INFO0940 - Operating systems

Project report

Step number 4

Romain Compagnie Olivier Moitroux
Group 19
Third year engineering bachelor

29 mars 2018

This report explains the different steps we have been through to implement the `sys_pfstat` system call.

1 Adding data to the task structure

In order to store the statistics about virtual memory page faults for one or multiple given process, we added two fields to the kernel process data structure, `task_struct`, found in */include/linux/sched.h*. This data structure contains various information about the processes. A `pfstat_status` field was added to keep track of the process pfstat mode, and the `pfstat` structure was added to store the process statistics. The `pfstat` structure was defined in the file earlier.

These fields are initialized in the `INIT_TASK` macro found in */include/linux/init_task.h*, which is used to initialize the first task table. They are also reset in the function `copy_process` found in */kernel/fork.c*, which is called when the function `do_fork` is executed to create a new child process. This was added to make sure that a child does not inherit his parent process statistics and pfstat mode.

2 Implementing the system call in the kernel

The system call is implemented at the end of the file */kernel/sys.c*. We made this choice because this file contains the implementation of system calls allowing the user to get information about the system, such as `gethostname` and `sysinfo`.

The implementation of the system call is straightforward, we print the entering system call message, we check the validity of the pfstat pointer passed as argument, we try to find the task corresponding to the given pid, we set the process to pfstat mode if it was not already in this state, we recover the pfstat data from the process and then we reset the pfstat data of the process to 0.

The system call being defined for a 32-bit x86 architecture, an entry was added in the system call table *arch/x86/entry/syscalls/syscall_32.tbl*.

An `asmlinkage` entry was added in the file *include/linux/syscalls.h*, to allow the kernel to look for the arguments on the kernel stack.

3 Updating the corresponding statistics

Upon a page fault, the CPU generates a page fault exception and calls the page fault handler `do_page_fault`, found in the architecture-dependant file */arch/x86/mm/fault.c*. This function consequently calls __do_page_fault located in the same file, which is responsible for handling the fault.

The fields of the pfstat structure of the process having triggered the fault are updated in various locations from this entry point, given the process is in pfstat mode. The task structure of this process can be accessed through the macro `current`, which returns the pointer to the current process in execution before the kernel was called.

- stack_low

  A page fault can be triggered to increase the stack. This is handled by the function `expand_stack`. The counter is therefore incremented in this function if no error occurs.

- transparent_hugepage_fault

  The `handle_mm_fault` function is called from the page fault handler. This function then calls `__handle_mm_fault`, where the counter is incremented after the missing huge page is created, or if the page exists but a fault happened anyway.

- anonymous_fault

  In case of a normal anonymous page fault, the function `do_anonymous_page` is called, and the counter is incremented in this function.

- file_fault

  A file-backed page fault is handled by the function `filemap_fault`, found in the file *mm/filemap.c*. The counter is incremented in this function.

- swapped_back
  A fault can occur if a page needs to be put back online by a read-from swap. This is handled by the function `do_swap_page`, which found in the file *mm/memory.c*. This function is called by the function `handle_pte_fault`. The counter is incremented just before the call to `do_swap_page`.
- copy_on_write
  A fault which backs a copy-on-write is handled in the function `do_cow_fault`, found in the file *mm/memory.c*. The counter is incremented in this function.
- fault_alloced_page
  The function `wp_page_copy`, found in the file */mm/memory.c*, handles the case of a page which we actually need to copy to a new page, which needs to be allocated. The counter is incremented in this function.

4 Shell

The shell has been updated with a new built-in program which calls the `sys_pfstat` system call. It has been updated to support background execution.