ELEC 377 – Operating Systems

Lab/Tutorial

Labs - General Procedure

- First Lab Period of a Given Lab Assignment
  1 Hand in PreLab documentation
  2.Start your linux virtual machine
  3.Log in using the account *student (password: student)*
  4 Check out initial code from the subversion server
  5 Edit and compile your code
  6 Edit test data files and add to subversion control
  7 submit everything to the svn server
  8 Test your program
  - (repeat from step 6 as necessary)
  9 Add output files to subversion control, submit
  10 log out of linux (exit, logoff)
  11 Log in as root and shut down linux machine
  12 Exit VMWare/VSphere player

Labs - Details

- Login to the Windows 7 machine using your NETID
- Open up Vsphere, on the desktop
- Under the list of existing virtual machines, chose ELEC 377
- For today only(1st time),
  Take a Virtual Machine snapshot
- Then click “Launch VM Console” button
  - A red screen may come up and at the bottom the prompt will say boot
  - Click on the window and press return.
  - To get your mouse back at any time use CNTRL+ALT

Labs - Details

- You can use Xwindows (command “startx”) which provides a graphical environment

- You can stay in console mode, and use alt-fn (function key) for alternate consoles.

- Editors :
  elvis (a vi clone), joe, jed
  xedit

Saving your Files

- Most of your labs will involve Linux Kernel Modules
  – these operate in kernel mode, with no protection
  – a bug in your code may crash your virtual machine and might corrupt your virtual hard drive
    - ◊ Save your code on the subversion server before you execute

- Subversion is a version control system. It keeps track of changes to each file.

- Your Repository Location:
  https://elec377.appsci.queensu.ca/svn/<netid>

Managing files

- svn --username *user* co *repositoryURL*
  - creates a directory in the current directory
  - copies contents of your repository (initially empty).
- svn add *filename*
  - file must be in the directory created above (or a subdirectory)
  - adds the file to svn management
  - does ***not*** send the file to the server
- svn add *directory*
  - directory must be in the directory created in the first command, adds all contents of the directory
  - can only be done once, new files added later must use “svn add filename”

Managing files

- svn ci -m “message”
  - must be run somewhere inside the directory from the first step
  - send the contents of the current directory (and all subdirectories) to the server
  - if in a subdirectory, does **not** send the contents of the parent directory to the server
  - messages is used to identify the version
- svn update
  - must be run somewhere inside the directory from the first checkout
  - downloads new versions from the server

Saving Files

- You can also check out the repository at home
  - svn clients for windows (tortoise svn), macintosh
- write documentation
  - add doc files and check in before assignment deadline

Look at Lab0 now

- Already checked out
- cd [netid]

- vi lab0mod.c

- three parts
  - my_read (main part of kernel)
  - init_module -- used to initialize data structures
  - cleanup_module -- used to remove pointers

Compiling Modules

- Sometimes a Makefile
  makefiles contain dependencies. Example:

```
all: lab0mod.o lab0user

lab0mod.o: lab0mod.c
	cc -c -Wall -DMODULE -D__KERNEL__ lab0mod.c

lab0user: lab0user.c
	cc -o lab0user lab0user.c
```

Inserting a Module

- Login as root
  - if in console mode, alt-fn to change consoles
  - use “su -” if running X windows
  - cd /home/student/[netid]/lab0
- use the following command to load the module
  insmod lab0mod.o
- you will get the message:
  Warning: loading lab0mod.o will taint the kernel:
  no license
  …
- After testing, unload the module with:
  rmmod lab0mod (note: no.o)

User Programs

- Some labs will also have a user level program
  - kernel: not all libraries are available!!
    - lab0mod: ctime() function formats a date
    - module returns a number
  - user level program that formats for ease of user understanding

% lab4user
The system was started September 17th at 3:40.

Testing

- Some labs are interactive
  - ◊ Interact with your kernel module

  – use script

  % script lab1_out1.txt

  Script started, output file is lab1_out1.txt

  % run your tests here

  ...

  % exit

  Script done, output file is lab1_out1.txt

- Some labs just have output

% cat /proc/lab1 > lab1_out1.txt

Testing

- Testing is your proof that the program works
  - ◊ TA's will not be running your code

  – industry standard in many domains, give customer test suites to provide confidence it works.

- Show input and output

  – Sometimes input is a kernel data structure

  – system commands such as ps, vmstat, iostat, and files in /proc give you some idea of what the state of the kernel is

  – think about this *before* you come to the lab

  – man pages available on the internet

Documentation

- Documentation are to be presented in text files or pdf. Do *not* commit word documents to the repository.