Kermit-20 5.3(277)-5 Maintenance Release

Thomas DeBellis, SLOGIN@TOMMYT.#DECnet (TOMMYT::SLOGIN, HECnet), 30-Aug-2024

Table of Contents

Summary

Version 5.3(277)-5 incorporates additional features, enhancements and fixes since the minor release of 5.3(255)-5 in March of this year (29-Mar-2024). These are as follows and can be identified in the source code with the edit number as the prefix of a comment.

The full 42 year, 7 month, 25 day, 8 hour, 32 minute, and 6 second edit history may be found in K20EDT.MAC. The file also contains suggestions and musings for (what was then) future functionality and makes for interesting reading.

This release's changes may roughly be classified into a few areas:

1. Fixes, typically to previous code when put in unexpected situations or stressed.
2. Speed enhancements, typically by more effective Tops-20 systems programming.
3. Informational.
4. Non-Kermit based transfers
5. More granular parity handling.

The environment of Tops-20 must be observed in context in order to determine a number of these enhancements. In particular, actual KL10 hardware now only exists in museums and, given the expense of maintaining it, and the cost of the electricity and cooling, since the advent of COVID, no systems are in actual operation.

Later hardware instantiations of the PDP-10 architecture do exist (such as the XKL Toad-1 and Toad-2 along with Systems Concept's SC-40), but these are relatively rare. There are no current guest ids on any of them for Kermit-20 backwards compatibility check out.

This makes the testing of serial line code a challenge, most importantly the parity code. Tops-20 will *not* generate parity on any network line whether TCP/IP, NRT, CTERM or LAT. Fortunately, it can be coaxed into doing so via the use of pseudo-terminals. Much of the coding and testing involves coming up with test cases to stress both side of the connection, which reveals error conditions, some in Kermit-20, other quirks in Tops-20 itself.

Parity has to be tested outside of regular Kermit protocol, with the TRANSMIT command. This is a highly valuable command for carefully sending files to systems which do not have Kermit. Typically, the files in question might be some kind of Kermit stub which can be run on the foreign system or source code which can be compiled.

Given this situation, the only way to test TRANSMIT was by writing a CAPTURE command. These pair of commands can then be used to test parity. Parity handling was given additional granularity to—instead of failing completely on a parity error—to translate the character with bad parity into some sort of a talisman that would not be expected to be found in the file, such as "~".

The advantage here is that searching the resulting file for "~" will show exactly where the error occurred and this turned out to be useful for chasing down edge cases in the various network line drivers. The parity changes cascaded through the rest of Kermit-20, including enhancing INPUT to respect parity when consuming characters from the foreign system.

It must be stressed that parity should **not** ever be used in place of the KERMIT protocol, which has far stronger error detection and correction. It is also unwise to depend on what an intervening system or terminal server might decide to do with parity (such as tossing it entirely).

Batch tests were updated to test the QUEUE% JSYS, a new interface to Galaxy.

Help has been written or updated as appropriate.

[256] /PARITY switch to ECHO for batch and other testing purposes

The ECHO command can be given a /PARITY switch to force parity on a line by line basis. The parity switch takes the following parameters: EVEN, MARK, NONE, ODD and SPACE.

The current setting of parity on the communications line will be ignored (even if no parity is being generated at all) and the supplied text will be echoed with the indicated parity. This is used to force parity errors when debugging.

Because the /PARITY switch is used for debugging and testing, it is invisible. You have to know that it's there to use it.

[257] Teach INPUT to respect parity, if asked

This can be coupled with edit 256 to produce forced parity error scenarios.

[258] SET PARITY action /ABORT, /PROCEED

Previously, Kermit did not check parity at all—it simply stripped it. The first implementation of parity checking handled a parity error aborting the entire transfer, which made debugging somewhat difficult.

It also ignored the fact that parity could have its uses for discovering issues in the operating system provided character stream. In this case, Kermit-20 can now be told to proceed and substitute a character for any character that was detected to have bad parity.

[259] Fix a few arcane bugs in the C constant expression expander

C constants for string expansion and recognition were introduced in edit [209] of Kermit-20 in 27-Jun-2022 and have seen enhancements and fixes since then. This edit caught another arcane edge case for octal numbers, which are now restricted to 36 bits, producing a maximum of 12 ASCII digits.

[260] Fix DEFINE to work (again) with ESCAPE and PARITY

From its very beginnings in the 1980's, Kermit-20 has supported macros. These have been enhanced in various ways, such as being able to be stored in binary format for far faster loading on Kermit startup.

However, macros impose a particular discipline when developing or updating commands. In certain cases, correct macro support was not properly coded for ESCAPE and PARITY.

[261] Decrease repeated parity error messing

Kermit would repeatedly issue warning about parity agreement and 8 bit prefixing. It only does this once, now.

[262] Remove the legacy INPUT code that uses BIN%'s

The legacy INPUT code was determined to no longer be working and was replaced with SIN% based network or line reads, like what regular Kermit transfer code uses. This significantly reduced processor time usage. As INPUT is the basis for Batch support, all Kermit-20 test jobs use it, so speed ups here have a detectable effect.

[263] Stop using a SIN% for a line at a time read

During the development of TRANSMIT and CAPTURE, it was found that the commands were using a form of SIN% to parse input files, stopping on line feeds. While convenient, this is not efficient. Regular Kermit transmits use PMAP%, for example.

Recalling that TRANSMIT is an uncommon use case, this is a completely reasonable trade off and decision.

For extremely high stress testing and edge case discovery, it begins to break down. The entire input file is now SMAP%'ed into a completely separate section and this single JSYS then allows Kermit-20 to use regular string instructions to consume the data with no further operating system calls.

The code was written in such a way as to eventually replace the multiple PMAP%'s used during a regular transfer with an SMAP% at some future point.

Again, the CPU savings would not be very noticeable in the 1980's with sub-kilobit dial up speeds of files of a few kilobytes. They are quite noticeable when dealing with multi-megabyte files over megabit network links.

This is a case of a space versus time tradeoff and is tuned for a hobbyist machines which rarely have more than one or two active users and thus have abundant memory, rarely swapping, if ever.

[264] Toggle session logging from command level

Usage `TOGGLE [ OFF | ON ]`

When connected to a remote system and logging the session, the user can issue the escape character followed by the letter Q to pause the log (or Quit Logging) or the letter R to Resume (active) Logging.

This is not possible if you are not directly connected via a terminal fork and are using the OUTPUT and INPUT commands to manage the remote job, instead. Such interaction with a remote system is common in a batch job.

If there is a session log and it is active, the OFF keyword will pause (and checkpoint) the session log. If there is a session log and it is NOT active, the ON keyword will resume logging.

Issuing a TOGGLE with no argument will pause (and checkpoint) the session log if it is active and resume it if it is inactive. It is not an error to toggle when session logging is not being done, but Kermit-20 will notify you of this.

[265] Fix TTY Input Buffer full when doing a TRANSMIT on a PTY

Doing a large write to a pseudo-terminal could trigger a monitor condition which would either force an error or wedge a transfer when using TRANSMIT. This was fixed by checking both the output buffer of the PTY and the input buffer of the respective associated TTY for full conditions and not writing until the data was consumed.

As per previous, this was discovered during an unreasonable edge case of absurdly large writes.

A number of commands were modified to specify the maximum number of characters in a line of text. This is for CAPTURE/TRANSMIT as regular packet sizes in Kermit have regularly handled this situation for decades.

[266] Turn TRANSMIT and CAPTURE increasingly hairy switches in SET parameters

As original coded, TRANSMIT and CAPTURE could put all their respective parameters on a single line, which became increasingly tedious to type. The parameters may now be specified in a separate SET command, and overridden on a case-by-case basis with the switches.

[267] Give effective data rate at end of TRANSMIT

At the end of the command, the effective data rate is now displayed, which is similar functionality to the STATUS command. This is useful information for tuning, writing stress tests and discovering edge cases. For example:

```
Local-Kermit-20>Transmit /silent /EOF ^Z F1: "Otto>>"

[KERMIT-20: Transmitting TOMMYT:<DOCUMENTATION>ALICES.PDP10.1
 If stuck, type:
 Carriage Return to send next line,
 ^P to resend current line,
 ^G^G to cancel.  Here goes...]

[KERMIT-20: Transmit of TOMMYT:<DOCUMENTATION>ALICES.PDP10.1
complete, 13841 characters,   82.0182 KC/s]
```

[268] CAPTURE is no longer hidden. Further document it

CAPTURE is used in Batch jobs to test transmit, so it made sense to not set it to be invisible and document it in a HELP command. CAPTURE has limited utility outside of TRANSMIT testing as a Batch job and session log will provide similar functionality.

[269] Review, update and correction of all help text

The amount of work being done indicated that it was time to reread all the internal help text and update it as well as write help text for new commands and old invisible commands which were no longer invisible.

[270] Enhance default for SET PROMPT to give some extra useful information

The amount of debugging being done necessitated managing a number of client/server Kermit-20 sessions. When connected and operating in LOCAL mode, Kermit-20 will default the text of the SET PROMPT command if asked. For example,

```
Kermit-20>conn pseudo-terminal /stay
[KERMIT-20: Loopback connection to VENTI2:: via PTY6: as TTY21:]
Kermit-20>show line

TTY for file transfer:  21 [PTY6:]
 (pseudo-terminal loopback to VENTI2::, KERMIT-20 is LOCAL)
  Handshake:         None
  Flow-Control:      None
  Escape Character: ^\
  Parity:       None
  Duplex:       Full
  Break Simulation: Disabled
  PTY Connection:   Online
  Controlling Type: PTY [Parity]
  Input Buffers:    1
  Output Buffers:   1

Kermit-20>set prompt
PTY6(21):Kermit-20>
```

[271] Fix TVT setting; it gets overwritten by automatic processing

A TVT is a Telnet Virtual Terminal. The Telnet protocol requires special programming (via either an MTOPR% or via the use of the IAC character) to put the line into binary mode which is necessary for efficient communication 8 bit data.

It was not possible to force TVT on or off as this was overwritten by checking. It is now possible to force TVT on or off and this is used to produce errors for TCP/IP transfers when Kermit-20 is in server mode or is otherwise REMOTE.

A /DISCOVER switch is now available to force discovery of the line type. This is useful debugging and double checking whether the line can be put into binary mode.

```
Kermit-20>set tvt /discover
[KERMIT-20: Turning TVT binary mode on, Pause, Turning TVT binary mode
off]

iMac supports setting binary mode
Kermit-20>
```

[272] SET TRANSMIT DEFAULT-PROMPT

Ongoing work to support [266]

[273] SET TRANSMIT CASE OBSERVE/IGNORE

This was a continuation of work begun in 266 to finish decoupling TRANSMIT from INPUT in the rare case where they are both being used. This was largely driven to support the ease of debugging.

[274] Fix SET INPUT to better support DEFINE

Earlier work had broken the ability of Kermit-20 to support macros with SET INPUT. This was a continuation of work begun in 260.

[275] SET TRANSMIT SETTINGS-DEFAULTS for backwards compatibility

See 273. This work was done to make debugging more convenient.

[276] Fix indirect recursion crash when reporting an error with REALLY short packets

Edge case testing with very short packet sizes of forty (40) characters showed that error handling could put Kermit-20 into an indirect error loop which would result in stack overflows.

Normally, Kermit-20 very carefully checks the length of the packet to send and it does this in multiple places. The only place where it wasn't done was when ERSTR% was being used to put the text of the Tops-20 error into a packet.

This would result in the error handling itself producing an error, which would also try to report the last Tops-20 error with ERSTR% and so on…

[277] Variable blips (so we don't get clobbered for very small packets)

```
SET BLIP [number]
```

A ‘blip’ is a real time indicator of packet traffic during a transfer. Specifically, it is the dot (or period) character (“.”), which is typed at specific packet intervals.

A blip can only be shown when Kermit-20 is in local mode. Were it to be typed in remote mode, the user could not see it and it would interfere with the data transfer.

Typing nothing (a simple confirm) restores the default, which is one blip every five packets. Typing a zero means to not type any blips, at all, ever.

When transferring a very large file with very small packet sizes, it may be useful to increase the blip interval so the terminal does not display thousands of blips. This may make sense in a batch job.

Negative numbers will be rejected.

The current blip interval is displayed by SHOW TIMING-INFO, but only when Kermit is in local mode.

The code is intended to serve as a basis for graphical ‘blips’, a future enhancement.

Kermit-20-Testing-Battery-5.3(277)-5 Updates

1. Various updates to support submission of batch jobs with the QUEUE% JSYS.
2. Info Fork/High for increased granularity on Kermit execution. This depends on a Tops-20 monitor modification to report runtimes in HPTIM% ticks, which are in multiples of 10 microseconds, the resolution of the DK10 hardware clock.

Updated Help

Additional help for above enhancements, other minor corrections.