Kernel module loading

- sudo insmod /path/of/module
- Internally calls init_module() syscall that invokes syscall implementation sys_init_module(),
- It does following things:
    - Check if user has enough permission (CAP_SYS_MODULE).
    - Invokes load_module() kernel API. It does following:
        - Loads module binary into kernel space in a temp memory.
        - Create struct module for this module and initialize its members.
        - Check version magic number of module against current kernel.
        - Replace relocatable address in module binary by the absolute address in temp memory.
        - Module params are copied from user space to kernel space.
        - Module state is changed to MODULE_STATE_COMING.
        - Actual location in kernel memory is allocated to copy code & data.
        - Unresolved symbols are resolved from kernel symbol table.
        - Pointer to struct module is returns.
    - Returned module struct is added into kernel module linked list.
    - Module initialization function is called i.e. module->init function pointer.
    - Release module initialization data.
    - Module state is changed to MODULE_STATE_LIVE

Kernel module unloading

- sudo rmmod module_name
    - Internally calls delete_module() syscall that invokes syscall implementation sys_delete_module().
    - It does following things:
        - Check if current user has enough permissions (CAP_SYS_MODULE).
        - Check if module is in use (using member source_list).
        - Check if module is already loaded or not i.e. state is MODULE_STATE_LIVE.
        - Update state of module to MODULE_STATE_GOING and call cleanup function i.e. module->exit func pointer.
        - Invoke kernel API free_module() is called. It does following:
            - Remove all kernel module references.
            - Perform arch specific cleanup (if any).
            - Unloads module from kernel i.e. release its memory.
            - Frees memory used by the module parameters.