# 1 Names

Dominic Giacoppe, Benny Zhao, Ryan Wood

# 2 A log of commands used to perform the requested actions.

1. Open two terminals and in the first terminal:
2. cd /scratch/spring2017/11-03
3. git clone git://git.yoctoproject.org/linux-yocto-3.14
4. git checkout v3.14.26
5. cp /scratch/spring2017/files/config-3.14.26-yocto-qemu .config
6. make menuconfig to get a new window
7. inside the window: press / and type in LOCALVERSION, hit enter
8. press 1, press enter and edit the value to -11-03-hw1
9. make -j4 all
10. cd ..
11. gdb
12. In the second terminal:
13. source /scratch/opt/environment-setup-i586-poky-linux
14. cd /scratch/spring2017/11-03
15. qemu-system-i386 -gdb tcp::5613 -S -nographic -kernel linux-yocto-3.14/arch/x86/boot/bzImage -drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio-enable-kvm -net none -usb -localtime –no-reboot –append "root=/dev/vda rw console=ttyS0 debug"
16. if you are getting file not found errors something got deleted when we weren't looking
17. else it should just hang here
18. Back to the first terminal
19. target remote :5613
20. continue
21. root, press Enter
22. you're in, do what you want in your linux environment.

# 3 Explanation of each flag in the listed qemu command line

1. -gdb :Waits for the gdb connection to device
2. tcp::5613 :Opens connection to port 5613
3. -S :Do not start CPU at startup (requires user to type "c/continue" in the monitor)
4. -nographic :Disables graphical output so QEMU is a simple command line application
5. -kernel ¡bzImage¿ :Use the specified bzImage as kernel image and kernel can be either a Linux kernel or multiboot format

6. -drive file=¡file¿, if=¡interface¿ :Define a new drive where file= is the disk image to use and if= defines which type of interface the drive is connected
7. -enable-kvm :Enables KVM full virtualization support
8. -net :Indicates that no network devices should be configured
9. -usb :Enables the USB driver
10. -localtime :Required for the correct date in MS-DOS or Windows
11. –no-reboot :Exit instead of rebooting
12. –append :Use cmdline as the kernel command line

# 4 Answer the following questions in sufficient detail (for the concurrency)

1. **What do you think the main point of this assignment is?**
   - The main point of this assignment was to engage the student with the concept of concurrency and how to think in parallel. By parallel, we mean to learn how threads work and how to synchronize them with the use of mutexes/locks and to prevent deadlock.
2. **How did you personally approach the problem? Design decisions, algorithm, etc.**
   - To approach the problem, our team first added the data structures we plan to use later in the functions and then we laid out the skeleton code of how we want our code to run. The initial design was to pass in arguments to create our producer and consumer threads. But after some debugging with gdb, we came to a dead end in what we were trying to achieve. Therefore, one of our group members went to get clarification from the professor about what exactly are we achieving here in terms of the programming I/O. We got back together as a group and discussed what changes we wanted to make towards our code design. The most recent design consisted of the producer function to continuously produce and our consumer function to consume when a resource is available independent of user input.
3. **How did you ensure your solution was correct? Testing details, for instance.**
   - To ensure our solution was correct, we used gdb to debug our initial code by checking when the consumer is attempting to consume an empty buffer and when the producer is trying to produce on a full buffer. We also developed the random generator implementation file independent of the main file, so that we could confirm there would be no issues from the random generator. Then we refactored our code according to the clarifications we were provided.
4. **What did you learn?**
   - For this assignment, we learned how to utilize pthreads in more depth and understanding how the locks fundamentally work in this problem context. Also, that globals do indeed have a place in the world, and to make sure to pester higher command when clarification on requirements is needed

# 5 Work Log

- 4/10 Went to lab, met each other, spend our hour trying to compile the kernel. Dominic took charge to finish it
- 4/12 Met after class, Dominic demonstrated how he compiled the kernel. Agreed to look over the concurrency homework separately over the weekend and come up with code.
- 4/18 Met after class. Started working on the baseline of the code until lab, then worked some more after lab. Agreed to have Ryan get the assembly bit working, and Dominic and Benny get the rest of it working.

- 4/19 Ryan gets the assembly working, Dominic creates the first draft that would turn out to be not what the assignment wanted. Benny gets it to compile and run, but we all agree to meet tomorrow in an attempt to fix it.
- 4/20 Team meets in Kelley after class. We spend 2 hours debugging it before speaking more with Kevin to determine that our solution was not what he wanted really, and that what he wanted was much easier. Dominic then spends 30 minutes revising it and gets a mostly working copy going.
- 4/21 Team meets to look over this document and check the code before submission. It goes well.