CS13B1040
Akshita Mittel

Operating System: Assignment 4

Inserting Kernel Modules

Kernel version: 3.19.0

Attachments: A directory consisting of the Kernel module code and the Makefile

How the module was constructed.

The module was made in the userspace and not in the kernel space. Creating the module in kernelspace will allocate permanent memory to the module.
The program for the module was written in link.c:
Instead of including the entire type.h directory, only a subset was included; namely list.h.
The list.h consists of definitions for all the kernel linked list modules.

First a structure was created to hold all the data required. A predefined list_head structure was included in the module. The list_head consists of a pointer to the previous node on the list and the following node. Note, that these nodes don't hold the data themselves, they are a part of different structure which hold the data. This is an ingenious method of describing a linked-list which is native to kernel doubly-circular linked-list.

There are two functions which will be called in macros at the bottom. The modules are as follows:

module_init(link_init):

This module is invoked whenever the command “insmod” is called, i.e. when the module is inserted and loaded. This macro calls the function link_init.
It function link_init is initialized by initializing the head using INIT_LIST_HEAD to null.
The 5 links are made iteratively, by initializing the head and then using the in-built macro list_add_tail; adding the current node to the head of the specified list. In this case the link is added to the beginning of the list. The space for each link is made using kmalloc. Which is equivalent to the non-kernel version malloc.

The macro list_for_each_entry iterates over the birthday list, where the ptr points to the current structure that the link is in. We then print the contents of each structure through the pointer ptr.

module_exit(link_exit):

This module is invoked whenever “rmmod” is called, i.e. when the module is to be deleted. The macro calls the function link_exit.
The function link_exit basically uses the macro list_for_each_entry_safe. This stores the next entry in a list, allowing us to delete the current link without additional steps and confusion. list_del removes the entry from the list, whereas kfree (the kernel equivalent of free) is used to free the kernel memory.
A message is printed to the kernel log, stating that the module has been unloaded.