

ITP30002 Operating System

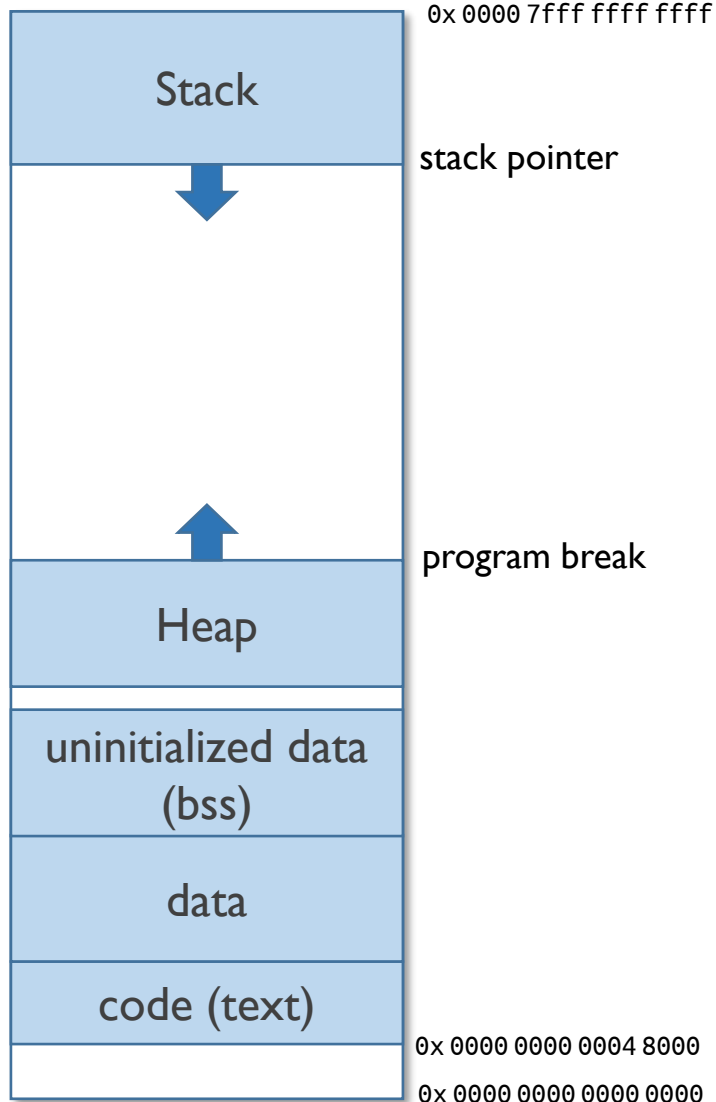
Homework 5

# **Smalloc: Simple Heap Memory Allocation Library**

# Overview

- Upgrade a given heap memory allocation library *smalloc* 1.0 to *smalloc* 2.0
  - Ver 1.0 <https://github.com/hongshin/OperatingSystem/tree/sysprog/Homework5>
    - basic APIs
    - first-fit algorithm for allocating memory slot
  - Ver 2.0
    - implement best-fit algorithm for allocating memory slot
    - merge unused continuous containers at free
    - implement `print_mem_uses()`
    - implement `srealloc()`
    - implement `sshrink()`
- Submission
  - Deadline: 11:59 PM, June 24 (Wed)
  - Deliverables
    - Source code files: source code file of *smalloc* 2.0
    - Write-up: 1 page (PDF) in the homework report template
  - Submit deliverables to Hisnet

# 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](https://en.wikipedia.org/wiki/X86-64#Virtual_address_space_details)

# Smalloc Version 1.0 - APIs

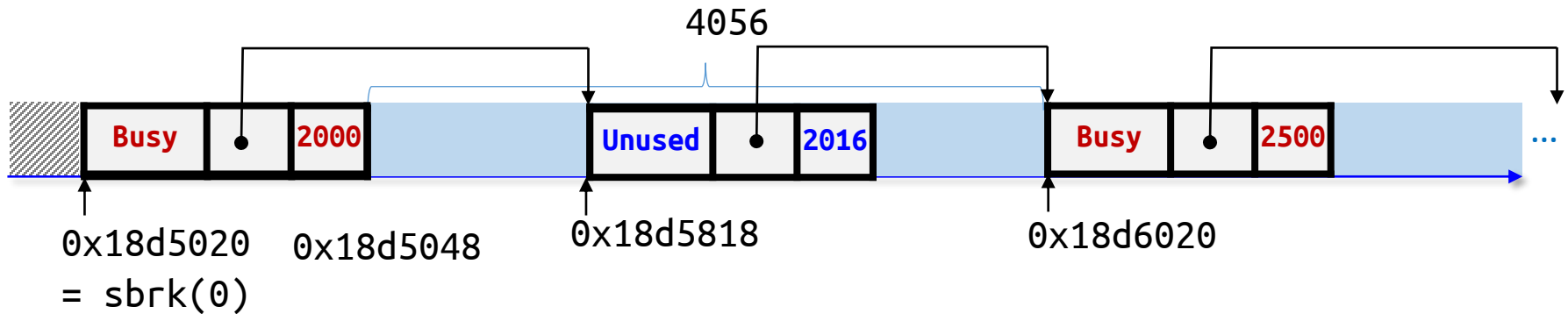
- `void * smalloc(size_t s)`  
smalloc allocates continuous s bytes in unused heap segment, and returns its starting address. Depending on memory use, smalloc may retain more heap memory to allocate s bytes. This function returns `null` if it fails at allocating s bytes.
- `void sfree(void * p)`  
sfree reclaims the memory region allocated by smalloc, 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 doubly linked list of `sm_container_t` objects
  - A `sm_container_t` object holds an allocable continuous memory region and its metadata
  - A list of `sm_container_t` objects fill out the memory retained by the smalloc library
- `struct sm_container_t`
  - `sm_container_status status ;`      `/* Busy or Unused */`
  - `sm_container_ptr next ;`      `/* sm_head at last element */`
  - `sm_container_ptr prev ;`      `/* sm_head at first element */`
  - `size_t dsize ;`      `/* the size of data in byte */`
- A `sm_container_t` object takes 40 bytes
  - i.e., `sizeof(sm_container_t)` is 40

# Example: test1.c

- `smalloc(2000) ;`
  - `retain_more_memory(2000) ;`
    - `sbrk(4096) ;`
  - `sm_container_split(hole, 2000) ;`
- `smalloc(2500) ;`



# Tasks for Version 2.0 (1/2)

- **Task 1**

Revise `salloc()` to select first best-fit unused container to allocate requested memory. In addition, construct a new test case `test4.c` on which the best-fit algorithm performs better than the first-fit algorithm (i.e., `salloc-1.0`)

- **Task 2**

Revise `sfree()` to merge adjacent unused containers if possible. Merge such adjacent unused containers as much as possible.

# Tasks for Version 2.0 (2/2)

- **Task 3**

Add a new API `print_mem_uses()`, according to the following description:

```
void print_mem_uses ()
```

`print_mem_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 4**

Add a new API `realloc(p, nsize)`, which resizes the memory allocated at pointer `p` as a new size `nsize`. Resize should be done without changing `p` if possible. Otherwise, it should give a new address after migrating the data to the new address.

- **Task 5**

Add a new API `sshrink()` which reduces the program break point (i.e., reduce allocated heap size) as much as possible at its execution time



# Evaluation

- Evaluation points

- Technical soundness      70%
- Presentation                15%
- Discussion                    15%
  - discuss possible improvements over smalloc 2.0

- Note

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