CPSC 457 – Winter 2017
Assignment 1 (System Calls & Affinity)
Due: January 27th, 2017 (11:59pm)
Total 40 marks; assignment worth 8% of final mark

Instructions

1. Apply `affinity_mask_assignment.patch` to the `master` branch in `kos` directory. Instructions to apply the patch is provided in a text file named "How to apply patch for Assign1.txt"
   The overview of the patch is that, it modifies `kos/src/runtime/Scheduler.cc` and adds a `yield()` function to the scheduler. It adds a new system call called `get_core_count()` which returns number of cores running. It modifies `kos/src/runtime/Thread.h` to add a new variable called "`affinityMask`" to the thread class.
2. Create and checkout to new branch called "`affinity_mask`" and start your assignment.
3. `schedAffinityTest.cc` is the user program that would be used to test your solution for this assignment.
4. paste `schedAffinityTest.cc` to `kos/src/user` to test your solution.
5. Find the sample output file in the support material of this assignment.

GIT guidelines that must be followed:

1. Start each assignment in a NEW BRANCH from the master branch.
2. After applying the patch provided with your assignment and pasting the provided file, please **`commit`** your changes
3. Command to `patch` of your assignment is: `git diff <final commit> <commit after which you started coding your assignment>`

Submission Instructions:

Submit a hardcopy of the below in the same order (only one submission per team is required)

1. Cover page Full name and student ID of each group member.
2. Answer to question 1.
3. `patch` of containing only assignment solution.
4. Any new files added (that does not mean that a new file must be added)

Upload in d2l

1. the patch file of your changes

Questions:

1. Answer each of the following questions in one sentence. What are processor modes? How many modes does Intel i5 core have? What is the mode of operation of an Intel i5

core when it executes a procedure call? What is the mode of operation of an Intel i5 core when it executes a system call?

2. For this assignment, you would add two linux system calls to KOS kernel. System call specifications are as below.

   int sched_setaffinity(pid_t *pid*, **size_t** *cpusetsize*, **cpu_set_t *****mask*);
   int sched_getaffinity(pid_t *pid*, **size_t** *cpusetsize*, **cpu_set_t *****mask*);
   (`cpu_set_t` defined as a 64 bit unsigned integer in `kos/src/include/kostypes.h`.)

   `sched_setaffinity` sets the affinity of the calling process to only those cores specified by the mask. Suppose the mask value is 0x11. Then, after the **`sched_setaffinity`** call, the calling process must be scheduled only on cores 0 and 1.

   `sched_getaffinity` sets the value of the mask pointer to the affinity of the calling process.
   (You have to modify `Scheduler::preempt()` so that the kernel schedules a process only on those processor cores that are allowed by the mask set by the process.)

   Detailed specifications about the system calls can be found here -
   http://linux.die.net/man/2/sched_getaffinity
   http://linux.die.net/man/2/sched_setaffinity

   For simplicity and viability purposes, the requirements of the above system calls are set as follows:

   1. EPERM and EINVAL are the only error conditions checked by `sched_setaffinity`.
   2. EPERM is the only error condition checked by `sched_getaffinity`
   3. The only accepted value for the first argument `pid` is 0. Any other value leads to EPERM error.
   4. Trying to set the bit of a non-existent processor (core) leads to EINVAL error. The default configuration is to run KOS with 4 cores.
   5. The masks can be read and write directly by the user program. Mask value of 0x1 refers to core 0, and mask value 0x3 refers to cores 0 and 1.

Notes and hints:

1. `Machine::getProcessorCount()` returns the number of cores. This call can be used only inside the kernel. For user programs, `get_core_count()` system call can be used. Indeed, `get_core_count()` system call internally used `Machine::getProcessorCount()`.

2. For "`SCHED TEST AFFINITY TEST 4`", the process would be scheduled in a mix of cores 1 and 2. Verify that you get similar output.

```
…
I am running on core 1
I am running on core 2
I am running on core 1
I am running on core 1
I am running on core 2
```

3. If the kernel crashes, try running running KOS twice or thrice to determine if the bug is deterministic or nondeterministic and get further help from your TA.
4. make debug runs KOS in debug mode with gdb. You can set breakpoints in the kernel using gdb debugger.

Coding guidelines:

- Comment the code appropriately.
- Use appropriate short names for variables.
- Use indentation for easy readability.

Administrative Information

Teams

You are advised, but not required, to work in a team of two members. Teams of more than 2 students are not allowed. You and your partner must be in tutorials that are taught by the same TA. One submission per team is required.

Academic Misconduct

Any similarities between assignment submissions will be further investigated for potential academic misconduct.

Code sharing with a different group is prohibited. Code sharing includes looking at others' code on paper and on the computer screen. Discussions with other groups can only be carried out at the concept level. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your team's original work. Any re-used code in excess of 10 lines must be acknowledged and cited. Violation of this policy may be considered academic misconduct. If unsure, always check with your instructor or TA.

Late submission policy

Late submissions are penalized with 12.5% deduction for each late day or portion of a day. Hence no submission are accepted 8 days after the deadline

D2L Marks

Any marks posted on D2L or made available using any other mean are tentative are subject to change (after posting). They can go UP or DOWN due to necessary corrections.

Happy Coding :)