In addition to the different levels of OS involvement in threads, different OSes (and different thread implementations for a particular OS) allocate CPU cycles to threads in different ways.

Cooperative multitasking systems have running threads give up control if one of two things happen. If a thread calls a yield function, it gives up control. It also gives up control if the thread does something that would cause it to block, such as perform I/O. In a cooperative multitasking implementation, one thread can starve all the others for CPU time if it so chooses.

Preemptive multitasking systems interrupt threads at regular intervals while the system decides which thread should run next. In a preemptive multitasking system, one thread usually won't monopolize the CPU.

On some systems, there can be cooperative and preemptive threads running simultaneously. (Threads running with realtime priorities often behave cooperatively, for example, while threads running at normal priorities behave preemptively.)

51.5 What kind of threads are perl threads?

If you have experience with other thread implementations, you might find that things aren't quite what you expect. It's very important to remember when dealing with Perl threads that Perl Threads Are Not X Threads, for all values of X. They aren't POSIX threads, or DecThreads, or Java's Green threads, or Win32 threads. There are similarities, and the broad concepts are the same, but if you start looking for implementation details you're going to be either disappointed or confused. Possibly both.

This is not to say that Perl threads are completely different from everything that's ever come before–they're not. Perl's threading model owes a lot to other thread models, especially POSIX. Just as Perl is not C, though, Perl threads are not POSIX threads. So if you find yourself looking for mutexes, or thread priorities, it's time to step back a bit and think about what you want to do and how Perl can do it.

51.6 Threadsafe Modules

The addition of threads has changed Perl's internals substantially. There are implications for people who write modules–especially modules with XS code or external libraries. While most modules won't encounter any problems, modules that aren't explicitly tagged as thread-safe should be tested before being used in production code.

Not all modules that you might use are thread-safe, and you should always assume a module is unsafe unless the documentation says otherwise. This includes modules that are distributed as part of the core. Threads are a beta feature, and even some of the standard modules aren't thread-safe.

If you're using a module that's not thread-safe for some reason, you can protect yourself by using semaphores and lots of programming discipline to control access to the module. Semaphores are covered later in the article. Perl Threads Are Different

51.7 Thread Basics

The core Thread module provides the basic functions you need to write threaded programs. In the following sections we'll cover the basics, showing you what you need to do to create a threaded program. After that, we'll go over some of the features of the Thread module that make threaded programming easier.

51.7.1 Basic Thread Support

Thread support is a Perl compile-time option-it's something that's turned on or off when Perl is built at your site, rather than when your programs are compiled. If your Perl wasn't compiled with thread support enabled, then any attempt to use threads will fail.

Remember that the threading support in 5.005 is in beta release, and should be treated as such. You should expect that it may not function entirely properly, and the thread interface may well change some before it is a fully supported,