CSE 451: Lab 2 Section Notes
1/18/2018

Synchronization and Processes

Synchronization Primitives: xk provides two types of locks, spinlocks and sleeplocks.

What are the pros/cons of using one vs the other?

What mechanism does xk use to implement “condition variables”? (hint: look at parameters for `sleep()`/`wakeup()`)

Fork: creating another copy of the current process.

What needs to be copied into the newly created process?

Where will the new process start executing in user mode? (and where is this information stored?)

To differentiate the new processes from the old process, we call the new process a child of the parent old process. The return value of `fork` is different between the child and the parent. The parent will return the current process id and the child will return 0.

How can we alter the return value of the fork function to simulate the situation above?

CSE 451: Lab 2 Section Notes
1/18/2018

Synchronization and Processes

Exit, Wait:
`exit` and `wait` are two important synchronization methods in xk. `wait` waits until a child has exited and returns the process id of the child. `exit` exits the current process.

xk cannot immediately free all the kernel data structures associated with the currently running process because the physical machine keeps track of the current process' virtual address space.

How can the kernel clean up the data structures associated with the currently running process when the process calls `exit`?

Pipes: mechanism to pass data from one process to the other

How would we implement a pipe? Which kernel functions give us the space we need to store the transferred data?

Exec: running a program given the program name and arguments

How do we pass the arguments from the program calling `exec()` to the new program?