EEE3096S - Tutorial 1
2023

Linux Basics, Virtual Machines and GDB Debugging

1 Overview

This tutorial sets out to familiarise you with some Linux basics and provide you with an introduction to Virtual Machines (VMs) and GDB debugging. Linux is used in many spheres of embedded systems development for both commercial companies and academic research. VMs offer tremendous versatility and the practice of testing platforms and solutions on VMs before investing in them is becoming ever more widespread.

2 Pre-tut Tasks

- Download and install VirtualBox. Windows users may first need to enable hardware virtualizatiom from within the BIOS setup. Instructions and more information on how to do that can be found here.
- Download Ubuntu 22.04 iso image.
- Get a basic understanding of git. For details regarding git and it's usage, read through http://wiki.ee.uct.ac.za/Git

3 Pre-tut Requirements

This section covers what you will need to know before starting the practical.

- Have VirtualBox installed on your laptop/PC.
- Have an Ubuntu 22.04 iso on your local drive.
- Have a basic understanding of Git.

4 Hardware Required

- Laptop/PC with VirtualBox installed

5 Outcomes of this Tutorial

You will learn about the following topics:

- Linux and Bash Basics
- Git
- GDB Debugging

6 Deliverables

At the end of this tutorial, you must:

- Submit the answers to the questions in a single PDF to Amathuba. Be sure to include all student numbers in the document name! A link to your shared repository with your partner should also be included.

7 Walkthrough

This tutorial consists of a programming task. Before doing that, we need to set up VirtualBox.

7.1 Setup

Using software like VirtualBox, we can split one physical computer into multiple virtual servers by allocating resources from the ”host” to the guest. This means we can run virtual machines with any OS without reinstalling the OS that is installed and running on the physical machine. We can thus install and use a virtual Linux OS on a Windows or Mac host machine. In this tutorial We will be using VirtualBox to create a virtual Ubuntu 20.04 machine.

With VirtualBox open, do the following:

1. Click on New to create a new VM.
2. Enter a name for your new VM. Change the Type to Linux and Version to Ubuntu (64-bit). Click on Next.
3. Allocate RAM to the VM. The default is 2GB. Increase this to 4GB (or more) if your machine can handle it. Click Next.
4. Select ”Create a virtual hard disk now” and click Create.
   4.1 Select VDI (VirtualBox Disk Image) and click Next.
   4.2 Select Fixed size and click Next.
   4.3 Enter the size of the virtual hard disk using the textbox or slider. I’d recommend allocating 10-15 GB. Click Create.
5. On the main VirtualBox window, click on your newly created VM and click on start.
6. In the ”Select start-up disk” window that opens select the Ubuntu iso you downloaded and then click on Start.

7. Using the directional keys, navigate to ”Try or install Ubuntu” and hit the Enter key on your keyboard.
8. Click on ”Install Ubuntu”.
9. Select ”Minimal installation” and ensure that ”Download updates while installing Ubuntu” is checked. Click Continue.
10. Ensure that ”Erase disk and install Ubuntu is selected and then click on ”Install Now” and then Continue in the pop up window.
11. Complete the Ubuntu installation and click on ”Restart Now” once its done. You may need to close the VM and restart it if it gets stuck at the ”Remove installation media” loading screen.
12. Open a new terminal by pressing CTRL+ALT+T and run the following:

    12.1 $ sudo apt update

    12.2 $ sudo apt upgrade

    12.3 $ sudo apt install git

7.2 Programming Task

For this task we'll be working with C/C++. In order to do so, you need to have a suitable compiler. Ubuntu and most linux Distros (including Raspbian on the Raspberry Pi) have this by default. On Windows however, you'd need MinGW (see this page on the EE Wiki). Luckily, we just installed an Ubuntu VM so this won't be necessary!

The solution of this task should pushed to a shared repository with your partner and a link to the repository should be included in the report.
By the end of this task, we expect you to:

- Have a basic understanding of Git
- Be comfortable working in a terminal/command prompt
- Understand real-time debugging
- Be familiar with at least 1 development and debugging tool-chain

Simple GDB Walkthrough

To learn about gdb, let's create a very simple[1] calculator.

1. Follow the instructions at http://wiki.ee.uct.ac.za/Git to configure Git on your Ubuntu VM.
2. Make sure you are in the home directory:
   $cd ~
3. Create a folder as follows (replace STDNUM001-STDNUM002 with your student numbers) and navigate into it:
   $mkdir -p 3096-Pracs-STDNUM001-STDNUM002/Prac1
   $cd 3096-Pracs-STDNUM001-STDNUM002/Prac1
4. Create a file called "main.c" on the command line:

   $touch main.c

5. Open the file in the nano editor and place the following code:

   $nano main.c

```
1  # include <stdio.h>
2  int main(){
3              int a, b, sum;
4
5              printf("Enter a value for a: ");
6              scanf("%d", &a);
7
8              printf("Enter a value for b: ");
9              scanf("%d", &a);
10
11             sum = a + b;
12
13             printf("The sum of a and b is %d \n.",sum);
14 }
```

7. Compile it with the debugging flag:$
   gcc -g main.c
8. Run it!
   $ ./a.out

[1] Incredibly simple.

If you try add two numbers - you'll see something really odd happen! Let's use gdb to figure it out

9. Open it in gdb:
   $ gdb a.out
10. We need to add breakpoints. Let's add them at points where we need to validate data:
    10.1 Line 7 - once we have a value for a
    break 7
    10.2 Line 10 - once we have a value for for b
    break 10
    10.3 Line 12 - once we've added them together
    break 12
11. Run the application inside gdb!
    run
    11.1 You'll be prompted to enter in a value for a. Enter in something simple, such as 3
    11.2 Once you hit the enter key, gdb will step in. Let's print the value of a to ensure it is 3
    print a
    11.3 If we're happy that it's the value we entered, we can continue execution by entering in c and pressing enter.
    11.4 We're prompted for a value for b. Let's do another simple number, 5
    (Please note Windows may not prompt you for the next number, but you can enter it anyway)
    11.5 gdb now steps in again. Let's validate that the value for b is indeed 5.
    print b
    It's not! Let's double check the value of a with print a The
    variable a was assigned the value of 5!
    11.6 We now know that when we enter out value for b, it's being assigned to a. We also know to look between lines 9 and 11 for our error. Close gdb by typing "q" and pressing enter. You will be prompted to kill the debugging session, do so by typing "y" and then pressing enter.
12. Look for the error.
    Do you see it? Here's a hint: It was caused by a copy-paste that went unedited.

That's right! Line 10 assigned the value entered for b to the variable a. As a result, a gets updated with the intended value for b, and, because b is never initialised or assigned, it holds a random value from whatever might be in memory.

13. Fix up the error, and run the calculator again. It should now work as expected.

Push your 3096S-Pracs-STDNUM001-STDNUM002 folder to your repository. Your submission on Amathuba should be a short PDF (around ~1 page) that includes your names, student numbers, a clickable link to your GitHub repo, and your corrected (working) code. Only one student needs to upload the code and submit their PDF on Amathuba.

Name the PDF submission as follows:

EEE3096S 2023 Tutorial 1 Hand-in STDNUM001 STDNUM002.pdf