

CS444 Homework 1

Lilliana Watts, Nicholas Skinner, Yong Ping Li

April 15, 2018

COMMANDS USED

Placeholder

QEMU FLAGS

A. Command Line:

-drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -net none -usb -localtime --no-reboot --append "root=/dev/vda rw console=ttyS0 debug".

Individual flags:

- **-gdb**: This flag is followed by a device (for this example localhost 5507) and signals a wait for a gdb connection on the device.
- **-S**: this tells it to not start CPU on startup. This is the reason why the command will halt until you type continue.
- **-nographic**: This disables the graphics, and makes qemu into a command line interface.
- **-kernel**: This signals to use the file that follows as the kernel image.
- **-drive**: This defines a new drive to use. The drive is used with the disk image that follows after the flag, for this setup, that is file=core-image-lsb-sdk-qemux86.ext4
- **-enable-kvm**: This flag enables full kernel based VM support, allowing us to use the linux kernel to create and run a VM.
- **-net none**: No network devices are being configured with our setup, so we use this flag.
- **-usb**: This flag enables the usb driver. We can use usb to upload files to the VM.
- **-localtime**: Boots up the kernel with the current UTC.
- **--no-reboot**: this flag makes it so that instead of allowing a reboot, the program will just exit.
- **--append**: This flag passes command line arguments to the booting kernel, it appears that in this example it is located in /dev/vda, it is using ttyS0 for the linux port, and the kernel is starting in debug mode.

CONCURRENCY QUESTIONS

B. What do you think the main point of this assignment is?

•This assignment is intended to refresh our knowledge of parallel programming, by having us implement a multi-threaded program with shared resources.

C. How did you personally approach the problem? Design decisions, algorithm, etc.

•We first researched the information provided in the Little Book of Semaphores[1], as well as the operation of mutex locks. The example provided by the book had been a useful starting point as it had a step-by-step approach. As the `rand` instruction is not supported on the provided os2 servers, we instead used the Mersenne Twister C Implementation[2] for our code. In our code, it was used by creating a function that generates a number within the given range passed to the function (two ints).

We used the Twister function `genrand real2` to acquire a double value between 0 and 1, then we multiply the maximum value by that double until a value above the minimum was obtained. The shared buffer that had been implemented was an array of 32 item structs. The buffer operates by placing an item in the first available space, which is denoted by an `Item.number` value of 0. Items are consumed by the buffer in the same manner, where it finds the first available non-zero item, saves the `Item.number` and `Item.wait` values in a new `Item`, then resetting said `Item`'s members to zero and returning its saved item. Consumer sleep time is read from the returning struct, and the producer sleep time is a randomly generated value.

D. How did you ensure your solution was correct? Testing details, for instance.

`Printf` statements were used frequently within the program's operation, like when a producer had created an item, when a consumer removed an item, as well as when a buffer had been accessed. the program had been left to run until 100 items had been both produced as well as consumed. and the output had been checked to determine if that had what was expected.

E. What did you learn?

•We learned how mutex locks and `pthread` waits operate, and how to implement both correctly. More specifically, we learned how `pthread cond` signals are used to unblock the threads, and when to call them.

GITHUB LOG

Detail	Author	Description
9138e68	Nicholas	Initial commit
c0384d2	Nicholas	Command Line Flags
05e52f4	Nicholas	FixFormat ConcQuestions
74e34ce	Nicholas	Add Table to Writeup

WORK LOG

Date	Name	Hours	Description
4/14	Nicholas	2	writeup
4/13	Nicholas	2	writeup
4/12	Lilliana	2	VM set up
4/12	Nicholas	2	writeup

REFERENCES

- [1] Makoto Matsumoto and Takuji. Nishimura. qemu-system-x86_64(1) qemu-system-x86 debian jessie debian manpages. <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/CODES/mt19937ar.c>. (Accessed on 04/13/2018).