Laboratory 3

Deadline: 1 week

Part I

Modify the Minix3 source code such that the string “PID <pid> swapped in” is printed, whenever a user-level process is brought in by the scheduler.

To do this, you will need to understand how the scheduler works. These pages https://minixnitc.github.io/scheduling.html , http://www.minix3.org/docs/scheduling/report.pdf give a good explanation. You have to look at the code in : *minix/servers/sched* and *minix/kernel/proc.c* .

Part II

You can test by using the UnixBench benchmark suite as instructed below:

1. Get the source code from ftp://ftp.iitdh.ac.in/opensource/tools/byte-unixbench-mod.zip and copy it to your home folder in the Minix3 VM.
   (Note that this is a modified version of the suite. The original benchmarks were designed as “fixed duration” benchmarks: they each repeat a certain pattern over and over until a timer expires. The modification was done to make some of the benchmarks (namely, *arith*, *fstime*, *pipe*, *spawn*, *syscall*) as “fixed work”: they do a certain fixed amount of work, regardless of the time it takes to do that work.).
2. Run the command “gmake” to build the benchmarks in the Minix3 VM.
3. The compiled benchmarks are in the pgms folder. The folder *workload_mix* contains some scripts: *workload_mix*.sh*. Each spawns a batch of UnixBench benchmarks. You may study the behavior of the scheduler by seeing the sequence of “PID” prints when these workloads are run. Feel free to create workload mixes of your own (you can write your own benchmarks other than the UnixBench ones as well).

Submit a study of the nature of the benchmarks in the UnixBench suite by analyzing the schedule orders. You may supplement your analysis by studying the source code of the benchmarks (in the folder *src*) and also studying online documentation of these benchmarks (https://www.ostechnix.com/unixbench-benchmark-suite-unix-like-systems/ , etc.). Use commands such as “time”. Note: refrain from using the graphics benchmarks in the suite.

Submit: a single zip file (format: <roll-number-1>_<roll-number-2>_lab3.zip) with all modified source C files and a shell script. The shell script must copy the modified source files to the correct directories, and build the system. The evaluator will simply run the shell script, reboot the system, and check for the desired behavior. The zip must also contain a report of your study of the Unixbench suite.

Tip: print statements from the kernel and scheduler are also saved in */var/log/messages*. Print statements from user programs can be simply redirected to a file for analysis.