CS347M : LAB 3

LAZY ALLOCATION

ROHAN BANSAL, 170070058

PROBLEM STATEMENT

In Lazy Allocation, pages are allocated to a process only when it tries to write to these pages, rather than allocating them write away. Aim of this lab is to implement it in xv6. Along with allocation, reclamation should also be successfully implemented when freeing up the memory.

This is demonstrated using the system call sbrk(). Sbrk is a xv6 system call which dynamically changes the amount of memory associated with a data segment.

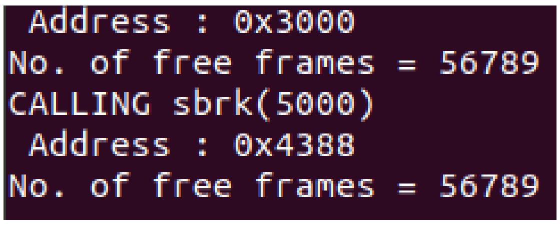
DESIGN DETAILS

When sbrk() is called, a new page is not allocated right away. We only increase the size of the data segment and let the page fault or trap to occur. Then we modify the code in trap.c so that the memory is manually allocated. Following steps sequentially explain the implementation.

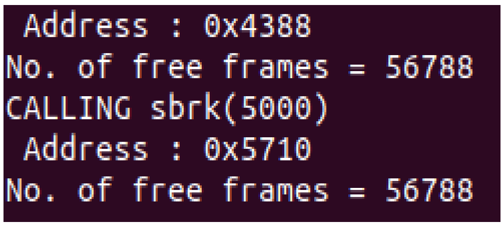
- **Changing the sbrk() function in sys_proc.c**
    - Increase the size of data segment
    - Delete growproc() which allocates pages right away
    - Insert deallocation function for reclamation i.e. calling sbrk(-).
- **Handling the fault occurred in trap.c**
    - Read the address at which page fault occurs from rcr2() register
    - Allocate memory using malloc()
    - Map the virtual address found from rcr2() register to the newly allocated memory using mappages()
- **CARRY OUT EXPERIMENTS**
    - Declare a variable which keeps track of the number of free pages in the memory. This variable is decremented after allocation and incremented after deallocation(modified in kalloc.c).
    - Call sbrk(+) and observe the number of free pages
    - Write on a memory segment and observe the number of free pages
    - Call sbrk(-) and observe the number of free pages.

EXPERIMENTS

- **CALLING Sbrk(+)**



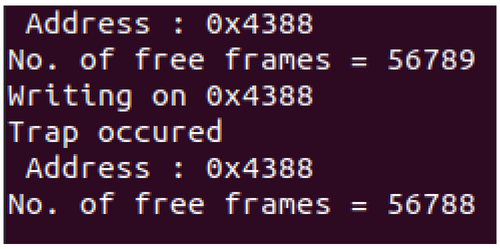
```
 Address : 0x3000
No. of free frames = 56789
CALLING sbrk(5000)
 Address : 0x4388
No. of free frames = 56789
```



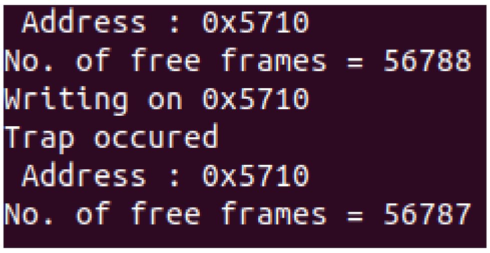
```
 Address : 0x4388
No. of free frames = 56788
CALLING sbrk(5000)
 Address : 0x5710
No. of free frames = 56788
```

After calling sbrk(), the number of free frames remains the same. The pages have not been allocated to the data segment right away.

- **WRITING TO MEMORY**



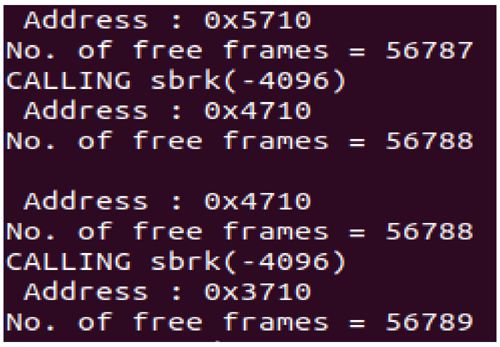
```
 Address : 0x4388
No. of free frames = 56789
Writing on 0x4388
Trap occured
 Address : 0x4388
No. of free frames = 56788
```



```
 Address : 0x5710
No. of free frames = 56788
Writing on 0x5710
Trap occured
 Address : 0x5710
No. of free frames = 56787
```

When writing to memory, we encounter the page fault. The page is allocated and the number of free frames decrease.

- **RECLAMATION**



```
 Address : 0x5710
No. of free frames = 56787
CALLING sbrk(-4096)
 Address : 0x4710
No. of free frames = 56788

 Address : 0x4710
No. of free frames = 56788
CALLING sbrk(-4096)
 Address : 0x3710
No. of free frames = 56789
```

sbrk(-4096) deallocates a full page because 1 page = 4096 bytes. The number of free pages increases after calling sbrk().

REFERENCES

- https://pdos.csail.mit.edu/6.828/2019/labs/lazy.html
- https://github.com/abhijeetkulkarni63/XV6_Lazy_Page_Allocation

SUBMISSION

Relevant files - kalloc.c, sysproc.c, lazy.c, usys.S, user.h, syscall.h, trap.c, syscall.c, Makefile, vm.c

Running the code - Run the terminal after cd into the submitted xv6 folder. Run 'lazy' command after make qemu-nox