Exercise 3:UNIX system programming

Dr. Nezer Zaidenberg

MTA – 2014-2015B term.

Bonus date: March 1st 2016

Due date: August 1th 2016

# Instructions

1. Please grok the Linux kernel-programming guide before trying to do this ex.

2. Please pay extra attention to section 2,4,7 of the linux programming guide.

3. Please send the compiling project in 1 ZIP or gzip file containing only.

The zip file name should be your “your ID number” i.e. a filename should be

0123456789.tgz or 9876543210.zip

Do not submit RAR or 7z etc.

4. I have ~~short attention span and low level reading comprehension~~ ADHD. This exercise is **complex to run**. Please state ~~using works of one syllable~~ clearly what needs to be done to compile and execute your work and in what order i.e.

1. First compile the project using make all
2. Insert the module to the kernel using “modprobe ./module/mymodule.ko”
3. Grep the major number from the kernel log using …
4. Start the daemon using ./daemon/daemon
5. Start the user application using ./usr/app
6. PLEASE explain all the steps as accurately as you can. Bare in mind I have no commonsense what so ever.
7. I run your code under Ubuntu 15.4 I will accept code running on other Linux flavors. However, if your code doesn’t run on my machine we may meet at very odd places and hours to demonstrate your code is running on your machine… please make an effort to make sure it does compile under 15.4!

5 Please ensure your project compiles with make all. Note that multiple makefiles and recursive make files may be required.

6. Please ensure make clean deletes your binary and any cruft.

7. Please add any additional instructions and comments to README.TXT file.

8. All dates refer to 23:59:59 Israel standard time on the date.

9. Submission by or before the bonus date will result in 10% grade BONUS

10. Submission by the due date is allowed.

11. Submission after the due date will be penalized by 25% for late submission

(automatic) + 10% per day after the second day. (ie. Late submission by 1 minute will be penalized by 25%. Late submission by 4 days will result by 55% penalty) – PLEASE you are responsible adults. Manage your time and meet the deadline.

12. You will receive an “acknowledged mail” for your exercise within 6 hours.   
If for any reason you did not receive mail please send another email as I may have missed your first email.

13. You need to implement this exercise on your own. You are allowed idea exchange ideas with your fellow students but not to exchange code. You are allowed to use web resources though. If you use any code you found on the Internet SPECIFY that in the README.TXT file.   
If two students submit identical work a disciplinary action will be taken. (Even if both copied the same source from the net)

14. It is your responsibility to ensure your work compiles and run on the first submission. 2nd submission (and 3rd and 4th etc.) will be penalized by 10% penalty per “re-submission.” exercises that do not compile will receive 0%.

15. Please follow KNF, 1TBS or BSD code style. Exceptionally good code will get 10% BONUS. Other coding standards (Allman, GNU etc.) are allowed as long as they are consistent but will not result in bonus. Kludge will receive 10% penalty. Students using Hungarian notation will ~~be shot, twice.~~  Will not be eligible for code quality bonus.

16. The code should be self-documented. You may comment critical parts of the code. Excessive commenting does not consist of good code and will be penalized.

# Learning objectives

1. Grok char devices

2. Grok kernel programming

3. Grok netlink sockets (bonus)

# Cover Story

You have two options for your work

1. Code a Virtual tape library using char \*\*.

Bonus

1. Code a Mouse/pointing device driver

Option 1 basic design

User space process (“backup software”) communicates with a kernel driver using char dev

Option 2 basic design

* A User process that is using the device (e.g. tape library or mouse)
* A kernel driver
* A back end daemon that emulate the physical device and communicates using netlink sockets.

# Option 1 - Tape drive

# Exercise 1 – canapé (10%)

Start with chapter 7 in the Linux kernel module programming guide and implement a char device with IOCTL

Use 10 strings of 1000 bytes (char [1000][10]) as your tapes.

Support the read/write/lseek commands to read from the current tapes.

Support IOCTLs to change tape and format tape,

# Exercise 2 – User code (10%)

Write any C backup and restore program that will backup some file to tape, restore some file from tapes, display current tapes status etc.

This is what you will use for unit testing and I will use to test your work.

Be creative

# Option 2 – Mouse – self learning is required!

This option can act as replacement or make up for students missing any past exercise.

Learn the generic netlink socket interface on your own.

1. Code an android/iphone application to act as a limited mouse-pointing device. (Sufficient is supporting just L/R mouse buttons) and gyro
2. The mouse mobile app should communicate with the daemon process (for example over TCP/IP)
3. The daemon process communicates with kernel using netlink it is your responsibility to learn the netlink socket intraface.
4. A second user process is reading the mouse movements from kernel.

The user process for unit testing communicate with the kernel using char device you must demonstrate read/write and ioctl (for example you can use ioctls to set sensitivity or to switch mouse buttons)

1. You don’t have to support GDM interface.