Operating Systems - Monsoon 2023

Dhruv Kumar

September 5, 2023

Assignment 2 (Total points: 150)

Due: September 19, 2023. Time: 23:59 Hrs. (Hard Deadline)

Instructions-

There should be ONLY ONE submission per group

Submit a *.zip* named RollNo1_RollNo2.zip file containing code and write-up.

This assignment requires you to spend a lot of time researching on the internet, so it may get a little overwhelming. But after completing this assignment on your own, you will become confident enough to read any documentation and learn any software or system.

Q1 Setting-up Artix OS VM and Compiling the OS kernel (Total points: 30).

The first question requires you to set up your testbench VM. The VM should not require too much resources – approximately 4 GB RAM, with 2 virtual CPU cores and about 20 GB of hard drive space. You need to install Artix-base (runit version) on that following the instructions on the site. The installation must be done using the text mode instructions (and not by using the GUI).

To make your life easier you would need to do the following after the installation:

1. Enable ArchLinux repositories.
2. Install the following packages: binutils, elfutils, gcc, gdb, make, automake, autoconf, yasm and vim.

Thereafter, you would require downloading the stock linux kernel (from https://www.kernel.org/) and compiling and installing it on your testbench. Ensure that it boots up!

Some Important Points:

Please use the attached kernel config while compiling your kernel.

- The attached kernel config supports Oracle Virtual Box, VMware, and Qemu.

Some Important Points:

- Once you have unpacked the tarball from kernel.org, run 'make mrproper' to clean up the build artifacts
- Copy ‘config-rev-9-gold’ into the unpacked directory and rename it to .config
- Make sure to run make nconfig. Now press Escape to update the config according to your kernel version
- Now run make -j$(nproc) and continue the kernel compilation

Info regarding Vmware (Optional):

- Open your VM settings and remove Printer, SoundCard, and USB Controller. Our VM does not require these devices.

Info regarding Virtual Box (Optional):

- Open your VM settings and disable soundcard by going to the settings of your VM.

Grading Rubric:

UEFI enabled VM booting Artix. The students should be able to show the correct partitioning of the virtual hard drive

VM with running stock kernel (that is compiled)

Q2 Process scheduling (Total points: 70).

This exercise is to show you how to use Linux's scheduling policies for different processes. You will be creating three processes, where each of the three processes will count from 1 to 2^32. The three processes should be created with fork() and thereafter the child processes should use execl() family system calls to run another C file which will do the counting mentioned above.
Reiterating you need to launch three process, each of which calls another process(the counting process)
The following should be the detailed specification of each of the process, to being with:
1. Process 1 : Uses SCHED OTHER scheduling discipline with standard priority (nice:0).
2. Process 2 : Uses SCHED RR scheduling discipline with default priority.
3. Process 3 : Uses SCHED FIFO scheduling discipline with default priority.
Each of these processes must time the process of counting from 1 to 2^32. To time the execution, the parent process could get the clock timestamp (using clock_gettime()) before the fork and after each process terminates (the event of which could be identified when the blocking system call waitpid() returns).
We require you to benchmark these three schedulers by changing the scheduling classes of the three processes. You would require to generate histograms showing which scheduler completes the task when, depending upon the scheduling policy. You may choose different colours for the histogram bars, with one axis showing the time to complete, and the other showing the scheduling priorities. (You may use python to plot the histogram.)
Hint: To correctly benchmark scheduling policy, remember that all the three processes should be in READY state and then it is the policy that decides which process to give the CPU first.

Grading Rubric:

Successful compilation of the program.
Appropriately used system calls to create processes with the system calls to set the scheduling discipline and their priorities.
Error Handling
Program runtimes and their variations due the different process scheduling policies and priorities along with their histogram plot showing the same. (With reproducible behaviour)
Makefile to compile the above programs.
README/Write-up describing the program logic used for achieving the above (no more than one page)and an explanation of the outcomes of the tests/measurements.

Q3 Module Programming (Total points: 50).

This exercise requires you to write your own small kernel module. You require
to implement a kernel system call as a module. The task of the system call
would be to read the entries of the process task_struct and print the number of currently running processes. The system call should be implemented in the kernel. It should be functional
only when the module is loaded, not otherwise.
The task_struct data structure requires the <**linux/sched/signal.h**> header file. Ensure that the required header files are on your system else install these header files by trying these commands:
sudo apt-get install linux-headers
sudo apt-get install linux-headers-generic
NOTE: You can show the module loading and unloading in both Artix Linux or any Linux Distribution of your choice. However, make sure you have installed all dependencies and given the permissions as required.

If you are using a Windows system you would need to use a VM with any Linux operating system of your choice. Module programming cannot be done in WSL.

What to submit:

1. Fully functional system call that runs only when the module is loaded, and not otherwise
2. A proper Makefile to compile the module.
3. A readme file describing the program logic.
4. A screenshot of the message displayed while loading and unloading the module.