Eric Grounds
C435
2-5-2013
Homework 1 – F4 Dump Write-up

Running and Testing

To test the F4 message count dump, simply boot and login into Minix. Then press the F4 key. The name of the sending process will be displayed. Directly below that, the receiving process name(s) will be displayed along with the amount of messages that were sent to that particular process. The message counts and receiving processes are displayed using up to three columns. If a process hasn't sent any messages, its name will be displayed, but there will be no receiving process name(s) or message count(s) displayed below. Each time the F4 key is pressed, the next sending process along with its destination processes and the amount of messages sent will be displayed on the screen.

Changes/Observations/Assumptions/Correctness

To start, knowing how messages are sent between processes is necessary. The proc.c file located in /usr/src/kernel essentially contains everything needed to handle messaging. The particular function of interest is the mini_send function. This function allows a process to send a message to another process. The sending process can send a message to the destination process if the destination is blocked waiting for the message. When this is the case, the message is copied to it and the destination process is unblocked. When the destination isn't waiting at all, or if it's waiting for another source, the sending process is queued. To handle keeping track of the messages sent between processes, an array that is as large as the number of tasks plus the number of processes running will be used. This array is declared in the proc.h file located in /usr/src/kernel. The C programming language automatically initializes array elements with the value 0. So that does not have to be handled in any other files. The value of element i in this array will tell how many messages the process i has sent. The value of element i in the array will be incremented right after the message is copied to the destination process. To display the number of messages sent between processes, the privileges_dmp function, which is executed by pressing the F4 button will be replaced with the messagecount_dmp function. The first thing to do is replace the privileges_dmp function prototype with the messagecount_dmp prototype. This change was made in the proto.h file found in /usr/src/kernel. The next change was in dmp.c, located in /usr/src/servers/is. The mapping of the F4 key was changed from executing privileges_dmp to execute messagecount_dmp instead. The final change made was to replace the privileges_dmp implementation with the messagecount_dmp implementation. Some variables from the privileges_dmp function were reused. Some code was also reused such as the code for obtaining a fresh process table, as well as the code used to allow paging. A nested for loop was used to display the number of messages that were sent. Each process, whether it has sent messages or not, will be displayed on the screen and be identified by its name. Directly below it are the names of the receiving processes along with the number of messages sent to that process. In the case where a process has sent no messages, there will be no receiving processes or message counts displayed below.

This assignment was very interesting. It provided a great opportunity to dig deep down within the operating system and see how interprocess communication works in Minix. It was difficult at first to grasp everything that was going on with the message passing between processes. But the more and more the code was looked at, the more it started to make sense.

It is believed that the work that was carried out to complete this assignment is correct, assuming that message passing only occurs in the mini_send function. With that assumption, it makes logical sense to increment the variable keeping track of the messages sent by a particular process right after the message is copied to the destination process. This implementation should display all of the messages sent to other processes as long as they are still in the process table.