5118020-03 Operating Systems

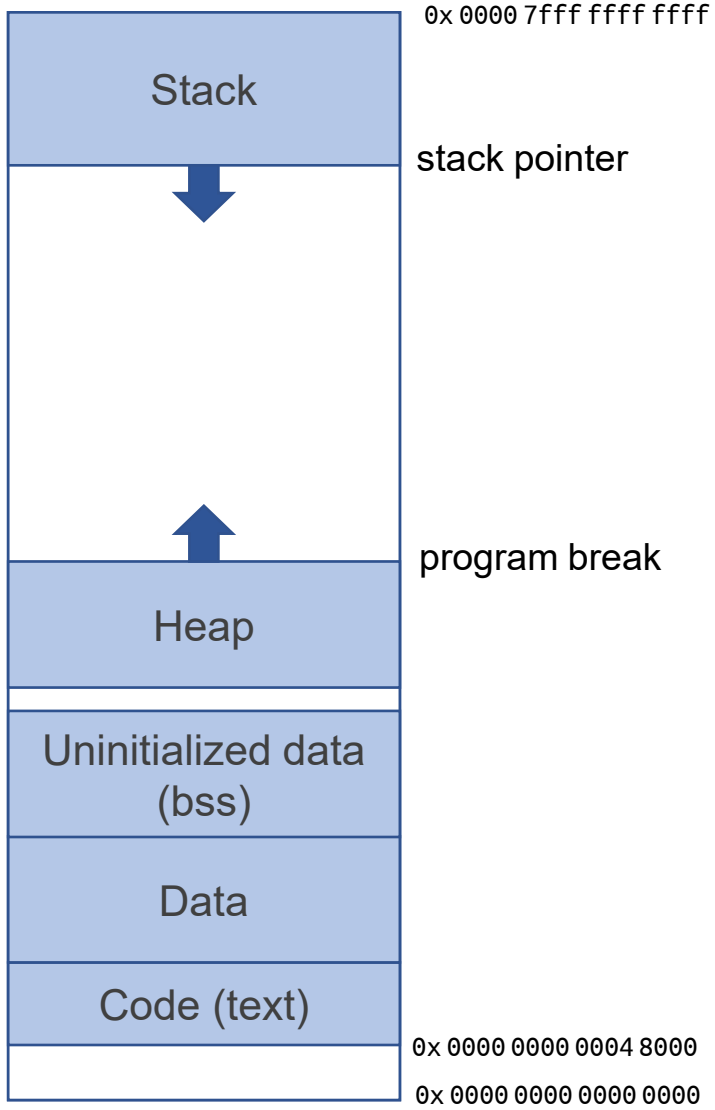
Homework 2. smalloc

Shin Hong

Overview

- Complete `smalloc.c`, an in-house heap memory allocator for C programs
- Point of study
  - memory management API: `mmap()`
- Timelines
  - Apr 30: First announcement & team arrangement
  - May 6-7: Help desks
  - May 13: Second test case releases
  - May 15, 9 PM: Artifact submission deadline (source code)
  - May 16, 9 PM: Presentation submission deadline (video record)

Background: Segmentation Layout (Linux, x86-64)



- `&etext` points to the first address past the end of the text segment
- `&edata` points to the first address past the end of initialized data segment
- `&end` points to the first address past the end of the uninitialized data segment

- `sbrk(0)` returns the first address past the end of the currently given heap segment
- `sbrk(s)` retains additional `s` bytes in heap and returns the starting address.
  - returns null when OS denies the request
- `getpagesize()` returns the number of bytes in a page

- c.f. https://en.wikipedia.org/wiki/X86-64#Virtual_address_space_details

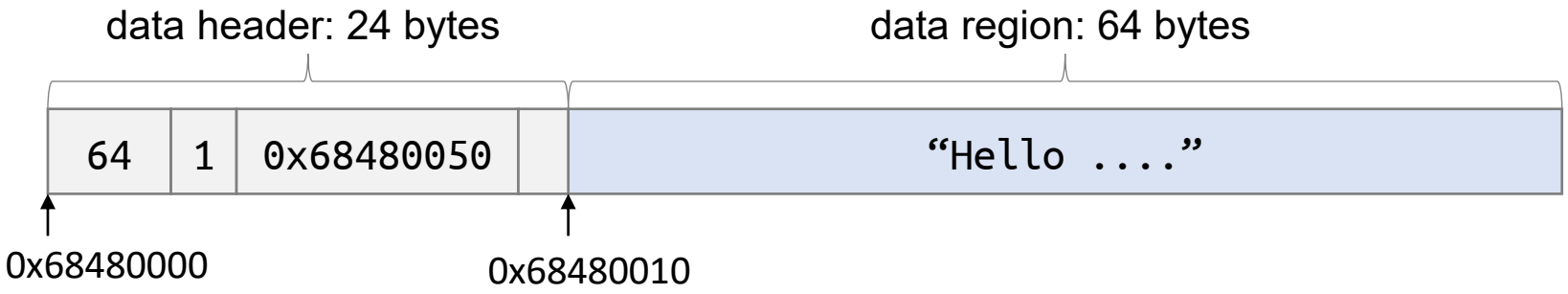
Background: `mmap()`

- `mmap()` adds one or multiple pages to the current process
  - the main use case is to set up a memory area for memory mapped I/O
  - `mmap()` with option `MAP_ANON` acquires a new pages while not connecting them with any file
  - the size of memory request must be a multiple of page size
    - `getpagesize()`: https://man7.org/linux/man-pages/man2/getpagesize.2.html

- `mmap()` manpage: https://man7.org/linux/man-pages/man2/mmap.2.html

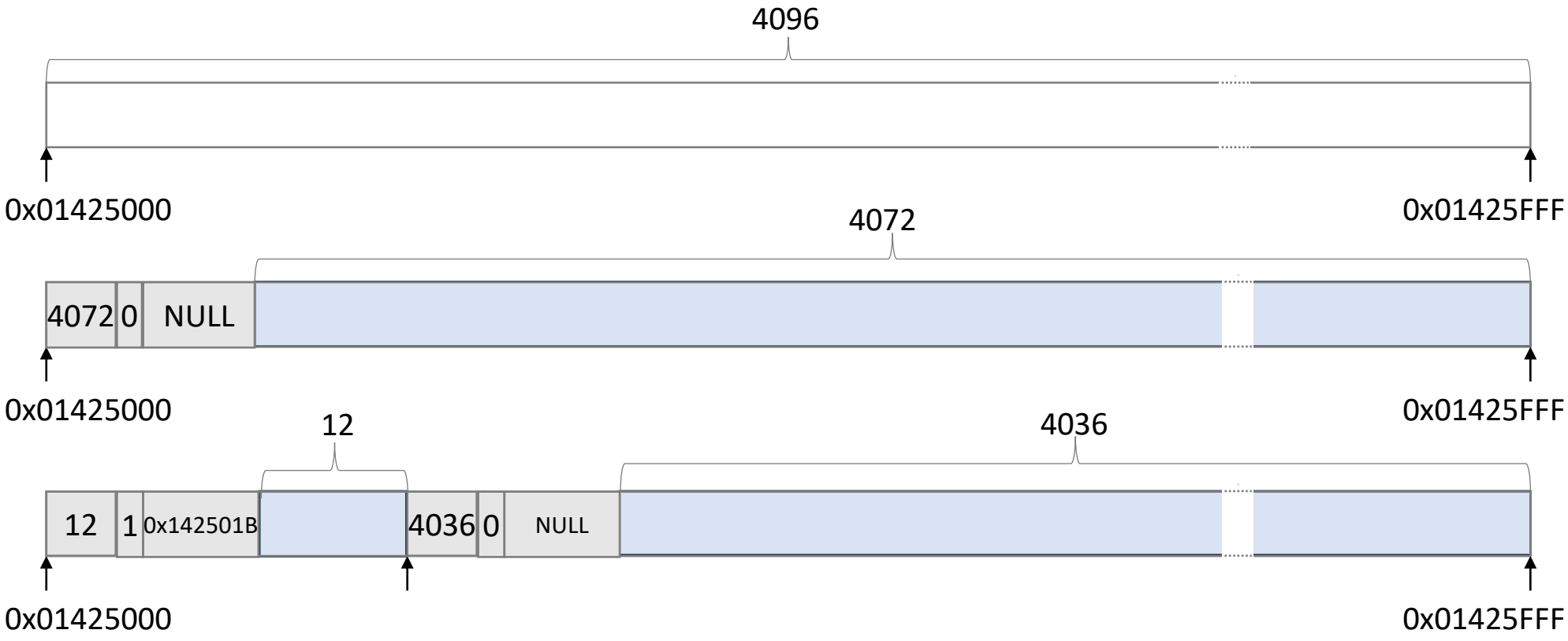
Memory Allocation Management (1/2)

- Use `mmap()` to acquire one or more pages as heap space
- Manage the information about the heap space as a linked list
  - a **data region** is a series of consecutive bytes to store user's data
  - a **data header** records the attributes of the corresponding data region, and it is located right before the data region in the memory address
    - the length of the data region
    - whether the data region is "used" or "unused"
      - used: allocated (given to the user) and not yet freed
      - unused: never used or freed
    - the pointer to the next data header (i.e., form a linked list)



Memory Allocation Management (2/2)

- Data regions and headers are placed inside the pages acquired by `mmap()`
  - embedded linked list
- Pages by `mmap()` are fully filled with data regions and their headers



Operations (1/2)

- `smalloc (size_t s)`
  - choose an unused data region whose size is greater than or equal to `s`
    - if no exists, use `mmap()` to add more pages
  - if the size of the chosen data region is greater than `s` + 24, split it into two
    - update the size of the chosen data region as s
    - add a new header for the remaining data region
  - return the first address of the data region

- `smalloc_mode (size_t s, smmode m)`
  - choose an unused data region with the specified mode `m`
    - mode: `bestfit, worstfit, firstfit`
  - the other behaviors are the same as `smalloc (size_t s)`

Operations (2/2)

- `srealloc (void * p, size_t s)`
  - find the header for the data region of pointer `p`
    - if it does not exist, abort the program execution (i.e., `abort()`)
  - resize the data region by s
    - change the data region if needed
    - split the data region if possible
- `sfree (void * p)`
  - find the header for the data region of pointer `p`
    - if it does not exist, abort the program execution (i.e., `abort()`)
  - mark the data region as unused at the header
- `smcoalesce ()`
  - if there exist two (or more) pairs of header and data region are unused and adjacent, merge them into one pair of header and data region

Submission

- All results must be submitted via LMS
  - Source code files
    - Submit all source code
    - You must provide a build script (e.g., bash script or Makefile) and its instruction document (e.g., README) if needed
  - Presentation
    - Submit the video record file; or, you can submit the URL to the presentation video on web
- No late submissions will be accepted

Video Presentation

- Take a 8-min video for reviewing the source code and testing the program
  - either in Korean or in English
  - every team member must take a part in presentation

- Your video must show how each API is implemented and used
  - devise scenarios for `smalloc_mode` and `smalloc_coalesce` and demonstrate their operations
  - suggest ideas to improve smalloc further

Notes

- Welcome your questions anytime on the Slack channel
- The team members must share the same responsibilities and take in charge of all tasks together
  - Peer evaluation follows immediately after the submission deadline
  - Inform me quickly if you keep fail to contact with your teammate
- It is strictly permitted to use auto-programming tools in any form