Running Pthreads Under Gforth

John E. Harbold – November 17, 2018

We present the method to run pthreads under Gforth in the Linux operating system.

Summary

Threading allow writing code to do a specific job without having to “juggle”, (e.g. cooperate), when one job needs servicing among other. Pthreads is an implementation of threading that runs under Linux. It allows the creation, control, and, ultimately the destruction of a thread. The control includes prioritizing threads, protecting thread access to resources, via mutexes, condition variables, read/write locks. Having such control allows the user to concentrate on what has to be done, instead of having to know when to switch between pieces of code to execute.

Gforth/C

Gforth threads have own stacks, via thread local storage (TLS).

Pthreads has its own stack memory distinct from Gforth.

Gforth employs (TLS), via GCC, to implement the above.

Gforth employs dynamic linking to access pthreads library, among others.

Gforth threads have initialization code before actual thread words.

Pthreads Books

Multithreaded Programming Guide,

Sun Microsystems, now Oracle

813-5137.pdf

Pthreads Programming

O'Reilly Media

Various Pthreads tutorials on the Web.

Pthreads

- Threads
- Thread Attributes
- Mutexes
- Mutex Attributes
- Semaphores
- Conditional Variables
- Conditional Variable Attributes
- R/W Locks
- R/W Lock Attributes
- Barriers
- Barrier Attributes
- Keys

Pthread Attributes

- Stack Address, Size and Guard Size
- Detach State
- CPU Affinity
- Schedule Priority
- Schedule Policy
- Resource Scope

Mutex Attributes

- Process Sharing
- Priority Ceiling
- Protocol
- Robust
- Type

Conditional Variable Attribute

Process sharing

Clock

Read/Write Lock Attribute

Process sharing

Kind

Barrier Attribute

Process sharing

Gforth/C Interface

libcc.fs – Interfaces Forth code to C code.

A small snippet of C code to relate what Forth will call to the actual call of the desired.

A Forth/C declaration that shows what Forth name will use and what the resulting C name that will be used. Also, what the Forth stack before and after the Forth name is executed.

The C code is compiled and linked, using libtool, into a shared library. Using dynamic loading, a compiled Forth word will cause this library to be opened and the resulting compiled C code executed.

Gforth/C Interface Example

```
c-library pthread

\c static inline int thread_mutex_size (void)
\c {
\c   return sizeof(pthread_mutex_t);
\c }
\c static inline int thread_mutexattr_init (Cell mutexattrPtr)
\c {
\c   return pthread_mutexattr_init ((pthread_mutexattr_t *) mutexattrPtr);
\c }

c-function pthread_mutexattr_size thread_mutexattr_size void -- n ( -- n )
c-function pthread_mutexattr_init thread_mutexattr_init n -- n ( addr -- r )
end-c-library
```

Gforth/C Interface Example

```
\ Create a variable of type pthread_mutex_t
: pthreadMutexAttrVariable ( "name" -- f ) pthread_mutexattr_size buffer: ;
pthreadMutexAttrVariable mutexAttr

mutexAttr pthread_mutexattr_init
dup 0= [IF]
    ." Mutex Attribute created!" drop CR
[ELSE]
    ." Mutex Attribute failed:  " mkstrerror type CR
[THEN]
```

Gforth/C Interface Example

```
pthread_mutexattr_t mutexAttr;

if(pthread_mutexattr_init(&mutexAttr) == 0) {

  printf("Mutex Attribute created!\n");

} else {

  printf("Mutex Attribute failed:  %s\n", strerror(errno));

}
```

Example

Multiple threads.

Thread synchronization via barriers.

Output managed through a mutex.

Future Work

More examples using the more obscure aspects of Pthreads.

Inter-process communication (IPC) objects such as mailbox, queue, events, etc.

Abandoning the eCOS porting effort due to lack of TLS.

Switching to RTEMS which does support both dynamic loading and TLS.

Future Work

* **Port GDB to Gforth**
  * Breakpoints, even on a thread basis
  * Single-stepping, over and into Forth words
  * Thread Information
  * Debug primitives in assembly language
* **GTK based GUI Debug Interface**
  * Display Forth Internals
    * Stacks
    * Decompile words
    * Show next word to be executed

Question?

Thank you!