



**IE4012**  
**Offensive Hacking Tactical and**  
**Strategic**  
**4<sup>rd</sup> Year, 1<sup>st</sup> Semester**

**< Shell Code >**

**< Report 02 >**

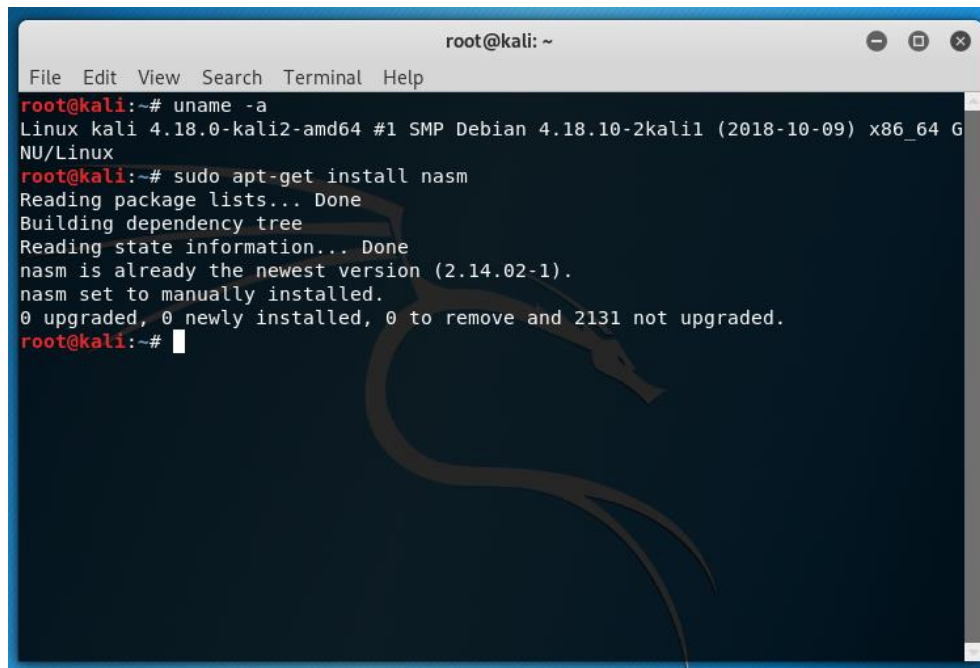
Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the  
Bachelor of Science Special Honors Degree in Information Technology

**<<08/03/2020>>**

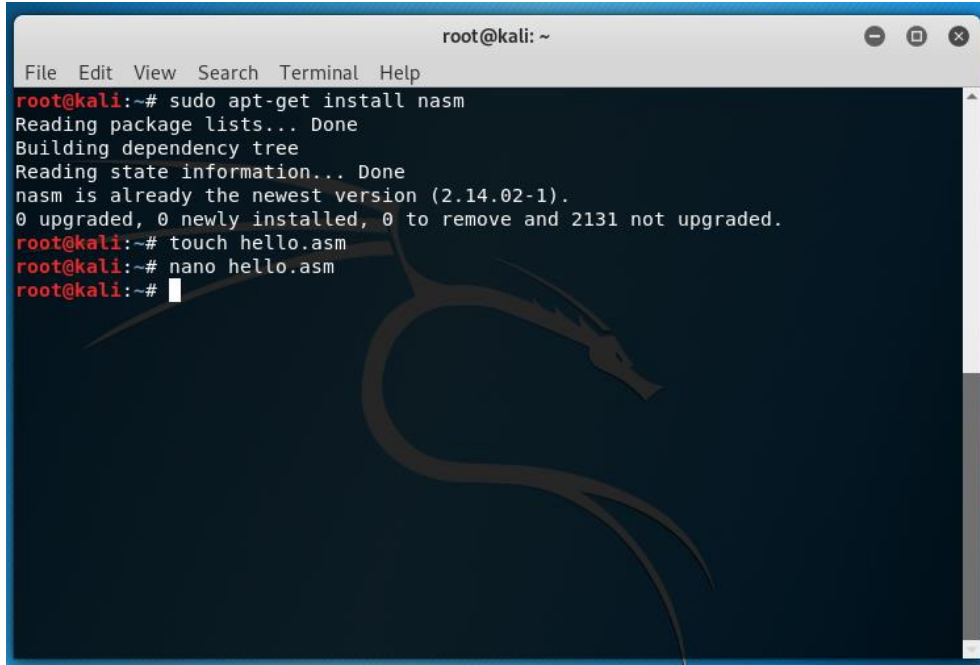
1. Install the nasm to the kali linux



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# uname -a  
Linux kali 4.18.0-kali2-amd64 #1 SMP Debian 4.18.10-2kali1 (2018-10-09) x86_64 GNU/Linux  
root@kali:~# sudo apt-get install nasm  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
nasm is already the newest version (2.14.02-1).  
nasm set to manually installed.  
0 upgraded, 0 newly installed, 0 to remove and 2131 not upgraded.  
root@kali:~#
```

Figure 1

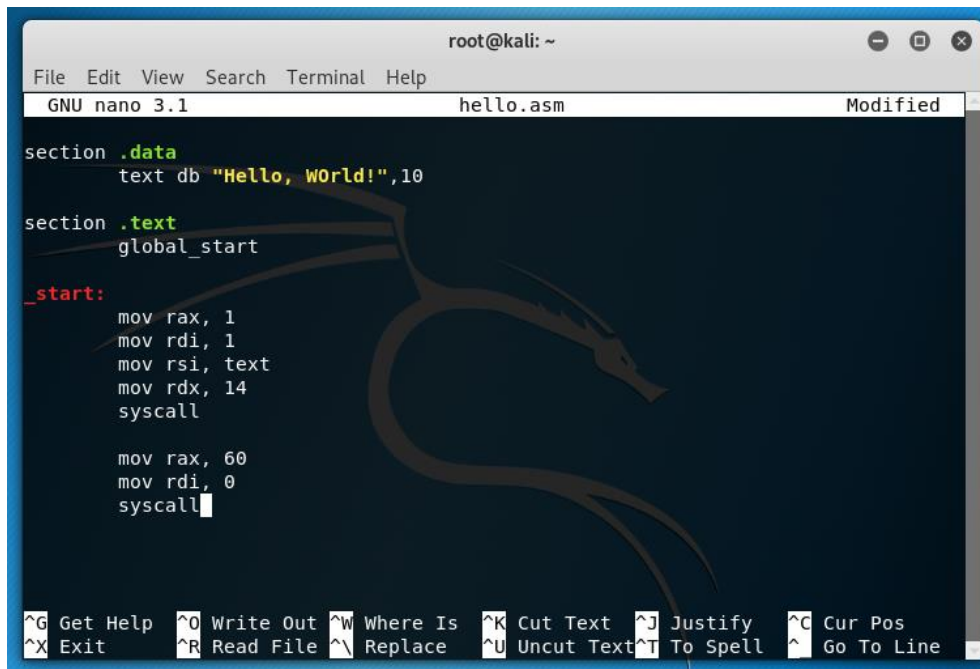
2. Open the **hello.asm**



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# sudo apt-get install nasm  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
nasm is already the newest version (2.14.02-1).  
0 upgraded, 0 newly installed, 0 to remove and 2131 not upgraded.  
root@kali:~# touch hello.asm  
root@kali:~# nano hello.asm  
root@kali:~#
```

Figure 2

3. Write a assembly code in **nano hello.asm**

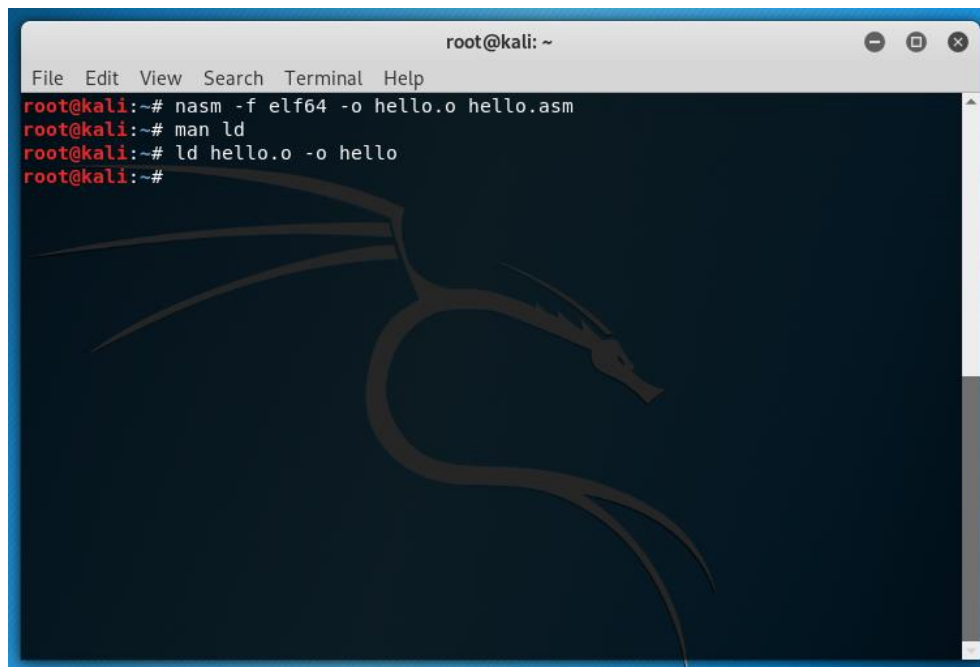


```
root@kali: ~  
File Edit View Search Terminal Help  
GNU nano 3.1 hello.asm Modified  
  
section .data  
text db "Hello, World!",10  
  
section .text  
global _start  
  
_start:  
mov rax, 1  
mov rdi, 1  
mov rsi, text  
mov rdx, 14  
syscall  
  
mov rax, 60  
mov rdi, 0  
syscall
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^\_ Go To Line

Figure 3

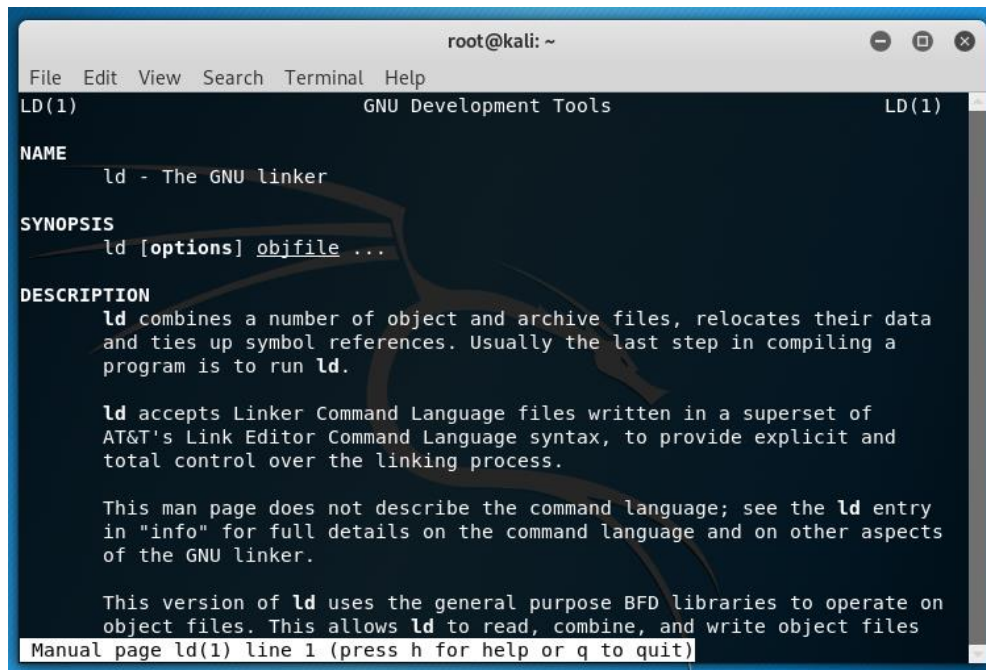
4. Compile this using
- Nasm -f elf64 -o hello.o hello.asm**
  - Man ld**
  - ld hello.o -o hello** these commands



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nasm -f elf64 -o hello.o hello.asm  
root@kali:~# man ld  
root@kali:~# ld hello.o -o hello  
root@kali:~#
```

Figure 4

5. Run the **man ld** command



The screenshot shows a terminal window titled 'root@kali: ~' with a menu bar containing 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal displays the output of the 'man ld' command. The output is formatted with section headers in all caps: NAME, SYNOPSIS, and DESCRIPTION. The NAME section identifies 'ld' as 'The GNU linker'. The SYNOPSIS section shows the command 'ld [options] objfile ...'. The DESCRIPTION section explains that 'ld' combines object and archive files, relocates data, and ties up symbol references. It also mentions that 'ld' accepts Linker Command Language files. At the bottom, a status bar reads 'Manual page ld(1) line 1 (press h for help or q to quit)'.

```
root@kali: ~
File Edit View Search Terminal Help
LD(1) GNU Development Tools LD(1)

NAME
  ld - The GNU linker

SYNOPSIS
  ld [options] objfile ...

DESCRIPTION
  ld combines a number of object and archive files, relocates their data
  and ties up symbol references. Usually the last step in compiling a
  program is to run ld.

  ld accepts Linker Command Language files written in a superset of
  AT&T's Link Editor Command Language syntax, to provide explicit and
  total control over the linking process.

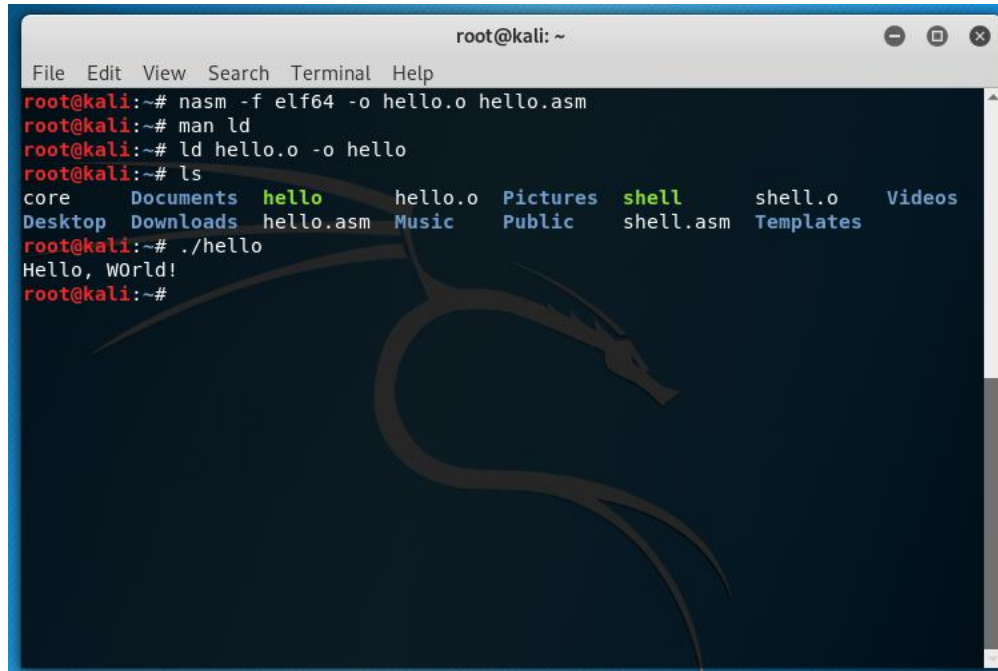
  This man page does not describe the command language; see the ld entry
  in "info" for full details on the command language and on other aspects
  of the GNU linker.

  This version of ld uses the general purpose BFD libraries to operate on
  object files. This allows ld to read, combine, and write object files

Manual page ld(1) line 1 (press h for help or q to quit)
```

Figure 5

6. Got the output – Hello, World!



The screenshot shows a terminal window titled 'root@kali: ~' with a menu bar containing 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal displays a sequence of commands and their outputs: 'nasm -f elf64 -o hello.o hello.asm' creates the object file; 'man ld' shows the linker manual page; 'ld hello.o -o hello' links the object file into an executable; 'ls' lists the files in the current directory, showing 'hello' and 'hello.o' among others; and './hello' executes the program, which outputs 'Hello, World!'.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nasm -f elf64 -o hello.o hello.asm
root@kali:~# man ld
root@kali:~# ld hello.o -o hello
root@kali:~# ls
core  Documents  hello      hello.o  Pictures  shell     shell.o  Videos
Desktop Downloads  hello.asm  Music    Public    shell.asm Templates
root@kali:~# ./hello
Hello, World!
root@kali:~#
```

Figure 6