The Queue Module Project

Giulio Zingrillo

March 2023

The project is made up of a kernel module, written in C, and a Makefile necessary to compile it. The code of the module is divided in three files: queue.c, implementing open, close, read and write functions, and dev.c, implementing the init and exit of the module.
When it is inserted, the kernel module registers a device file called queue_device and initializes a queue (FIFO principle), implemented with the kernel list, a mutex and a waitqueue. Morever, two parameters are set by the user: period (default 1000 ms) and max_elems (default 5). Period represents the time the task sleeps while reading the device file after printing each element of the list; max_elems sets the maximum number of elements the queue can contain. A counter, representing the elements that currently are in the queue, is globally allocated.
The write function adds an element to the queue with the string provided. If the string is too long (more than 30 chars) an error is returned. If the queue is full, the task is added to a waitqueue (cooperative synchronization), waiting for the number of elements in the queue to decrease. Signals are handled and mutex are used.
The read function, instead, prints each element of the queue. After each element, the task is put in an interruptible state, the scheduler is invoked and signals are handled. Moreover, the first element of the list is deleted (and this allows to remove a task from the waitqueue). If the queue is empty, nothing is done. The read function returns the value of the first element of the queue, and end of file if the queue was empty. Also this function is protected with mutexes.
Finally, when the module is removed it deletes every element of the list and the dynamic counter of the elements. The misc device is deregistered and the number of elements of the former queue is printed.