Name: Benjamin Denison
Username: bdenison

Test cases

Case 1: All priorities the same.

I originally had trouble getting this one to work because of how priority worked. Setting each to a higher prio than main would block main from creating the other processes. Setting them to a lower priority would allow main to continue executing and it'd spawn too many shell processes. I just decided to get rid of the shell at the end altogether and set each process to a priority of 15. The test went as expected after that. The print statements were mixed up and they finished in a random order. I added a mutex at the end so reading the CPU usage at the end would be slightly less painful. Each process consumed about the same amount of CPU time, and the proportion was right for the number of processes.

Case 2: Processes created in ascending order of priority

After I got the first test working, this one went a lot more smoothly. I expected that despite being the last process created, the process with the highest priority would completely finish before all the others. This was correct. The times were also accurate.

Case 3: Executing a shell in the midst of it

For this I made the wrong assumption that the shell would block the other processes from running, but instead got reminded that the shell will let other processes run while it waits for input. It gets put into a waiting state, which allows other processes to run. So instead of the printing being stopped by the higher-priority shell, the shell just prints and begins waiting while the test processes spin.

Case 4: Starving a process

This one I just let main starve a process.

Case 5: Waiting

For this I tested both the behavior of main and how much time it consumed. I had it print a few lines and wait in a while loop while the other process ran. Once the process had finished, I had main print its CPU consumption. It printed that it was active for only a millisecond. I originally thought this was due to the prctxswbeg variable being reset every context switch, and main was only remembering the beginning time for the very last context switch, so I made prctxswbeg a field in the proctab. I now realize that is very wrong and it's the reason that it's a field in the proctab right now. I'll fix that later. Now I think the reason is just lack of precision.