Operating Systems Project 01

Hogge , Louis, s192814
louis.hogge@student.uliege.be

Romoli, Raphael, s182115
raphael.romoli@student.uliege.be

1 Implementation

To help us in our implementation we used several functions in addition to the principal one:

1.1 Functions and structures

1. The `get_process_info` function is designed to find and return a pointer to a `process_info` structure in the global hash table, given a process name and its length.(used in several functions)
2. `is_tracked_process` function is designed to check if a given process (identified by its name and length) is being tracked in the global hash table of process information. It returns true if the process is tracked and false otherwise.
3. `find_task_by_comm` function is designed to search for a task (a process) with the given process name. It iterates through all tasks in the system and returns a pointer to the `task_struct` of the target process if it's found, or NULL if the process is not found. (used in the `update_process_vma`)
4. The `update_process_vma` function updates the VMA information for a given process.
5. The `update_process_cow_page_faults` function updates the statistics of Copy-On-Write (COW) page faults for a given process. It increments the total count of COW page faults and then iterates through the VMAs of the process. If the provided address lies within a VMA, it increments the fault count for that VMA and exits the loop.
6. The struct `process_info` defines a data structure to store information about a process. It contains the process name, name length, a `pf_stat` structure for storing page fault statistics, and a `hlist_node` for organizing the data in a hash table.

1.2 `pf_set_param`

Initialize the hash global table if not already done.
Copy the process name from user space to kernel space.
Check if the process is already being tracked using `get_process_info` function.
If the process is not being tracked, allocate memory for a new `process_info` struct and populate it.
Add the new `process_info` struct to the global hash table.

1.3 pf_get_info

Copy the process name from user space to kernel space.
Find the process_info struct using get_process_info function.
Copy the pf_stat structure from the process_info struct in kernel space to the user space memory pointed to by pf.

1.4 pf_cleanup

Iterate through the hash table, performing the following actions for each process_info struct: Remove the process_info struct from the hash table.
Free the memory allocated for the process_info struct.
Set the cleaned_up flag to true, indicating that at least one process_info struct has been cleaned up.
Reset the hash_table_initialized flag to false, allowing the hash table to be reinitialized upon the next use.

1.5 Others

- We used a new file to write all of our syscalls, this file is located in the kernel folder. We thus added this file in the corresponding Makefile.
- We added our 3 systems calls to /arch/x86/entry/syscalls/syscall_32.tbl.
- We added a file in /include/linux called pf_syscalls.h to store our prototypes.
- We modified the file /arch/x86/mm/fault.c to use our function update_process_vma and update_process_cow_page_faults with respect to those flags : (FAULT_FLAG_WRITE, FAULT_FLAG_USER,VM_HUGEPAGE)

1.6 Memory management

When a process requires additional memory, it sends a request to the kernel for allocation. In response, the kernel creates a new Virtual Memory Area (VMA) that satisfies the necessary size and permission criteria. Subsequently, the kernel updates the process's page table, which is responsible for translating virtual addresses employed by processes into corresponding physical memory addresses. VMAs represent contiguous regions within a process's virtual memory address space, with each VMA possessing a distinct start address, end address, and a set of attributes. This mechanism enables efficient memory management and allocation for processes.