RTX P2

Things we have learned and mistakes we have made

- User and system processes **must use syscalls** instead of executing kernel level methods. When calling these methods, the processes are executing in thread mode. If they call the kernel methods directly, rather than via the syscall interface, a hard fault occurs when we try to switch to another process, since that expects us to be in handler mode.
- Interrupts occur in handler mode. Syscalls **should not** be used for kernel level methods and those methods should be called directly to avoid hard faults
- **Clear values** of memory blocks before using them. Without this, issues arise in occasions such as when using reusing elements with pointers. An element could potentially point back to itself if the next pointer is not cleared
- Our **heap blocks** contain header information accessible only from the kernel level. This is useful for message passing when storing the process source, process destination and message type in the header
- Characters can be outputted on the UART either through **polling** or **via interrupts**
- Interrupts are triggered by setting **register values**. For example, we set the IER_RBR, IER_THRE, and IER_RLS bits to use our UART i-process
- Interrupt handler methods should **never** be preempted
- All kernel code executed should be **non-blocking** within interrupts
- Every process should have a **mailbox** to store all of its received messages. This includes the UART i-process for messages sent to it by the CRT process
- A **hard fault handler** is extremely useful for debugging purposes
- When outputting a carriage return character (‘\r’) we should also output a **newline character** (‘\n’) for better user interaction in the UART
- **Interprocess communication** is accomplished through message passing
- **System processes** exist to arbitrate between user processes and kernel level code. We use them to accomplish tasks such as printing