**CSC595 HW3 Part A:**

1. Create a new directory named “new-root”.
2. Determine all dependencies for bash and copy them under “new-root”.
3. chroot to “new-root”. Show that the command bash works by calling bash.
4. Check if cd, cd . , pwd, and less can be run in this bash.

You must submit a shell script with the commands used in (i)-(iv). In a Word document, answer the following: Which of the commands in (iv) ran and why?

**Commands that ran:**

“cd” , “cd . “ , “ls” , “pwd” --> These commands ran because they were either builtints or I copied the proper dependencies into the proper library paths

**Commands that did not run:**

“ps” , “less” --> I know ps does not run because there are certain dependencies I don’t have. However, I do not know why ‘less’ does not run because I properly copied the dependencies gathered from the ‘ldd less’ command

**Script Content:**

#!/usr/bin/env bash

sudo mkdir /new-root

sudo mkdir /new-root/{bin,etc}

sudo mkdir -p /new-root/{lib/x86\_64-linux-gnu,lib64}

sudo cp /bin/bash /new-root/bin/

sudo cp /bin/ls /new-root/bin/

**for** i **in** /lib/x86\_64-linux-gnu/libtinfo.so.5 /lib/x86\_64-linux-gnu/libdl.so.2 /lib/x86\_64-linux-gnu/libc.so.6 ; **do** sudo cp -v $i /new-root/lib/x86\_64-linux-gnu/; **done**

sudo cp /lib64/ld-linux-x86-64.so.2 /new-root/lib64/\

**for** i **in** /lib/x86\_64-linux-gnu/libselinux.so.1 /lib/x86\_64-linux-gnu/libc.so.6 /lib/x86\_64-linux-gnu/libpcre.so.3 /lib/x86\_64-linux-gnu/libdl.so.2 /lib/x86\_64-linux-gnu/libpthread.so.0 ; **do** sudo cp -v $i /new-root/lib/x86\_64-linux-gnu/; **done**\

**for** i **in** /lib/x86\_64-linux-gnu/libc.so.6 /lib/x86\_64-linux-gnu/libtinfo.so.5 ; **do** sudo cp -v $i /new-root/lib/x86\_64-linux-gnu/; **done**

**echo** "PS1='JAIL $ ' " | sudo tee /new-root/etc/bash.bashrc

sudo chroot /new-root/ /bin/bash

**echo** "You Are Now in Jail, Enjoy!"

**CSC595 HW3 Part B:**

I was able to break out of the jail. However I was not able to execute the last bin/bash in interactive mode exec() call. I think it is because my re-traversal of the file directory path. There is also an issue with having accidentally set my root privileged access point of the filesystem so a duplicated ‘/proc’ under the name of ‘ps’. I did this by mistake when playing around with mount() system call programs. I am going to come back to this, but the main success was breaking out of the jail and I was able to validate this through print statements. Source code provided in zipfile.

**CSC595 HW3 Part D:**

1. NAMESPACE column has the namespace ID’s of each currently open namespace. TYPE column has the namespace type which [mnt, pid, uts, net, user, ipc, cgroup]. NPROCS column has the amount of proc filesystems being used by each namespace ID and its type. The USERS column has the UID of the user initializing the namespace. Lastly CMD column has the system command that created the namespace.

4026531836 pid 117 0,1,100,104,107,1000 /sbin/init

1. PID column has the numbered process ID which acts as a primary key. PPID column houses the PID of the parent process to this rows process. NAME column has the name of the process the row is for. CMD column has the command that instantiated the process. NTHREADS column is how many threads this process is currently running. CGROUPS column has information regarding the different devices in /proc/<PID>/cgroups.

PID PPID NAME CMD NTHREADS CGROUPS STATE

1 0 systemd /sbin/init 1 11:blkio:/init.scope S (sleeping)

10:cpuset:/ 9:perf\_event:/

8:devices:/init.scope

7:freezer:/

6:cpu,cpuacct:/init.scope

5:memory:/init.scope

4:pids:/init.scope

3:net\_cls,net\_prio:/

2:hugetlb:/

1. This just outputs two columns from previous calls. The NAMESPACE column has the namespace ID and the TYPE column has the namespace type between the choices of uts, mnt, ipc, pid, net, and user.
2. I entered commands into the terminals while monitoring them in another terminal and the monitoring is real-time with every command entered and process(es) run. You can view the memory usage of the process as well as it’s storage and the namespaces it occupies. This is the command that yields the most interesting information.

cinf PID [1] PPID [0] CMD [/sbin/init]

UIDS [0 0 0 0] STATE [S (sleeping)]

NAMESPACES [[{mnt 4026531840} {uts 4026531838} {ipc 4026531839} {pid 4026531836} {net 4026531993} {user 4026531837}]]

CONTROLFILE VALUE

memory.usage\_in\_bytes 5857280

cpuacct.usage 3829241301

blkio.throttle.io\_service\_bytes 202:0 Read 5296128 202:0 Write 0 202:0 Sync 0 202:0 Async 5296128 202:0 Total 5296128 Total 5296128

cinf PID [24841] PPID [24840] CMD [/bin/bash]

UIDS [1000 1000 1000 1000] STATE [S (sleeping)]

NAMESPACES [[{mnt 4026532167} {uts 4026532168} {ipc 4026532169} {pid 4026532170} {net 4026532172} {user 4026532162}]]

CONTROLFILE VALUE

memory.usage\_in\_bytes 72728576

cpuacct.usage 46180313

blkio.throttle.io\_service\_bytes 202:0 Read 134082560 202:0 Write 875413504 202:0 Sync 24395776 202:0 Async 985100288 202:0 Total 1009496064 Total 1009496064