Kermit-20 5.3(235)-5 Maintenance Release

Thomas DeBellis, SLOGIN@TOMMYT.#DECnet (`TOMMYT::SLOGIN`, HECnet), 11-Jun-2023

Table of Contents

Summary 2
[231] Fix RECEIVE with no file name 2
[232] 36 bit byte file sizes 2
[233] Transaction Log and Debug log fixes and enhancements 3
  Transaction logging 3
    Garbled Text 3
    Write-Protection Failures 3
    Enhancements 3
  Debug Log Decode Fix 3
[234] Error messages may not be seen if displayed in remote server 4
[235] Properly signal and handle file errors in server mode 4
Additional Batch Tests 4
  `K2036P`: Kermit-20 36 Bit Mode via pseudo-terminal 4
  `K2036C`: Kermit-20 36 Bit Mode via pseudo-terminal with parity 4
Updated Help 5

Summary

Version 5.3(235)-5 incorporates all fixes since the major release of 5.3, Edit 230 in January of this year. These are as follows and can be identified in the code with the edit number as the prefix of a comment.

[231] Fix RECEIVE with no file name

Previously, it was possible to issue the `RECEIVE` command without a file specification in order to transfer multiple files to Tops-20. It was also possible to do this directly from the Tops-20 EXEC by typing the following at the EXEC's at-sign command prompt, viz: `@KERMIT RECEIVE`.

Since the update, Kermit-20 appeared to require a filename:

```
Kermit-20>receive
?Filename was not specified
```

From the EXEC:

```
@KERMIT RECEIVE
Filename was not specified
?Not a KERMIT command -
```

This was the result of a missing symbol; an editing oversight.

## [232] 36 bit byte file sizes

This enhancement was driven by the new need to distribute executables between Tops-20 hosts.

It may seem strange that Kermit-20, running on a PDP-10 with its 36 bit architecture has never itself had a 36 bit byte file size even though the file system directly supports this. Because Kermit was primarily used for transferring files between microcomputers and mainframes, this was largely unnecessary.

Since its inception, however, Kermit-20 has always been able to send and receive DEC-20 binary files perfectly with absolutely no loss of data, as can be verified by doing a binary comparison using `FILCOM` with the /W switch or comparing checksums via the EXEC's `Checksum By-Pages` directory listing option.

It did this by adopting a kind of 'trick' that Tops-20 itself uses when writing a 9-track ANSI-ASCII magnetic tape. The file is read using a 7 bit byte pointer, which guarantees that bit 8 (or the parity bit) can NEVER be set. Since it is unused, this bit may be considered as a kind of undocumented 'spare' and is used to store bit 35 in the following way.

The trick is to set this bit 8 (or the parity bit) of every 5th character to the value of bit 35 of the PDP-10 word it came from on a send and check for it on a receive.

This allows for Tops-10/20 .EXE's, other 36-bit binary files, and SOS-line numbered files, to be sent to 8-bit systems and retrieved intact. Clearly, no adverse effects can be suffered by ordinary text files.

That being said, the problem is that a received file will be stored with a size byte of 7 instead 36 and a byte count that will be off by a factor of five. This can cause undesired behavior in programs which rely on or otherwise pay attention to either one.

Setting a byte size of 36 forces a 7 bit transfer but then causes special post-receive processing to be done to properly set the byte size to 36 bits and calculate the byte count accordingly.

[233] Transaction Log and Debug log fixes and enhancements

Transaction logging

Garbled Text

As a multi-section program executing in section 0, Kermit-20 uses one word global pointers (`OWGP`'s) to write text that has been moved to section 1 to free up executable memory. Such text will display perfectly when written to a terminal and can be uneventfully transferred with a traditional `ILDB`/`IDPB` loop.

The behavior is not consistent because the exact same one word byte pointer will fail in any string instruction or for String OUT (`SOUT%`) which writes a file. The latter case resulted in abject junk—a series of `^@`'s (`NUL`'s)—unexpectedly showing up in transaction logs. This was an omission as other logs work properly because an inter-section call to section 1 is used to write the data.

Write-Protection Failures

A major feature of the 5.3 address space rearrange was to no longer use self-modifying code and to write-protect all non-data spaces. A stray byte store from an uninitialized register into the (now write-protected) code segment resulted in the expected 'desired' of Kermit immediately failing

In this case, 'desired' means that the store was no longer quietly overwriting memory, resulting in the occasional undiagnosable bug. In this case, the store was immediately identified and the register initialization done.

Enhancements

Previously, a transaction was defined as any file transfer. Kermit-20 has been enhanced to log anything that involves sending a data packet as a response or non-trivial activity. This allows the logging of TYPE commands and wildcarded DELETE's while still filtering directory connects.

More importantly, errors can be logged both remotely and locally, which aids for trouble-shooting transfer errors and subsequent debugging.

Debug Log Decode Fix

Version 5.3 introduced some amount of packet decoding, along the same lines as Wireshark. This is useful for individuals who may not have the protocol memorized or are trying to debug an issue or merely trying to learn it.

In some cases, Kermit-20 would appear to not be able to decode what was known to be a perfectly good packet.

Decoding is based on the Kermit packet type character, which is always 7 bits. However, when parity is being used, bit 8 may be set, resulting in the decoder not recognizing the packet type. The fix is to strip the parity bit before doing the decode.

[234] Error messages may not be seen if displayed in remote server

When Kermit-20 is operating in SERVER mode, it must use the Kermit protocol to signal all errors and messages. Anything that is simply typed on the terminal without such encapsulation will silently be discarded as line noise, at most resulting in a packet retransmission.

This can result in problems not being noticed and hangs.

The fix is to not put anything on the terminal if Kermit-20 is performing the role of the remote server, but rather putting them into the transaction log and signaling them with an error ('E') packet.

[235] Properly signal and handle file errors in server mode

Under Tops-20, it is possible to get a handle on a file (via `GTJFN%`) which you do not have permission to open. This is known as having list-only access. When handling a wildcarded file transfer, a file opening (`OPENF%`) error would be displayed on the remote server's line, not signaled and the wildcarded JFN would continue to be stepped (via `GNJFN%`).

The user visible behavior would be an apparently entirely unremarkable transfer with certain files simply failing to show up at the target system.

This is a protocol error. Kermit-20 has been fixed to be compliant by forcing itself into 'A' state (also known as, `c`**`A`**`ncel`), responding with an 'E' packet and going back to top-level server command processing.

Additional Batch Tests

These are also listed in an updated Kermit-20 Testing Battery file. All tests were successfully completely prior to release except for Tops-10, due to unrelated networking issues.

`K2036P`: Kermit-20 36 Bit Mode via pseudo-terminal

1) Same as `K20PTY`, except no large files, all files being executables
2) Demonstrate correct and enhanced transaction logging
3) `FILCOM/E` demonstrates transfers correct to bit level
4) Double checked against EXEC directory, `CHECKSUM BY-PAGES`
5) Pre and post transfer file byte sizes and counts visually compared to be correct

`K2036C`: Kermit-20 36 Bit Mode via pseudo-terminal with parity

1) Same as `K2036C`, but with parity sending on terminal and packets and checking being performed
2) Demonstrate correct and enhanced transaction logging which is transparent to parity
3) Demonstrate proper error handling when a file cannot be opened by the Kermit-20 server;

a. That server state is updated,
b. That an ‘E’ packet is sent,
c. That the ‘E’ packet is properly interpreted by the Kermit-20 client, and
d. That no-remnants of the file are left in the testing directory.

Updated Help

Additional help for 36 bit mode and transaction logs, other minor corrections.