Project 1 Report: Process and Thread
CSE 3320:Operating Systems
GROUP 3

Hamilton Nguyen 1000538439

February 20, 2020

Date Performed:	February 1, 2020
Partners:	Marvin Willington 1001660133

1 Assignment 0: Build the Linux kernel

According to the specifications stated in assignment 0 for project 1, all steps were carried out successfully on a 64-bit personal computer using a ubuntu via virtual box. However, there are some technical issues that were encountered in step 3 and step 6. In step 3, there were some error printout during the command execution using "sudo make -j4", later during the installation the best corrective action to avoid the printout errors is to switch the kernel version from kernel 5.3.0 to 5.0.1. . In step 6, there are still prersistent failed operations when going into reboot mode in which it never successfully reboot. The best corrective action to solve the software error in step 6 is to "sudo poweroff" and turn the Virtual Machine Environment on manually.Also to note that using command 'sudo' for each procedure are required but using 'sudo su -' and navigating to the LINUX SOURCE folder is also another solution to execute the step procedures. Refer to figure 1 for kernel version printout. Refer to the following attached kernel and user-level source codes.

2 Assignment 1: Add a new system call into Linux Kernel

According to the specifications stated in assignment 1, all steps were carried out successfully. In addition, a 64-bit Linux system call table were used for this particular assignment as required for the hardware that I have in possession. There are no errors that were encountered in the following steps procedures. Refer to figure 2 for "helloworld" printout using a system call.

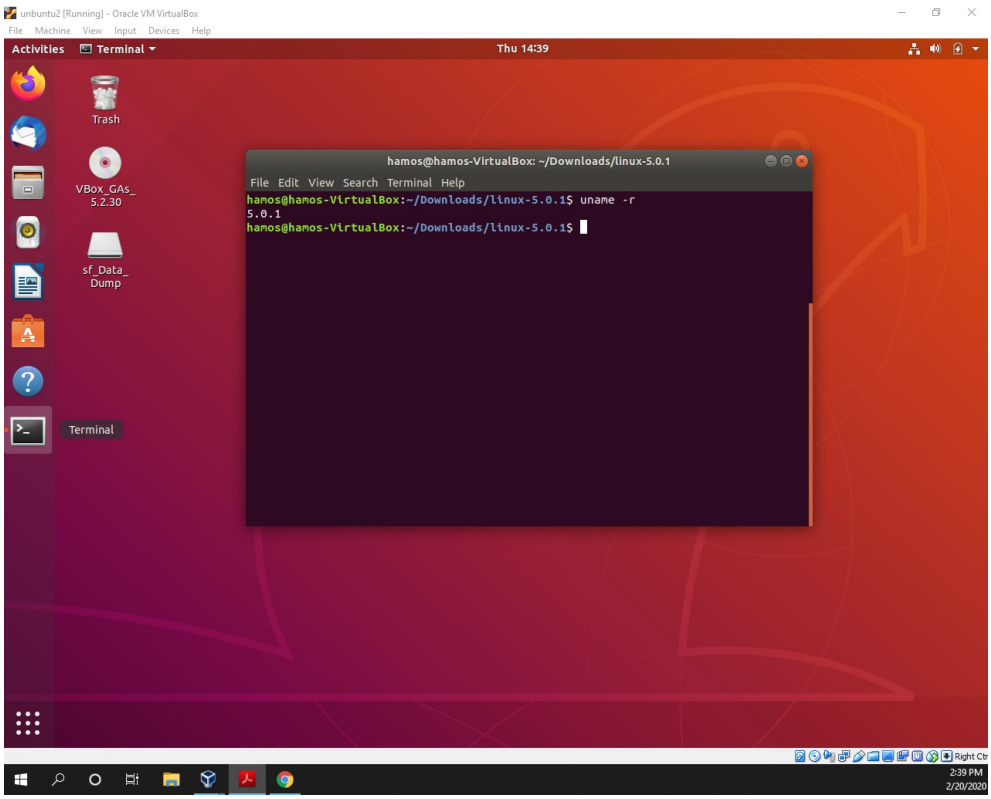

Figure 1: Assignment 0: step 6 printout of new kernel

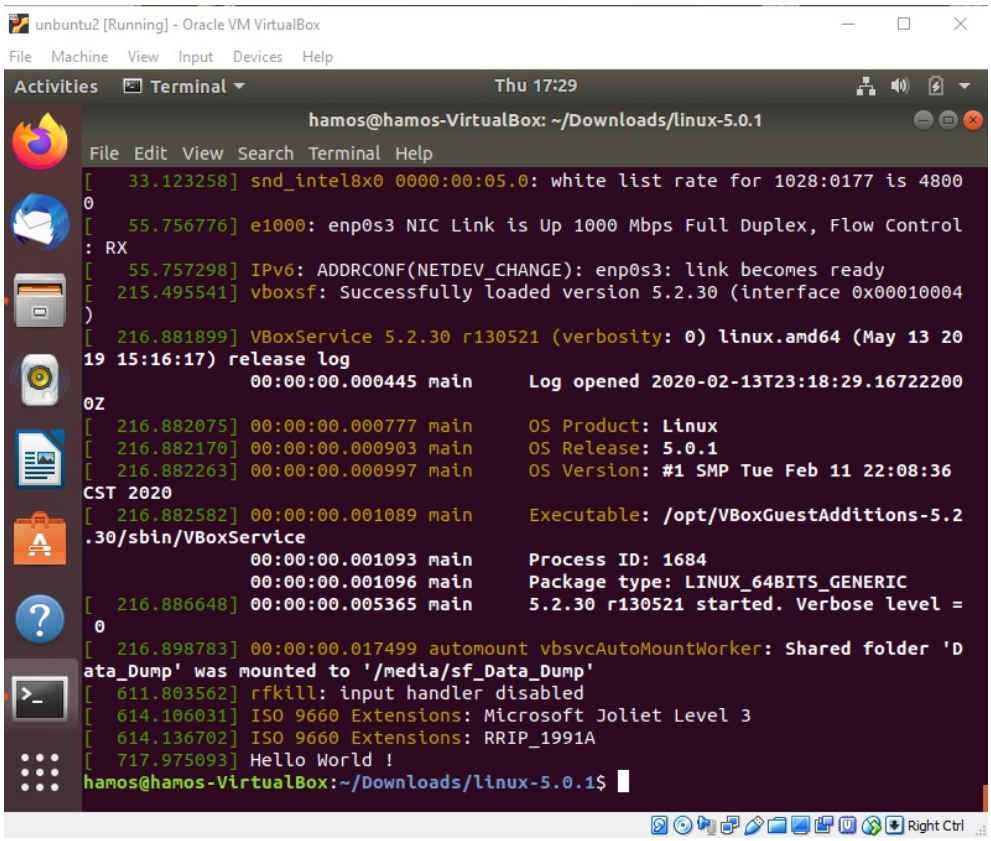

Figure 2: Assignment 1: "helloworld" system call printout using dmesg

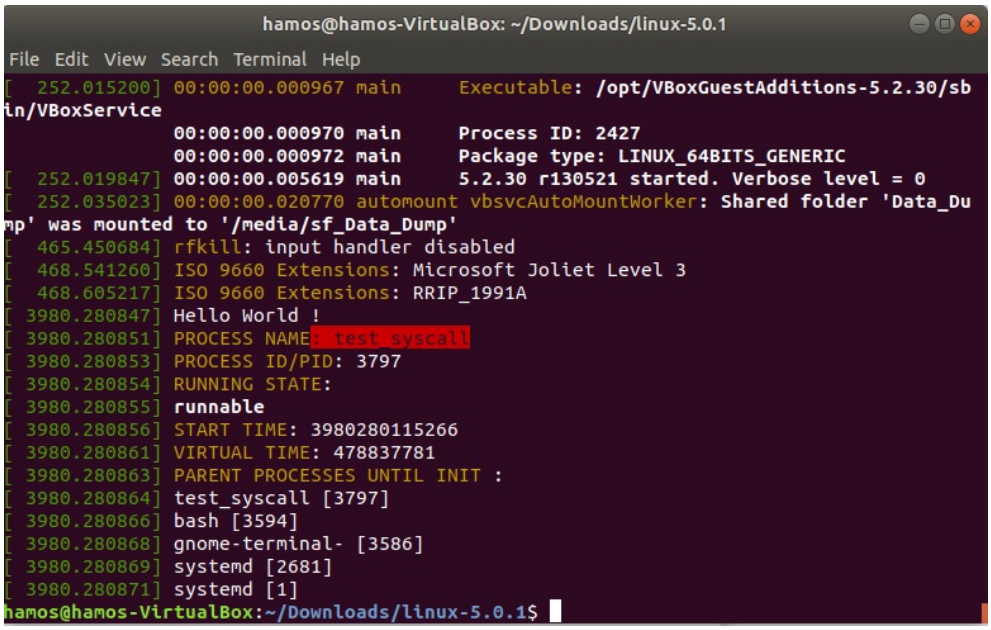

Figure 3: Assignment 2: information printout of a new system process from a system call print other

3 Assignment 2: Extend your new system call to print out the calling process information

According to the specifications stated in assignment 2, all steps were carried out successfully. A new system call, print self, was implemented in the kernel and user-level source code in which it print out various information of the process such as Process id(refer to as pid), running state, program name, start time, virtual time, and parent processes until init. The user-level code algorithms are implemented using a resource from a following website as stated in the step procedures. Refer to figure 3 for process information printout using a new system call. Refer to the following attached kernel and user-level source codes.

4 Assignment 3: Extend your new system call to print out the information of an arbitrary process identified by its PID

According to the specifications stated in assignment 3, the implementation of a new system call name print other was successful. The print other system call has sucessfully print the information for an arbitrary process. The mechanics of this system call functions when the client user code in the sys test call c file

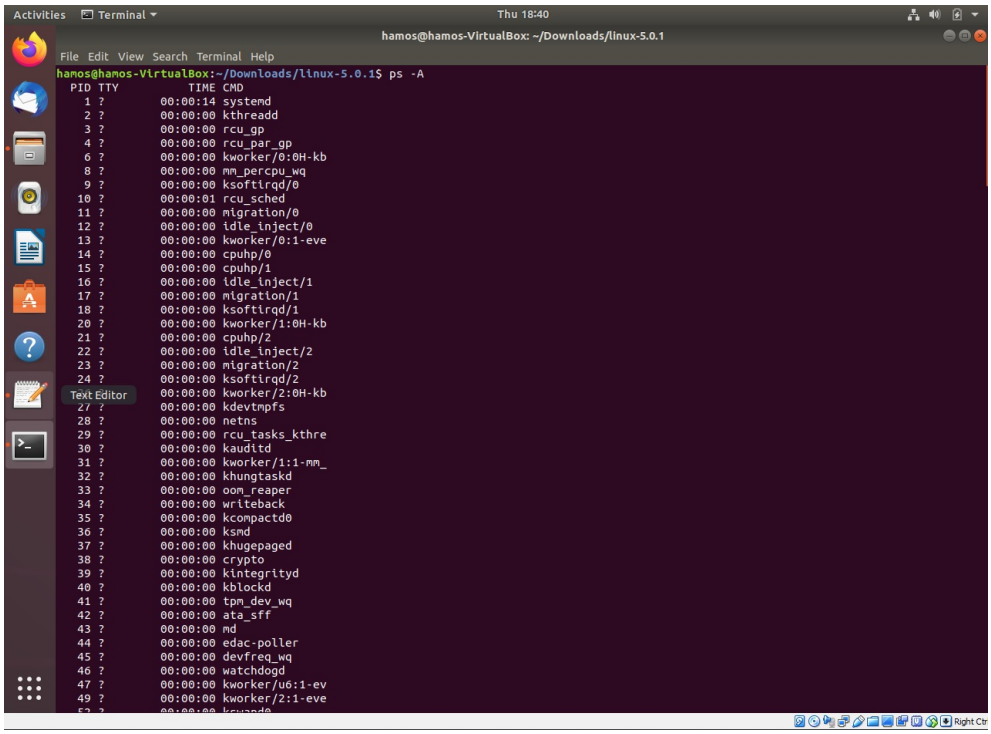

Figure 4: Assignment 3: pid hash table of current running system before compiling sys test call c file.

prompts the user to enter the Process Id (pid) number. Moreover, the user will enter the pid number and press enter, then the system will process, fetch, and print the process identified by the pid number of interest. The user can view all processes through the dmesg command through the Linux terminal. In addition, a new software application was installed for this particular assignment, the is application is name htop (using the command sudo apt-get install htop and ps -A). The htop software application accesses and create a table list of all run time pid numbers of the ubuntu operating system. From this software, determining the pid number of interest when using the client side prompt will help execute the arbituary process of the pid number and print out its process in the dmesg. Refer to figure 4, figure 5, figure 6 figure 7 and figure 8 for the step procedural process of this assignment. Refer to the following attached kernel and user-level source codes that accompanies this report.

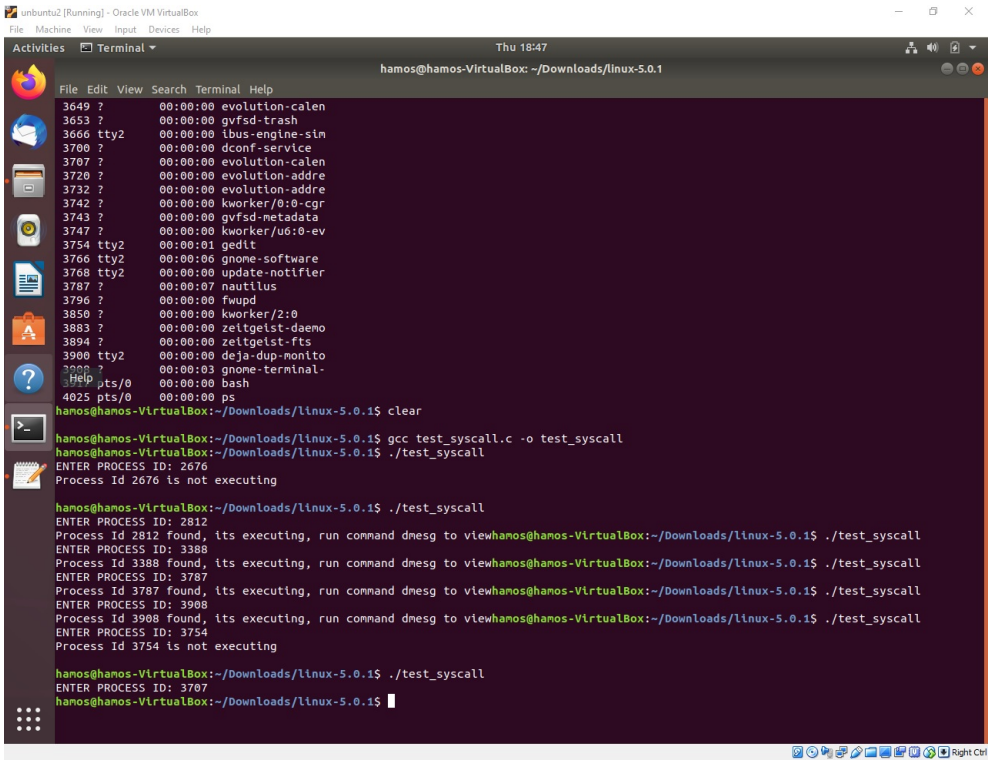

Figure 5: Assignment 3: compiling and executing sys test call c file in order to use user prompt.

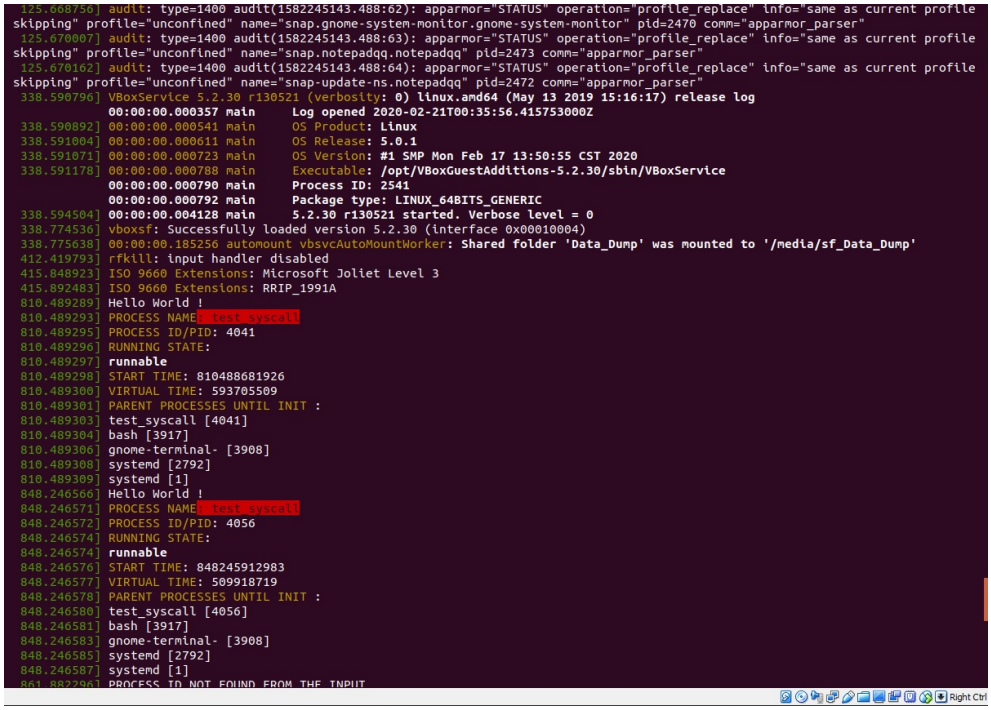

Figure 6: Assignment 3: dmesg printout part 1 after execution of sys test call c file.

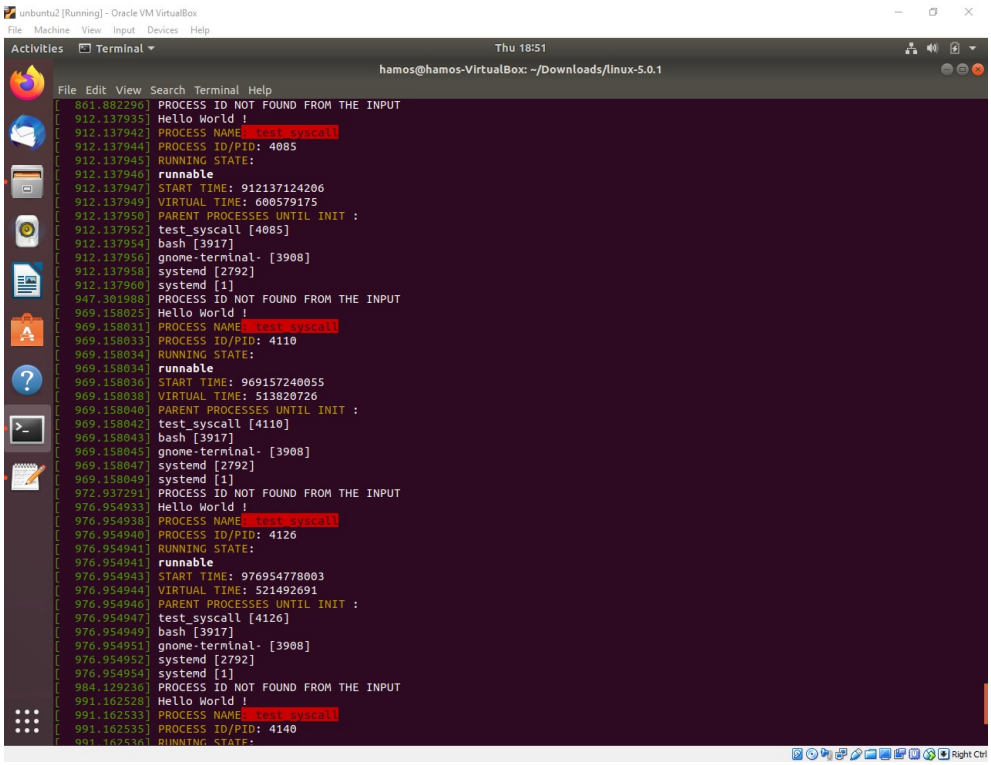

Figure 7: Assignment 3: dmesg printout part 2 after execution of sys test call c file.

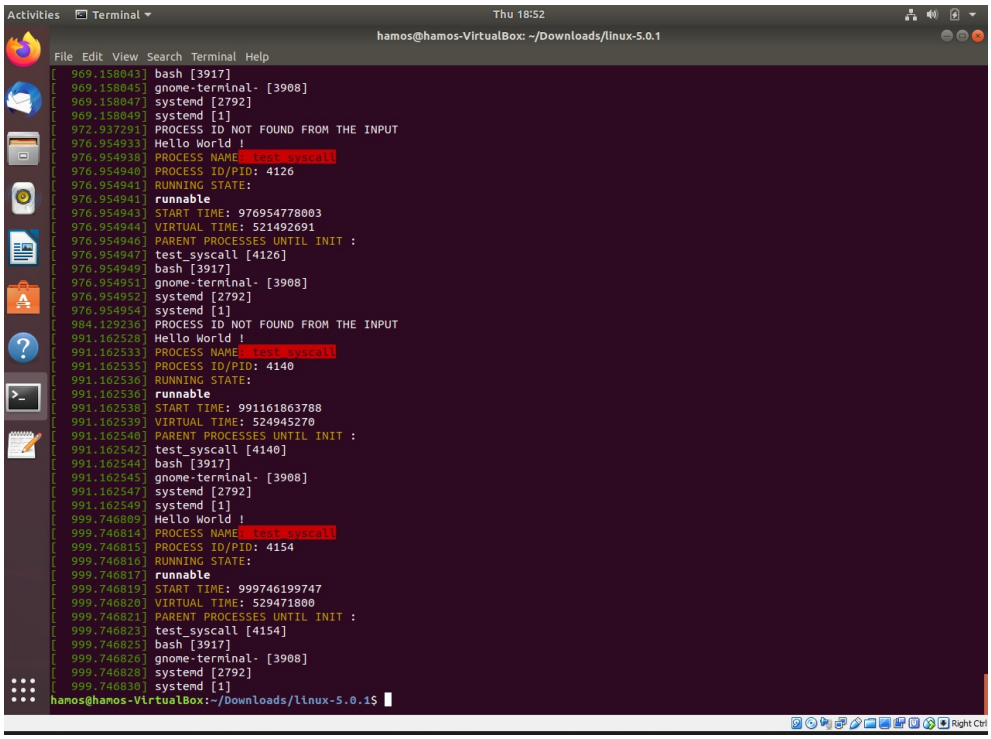

Figure 8: Assignment 3: dmesg printout part 3 after execution of sys test call c file.