Subject: Re: Stop a thread
Posted by mr_ped on Fri, 06 Mar 2009 07:34:46 GMT
View Forum Message <> Reply to Message

darthspawn wrote on Thu, 05 March 2009 17:12"volatile atomic"
volatile = C/C++ keyword, it tells compiler the value can change from "outside" in memory while the current code is running. With general variables like in:

```
bool run = true;
while ( run ) {
  run = false;   //change "inside" code
}
```

The compiler can optimize the whole code to read the content of variable from memory into register once, and then work only with register and ignore the memory.
So if the value would change only in the memory outside of the current code, the final code will not catch it and would run in infinite loop.
With "volatile" the compiler knows the value has to be re-read from memory all the time to keep it fresh and allow modifications from "outside" to be catch ASAP.

Atomic is defined as "long" in Mt.h, and that's because reading/writing of long at x86 is atomic operation, i.e. you either did read/write whole long, or you didn't at all, you can't get interrupted in between. Actually there's special machine code instruction to compare+exchange 8/16/32b value atomically (CMPXCHG and CMPXCHG8B), which allows you to write some new value into memory only if the old value didn't change yet (very handy in MT where more then one thread access and change the same memory variable and need to cooperate in some way, like doing auto-increment of IDs for example).