

Swappin' Pages

Swappin' Pages:

What are the page fault trap frame error bits, `tf->err`, for a page that isn't in main memory. How do we distinguish between a page that is on disk and a page that has never been allocated?

When a copy-on-write trap occurs on page *A*, why can we guarantee that page *A* is present and exists in physical memory?

If you call `vspaceinvalidate` for a `vspace` that just set the present bit for one `vpage_info` struct, what's the maximum number of free pages required?

Note: With a 4 level page table, 1 new mapping can result in 3 `kalloc` calls.

How should copy-on-write fork handle a page that is in the swap region of disk (the page is not in physical memory) during the copy process?

What are the `dst vpage_info`'s fields set to?

How can we signify that the reference to that page has increased?

When handling a copy-on-write page fault, we call `kalloc` for the new page.

How do we ensure we do not evict the page we plan to copy during a copy-on-write?