Ryan Citron
Hw1

This first assignment was difficult since Minix was a new operating system and the first operating system I have edited/written code for. In total, I probably put in a total of 40 to 50 hours into this assignment. I found that once you started getting invested into how everything works it was really easy to dial into. I even pulled an all-nighter because I enjoyed doing it. I was able to complete the basic task. This included sending information from the kernel to the servers to print out the number of messages sent and also get the formatting mostly correct. I was able to also get the table to tab to the right until all of the processes were shown and then scroll down to see the next set of processes. Unfortunately, I did have a few problems with this assignment. For certain processes messages were sent to, the sender/caller would have the max data type value as how many messages was sent to that process. This happened twice (for inet and pci) which threw off a few formatting areas. Another problem was on the last tab it would not print out all of the rows of the table. I assume this was due to some pointer issues as I created my own pointers as well as using the ones that were already being in use. In total, the pros outweighed the cons and I can fully explain what is going on from what we needed to modify and what those certain files did.

Ways this this was tested was just browsing through Minix, opening different files, and just waiting. Doing this if cycling through the F4, it is easy to see numbers changing or updating.

Files changed:

/usr/src/kernel/:

main.c: On Minix startup, Minix clears the process table. During this time we initialize the array of messages to 0 at all locations. Line 46.

proc.h: In proc.h I added my unsigned long array called messages_sent to keep track of the total number of messages sent. The full number of elements was the NR_PROCS + NR_TASKS. Line 45.

proc.c: Using pointers given in mini_send, I created the way to increment the array in given locations depending on what was the message sender/caller and where it was going using the caller_ptr and the dst_ptr. Line 292.

/usr/src/servers/is:

dmp.c: This is where I established my new F4 command in the hooks for my newly created function message_dmp. Removed priviledges_dmp that was using F4 already. Line 25.

proto.h: Created the prototype for message_dmp. Removed privileges_dmp. Line 13.

dmp_kernel.c: This is where the magic happens. The main section of the code created to allow this all to mesh together is located here in message_dmp. This function uses 4 pointers and 4 ints. From there, this function creates a table that allows “tabs” and “scrolling” depending on which processes need to be shown. This is also the location in which the data from the array gets sent. The comments explain then entire function throughly. Incorporated in also is some of the original programmers code. Using ideas from the programmer we are able to construct a nice looking table and a nice way of retrieving data and printing it to console. Line 334.