<u>Project #4 Design</u>

Ayushi Rajendra Kumar

Suhas Keerthi Raju

Physical memory layout

```
--------------------------------------------0x4400000

          SWAP AREA                              Size= 32MB

--------------------------------------------0X2400000

          FFS- VIRTUAL HEAP                      Size=16MB

------------------------------------------- 0X1400000

          PAGE TABLE                                       Size=4MB

------------------------------------------- 0X1000000

          XINU STACK

          XINU HOLE

          XINU HEAP

%%%%%%%%%%%%%%%%

          XINU BSS                               Size=16MB

%%%%%%%%%%%%%%%%

          XINU DATA

%%%%%%%%%%%%%%%%

          XINU TEXT

---------------------------------------- 0X000000
```

Initialization of page directories and page tables

Each process will have its own page directory. All system calls will have a common shared page directory. (Page Directory - PD).

Page directories are initialized in the create (PD0) and vcreate(all other page directories) function calls

PD0 - corresponds to page directory shared by the system calls, having 1st few entries pointing to the global page tables

PD8- page directory of process 8, having 1st few entries pointing to the global page tables. On vcreate, more page table entries get created

PD9,PD10,....

Page table space in memory

PD6

PT5 -(vcreate in Process 5)

PD5(1st few entries pointing to PT1,PT2,...)

PT4

PT3

PT2

PT1

PD0

Total number of pages in xinu default = (16*1024*1024)/4096 = 4096 pages, since each page table entry is of 4 bytes, 1024 pages entry can be stored in 1 page table, therefore, we will have 4 page tables (4096/1024) for default xinu processes. PD0 will have 1st 4 entries pointing the these 4 page tables. All page directories will have the entries pointing to these 4 entries.

When a vcreate process requests for a vmalloc, new page table is created to map these entries.

A check is made to limit the page table area, so that the page tables and directory mappings do not go beyond the intended area. Before filling the page table entry, a check is made if the address is within the virtual space and is not in the swap area.

System Initialization

Paging is enabled right after sysinit in initialize.c. It is enabled by writing on CR0 register and setting the PE flag(0th) to enable protection before turning on paging, and then setting the 31st bit of CR0 to enable paging. Note: Before enabling paging, in the page that has same physical and logical address, it is needed to initialize the default page directory and page tables for all default xinu processes

Process Creation

In process creation-create(), to support paging, CR3 register needs to be set to the base address of the page directory corresponding to default xinu processes.

For processes created using vcreate(), for correct mapping from logical address to physical address, vcreate function should initialize the page directory for the new process and CR3 register(PDBR) to point to newly created page directory. The 1st few entries of the page directory should be the page table entries of default xinu pages tables. When a process calls vmalloc, a new entry to the page directory should be made pointing to the virtual heap space allocated

Process Termination

In the kill function, release the memory/ clear the page directory entry by marking it as available for use.

Context Switch

To support paging, at context switch, the PDBR /CR3 register must be updated with the respective process's page directory base information from the process control block.

Heap allocation, deallocation and access

When the heap allocation is called, only the addresses of the memory must be saved, so that when **memory is accessed,** only the amount of memory accessed is allocated memory. On access to memory, page fault is generated on the 1st usage. The page fault handler should then take care of allocating the requested memory. At deallocation of heap, all the used memory should be free'd and the saved addresses must be unsaved.

Vmalloc returns the virtual address and not the physical address. Free list check for virtual memory to be done. Initialize the valid bit in PTE to indicate that it is a valid address, present bit should be 0 as there is no physical memory allocated.

Page Fault Handler Design

1. In which circumstances will the hardware raise a page fault?

1. If the address is not valid or the process is trying to access memory that is not allocated.
2. While accessing the memory for the 1st time i.e. disk bit is set to 0 and the address is valid
3. If the address is valid and the disk bit is set to 1, ie the address is not present in the disk and has been evicted.

2. What operations should be performed by the page fault handler depending on the circumstances under which it is invoked?

1. If the address is not valid or the process is trying to access memory that is not allocated. ---> Segmentation fault
2. While accessing the memory for the 1st time i.e. disk bit is set to 0 and the address is valid ---> fetch the page from physical memory
3. If the address is valid and the disk bit is set to 1, ie the address is not present in the disk and has been evicted. - fetch the page from disk

Swapping Design [only students taking course at graduate level]

Operations:

In PTE- assign a reference bit, in order to keep a track of least recently used page. Every time the clock ticks- in clockhandler.c, the reference bit of all the pages in the page table is checked. If the reference bit is 1, that page cannot be evicted as it was recently used. Set this bit to 0.

If the reference bit is 0, check for dirty bit before making decision to evict. If dirty bit is not set, evict the page and set the reference bit to 1. If all dirty bit is set to 1, then evict one of them. The 1st preference is given to page whose dirty bit is not set as it is lest costly operation.

Data structure:

In order to make the swapping decision, each page in the disk should have the following fields;

1. Virtual page number
2. Physical frame number
3. Pid of the process

These 3 fields provide a link between the virtual heap and the swap area.