Lab 4 Design Doc: On-demand paging

Overview

The goal of this lab is to implement on-demand allocation of pages of physical memory. This is a way to increase space efficiency, as we won't allocate pages if the process is not going to use them.

Major Parts

Grow user stack on-demand: Whenever a process needs additional stack pages (reads or writes to user stack/page fault), allocate physical memory at run-time up to a limit of 10 pages.

Create a user-level heap: A process can explicitly request for more virtual addresses to be allocated in the heap by calling sbrk.

In-depth Analysis and Implementation

Grow user stack on-demand

kernel/pgfault.c:handle_page_fault

- Check if the address is within a valid memory region
    - as_find_memregion
- Check if proper permissions
    - If present is not null, return.
    - If write, and memory is read only (MEMPERM_UR or MEMPERM_R), return.
- Allocate a physical page.
    - Translate the physical address to a kernel virtual address (kmap_p2v).
        - Memset the allocated physical page to 0.
- Map it to the process's page table and return.
    - Call vpmap_map.

kernel/proc.c:stack_setup

- Allocates one page of physical memory (pmem_alloc)
- Adds a one-page memory region (segment) to the process's address space (as_map_memregion with third arg as USTACK_PAGES*pg_size).

- Adds a page table entry mapping the virtual address of the stack's page to the address of the newly allocated physical page (vpmap_map, with argument n = USTACK_PAGES).

Create a user-level heap

kernel/syscall.c:sys_sbrk

- Get increment from agrs.
- Call memregion_extend, passing old_bound pointer.
- Return old_bound or error code returned by memregion_extend.

kernel/mm/vm.c:memregion_extend

- Set old_bound to be region_end.
- Update region->end.
- If region->end is before (<) region->start, return ERR_VM_INVALID
- If the extended region overlaps with other regions in the address space return ERR_VM_BOUND.

Risk Analysis

Unanswered Questions

- Nope we're good :)

Staging of Work

First, we will update stack_setup, then we will implement the page fault handler. Finally, we will make the changes to support dynamic heap growth - sbrk and memregion_extend.

Time Estimation

- ❖ Grow user stack on-demand (10 hours)
- ❖ Create a user-level heap (10 hours)
- ❖ Edge cases and error handling (10 hours)