, you need to consider this possibility, and implement a way to add more memory

Exercise 1 demo

- Run your code:
  - Start your ex0 program (with some arbitrary seed)
  - Ensure that the runner prints:
    - Child A received data successfully
    - Child B received data successfully
- Show me your code in mmf.h and mmf.c, explain it to me, and answer some questions about it