Enduro/X Core - Feature #113

Add unix mode with assuming that mqd_t is file descirptor

04/04/2017 02:13 PM - Madars

Status:	Closed	**Start date:**	04/04/2017
Priority:	Normal (Code 4)	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

We shall support the configuration when we run in unix mode, but assuming that mqd_t is file descriptor, thus we will be able to use direct poll() call on queues. Thus heavily increase the system throughput...

Needs research with unixes support this.
Linux is doing this for sure, but how about others?

1) It is freebsd: https://www.freebsd.org/cgi/man.cgi?mq_open

> NOTES
> FreeBSD implements message queue based on file descriptor. The descriptor is inherited by child after fork(2). The descriptor is closed in a new image after exec(3). The select(2) and kevent(2) system calls are supported for message queue descriptor.

2) Needs to check AIX & HP-UX

3) Solaris seems not to use this approach

History

#1 - 04/04/2017 02:14 PM - Madars

- Description updated

#2 - 04/13/2017 12:06 PM - Madars

Tests:

file q.c:

```
==================
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/epoll.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <errno.h>
#include <mqueue.h>

#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)

#define usageErr(msg, progName) \
                    do { fprintf(stderr, "Usage: "); \
                        fprintf(stderr, msg, progName); \
                        exit(EXIT_FAILURE); } while (0)

#ifndef EPOLLEXCLUSIVE
#define EPOLLEXCLUSIVE (1 << 28)
#endif
```

```
#define MAX_SIZE 10

int
main (int argc, char *argv[])
{
  int epfd, nready;
  struct epoll_event ev, rev;
  mqd_t fd;
  struct mq_attr attr;
  char buffer[MAX_SIZE + 1];
  int cnum;

  /* initialize the queue attributes */
  attr.mq_flags = 0;
  attr.mq_maxmsg = 5;
  attr.mq_msgsize = MAX_SIZE;
  attr.mq_curmsgs = 0;

  /* cleanup for multiple runs... */
  mq_unlink ("/TESTQ");

  /* create the message queue */
  fd =
    mq_open ("/TESTQ", O_CREAT | O_RDWR | O_NONBLOCK, S_IWUSR | S_IRUSR,
        &attr);
  if (fd == -1)
    errExit ("open");

  for (cnum = 0; cnum < 3; cnum++)
    {
      switch (fork ())
    {
    case -1:
      errExit ("fork");

    case 0:         /* Child */
      epfd = epoll_create (2);
      if (epfd == -1)
        errExit ("epoll_create");

      ev.events = EPOLLIN | EPOLLEXCLUSIVE;
      if (epoll_ctl (epfd, EPOLL_CTL_ADD, fd, &ev) == -1)
        errExit ("epoll_ctl");

      printf ("About to wait...\n");
      nready = epoll_wait (epfd, &rev, 1, -1);
      if (nready == -1)
        errExit ("epoll-wait");

      printf ("Child %d: epoll_wait() returned %d\n", cnum, nready);
      exit (EXIT_SUCCESS);

    default:
      break;
    }
    }
  sleep (1);
  /* send a msq to Q */
  memset (buffer, 0, MAX_SIZE);
  if (0 > mq_send (fd, buffer, MAX_SIZE, 0))
    errExit ("mq_send");
  printf ("msg sent ok...\n");

  wait (NULL);
  wait (NULL);
  wait (NULL);

  exit (EXIT_SUCCESS);
}
==================

$ gcc q.c -lrt
$ ./a.out
About to wait...
```

```
About to wait...
About to wait...
msg sent ok...
Child 2: epoll_wait() returned 1
^C
$
```

#3 - 04/13/2017 12:07 PM - Madars

Platform checks could be done in CMakeLists.txt, with try_compile+try_run

#4 - 04/23/2017 09:48 AM - Madars

Seems like freebsd only supports select or kqueue event notifications. Poll seems to corrupt the descriptors or queues...

#5 - 04/23/2017 11:28 AM - Madars

Seems like for BSD it is pointer to FD.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <mqueue.h>
#include <errno.h>
#include <fcntl.h>

int main() {
  struct mq_attr attrs;
  attrs.mq_maxmsg = 10;
  attrs.mq_msgsize = sizeof(int);

  const char name[] = "/test-queue";

  mqd_t q = mq_open(name, O_CREAT | O_RDWR, 0600, &attrs);
  mqd_t q1 = mq_open(name, O_CREAT | O_RDWR, 0600, &attrs);
  mqd_t q2 = mq_open(name, O_CREAT | O_RDWR, 0600, &attrs);
  mqd_t q3 = mq_open(name, O_CREAT | O_RDWR, 0600, &attrs);
  mqd_t q4 = mq_open(name, O_CREAT | O_RDWR, 0600, &attrs);
  if (q == (mqd_t)-1) {
    perror("mq_open");
    exit(EXIT_FAILURE);
  }
 void *x = q4;

 int *y = (int *)x;

 fprintf(stderr, "descr=%d\n", *y);
  mq_unlink(name); // it doesn't matter if I do this at the end or not

  if (fork()) {
    int msg = 666;
    if (mq_send(q, (const char *)&msg, sizeof(msg), 1)) {
      perror("mq_send");
      exit(EXIT_FAILURE);
    }
  } else {
    int msg;
    unsigned priority;
    if (mq_receive(q, (char *)&msg, sizeof(msg), &priority) == -1) {
      perror("mq_receive");
      exit(EXIT_FAILURE);
    }
    printf("%d\n", msg);
  }
```

```
  mq_close(q);

  return 0;
}
```

```
cc -std=c99 -Wall -o mqtest test.c -lrt

$ ./mqtest
descr=7
```

#6 - 05/03/2017 11:13 AM - Madars

- Status changed from New to Resolved

#7 - 05/03/2017 11:13 AM - Madars

- Status changed from Resolved to Closed