Subject: SIGPIPE problem
Posted by zsolt on Wed, 26 May 2021 03:19:18 GMT
View Forum Message <> Reply to Message

I'm writing an HTTP App server currently and I'm using some libraries, such as libpq.
My problem is, that some of the libraries are switching SIGPIPE signal handler on and off, and I have no control over that.
So when a client disconnects while my server is sending the HTTP response, sometimes it gets a SIGPIPE and my app stops at that point in debugger.
I'm running tons of unit tests so this is not very convenient, as some tests are doing this.

I fixed this by changing the flags argument of send() from 0 to MSG_NOSIGNAL in TcpSocket::RawSend() method.

Do you have any better idea?

```
 int TcpSocket::RawSend(const void *buf, int amount)
 {
+#ifdef PLATFORM_POSIX
+ int res = send(socket, (const char *)buf, amount, MSG_NOSIGNAL);
+#else
  int res = send(socket, (const char *)buf, amount, 0);
+#endif
  if(res < 0 && WouldBlock())
   res = 0;
  else
  if(res == 0 || res < 0)
   SetSockError("send");
  return res;
 }
```

Subject: Re: SIGPIPE problem
Posted by mirek on Thu, 27 May 2021 07:14:16 GMT
View Forum Message <> Reply to Message

zsolt wrote on Wed, 26 May 2021 05:19I'm writing an HTTP App server currently and I'm using some libraries, such as libpq.
My problem is, that some of the libraries are switching SIGPIPE signal handler on and off, and I have no control over that.
So when a client disconnects while my server is sending the HTTP response, sometimes it gets a SIGPIPE and my app stops at that point in debugger.
I'm running tons of unit tests so this is not very convenient, as some tests are doing this.

I fixed this by changing the flags argument of send() from 0 to MSG_NOSIGNAL in TcpSocket::RawSend() method.

Do you have any better idea?

```
 int TcpSocket::RawSend(const void *buf, int amount)
 {
+#ifdef PLATFORM_POSIX
+ int res = send(socket, (const char *)buf, amount, MSG_NOSIGNAL);
+#else
  int res = send(socket, (const char *)buf, amount, 0);
+#endif
  if(res < 0 && WouldBlock())
   res = 0;
  else
  if(res == 0 || res < 0)
   SetSockError("send");
  return res;
 }
```

Sounds good, applied, thank you.

Mirek

Subject: Re: SIGPIPE problem
Posted by zsolt on Thu, 27 May 2021 14:41:55 GMT
View Forum Message <> Reply to Message

Thank you!

Subject: Re: SIGPIPE problem
Posted by Novo on Fri, 28 May 2021 18:41:40 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 27 May 2021 03:14

Sounds good, applied, thank you.

Mirek
MacOS 10.15: ./umk uppsrc ide CLANG -bus
/Users/ssg/worker0/m-upp/build/uppsrc/Core/Socket.cpp:565:52: error: use of undeclared identifier 'MSG_NOSIGNAL'
        int res = send(socket, (const char *)buf, amount, MSG_NOSIGNAL);
                                                  ^
1 error generated.

Subject: Re: SIGPIPE problem
Posted by mirek on Mon, 31 May 2021 07:44:47 GMT
View Forum Message <> Reply to Message

Ok, needs to be PLATFORM_LINUX instead... (commited).

Mirek

EDIT: "That said, question for zsolt: What about using signal(SIGPIPE, SIG_IGN) instead?" - already answered in the first post...

Subject: Re: SIGPIPE problem
Posted by zsolt on Mon, 31 May 2021 09:01:13 GMT
View Forum Message <> Reply to Message

Thank you! signal(SIGPIPE, SIG_IGN) is not working well.
But BSD (FreeBsd at least) also supports MSG_NOSIGNAL.

Subject: Re: SIGPIPE problem
Posted by mirek on Mon, 31 May 2021 09:21:15 GMT
View Forum Message <> Reply to Message

OK, can I get a verified list for that conditional compilation? :)

Mirek

Subject: Re: SIGPIPE problem
Posted by Oblivion on Mon, 31 May 2021 11:11:28 GMT
View Forum Message <> Reply to Message

Hi,

On BSD-based systems there is a socket option called SO_NOSIGPIPE, instead:

FreeBSD >= 9.0: https://www.freebsd.org/cgi/man.cgi?query=getsockopt&sek
tion=2&manpath=FreeBSD+9.0-RELEASE
MacOS X: https://developer.apple.com/library/archive/documentation/Sy
stem/Conceptual/ManPages_iPhoneOS/man2/getsockopt.2.html

Note: Apparently neither MSG_NOSIGNAL nor SO_NOSIGPIPE is supported on Solaris (if we support that OS at all...).

Best regards,

Oblivion

Subject: Re: SIGPIPE problem
Posted by zsolt on Mon, 31 May 2021 15:05:04 GMT
View Forum Message <> Reply to Message

Is this OK?

```
@@ -557,16 +557,22 @@ bool TcpSocket::WouldBlock()
  }
  return c == SOCKERR(EWOULDBLOCK);
 #endif
 }

 int TcpSocket::RawSend(const void *buf, int amount)
 {
-#ifdef PLATFORM_POSIX
- int res = send(socket, (const char *)buf, amount, MSG_NOSIGNAL);
+#ifdef MSG_NOSIGNAL
+#define UPP_NOSIGPIPE_OPTION MSG_NOSIGNAL
+#endif
+#ifdef SO_NOSIGPIPE
+#define UPP_NOSIGPIPE_OPTION SO_NOSIGPIPE
+# endif
+#ifdef UPP_NOSIGPIPE_OPTION
+ int res = send(socket, (const char *)buf, amount, UPP_NOSIGPIPE_OPTION);
 #else
  int res = send(socket, (const char *)buf, amount, 0);
 #endif
  if(res < 0 && WouldBlock())
   res = 0;
  else
  if(res == 0 || res < 0)
```

Subject: Re: SIGPIPE problem
Posted by zsolt on Mon, 31 May 2021 15:06:16 GMT
View Forum Message <> Reply to Message

The complete method:

```
int TcpSocket::RawSend(const void *buf, int amount)
```

```
{
#ifdef MSG_NOSIGNAL
#define UPP_NOSIGPIPE_OPTION MSG_NOSIGNAL
#endif
#ifdef SO_NOSIGPIPE
#define UPP_NOSIGPIPE_OPTION SO_NOSIGPIPE
# endif
#ifdef UPP_NOSIGPIPE_OPTION
 int res = send(socket, (const char *)buf, amount, UPP_NOSIGPIPE_OPTION);
#else
 int res = send(socket, (const char *)buf, amount, 0);
#endif
 if(res < 0 && WouldBlock())
  res = 0;
 else
 if(res == 0 || res < 0)
  SetSockError("send");
 return res;
}
```

Subject: Re: SIGPIPE problem
Posted by zsolt on Mon, 31 May 2021 15:10:40 GMT
View Forum Message <> Reply to Message

Or this one is better:

```
int TcpSocket::RawSend(const void *buf, int amount)
{
#ifdef MSG_NOSIGNAL
#define UPP_NOSIGPIPE_OPTION MSG_NOSIGNAL
#endif
#ifdef SO_NOSIGPIPE
#define UPP_NOSIGPIPE_OPTION SO_NOSIGPIPE
#endif
#ifndef UPP_NOSIGPIPE_OPTION
#define UPP_NOSIGPIPE_OPTION 0
#endif
 int res = send(socket, (const char *)buf, amount, UPP_NOSIGPIPE_OPTION);
 if(res < 0 && WouldBlock())
  res = 0;
 else
 if(res == 0 || res < 0)
  SetSockError("send");
 return res;
}
```

Subject: Re: SIGPIPE problem
Posted by Novo on Mon, 31 May 2021 15:18:51 GMT
View Forum Message <> Reply to Message

I checked against the latest revision.
Everything compiles fine now, including FreeBSD 12.

Subject: Re: SIGPIPE problem
Posted by Oblivion on Mon, 31 May 2021 15:23:25 GMT
View Forum Message <> Reply to Message

Hello zsolt,

It looks nice but I'm afraid that won't work. SO_NOSIGPIPE should be set using the setsockopt() function, preferably at socket initialization.

E.g.

```
#ifdef SO_NOSIGPIPE
  const int val = 1;
  setsockopt(socket, SOL_SOCKET, SO_NOSIGPIPE, &val, sizeof(val));
#endif
```

Best regards,
Oblivion

Subject: Re: SIGPIPE problem
Posted by mirek on Tue, 01 Jun 2021 07:19:48 GMT
View Forum Message <> Reply to Message

Yeah, great, getting complicated pretty fast... :)

Oblivion wrote on Mon, 31 May 2021 17:23Hello zsolt,

It looks nice but I'm afraid that won't work. SO_NOSIGPIPE should be set using the setsockopt() function, preferably at socket initialization.

E.g.

```
#ifdef SO_NOSIGPIPE
  const int val = 1;
```

```
   setsockopt(socket, SOL_SOCKET, SO_NOSIGPIPE, &val, sizeof(val));
#endif
```

Best regards,
Oblivion

Is it guaranteed to be #define though?

But perhaps even if it is not, it would be worth it...

Mirek

Subject: Re: SIGPIPE problem
Posted by Oblivion on Tue, 01 Jun 2021 11:15:37 GMT
View Forum Message <> Reply to Message

Well, the SIGPIPE issue is a relic network/socket programmers have to deal with, dating back to the implementation of TCP sockets in unix in early 80s.

AFAIK, there is no perfect solution. While the posix manual seems to define MSG_NOSIGNAL from some point on, it is not adopted by every camp yet.

Another workaround seems to be this, which I didn't know either:

Quote:
The most general solution, for when you are not in full control of the program's signal handling and want to write data to an actual pipe or use write(2) on a socket, is to first mask the signal for the current thread with pthread_sigmask(3), write the data, drain any pending signal with sigtimedwait(2) and a zero timeout, and then finally unmask SIGPIPE. This technique is described in more detail here. Note that some systems such as OpenBSD do not have sigtimedwait(2) in which case you need to use sigpending(2) to check for pending signals and then call the blocking sigwait(2).

Read the full blog, here:

https://www.doof.me.uk/2020/09/23/sigpipe-and-how-to-ignore- it/

Workaround (explained):
https://web.archive.org/web/20200126153413/http://www.microh owto.info:80/howto/ignore_sigpipe_without_affecting_other_th reads_in_a_process.html

Best regards,
Oblivion
