Subject: : AmigaOS4
Topic: : CSNode - file transfer over socket made as simple as possible
Re: CSNode - file transfer over socket made as simple as possible
Author: : alfkil
Date: : 2021/9/18 14:22:58
URL:

After having done long term testing on CSNode for some days, I have come up with a test, that shows something fishy about newlib select().

To cut it short : select() will return 0 (zero) after some time of running the test, and this will be not caused by a timeout but by something else.

Posix tells, that select() returns 0 on timeout, a positive number on successfull selection on the file descriptors given as input. It returns -1 in case of an error.

In this case, something has gone wrong, since after abandonning reception of data and re-binding the host socket, the client process (another computer) is unable to call the host (the Amiga) over the network.

Also it is noticable, that select() returns zero endlessly when this state is reached. The only way out of the loop is through a count and break;

Can this behavior somehow be explained??! I know this is not very core, and the obvious answer would be to suggest switching to bsd-sockets. But there is a possible solution hidden here, I feel.

Here is the code for the testing loop:

```
try {
    while (bytesReceived < size && bytesReceived == bytesWritten) {
        int i;
        for(i = 0; i < 5; i++) {
            if(i > 0) printf(",-");
            fd_set rfds;
            struct timeval tv;
            tv.tv_sec = 5;
            tv.tv_usec = 0;
            FD_ZERO(&rfds);
            FD_SET(connection->connectionSocket, &rfds);
            int ret = select(connection->connectionSocket+1, &rfds, 0, 0, &tv);
            if(ret == 0) printf("select returned zero.n");
            if(ret < 0) {
                throw(1);
            }
```

```
        if(FD_ISSET(connection->connectionSocket, &rfds)) {
            int bytes = 0;
            while (bytes == 0)
                bytes = recv(connection->connectionSocket, buffer, MIN(bufSize, size-bytesReceived), 0);
            if (bytes < 0) {
                if(errno == ECONNRESET) {
                    connection->isValid = false;
                }
                throw(0);
            }
            bytesReceived += bytes;
            //first fill the buffer (in case it was already non-empty)
            connection->readBuffer.fill(buffer, bytes);
            //...then extract from it
            int bytes1, bytes2;
            while((bytes1 = connection->readBuffer.readBytes(buffer, MIN(bufSize, size - bytesWritten))) > 0) {
                bytes2 = write(fd, buffer, bytes1);
                bytesWritten += bytes2;
            }
            break;
        }
    }
    if(i == 5) throw(2);
    cout << ".";
  }
  cout << "ok.n";
} catch(int kind) {
  if(kind == 1) perror("select");
  else if(kind == 0) perror("recv");
  else printf("Trials max reached. Abandonning recv.n");
}
```

EDIT: A partial explanation would be, that this is some kind of limitation on the local network. The test runs both computers in a loop sending the same 30mb file from the PC to the Amiga. At intervals of minutes the communication will break off and come to a standstill. Now, after a timeout on the PC side, it will tell you, that it cannot find the Amiga host (no route to host). The cute thing is, that when I ping google.com on the Amiga side, the two computers - host and client - will find each other again and recommence where they left. Explain!