Subject: Re: [PATCH v3 08/10] IPC: message queue copy feature introduced
Posted by Manfred Spraul on Sun, 12 Aug 2012 09:48:18 GMT
View Forum Message <> Reply to Message

Hi Stanislav,

2012/8/11 Stanislav Kinsbursky <skinsbursky@parallels.com>:

>> a) What about the simpler approach:
>> - if MSG_COPY is set, then @mtype is interpreted as the number of the
>> message that should be copied.
>> If there are less than @mtype messages, then -ENOMSG is returned.
>
>
> Hi, Manfred.
> Your approach is simplier, but makes the call less generic and adds
> limitations.
> I.e. sys_msgrcv() allows you to receive message by type. And from my pow
> this logic have to be preserved - you can specify type and then copy all the
> messages of specified type.
>

Your implementation adds the ability to select a message for MSG_COPY by id.
But I think the price is way too high:
a) added complexity
b) I didn't notice it immediately:
The implementation means that MSG_COPY cannot be used at all by
multiple processes:

task 1: msgrcv(id,buf,len,0,MSG_COPY).
task 2: msgrcv(id,buf,len,0,MSG_COPY).

It is unpredictable if a task receives the first or the 2nd message
from the queue.

task 1: int msgnr=0;
         msgctl(id,MSG_SET_COPY,&msgnr)
         msgrcv(id,buf,len,0,MSG_COPY).
task 2: int msgnr=0;
         msgctl(id,MSG_SET_COPY,&msgnr)
         msgrcv(id,buf,len,0,MSG_COPY).

Doesn't work either, it's a race condition.

>
>> b) I do not understand the purpose of the decrease of msq->q_copy_cnt:
>> Do you want to handle normal msgrcv() calls in parallel with
>> msgrcv(|MSG_COPY) calls?

>
>
> Actually, I'm not going to copy a message from a queue, when somebody is
> reading from it. But better to handle this case by decreasing
> msq->q_copy_cnt, because otherwise this counter becomes invalid in case of
> somebody is reading from queue. And this logic is similar to new "peek"
> logic for sockets (introduced in 3.4 or 3.5).
> But I understand, that in case of queue with messages with different types
> this approach works only if mtype is not specified for copy operation.
> Otherwise result is unpredictable.
>
a) If the result is unpredictable when mtype is used, does it make
   sense to implement msgcrl(id,buf,len,id=<x>,MSG_COPY)?

b) The result is also unpredictable if mtype is used for the "normal"
   msgrcv() (i.e. without MSG_COPY) call.
>
>> I don't think that this will work:
>> What if msq->q_copy_cnt is 1 and and msgrcv() call receives the 20th
>> message in the queue?
>
>
> By "receives" you mean "copied"? If so, then it can happen only if mtype was
> specified. And this logic is a part of current implementation.
>

I was thinking about the following case:

```
task 1: /* copy all entries */
        errno=0;
        for (i=0;errno<>0;i++)
              msgrcv(msgid,buf[i],buflen,0,MSG_COPY);

task 2: /* receive a message with a given ID */
        msgrcv(msqid,buf,buflen,123,0);
```

Now suppose that task 1 has read 5 messages from the queue and then
task 2 tries to receive the message with the id=123. Suppose this is
the 20th message in the queue.

Result: task 1 will copy the message 5 twice.

I would keep it simple - unless there is a clear use case where "peek
by id" is useful.

Or - since MSG_COPY is linux specific anyway:
What about storing the number of the message that should be returned in *msgp?
Store it as "int64", just to avoid any 32/64 bit issues.

--
Manfred