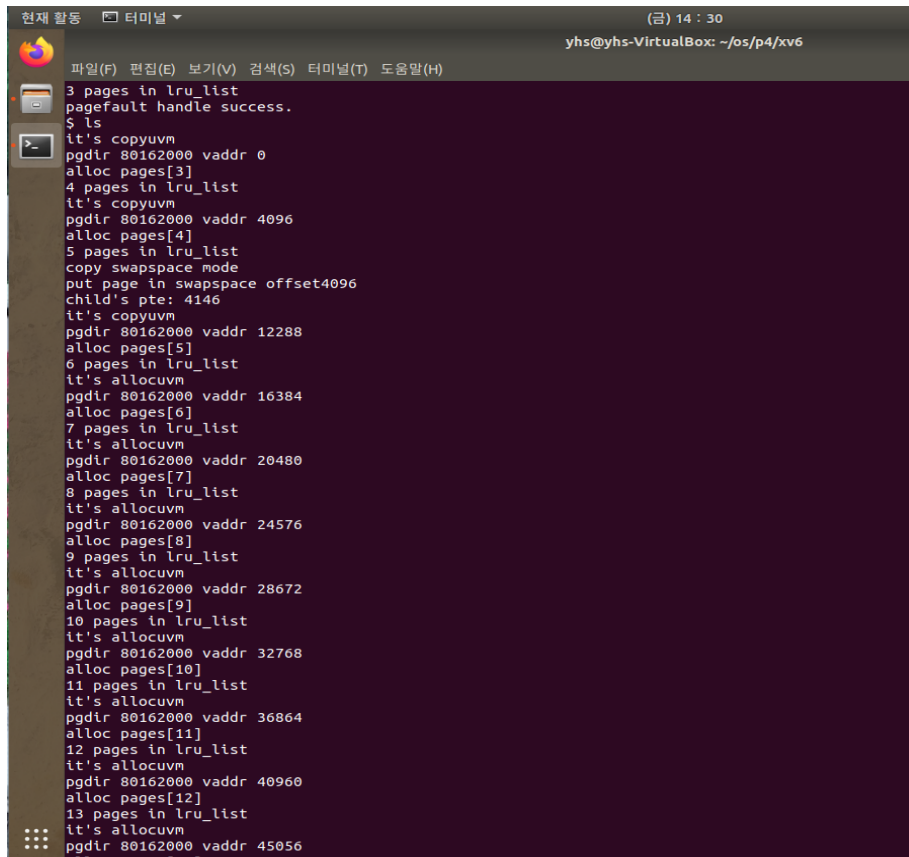


2016312761 여혁수 project4 : swap 레포트

I test my own code by entering user command **ls** & **sbrk()** system call, and monitoring lru list length, and swapread swapwrite operations.

<my result>



```
현재 활동  터미널 (금) 14 : 30
yhs@yhs-VirtualBox: ~/os/p4/xv6

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

3 pages in lru_list
pagefault handle success.
$ ls
it's copyvm
pgdir 80162000 vaddr 0
alloc pages[3]
4 pages in lru_list
it's copyvm
pgdir 80162000 vaddr 4096
alloc pages[4]
5 pages in lru_list
copy swapspace mode
put page in swapspace offset4096
child's pte: 4146
it's copyvm
pgdir 80162000 vaddr 12288
alloc pages[5]
6 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 16384
alloc pages[6]
7 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 20480
alloc pages[7]
8 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 24576
alloc pages[8]
9 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 28672
alloc pages[9]
10 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 32768
alloc pages[10]
11 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 36864
alloc pages[11]
12 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 40960
alloc pages[12]
13 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 45056
...
```

ls 커맨드를 실행한 직후 결과입니다. 일단 Copyvm을 4번 진행하는데 세 번째 copyvm에서 부모(sh)의 페이지가 swapspace에 있어서 child역시 swapspace에 복사를 합니다. 그 후 많은 allocvm이 실행되는 것을 볼 수 있습니다.

```
현재 활동  터미널 (금) 14 : 30
yhs@yhs-VirtualBox: ~/os/p4/xv6

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

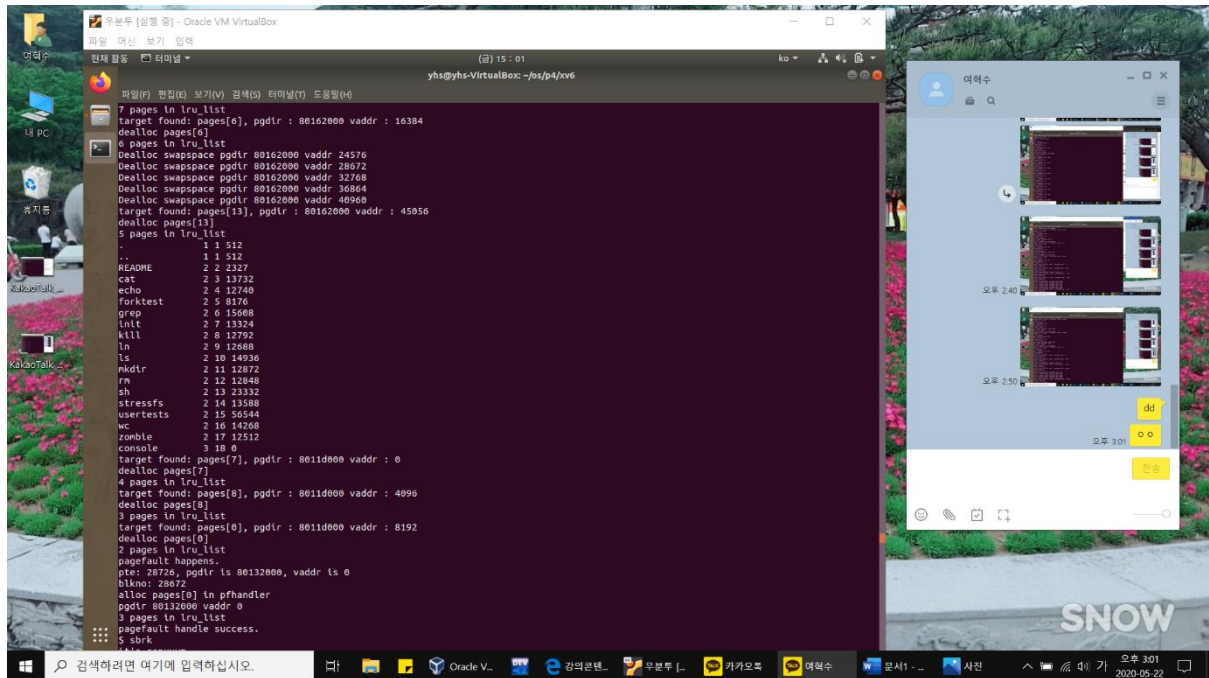
8 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 24576
alloc pages[8]
9 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 28672
alloc pages[9]
10 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 32768
alloc pages[10]
11 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 36864
alloc pages[11]
12 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 40960
alloc pages[12]
13 pages in lru_list
it's allocvm
pgdir 80162000 vaddr 45056
alloc pages[13]
14 pages in lru_list
reclaim happens.
check pgdir 80132000, vaddr 0
check pgdir 80132000, vaddr 4096
check pgdir 80132000, vaddr 12288
check pgdir 80162000, vaddr 0
check pgdir 80162000, vaddr 4096
check pgdir 80162000, vaddr 12288
check pgdir 80162000, vaddr 16384
check pgdir 80162000, vaddr 20480
pte value: 126007
write to swapspace in offset4096
swapout which of pgdir is 80162000, vaddr is 20480
13 pages in lru_list
reclaim happens.
check pgdir 80162000, vaddr 24576
pte value: 136007
write to swapspace in offset8192
```

Allocvm이 진행되다가 free_list가 줄 페이지가 없어 reclaim이 발생합니다. Lru_list에서 PTE_A가 0인 페이지를 탐색하는 것을 볼 수 있습니다. Victim이 정해지면 그 페이지를 swapspace로 옮깁니다.

```
현재 활동  터미널 (금) 14 : 50
yhs@yhs-VirtualBox: ~/os/p4/xv6

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
swapout which of pgdir is 80162000, vaddr is 36864
9 pages in lru_list
it's allocvm
pgdir 8011d000 vaddr 0
alloc pages[7]
10 pages in lru_list
reclaim happens.
check pgdir 80162000, vaddr 40960
pte value: 11f007
write to swapspace in offset24576
swapout which of pgdir is 80162000, vaddr is 40960
9 pages in lru_list
it's allocvm
pgdir 8011d000 vaddr 4096
alloc pages[8]
10 pages in lru_list
reclaim happens.
check pgdir 80162000, vaddr 45056
check pgdir 80132000, vaddr 0
pte value: 119007
write to swapspace in offset28672
swapout which of pgdir is 80132000, vaddr is 0
9 pages in lru_list
it's allocvm
pgdir 8011d000 vaddr 8192
alloc pages[0]
10 pages in lru_list
target found: pages[3], pgdir : 80162000 vaddr : 0
dealloc pages[3]
9 pages in lru_list
target found: pages[4], pgdir : 80162000 vaddr : 4096
dealloc pages[4]
8 pages in lru_list
Dealloc swapspace pgdir 80162000 vaddr 8192
target found: pages[5], pgdir : 80162000 vaddr : 12288
dealloc pages[5]
7 pages in lru_list
target found: pages[6], pgdir : 80162000 vaddr : 16384
dealloc pages[6]
6 pages in lru_list
Dealloc swapspace pgdir 80162000 vaddr 24576
Dealloc swapspace pgdir 80162000 vaddr 28672
Dealloc swapspace pgdir 80162000 vaddr 32768
Dealloc swapspace pgdir 80162000 vaddr 36864
Dealloc swapspace pgdir 80162000 vaddr 40960
target found: pages[13], pgdir : 80162000 vaddr : 45056
dealloc pages[13]
5 pages in lru_list
```

Free_list의 공간이 확보되었으면 allocvm을 한 페이지 실행합니다. 다시 reclaim이 일어나고 alloc page를 반복합니다. 그 후 dealloc이 되기 시작하는데 페이지 구조체에 있다면 그냥 빠지지만 swapspace에 있다는 것이 제가 설정한 swap bit flag를 통해 밝혀지면 swapspace offset에 맞는 비트맵 값을 1에서 clear해줍니다.



여러 시스템 콜의 변수 값(?)들이 나오고 dealloc이 마저 되는 것을 볼 수 있습니다. alloc했던 페이지를 접근하다가 페이지 폴트가 났다면 그 페이지는 swap space에 있는 것이므로 trap.c의 핸들러 함수로 가서 swapread를 하는 것을 볼 수 있습니다.

```
현재 활동 터미널 (금) 14 : 17
yhs@yhs-VirtualBox: ~/os/p4/xv6

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

target found: pages[10], pgdir : 80135000 vaddr : 32768
dealloc pages[10]
6 pages in lru_list
target found: pages[11], pgdir : 80135000 vaddr : 36864
dealloc pages[11]
5 pages in lru_list
target found: pages[12], pgdir : 80135000 vaddr : 40960
dealloc pages[12]
4 pages in lru_list
target found: pages[13], pgdir : 80135000 vaddr : 45056
dealloc pages[13]
3 pages in lru_list
$ sbrk
it's copyvm
pgdir 8011f000 vaddr 0
alloc pages[3]
4 pages in lru_list
it's copyvm
pgdir 8011f000 vaddr 4096
alloc pages[4]
5 pages in lru_list
copy swap space mode
put page in swap space offset 4096
child's pte: 4146
it's copyvm
pgdir 8011f000 vaddr 12288
alloc pages[5]
6 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 16384
alloc pages[6]
7 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 20480
alloc pages[7]
8 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 24576
alloc pages[8]
9 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 28672
alloc pages[9]
10 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 32768
alloc pages[10]
11 pages in lru_list
```

sbrk 실행 후 copyvm이 진행되는데 swap space에 있다면 swap space에 복사하는 것을 볼 수 있습니다.

```
현재 활동  터미널 (금) 14 : 24
yhs@yhs-VirtualBox: ~/os/p4/xv6

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
alloc pages[11]
12 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 40960
alloc pages[12]
13 pages in lru_list
it's allocvm
pgdir 8011f000 vaddr 45056
alloc pages[13]
14 pages in lru_list
exec: fail
exec sbrk failed
target found: pages[3], pgdir : 8011f000 vaddr : 0
dealloc pages[3]
13 pages in lru_list
target found: pages[4], pgdir : 8011f000 vaddr : 4096
dealloc pages[4]
12 pages in lru_list
target found: pages[5], pgdir : 8011f000 vaddr : 12288
dealloc pages[5]
11 pages in lru_list
target found: pages[6], pgdir : 8011f000 vaddr : 16384
dealloc pages[6]
10 pages in lru_list
target found: pages[7], pgdir : 8011f000 vaddr : 20480
dealloc pages[7]
9 pages in lru_list
target found: pages[8], pgdir : 8011f000 vaddr : 24576
dealloc pages[8]
8 pages in lru_list
target found: pages[9], pgdir : 8011f000 vaddr : 28672
dealloc pages[9]
7 pages in lru_list
target found: pages[10], pgdir : 8011f000 vaddr : 32768
dealloc pages[10]
6 pages in lru_list
target found: pages[11], pgdir : 8011f000 vaddr : 36864
dealloc pages[11]
5 pages in lru_list
target found: pages[12], pgdir : 8011f000 vaddr : 40960
dealloc pages[12]
4 pages in lru_list
target found: pages[13], pgdir : 8011f000 vaddr : 45056
dealloc pages[13]
3 pages in lru_list
$ QEMU: Terminated
yhs@yhs-VirtualBox:~/os/p4/xv6$ vim vm.c
yhs@yhs-VirtualBox:~/os/p4/xv6$
```

sbrk () 는 커맨드로 실행해보았는데 작동하지는 않지만 많은 alloc이 일어나고 후에 dealloc이 그만큼 일어나는 것을 볼 수 있습니다.

<Code Description>

1. main.c : 저는 main.c에서 kinit1의 범위를 1024*1300까지 잡았고, kinit2를 그 후부터 PHYSTOP까지로 잡았습니다. Memlayout.h에서 설정한 PHYSTOP의 값은 1024*1440입니다. 이 때 freelist는 약 15페이지 정도 있는 것 같습니다.

2. vm.c : Lock should be considered with shared resource for synchronization. 라는 피피티의 문구를 보고 lru_list를 여러 프로세스가 공유하니까 lock을 해줘라는 것으로 이해했습니다. 일단 initvm, allocvm, deallocvm, copyvm 이렇게 4개의 함수에서 lru_list를 관리하므로 각 함수에 acquire(&lru_lock), release(&lru_lock) 이런 방식으로 lock을 걸었습니다. Initvm, allocvm, copyvm 이 3개의 함수에서는 pages구조체에 페이지를 추가를 해줍니다. Copy의 경우 이미 있는 페이지를 복사하는 것이므로 복사하려는 페이지가 pages구조체에 없고 swapspace에 있는 경우가 생길 수

있는데, 그 때는 fs.c에 만든 swapcopy라는 함수를 통해 swapspace의 다른 빈 공간으로 복사합니다. 마지막 deallocvm함수에서는 페이지를 제거하는데, 마찬가지로 swapspace에 들어간 페이지를 제거하려고 하는 경우가 생길 수 있는데, 그 때 pte값에서 swapspace의 blkno offset을 가져와서 비트맵에 그 값에 해당하는 인덱스를 0으로 clear합니다. 그리고 pte안에 flag bit를 0으로 초기화합니다.

3. fs.c : 앞서 말한 swapcopy라는 함수를 설명하자면, 일단 부모의 페이지를 가져올 buf*와 자식의 공간을 가리키는 buf*를 선언합니다. bread()를 통해 부모의 데이터를 버퍼에 담은 다음, memmove로 자식의 버퍼 데이터 공간으로 옮깁니다. 그리고 부모의 pte를 자식의 pte로 복사하고, flag만 살리고 나머지는 초기화합니다. 그 다음 offset값을 pte에 저장합니다.

4. kalloc.c : kinit2에서 freerange해주는 범위 만큼의 값을 total_pages라는 변수에 저장합니다. 이 변수는 나중에 페이지 구조체에서 반복문을 통해 무언가를 찾을 때 반복문 조건으로 쓰이게 됩니다. kalloc함수가 실행되었는데 freelist에 빈 페이지가 없다면 reclaim()을 수행합니다. 페이지 구조체를 탐색하면서 PTE_A가 1이면 0으로 바꾸고, 0이면 그 페이지를 evict target으로 삼습니다. 비트맵에서 사용하지 않는 칸을 찾고 그 칸을 1로 set합니다. 그리고 그 인덱스를 swapwrite의 blkno 인자로 사용합니다. Swapwrite로 physical address를 swapspace에 쓴 다음에 Kfree함수로 physical address를 free합니다. Pte값은 flag비트만 살리고 초기화합니다. 그리고 blkno를 pte에 저장하고, PTE_P비트를 clear하고, PTE_A비트를 set합니다. swapout되었음을 알리는 저만의 PTE_SW비트도 set해줍니다. 그 후 구조체에서 해당 페이지를 제거하고 reclaim은 끝이 납니다.

5. trap.c : 페이지 폴트가 일어난 경우 pfhandler()라는 함수에 들어갑니다. 페이지 폴트를 일으킨 프로세스의 페이지 디렉토리를 받고, 정확히 페이지 폴트가 일어난 virtual address인 rcr2()를 받아 walkpgdir을 통해 해당 페이지의 pte값을 받아냅니다. 아마 그 pte값에는 physical address와 연결을 하고있는 어떤 값이 아닌 swapspace blkno의 offset을 가지고 있을 것입니다. 그 offset을 받아서 swapread를 호출해서 kalloc()을 통해 새로 받은 physical page에 내용을 넣어줍니다. 그러면 이제 이 physical page는 페이지 디렉토리나 virtual address와 연결이 된 것이므로 페이지 구조체에 정보를 추가합니다. 비트맵에 해당하는 인덱스 값도 1로 되어있을텐데 0으로 clear합니다.

Thank you for reading long and poor article.