Assignment-0

Shriram R.
06-02-01-10-51-18-1-15763

1 System Call Implementation

The *my_precious()* has been implemented by declaring the new call in *include/linux/syscall.h* and defining inside *kernel/sys.c*. Also, new entries were added to *x86* architecture syscall tables (32 and 64 bit) with new syscall numbers and function mappings. A new entry was also added to *kernel/sys_ni.c* and *unistd.h*.

2 Benchmark - Setup

Benchmarks were run on a bare metal desktop consisting of Quad core Intel Core i5-6500T CPU clocked at 2.50GHz and 8GB RAM. The kernel was loaded within Ubuntu 18.04.1 OS. The benchmark code was compiled using GCC v7.4 and without optimization flags. Each benchmark was run in a loop of 100 iterations. The programs used for benchmark has 100MB allocated heap memory. The clock cycles were measured by reading TSC register (RDTSC instruction) using inline assembly code.

3 Benchmark - Analysis

The following table provides the mean, median, min, max and standard deviation of clock cycles for *fork()* and *my_precious()* system calls.

Call	**Mean**	**Median**	**Min**	**Max**	**Std. Dev.**
fork()	3218707	2372842	1611556	8150548	1636224
my_precious()	650	628	622	2658	200

We can observe that *fork()* takes about 3 orders of magnitude more cycles than *my_precious()*. The standard deviation for both calls is significant. This can be partially explained due to context switches during system call execution. For *my_precious()*, only the first iteration took more cycles and the rest showed negligible variance. This could be explained due to cache misses happening during the first invocation of the call. In general, median cycle counts should be taken as the representative value since it is not skewed significantly by outliers.

4 Reference

1. https://kernelnewbies.org/KernelBuild
2. https://www.kernel.org/doc/html/v5.0/process/adding-syscalls.html
3. Intel 64 and IA-32 architectures software developer's manual - Instruction set reference