CSE 451: Section 6 Handout
10/31/2019

Page Faults and COW (MOOOOOOOO)

Page Faults:

A trap 14 defines a page fault, this means that the memory address was a not a valid page for the client to manipulate.

Can the kernel cause a page fault? If so, how?

For a user process, how will you know if the page fault was caused by attempting to access the stack region of its virtual address space?
Hint: trap.c has a variable `addr` which is the address the user process tried to access.

The trapframe error code can be read with `myproc()->tf->err.`
What will the error code be if the page fault was from touching the stack region of memory?

Can the kernel cause a page fault that was meant for stack growth?

CSE 451: Section 6 Handout
10/31/2019

Page Faults and COW (MOOOOOOOO)

copy-on-write fork:

What is the purpose of copy-on-write fork?

What do the fields of a page (`struct vpi`) need to be after a copy-on-write fork?

What needs to be changed in the `core_map_entry` to support copy-on-write fork?

What will the error code be if the page fault occurred from touching a copy-on-write page?

Can the kernel cause a copy-on-write page fault?

What can happen if a copy-on-write fork is not synchronized?

When is copy-on-write less efficient than a deep copy fork?