Marshall Hairston

Sasha Jouravlev

Professor Skalak

Operating Systems

Project 6 Write-up

Overview: For project 6 we decided to implement a new function rather than work on fixing the broken issues in our code. The main issues with our project 5 was the functionality of kill in the shell, show processes in the shell, dir in the shell, and previously being able to add more to the shell. Once we changed which order we ran the main function in, the shell went back to normal functionality despite how many memory segments it ran through. Once we figured out this simple fix to the issue, we thought that we did not have enough bugs to warrant spending the time going over our code to fix the errors. Additionally we thought that we had a pretty clear understanding of the concepts, it was just a matter of making the code work. For these reasons we chose to implement the sleep function in the kernel and the function in the shell. We created a program called uprog4 to demonstrate the functionality of our sleep function while using the system call from our user library to do so.

Implementation: Starting to write the sleep function we had some issues wrapping our heads around how this would work without being able to call the wait function due to the limitation of our compiler. After a brief discussion with Professor Skalak, we determined the best course of action. Our sleep function takes in one parameter which determines how long the function should sleep before starting to run again. To do this we changed our proc.h to include a sleeping state and a sleep counter in order to track of the amount of time it has been asleep. We changed the state of the process to sleeping which takes it out of the ready queue and does not put it back in until the sleep counter reaches 0. In the handleInterruptTimer, we decrement the sleep counter if the state of the process is sleeping. Once the sleep counter is 0 we change the state to ready and put it back in the ready queue. Our sleep counter is originally set to 125 times the number of seconds entered in which we determined by trying it out multiple times to get the correct time.

Next we tried to implement a new shell command in the shell. We ran into many issues, originally trying to implement a function that kept track of previous commands using the up arrow, and adding the & to the end of execute to make the shell printout cleaner, but for both of these attempts our previous code was making it difficult to come up with a solution. Finally we settled on creating a help method which displays all of the commands that you can run inside the shell. With our extra time we decided to see if we could fix some previous errors in our shell such as dir, kill, and ps, but were only able to fix the dir command and the ps command for the most part. Ps now will only print out the shell but we are not sure why it will not continue onto the other processes. We believe that kill is being affected by the way we set up our handleInterruptTimer, making the fix much more complex than we originally believed.

Tour: In order to see our enhancement in action it is quite simple. Just run the program like you normally would and execute uprog4 to see the sleep function in action. There are two lines being printed out and it should be about 2 seconds between each one printing out. To see the help in action, once the shell loads just type help to receive a friendly set of instructions on the various things you can do. If you would like to see the working dir command just type dir.

Bugs and Limitations: As far as we are aware, there is no way to make the sleep function crash or the help call in the shell crash.