

---

Subject: Re: Stopping ReadStdIn() function  
Posted by [Oblivion](#) on Mon, 17 May 2021 08:46:45 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Hello Xemuth,

You are insalling a SIGINT hook and from that point on, it is your responsibility to send a SIGINT to the foreground process:

```
std::signal(SIGINT,static_cast<__p_sig_fn_t>([](int s)->void{serverPtr->StopServer();}));
```

One way might be to set the SIGINT hook to default after you've clean up the application:

```
static void SecureStop(int)
{
    // Cleanup the app here...

    signal(SIGINT, SIG_DFL); // unsintall your hook.
    raise(SIGINT);          // Let OS handle it.
}
```

This way, you shouldn't even need "done" flag...

Example:

```
static std::atomic<bool> done(false);
static void SecureStop(int)
{
    // Do app cleanup here..
    done = true;
    signal(SIGINT, SIG_DFL); // unsintall the hook.
    raise(SIGINT);
}
```

```
CONSOLE_APP_MAIN
{
    StdLogSetup(LOG_COUT|LOG_FILE);
```

```

auto worker = Async([] { // Simulate the signal request...
    int timeout = 5000;
    int t = msec();
    while(msecs(t) < timeout) {
        if(CoWork::IsCanceled())
            return;
        Sleep(10);
    }
    if(!done) {
        RLOG("rasigin SIGINT with custom hook...");
        raise(SIGINT);
    }

});
signal(SIGINT, &SecureStop);
String s;
while(s != "exit") {
    s = ReadStdIn();
}
worker.Cancel();

}

```

Please note that this may work, because the code you posted is very simple. This isn't a good way to handle it if it get complex, as you need to make your cleanup-code thread-safe.

Best regards,  
Oblivion

---