ECE30021/ITP30002 Operating System

Programming Assignment 4

Smalloc: Simple Dynamic Memory Allocation Library

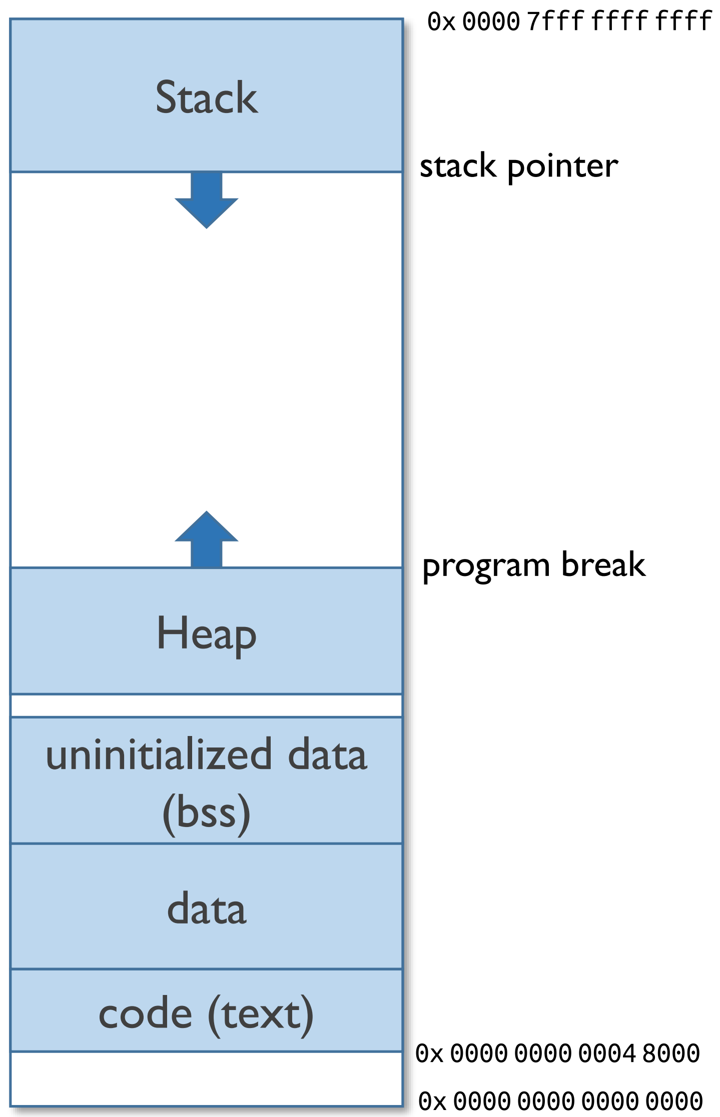
Overview

- Upgrade a dynamic memory allocation library *smalloc* version 1.0 to version 1.1 to version 1.2 as requested
  - Version 1.0 (given)
    - basic APIs
    - first-fit algorithm for allocating memory
  - Version 1.1
    - memory usage report
    - best fit algorithm for allocating memory
  - Version 1.2
    - fast allocation with unused container list
    - merge unused continuous containers at free

Notes

- PA 4 is an individual assignment (no partner)
  - You must not collaborate, discuss, or share your results with anyone
- The source code of `smalloc` version 1.0 can be found at https://github.com/hongshin/OperatingSystem/tree/sysprog/PA4
- It is recommended to use peace in doing this assignment because your programs will be built and tested for evaluation on peace
- Deliverables
  - Source code files
    - source code file for each version
    - two archive files must be submitted (e.g., ver1.1.tar, ver1.2.tar)
  - Write-up: up to 3 pages, in PDF

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

Smalloc Version 1.0 - APIs

- `void * smalloc(size_t s)`

  smllaoc allocates unused, continuous s bytes in the heap segment, and returns its starting address. Depending on the memory use, smllaoc may retain more memory to allocate s bytes. Or, this function returns null if it fails at allocating s bytes.

- `void sfree(void * p)`

  sfree reclaims the continuous memory region allocated by smllaoc, which starts from memory address p.

- `void print_sm_containers()`

  print_sm_containers displays the internal status of memory management by the smalloc library. It prints out to standard error the details of the sm_container linked list. Note that print_sm_containers must not be changed over version-ups.

Smalloc Version 1.0 – Data Structure

- The smalloc library manages the retained memory locations with a linked list of `sm_container` objects
  - A `sm_container` object holds an allocable continuous memory region and its metadata
  - A set of `sm_container` objects fill out the memory retained by the smalloc library
  - The first elements of the `sm_container` linked list is indicated by `sm_first`

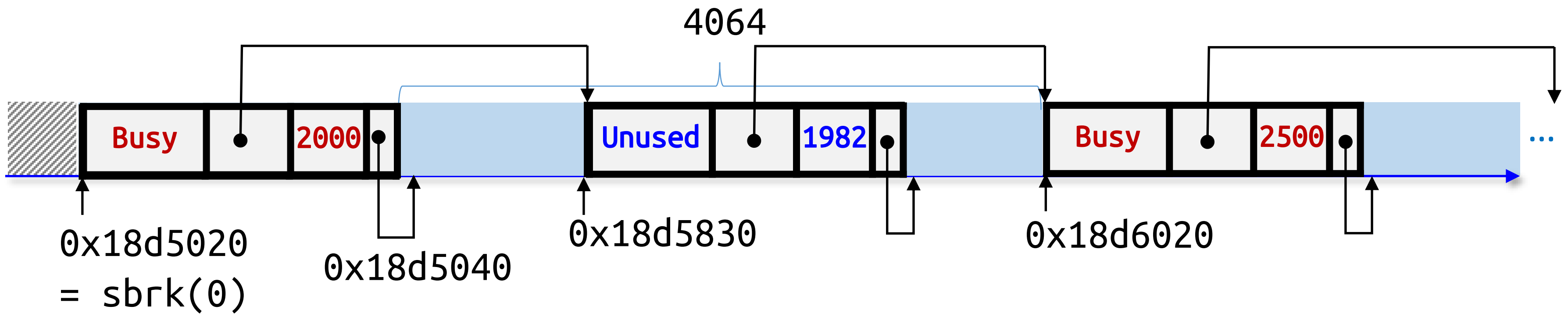
- struct `sm_container_t`
  - `sm_container_status status ; /* Busy or Unused */`
  - `sm_container_ptr next ; /* Null for the last element */`
  - `size_t dsize ; /* the size of 'data' in byte */`
  - `void * data ; /* memory region to allocate */`

- A `sm_containter_t` object takes 32 bytes (i.e., `sizeof(sm_container_t)` is 32)

Example: test1.c

- smalloc(2000) ;
  - sm_retain_more_memory(2000) ;
    - sbrk(4096) ;
  - sm_container_split(hole, 2000) ;

- smalloc(2500) ;



Version 1.1

- **Task 1.1**

  Add one more API, `print_sm_uses()`, according to the following description:

  > `void print_sm_uses()`
  >
  > `print_sm_uses` prints out the following information to standard error: (1) the amount of memory retained by `smalloc` so far, (2) the amount of memory allocated by `smalloc` at this moment, (3) the amount of memory retained by `smalloc` but not currently allocated.

- **Task 1.2**

  Modify `smalloc()` to find a best-fit unused container for allocating requested memory

- **Task 1.3**

  Construct a new test case test4.c on which the best-fit algorithm performs better than the first-fit algorithm (i.e., smalloc-1.0)

Version 1.2

- **Task 2-1**

  Revise `smalloc()` to maintain a linked list of unused container starting with `sm_unused_containers` and use this linked list to find a fitting unused container. Implement this feature by using the `next_unused` filed of `sm_container` and a global variable `sm_unused_containers`

- **Task 2-2**

  Revise `sfree()` to merge adjacent continuous unused containers if possible

- **Task 2-3**

  Give at least two ideas of improving smalloc beyond version 1.2, in your write-up

Evaluation

- Evaluation points

Technical soundness	70%
Presentation	15%
Discussion	15%

- Note
  - Your programs will be executed with testcases for evaluation
  - TAs will test the submitted files on the peace server

Submission

- Deadline: 11:59 PM, 15 June (Sat)
  - late submissions will be accepted by 11:59 PM, 17 June (48 hours) at penalty of 15% off of the total score

- Your submission must include the followings:
  - Write-up: up to 3 pages (either in single- or double-columns)
  - Two archives of the source code files

- How to submit
  - upload your files to a homework repository in Hisnet