Dennis McWherter
9 September 2011
CS 241
dmcwhe2

In MP2, I implemented a simple shell. In implementing the shell, I had to create a separate process for commands which I did not implement (i.e. I delegated the work to the system() command). I had to create a new process in order to ensure that the binary I was calling would run within its own address space to guarantee there was no shared memory, etc. Furthermore, if I created a separate process to run the cd command, when the child process returns (i.e. "/bin/cd") there will have been no change on the parent process (i.e. shell) due to running in separate address spaces.

In the implementation of MP2, I ran into few bugs. However, bugs I encountered took deep thought to solve. For instance, when accepting input, I had to devise a way to appropriately clear the buffer if needed. With that in mind, I needed to find a way to determine if "stdin" still had information in it; if not, my method to clear the buffer would wait for the next input and the next input string would never be parsed. Also, in tokenizing my input, I found I had made a very naïve mistake in attempting to store pointers in the vector. That is, I was attempting to store the original pointer to a command directly which was repeatedly being overwritten by the input string. The remedy to this issue was to create a deep copy of this data.

Finally, the MP description could have been made more clear through a better description of the !N command. For instance, if you attempt to view your history (!#) and count the list, you would have to add one to re-execute the instruction. Though this can be considered a feature, it is somewhat counter-intuitive. That said, my original implementation did not add !# to the history (even though it does in the final solution).