%INTRO

Interrupts are the performance efficient alternative to polling for any operating system level process or user-land program. They offer programmers the ability to notify a CPU at anytime indicating a program/kernel critical change that needs attention and quantum time from the CPU. synchronization within the kernel are essentially an abstraction of interrupts on the kernel and user land levels. Synchronizing allows multiple concurrent processes to share data between one another without potentially corrupting it by altering it mid read or write. These abstractions of interrupts are implemented in the form of mutexes, semaphores, spin locks, and many others. These implementations however vary slightly between operating system to operating system, either by varying syntax with the underlying semantics being relatively the same. With a concise definition of interrupts as well as synchronization have been established, it will help aid in the development of the comparisons that will be made between Windows, FreeBsd, and Linux.

%FREEBSD

In FreeBSD interrupts are handled by two different methods system calls, or the more traditional trap handlers. Trap handlers normally occur because of "unintentional errors, such as division by zero or indirection through an invalid pointer"\cite{freeBsdBook}. When a trap handler catches an interrupt it starts by saving the current state of the erroring process, determines interrupt information, checks higher-priority processes if they are affected, and exits safely. **However** when interrupts are created from system calls instead of kernel level errors in the system they can block for data causing mutual exclusion within the kernel. In the FreeBSD Kernel software interrupts have the highest priority, and are typically used for network devices because of their random retrieval of packets\cite{freeBsdBook}, while hardware level interrupts are serviced after the fact. When any interrupt is serviced they are referred to as heavyweights, because a full service switch is require by the processor in order to begin assigning quantums\cite{83Genera16:online}.

However there are options to the programmer for quicker lighter weight interrupts using the INTR\\_FAST flag, this flag was mainly used in earlier version of the FreeBSD kernel, and generally unused because its requires additional used memory and processor time\cite{83Genera16:online}. Synchronization in FreeBSD unlike windows contains many of the same naming schemes as its Linux counterpart, however many of the mechanics it abstracts are very different. For example in user space programs, programmers have the option to use mutexes to ensure only one thread at a time can access data. Within FreeBSD there are rules set to prevent deadlocking of processes, to achieve this threads with immediate access to multiple pools of data may only lock one mutex at a time\cite{43Contex5:online}. This fail safe however is not implemented by the kernel in Linux, therefore requiring elegant solutions like those in the dining philosophers problem.

%WINDOWS

Interrupts in windows are typically handled by a trap dispatching mechanism. This allows for a driver to force a processor to temporarily suspend what it is currently doing in favor of a newly executing thread. These trap handlers help the processor distinguish if a given interrupt can be run synchronously vs asynchronously, as well as determine the type of interrupt (hardware or software related). Once a interrupt is caught by the trap handler in windows it is sent to be dispatched by Interrupt service routines much like Linux. Additionally interrupts are indexed to the IDT (interrupt dispatch table), much like Linux's system call table it allows programs in user space to create various calls to a function by its interrupt number to retrieve data from the kernel or run various kernel routines. **However** the difference between the systems is its limitation of two hundred and fifty six entries per processor (logical core) for a windows IDT\cite{windowsbookpt1}. Many interrupts in the Windows system are reliant on synchronization, because they may edit or read data that can be altered by another process at the same time.

This works differently that High-IRQL's that they don't interrupt the processors in any way, and are restricted to user-level programs only\cite{windowsbookpt1}.

%CONCLUSION

Implementation of interrupts and synchronization between Windows, FreeBSD, and Linux generally use similar constructs. With the only major difference between them being implementation of varying locks, and method of creating interrupts themselves. For Windows its biggest difference is the use of IRQs and how trap handlers relay information about a given interrupt. For FreeBSD it is mainly trap handler memory and CPU usages, otherwise in comparison to Linux they show similar traits.\\

Software requests, though less common than hardware interrupts, have an important place on the Windows operating system. Types of tasks that that are facilitated by software interrupts are thread dispatching, asynchronous I/O, and timer expiration, among others. One specific type of software interrupt is called a Dispatch or Deferred Procedure Call, or DPC. These types of software interrupts are enacted when a thread is sleeping or has terminated. The interrupt request is send in order to facilitate a context switch, as discussed in the CPU scheduling section \cite{win}.