Protothread

A **protothread** is a low-overhead mechanism for concurrent programming.

Protothreads function as stackless, lightweight threads, or coroutines, providing a blocking context cheaply using minimal memory per protothread (on the order of single bytes).

Protothreads are used to accomplish a non-preempted form of concurrency known as cooperative multitasking and, therefore, do not incur context switch when yielding to another thread. Within a protothread, yielding is accomplished by utilizing Duff's device within a thread's function and an external variable used in within the switch statement. This allows jumping (resuming) from a yield upon another function call. In order to block threads, these yields may be guarded by a conditional so that successive calls to the same function will yield unless the guard conditional is true.

A feature of protothreads relative to other implementations of coroutines, or proper threads, is that they are stackless. This has advantages and disadvantages. A disadvantage is that local variables within the protothread cannot be trusted to have retained their values across a yield to another context. They must retain their state through the use of static or external, often global, variables.[1] An advantage is that they are very lightweight and therefore useful on severely memory constrained systems like small microcontrollers where other solutions are impractical or less desirable.

Tom Duff, of Duff's device fame, had this to say about the shortcomings of the method: "a similar trick for interrupt-driven state machines that is too horrible to go into. [...] I never thought it was an adequate general-purpose coroutine implementation because it's not easy to have multiple simultaneous activations of a coroutine and it's not possible using this method to have coroutines give up control anywhere but in their top-level routine. A simple assembly-language stack-switching library lets you do both of those."[2]

The protothread concept was developed by Adam Dunkels and Oliver Schmidt,[3] based on prior work by Simon Tatham[4] and Tom Duff.[2]

See also

- Coroutine
- Microthread

References

1. A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, *Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems* (http://doi.acm.org/10.1145/1182807.1182811), Proc. ACM SenSys, Boulder, CO, USA, Nov 2006. (PDF (http://dunkels.com/adam/dunkels06protothreads.pdf), Presentation slides (http://dunkels.com/adam/dunkels06protothreads.ppt))
2. "Brainwagon » Coroutines in C" (https://brainwagon.org/2005/03/05/coroutines-in-c/#comment-1878).
3. Adam Dunkels. "Protothreads - Lightweight, Stackless Threads in C" (http://dunkels.com/adam/pt/). *Dunkels.com*. Retrieved April 21, 2017.
4. "Coroutines in C" (https://www.chiark.greenend.org.uk/~sgtatham/coroutines.html).

External links

- Protothread library in C (http://dunkels.com/adam/pt/) used by Contiki
- Using Protothreads for Sensor Node Programming (http://dunkels.com/adam/dunkels05using.pdf) A PDF with some in-depth discussion about the use of Protothreads
- Protothread library (https://code.google.com/p/protothread/) requiring GCC, includes a thread scheduler modeled on the UNIX kernel (a simplified form of POSIX condition variables). This source base also includes a version that supports multiple CPU cores (processors).
- eigenclass.org performed a comparison of protothreads against POSIX threads, and found a 400x improvement in speed [1] (https://web.archive.org/web/20100725045509/http://eigenclass.org/hiki/threadring-with-protothreads) Retrieved from Archive.org Oct 2014
- C++ Protothread (https://web.archive.org/web/20140604192611/http://www.manishshakya.com.np/index.php/page/code)
- Symbian C++ Protothread Implementation (https://web.archive.org/web/20160419190507/http://www.mind-flip.com/theBlog/2007/12/12/protothreads-removing-the-state-from-state-machine/)