# Assignment 1

Chris Bakkom and Marie Bomber

October 9, 2017

## 1 Building qemu

1. A log of commands used to perform the requested action

```
cd /scratch/fall2017
mkdir 33
cd 33
/scratch/bin/acl_open .  bakkomc
git clone git://git.yoctoproject.org/linux-yocto-3.19
cd linux-yocto-3.19
git checkout 'v3.19.2'
git branch Group33
cd ../../../files
cp environment-setup-i586-poky-linux ../fall2017/33/linux-yocto-3.19/
cp core-image-lsb-sdk-qemux86.ext4 ../fall2017/33/linux-yocto-3.19/
cp config-3.19.2-yocto-standard ../fall2017/33/linux-yocto-3.19/.config
cd ../fall2017/33/linux-yocto-3.19
source environment-setup-i586-poky-linux
make -j4 tags
make menuconfig
general Setup/Local version 33-HW1
qemu-system-i386 -gdb tcp::5533 -S -nographic -kernel bzImage-qemux86.bin -drive file=core-image-lsb-sdk-qe
-enable-kvm -net none -usb -localtime --no-reboot --append "root=/dev/vda rw console=ttyS0 debug"
```

And then in a new terminal:

```
gdb
connect remote localhost:5533
continue
```

2. An explanation of each and every flag in the listed qemu command-line

```
 -S: This specifies that the vm is to be used with gdb and will wait for a connection.

-nographic:  Qemu can be compiled with support for a graphics window.  This command runs qemu as a text
based command line application.

-kernel:  This is the bzImage, which will provide the Linux kernel image.  Following this specifies the
Linux kernel command arguments which is the argument bzImage-qemu.bin

-drive:  This defines the disk image where the drive will be.  The argument for this "file=core-image-lsb-s
is the file name that defines the disk image.  The second argument defines the interface for the drive.
In this case if=virtio which is a virtual interface over the driver.

-enable-kvm:  If qemu is compiled with kvm then this enables kvm virtualizations support

-net:  This specifies the virtual network that the vm will use which in this case is "none" but you can
specify a user mode network stack

-usb:  This enables the usb driver for the vm.

-localtime:  This starts the vm in local time and date.

-no-reboot:  This specifies that the vm exit instead of rebooting

-append:  This gives command arguments to the linux kernel.  In this case it specifies the "root=/dev/vda
rw console=ttyS0 debug" which loads a linux kernel inside Qemu and configures the console to use the serial
port 0 and in debug.
```

## 2 Concurrency

**A**nswer the following questions in sufficient detail (for the concurrency):

1. What do you think the main point of this assignment is?
   Marie: I feel the main point of the assignment was to reacquaint ourselves with threading and start getting familiar with assembly calls. It turned out to be a fun exercise in getting to know C again (and remember that objects aren't always going to be around to save us).
   Chris: I think that this assignment was to refresh ourselves with C programming, getting comfortable with threads and implementing code we would have had to research. I had never heard of the Bull Mountain project and it was a nice exercise in implementing something I was unfamiliar with.

2. How did you personally approach the problem? Design decisions, algorithm, etc.
   At first, the problem presented felt overwhelming, but we found that breaking the problem down to the functions we would need (thus breaking the project into smaller parts) made the task much easier to conceptualize and then understand what components were needed where. From there, it was just reminding ourselves how pthreads and mutexes work, and piecing each element together.
   The key here is just to take it one step at a time. Lay the frame work for the objects necessary, then write some testable functions and then piece them together one at a time.

3. How did you ensure your solution was correct? Testing details, for instance.
   As each function was being built, it was tested in a short main to confirm it was acting as intended, and then, once all the pieces were put together, a "debug mode" was created, where enabling the debug flag would generate print statements confirming the intended actions were being taken.

4. What did you learn?
   Beyond the basic "this is how to call assembly instructions and oh yeah, threads are something you need to be thinking about again", we learned that yes, anything more than 3 levels of indentation means things were probably getting to complicated and should be broken out (with the exception of debug mode). Also, sometimes you need to read the documentation three to four times before you truly understand what you're working with.
   Specifically, we learned about thread conditions and condition signals, how to write inline assembly and the intel rdrand function and how to tell if your processor supports it. Just for note purposes, it will set the 30th bit in the ecx register after calling cpuid.

5. Version control log (formatted as a table) – there are any number of tools for generating a TeX table from repo logs
   Note - Apologies for the short logs, we had forgotten this question when working on the concurrency assignment and as the assignment was primarily worked on the OS2 server with separate files being created and then merged into the core concurrency file, git was not initialized until late in the project.

   sha User Date/Time Comment 559aa5c Marie Bomber Sun Oct 8 22:42:57 2017 -0700 Added a few more debug statements
   8ccda9e Marie Bomber Sun Oct 8 22:30:59 2017 -0700 Fixed rdrand version
   1092c80 Marie Bomber Sun Oct 8 22:28:12 2017 -0700 Initial version

6. Work log. What was done when? Be detailed.
   2017-10-03 10:00-11:00 Marie - Setup inital group folder, imported linux yocto, began initialization
   2017-10-05 10:00-11:00 Marie - Finished yocto setup, compiled, helped classmates
   2017-10-07 09:30-11:00 Marie - Confirmed running of yocto and attaching GDB, created LaTex file and created master git repos
   2017-10-07 14:30-15:00 Chris - Ran yochto and gdb attached
   2017-10-07 15:00-16:30 Chris - Gathering flag definitions
   2017-10-07 14:45-18:00 Marie - Began concurrency implementation
   2017-10-07 20:45-23:15 Marie - Continued on function creation for concurrency implementation
   2017-10-08 10:00-11:30 Marie - Finished function creation and added Mersinne Twister, all code running except ASM random number generation
   2017-10-7 20:00-0:00 Chris - Working on concurrency implementation
   2017-10-8 11:00-14:00 Chris - Implementing asm function and rdrand
   2017-10-8 21:30-23:10 Marie - Debugging asm function, cleaning up debug mode, writing report.
   2017-10-8 22:00-23:00 Chris - Helping debug the asm function
   2017-10-9 00:00-01:00 Chris - Adding some notes to the question answers
   2017-10-9 20:00- Marie - Finished write up, including this log, transferred git logs, tar-balled assignment and submitted.