Enduro/X Core - Feature #281

Feature # 328 (Closed): Enduro/X Version 6.0 master task

System V message queue support

02/07/2018 08:10 PM - Madars

Status:	Closed	**Start date:**	02/07/2018
Priority:	High (Code 3)	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

- Each server have two treads, one for main command receiver, with msgrcv,
- Other thread will wait for admin commands on own queue
- when admin receives shutdown and it knows that main thread is waiting on Q (synced via global flags), it should set some globals that shudown is requested, and perfrom pthread_kill on main thread.
- when there is incoming ping, the admin thread checks the main thread, if it is doing msgrcv, the reply to ping ok, if not , then save in globals that ping reply is pending and let main thread at reach of next msgrcv reply on ping
- we shall support rqaddr, meaning that server either listens on single queue or it's own queue. So unix mode on shared mem should be enabled by default (allowing servers run on their own queues or on shared queue)
- ndrxd should act correspondingly - if shared, then remove Q only if last servant is off
- there should be global read/write locks protected dictionary of System V queues and Posix queue names, so that everything else does not break on Enduro/X

History

#1 - 02/14/2018 04:10 PM - Madars

For timeout handling, we shall also use pthread_kill() by background thread. This means that client process will require background thread too. As it might perform timed calls (like tpcall with timeout). As there could be many user threads, the timeout thread shall perform scheduler for them all.

#2 - 03/23/2018 08:50 AM - Madars

For socket processing (fd add extension) I guess we need another thread. So that if message comes in, we put big lock, so that other queues hold on their processing. While we get something. And basically the main thread while doing polling basically should wait for message to appear in some kind of buffer. Locks would be faster than pipes.

#3 - 09/14/2018 07:38 AM - Madars

Probably we need following tables

1. Mapping for string names -> msgqid, with rwlocks segmented... Also needs to think by which param it will be indexed. probably by q string name

The string names could include RQ address path, or unique RQ address path per binary. if RQA not used. Then binaries needs to register RQADDR strings in the service shared memory entries. Length is 1...30

So Enduro/X service shared mem block, pointers to either one or multiple RQADDRs. Either binaries share RQA or each generates own RQA. The RQA strings are generic Q strings mapped to MSGID.

#4 - 09/14/2018 10:10 AM - Madars

Regarding message dispatching.

So we get 3x threads and one lock W and one Job queue. In job queue messages from admin Q or poll() thread are put. Also we have

Main thread does following:

1. init
1. START:
1. lock S
1. goes to msgrcv waiting for service dispatching.
1. unlock S
1. if any message received or msgrcv was interrupted -> lock W
1. if message was received put that in job Q.
1. process each message in Job Q.
1. release lock W
1.. continue START

thread 2, admin Q:
2. init
2. START: goes into msgrcv
2. if message received, we lock W
2. put message in job Q
2. unlock W
2. while trylock S and it is locked already, do pthread_kill to main thread so that w guarantee that we interrupt the msgrcv of 1., reschedule our thread, and repeat while we get the lock... If we get the lock then continue to work to next step
2. continue START

thread 3, poll & timeouts:
3. init
3. START: goes into poll() with timed option in case if periodic callback is set
3. if message received, we lock W
3. put message in job Q
3. unlock W
3. while trylock S and it is locked already, do pthread_kill to main thread so that w guarantee that we interrupt the msgrcv of 1., reschedule our thread, and repeat while we get the lock... If we get the lock then continue to work to next step
3. continue START

#5 - 09/14/2018 10:13 AM - Madars

- Parent task set to #328

#6 - 09/14/2018 11:21 AM - Madars

- File sisi.c added

#7 - 09/14/2018 01:32 PM - Madars

For worker threads if we need to generate TPETIME when doing tpcall, then we have to assign our thread id and timeout parameter to thread 3. so that it can interrupt our msgrcv when timeout is reached. Also this must be deliver to thread 3. in some atomic fashion, also wait for mutex to clear, send kills, till second mutex becomes free.

#8 - 09/20/2018 07:30 AM - Madars

https://www.softprayog.in/programming/interprocess-communication-using-system-v-message-queues-in-linux

#9 - 09/20/2018 03:23 PM - Madars

RW sempahore: https://cboard.cprogramming.com/c-programming/95848-reader-writer-locks-shared-mem-using-semaphores.html

#10 - 09/21/2018 08:24 AM - Madars

- Priority changed from Normal (Code 4) to High (Code 3)

#11 - 09/24/2018 06:18 PM - Madars

Tuxedo probably for timeout processing uses BBL to send timeout messages to clients when they do timeout. Enduro/X uses per process internal mechanisms for timeout processing and does not relay on BBL... thus Enduro/X is able to crash survival.

#12 - 10/09/2018 10:48 AM - Madars

Registered issue for Ubuntu 18.04, https://bugs.launchpad.net/ubuntu/+source/gcc-defaults/+bug/1796858 - Building process fails with hidden symbol `pthread_atfork'

#13 - 11/18/2018 04:54 PM - Madars

- Status changed from New to Resolved

#14 - 11/18/2018 04:54 PM - Madars

- Status changed from Resolved to Closed

#15 - 12/03/2018 07:10 AM - Madars

- % Done changed from 0 to 100

Files

sisi.c	1.73 KB	09/14/2018	Madars