Programming Assignment 6: Paging
CO21BTECH11004

Part-1

Implementation: -

- For only allocating read-only code/date, change the size of memory allocated by allocuvm in exec.c : -

```
read_only = ph.filesz;
dynamic_allocated = ph.memsz - ph.filesz;

// if((sz = allocuvm(pgdir, sz, ph.vaddr + ph.memsz)) == 0)
if ((sz = allocuvm(pgdir, sz, ph.vaddr + read_only)) == 0)
  goto bad;
sz += dynamic_allocated;
```

- Here ph.filesz refers to the size of the segment in the executable file (read-only) while ph.memsz the size of the program segment in memory (includes read-only and dynamic data).
- We need to modify trap.c to handle the trap, as now the trap function exits with an error. Include the case for T_PGFLT (page fault trap) while calling the page fault handler function: -
    - Prints the virtual address of the page that had page fault (rc2()).
    - Checks if the faulting address is a valid address in the virtual memory range of the process.
    - If it is invalid, print the same error message coming before the page fault trap handler.
    - If it is valid, first zero out a physical page (memset), then assign and map it to the page directory/page table.

Observation: -

- Whenever there is access to a page containing the array data and if the page is not in the page table, a page fault occurs, the page is brought, and page table entry increases on printing the page table.
- When the array size increases, the number of page faults increases because more pages must be brought from memory as more pages are required to store array data.
- Initially, a one-page table entry with virtual address 0 is in the page table because read-only data is allocated.

Part-2

Implementation: -

- To prevent the making of copies of parent's pages for a child, copyuvm is modified in vm.c.
    - Parent pages are marked read-only.
    - Both parent and child should point to the same physical page.
    - TLB flush
- Modify trap.c to handle page fault. Include T_PGFLT case and call page fault handler function: -
    - Get the fault virtual address using the rc2() command.
    - If the address is in the illegal range, throw error.
    - Get reference count, handle two cases: - if the reference count is one and greater than one.
    - Allocate a new page with a copy of contents with a set write bit.
    - Change the reference count.
- To keep track of processes that map a page into their virtual address page, the reference count is used, which is added in the struct kmem in and all the related functions where kmem is modified in kalloc.c. Now to maintain the reference count, we need to define the following functions to calculate (calcRefrenceCount), increase (setRefrence), and decrease (removeRefrence).
- Write myCOW.c to test the implementation of Copy-on-Write.

Observations: -

- In the child process, before updating the array, pgtPrint, entries have the write bit set to 0, but after updating all array elements, pgtPrint, entries have the write bit set to 1 because the new page with copy is added with set Write bit in child's process page table.
- After updating, the physical address of entries also changes, this is because the new page is allocated with the copy in the child's process page table.
- If the whole array is not updated, the entries corresponding to the updated page table have a change in the write bit and physical address because, for only modified pages, a new page is allocated with a copy in the child's process page table.

Submission: - Two different tar files are submitted, one for part-1 and the other for Part-2. For part-1, run mydemandPage on the terminal. For part-2, run myCOW on the terminal.