Debugging Linux Applications – LABS

Lab 1 – Linking Errors

1. open /opt/labs/lab1 workspace
   demo1, demo2 are shared libraries
   uselibs is a client project

2. try to run uselib, why it fails?
3. Fix the problem using 2 ways:
   - change the code to include the path to the library
   - define the LD_LIBRARY_PATH environment variable

4. change uselib code to load demo2 library
5. use nm to find why it fails
6. fix the problem by:
   - changing the client code to load the correct symbol name
   - changing the library code to export the symbols for C clients

Lab 2 – Debugging using GDB/Eclipse

1. Open /opt/labs/lab2 workspace
   The program should get 2 numbers and divide the first by the second, it should print the result and if the result is greater than 25,000 it should print “big” , otherwise print “small”

2. use GDB to load and debug the program. Correct the problems and check again

3. use Eclipse to debug the code

4. change the project settings to cross compile and run the code for ARM architecture
   - change the compiler and linker to arm-linux-gcc
   - copy the binary file to /opt/armsystem/output/rootfs
   - in directory: /opt/armsystem/output , run the script ./run_eclipse to load the emulator
   - use gdbserver and gdb/eclipse to remote debug the application

Lab 3 – Debug stack errors

open /opt/labs/lab3 workspace
The program gets a file name (in the command line) and print it to the screen line by line
find and fix all bugs and security problems

Lab 4 – Debug memory errors

part 1

1. open /opt/labs/lab4 workspace
2. find and fix all the bugs

part 2

1. write 2 functions to replace malloc and free with a safe version

   for each allocation ask 8 bytes more:

Size	The user allocated data	Magic

   Size – the size of the user part
   Magic – 0x1234
   in malloc you should write the user size and the magic number and return the address of the starting user part
   in free check the size and the magic number and print message on error

2. test your functions on the first part

Lab 5 – Debug multiple processes

1. open /opt/labs/lab5 workspace
   the demo application create 3 child processes and wait for any child exit.
2. Run the application and see the results
3. debug one of the processes by attaching a debugger
   - run gdb
   - attach [pid]
4. run 2 more debuggers to debug the other child-processes
5. quit all tasks

part 2: debug from start

1. run the application with the debugger (gdb ./lab5ex)
2. add a breakpoints in main and in writeDataTask.c line 71
3. continue to run.
4. Start again but now before step 3 run:
   set follow-fork-mode child
5. try to set a breakpoint on tcpServerTask