Paging Note

Ruhul Azgor

February 2023

1 STEP-1:TRACK LIVE PAGES

Please read the whole note first. And think about why should this work.If you think this should work then go ahead. Firstly you need to need a structure. I thing Something like this would be fine.

```
pagetable_t p; // this is must needed
uint64 va; // also this
uint64 pa; // This also easy some task
uint64 serial; // for FIFO.
uint32 valid; // This is also needed
uint32 last_used; // not necessary
uint16 flag; // Synchronization purpose
```

Then Keep a gobal array of maxsize Of this structure . Initialized The array. You can do this step in any file but better on a new file.

```
page_t used_pages[MAXPHYPAGES];
uint16 live_page_count=0;
uint64 serial = 0;

void init_paging() {
  for (int i=0; i<MAXPHYPAGES; i++) {
    used_pages[i].valid = 0;
    used_pages[i].pa = 0;
    used_pages[i].flag = 1;
  }
  live_page_count = 0;
  swapinit();
}
```

Now you are ready to add Pages to this array. write a function to add page

```
  add_page(pagetable_t pagetable, uint64 va, uint64 pa) {
  if (live_page_count >= MAXPHYPAGES) {
    page_swap_fifo();
  }

  int i;
  for (i=0; i<MAXPHYPAGES; i++) {
    if (used_pages[i].valid  == 0) {
        used_pages[i].valid = 1;
        used_pages[i].flag = 1;
        used_pages[i].pa = pa;
        used_pages[i].va = va;
        used_pages[i].serial = serial++;
        used_pages[i].p  = pagetable;
        break;
    }
  }
```

```
  live_page_count++;
```

Now you should add page from uvmalloc.

```
    if((xperm&PTE_X) == 0 ){
    add_page(pagetable, a, (uint64)mem);
    }
```

Add a remove function Now .. You should write two function one for PA AND One for PAGETABLE. This is a example for PA remove.

```
    void remove_page(uint64 pa) {
  for (int i=0; i<live_page_count; i++) {
    if (used_pages[i].valid && used_pages[i].pa == pa) {
      used_pages[i].valid = 0;
      used_pages[i].pa = 0;
      live_page_count--;
      return;
    }
  }
}
```

Now go to kfree func. Call the above function when you are freeing a page. And call pagetable version from freewalk.

2 SWAPING OUT

Sooo Now you need to swap out a page to the disk.Check out You have been already called it. Find a index of the array which page you want to swapout. Serial will help you. Swap out the page and keep that pointer to the pageable entry. Think about it. Now check the code below

```
    used_pages[min_index].flag = 0;
    struct swap *s = swapalloc();
    swapout(s, (char*)used_pages[min_index].pa);
    used_pages[min_index].valid = 0;
    used_pages[min_index].flag = 1;
    live_page_count--;
    pte_t *pte = walk(used_pages[min_index].p, used_pages[min_index].va, 0);
    uint16 flags = PTE_FLAGS(*pte);
    flags &= ~PTE_V;
    flags |= PTE_PG;
    kfree((void*)PGROUNDDOWN(used_pages[min_index].pa));
    uint64 new_pte = (((uint64)s)<< 10) | flags;
    *pte = new_pte;
```

Dont forget to do something crazy. Go to the wait func. and change it into the following code.This will save you from sched lock (For happy path). There can be other reason from which you can get it.

```
        release(&pp->lock);
           release(&wait_lock);
           freeproc(pp);
```

3 SWAPIN

Dont forget the swapped out pages. You will get page fault for accesing swaped out pages. Scause will be 15(sw pagefault) and 13(lw page fault). So how to swap in a swapped out page? Think about it. Now check out the below check section.

```
else if(((*pte) & PTE_PG)){
    struct swap *s= (struct swap*)((*pte)>>10);
    char *mem=kalloc();
    swapin(mem,s);
    uint64 flags=PTE_FLAGS(*pte);
    flags=flags&(~PTE_PG);
    flags=flags|PTE_V;
    flags&=~PTE_COW;
    flags|=PTE_W;
    *pte=0;
    sfence_vma();
    mappages(p->pagetable,va,PGSIZE,(uint64)mem, flags);
    add_page(p->pagetable,va,(uint64)mem);
    if(s->ref_count>1){
      s->ref_count--;
    }
    else{
      printf("_Freeing swapped out page\n");
     swapfree(s);
    }
```

Wait. What is s->ref_count? This is for copy on write perpose. A swap out page can be in parents and child's pagetable. So don't call swapfree(). When ref count is greater than 1.Also think about other combination. ONLY COW PAGE NOT PG, BOTH COW AND PG PAGE.

4 FORK

Life would be easier if there won't be any fork in life. Sad. :(. So if you are wanted to implement with cow, keep your old code but when you are copying the swaped out page, increase reference count(Add a field in swap stract). Also **Don't use mappages() function for maping. Update by using pointer.** Also mark every page as Cow page. From now you will not swap out any cow page to disk.
Now if you are targeting to implement without cow. This is easy ig.Copy everything from memory which is on the memory. You can also read swapped out page using swapin. but don't call swap free here. Parent funtion will still need this. also call add_page() function here.

5 I THINK YOU ARE FORGETING SOMETHING

When you are freeing child's memory you need to free those swaped out pages. So go to uvmunmap() and add something like this

```
if((*pte & PTE_PG) != 0)
    {
      struct swap *s = (struct swap*)((*pte)>>10);
      if(s->ref_count >1){
        s->ref_count--;
      }
      else{
        swapfree(s);
      }
```

6 DON'T GIVE UP.

Upto this point everything should work without file read write. You may wonder what else you need to change.Go to COPYOUT AND COPYIN FUNCTION. AND ADD CODE FOR SWAPPED OUT PAGE.I think this will help you.If not then sorry the time you have spent on this.