Subject: Re: [RFC][PATCH][cryo] Save/restore state of unnamed pipes Posted by Sukadev Bhattiprolu on Wed, 18 Jun 2008 21:56:12 GMT View Forum Message <> Reply to Message

Matt Helsley [matthltc@us.ibm.com] wrote: > So let me rephrase the problem. > > Suppose the checkpointed application was using fds in following > "orig-fd-set" > > { [0..10], 18, 27 } > > where 18 and 27 are part of a pipe. For simplicity lets assume that > 18 is the read-side-fd. so orig pipefd[0] == 18and orig\_pipefd[1] == 27 > We checkpointed this application and are now trying to restart it. > > In the restarted application, we would call > > dup2(fd1, fd2), > > where 'fd1' is some new, random fd and 'fd2' is an fd in 'orig-fd-set' Even if they were truly random, this does not preclude fd1 from having the same value as an fd in the remaining orig-fd-set -- such as one of the two we're about to try and restart with pipe().

I agree. fd1 could be an hither-to-unseen fd from the 'orig-fd-set'.

> (say fd2 = 18).
fd1 = restarted\_pipefd[0]
fd2 = restarted\_pipefd[1]
In my example fd1 == 27 and fd2 == 18
> IIUC, there is a risk here of 'fd2' being closed accidentally while
> it is in use.
Yes, that's the risk.
> But, the only way I can see 'fd2' being in use in the restarted process

> But, the only way I can see 'fd2' being in use in the restarted process
 > is if \_cryo\_ opened some file \_during\_ restart and did not close. I ran

Both file descriptors returned from pipe() are in use during restart and closing one of them would not be proper. Cryo hasn't "forgotten" to close one of them -- cryo needs to dup2() both of them to their "destination" fds. But if they have been swapped or if just one is the "destination" of the other then you could end up with a broken pipe.

Ok I see what you are saying.

The assumption I have is that we would process the fds from 'orig-fd-set' in ascending order. Its good to confirm that assumption now :-)

proc\_readfd\_common() seems to return the fds in ascending order (so readdir() of "/proc/pid/fd/" would get them in ascending order - no ?)

If we process 'orig-fd-set' in order and suppose we create the pipe for the smaller of the two fds (could be the write-side). Then the other side of the pipe would either not collide with an existing fd or that fd would not be in the 'orig-fd-set' (in the latter case it would be safe for dup2() to close).

> into this early on with the randomize\_va\_space file (which was easily> fixed).

This logic only works if cryo only has one new fd at a time. However that's not possible with pipe(). Or socketpair(). In those cases one of the two new fds could be the "destination" fd for the other. In that case dup2() will kindly close it for you and break your new pipe/socketpair! :)

That's why I asked if POSIX guarantees the read side file descriptor is always less than the write side. If it does then the numbers can't be swapped and maybe using your assumption that we don't have any other fds accidentally left open ensures dup2() will be safe.

I don't think POSIX guarantees, but will double check.

> Would cryo need to keep one or more temporary/debug files open in the > restarted process (i.e files that are not in the 'orig-fd-set').

There's no need to keep temporary/debug files open that I can see. Just a need to be careful when more than one new file descriptor has been created before doing a dup2().

> If cryo does, then maybe it could open such files:

> - after clone() (so files are not open in restarted process), or >

> - find the last\_fd used and dup2() to that fd, leaving the

- > 'orig-fd-set' all open/available for restarted process
- >

> For debug, before each 'dup2(fd1, fd2)' we could 'fstat(fd2, &buf)'

> to ensure 'fd2' is not in use and error out if it is.

fstat() could certainly be useful for debugging dup2(). However it still doesn't nicely show us whether there are any fds we've leaked that we forgot about unless we fstat() all possible fds and then compare the set of existing fds to the orig-fd-set.

Yes, was suggesting fstat() only to detect collisions, but yes, to detect leaks, we have to do more.

Containers mailing list Containers@lists.linux-foundation.org https://lists.linux-foundation.org/mailman/listinfo/containers

