Subject: MCR10090, GTSS Initialization Fixes

Author: Gary Dixon

Date: 2021-05-27

Multics Ticket: http://multics-trac.swenson.org/ticket/124

The GTSS (GCOS Time Sharing System) command available on Multics reports an error when attempting to save a session (such as a Basic language programming session) in a file for later reuse. The script below demonstrates this error.

```
gtss
GTSS 4JS3 (4.0)
*old MRGD
<50>FILE MRGD -- NONEXISTENT
*bye
r 06:57 0.466 9
```

The above test shows GTSS file named MRGD does not exist. The next step begins a brief Basic language session, and attempts to save that session in an MRGD file.

```
gtss
GTSS 4JS3 (4.0)
*system basic
*10 print "Multics Rulez, GCOS Droolz"
*run

Multics Rulez, GCOS Droolz

ready
*save MRGD

Error:  conversion condition by any_to_any_$formline_|6213
(>system_library_1>bound_library_wired_)
onsource = "chain", onchar = "c"
Invalid character follows a numeric field.
system handler for error returns to command level
r 06:58 0.606 45 level 2
```

Tracing the stack shows that the conversion condition occurs in the GTSS derail handler subroutine gtss_mcfc_$open when it calls the com_err_ subroutine to report an error.

```
trace_stack -brief
error condition:
234|21320   default_error_handler_$wall|2671 (bound_error_handlers_|2671)
234|21220   initialize_process_$any_other.2|426 (bound_process_init_|426)
234|20660   signal_$signal_|1467 (bound_library_1_|11205)
conversion condition:
234|17020   pl1_signal_$help_plio2_signal_|776 (bound_library_1_|6572)
234|16440   pl1_signal_conversion_$pl1_signal_conversion_|170 (bound_library_1_|7166)
234|15500   any_to_any_$formline_|6213 (bound_library_wired_|53565)
234|15040   ioa_$general_rs|1147 (bound_library_wired_|16621)
234|14200   com_err_$com_err_|573 (bound_library_1_|26443)
234|12220   gtss_mcfc_$open|6035 (bound_gcos_tss_|252745)
234|11700   gtss_ios_open_$gtss_ios_open_|1226 (bound_gcos_tss_|244306)
234|11020   gtss_filact_funct04_$gtss_filact_funct04_|1032 (bound_gcos_tss_|204150)
234|10600   gtss_drl_filact_$gtss_drl_filact_|2247 (bound_gcos_tss_|106413)
234|10360   gtss_derail_processor_$gtss_derail_processor_|2570 (bound_gcos_tss_|57432)
234|10200   sct_manager_$sct_manager_$call_handler|123 (bound_library_1_|11461)
234|7660    signal_$signal_|301 (bound_library_1_|10017)
234|7240    return_to_ring_0_|0
derail condition:
234|7100    !BBBKPzjKbXphfz.temp.0360|2753
234|6320    gtss_run_subsystem_$gtss_run_subsystem_|1503 (bound_gcos_tss_|24605)
234|6140    gtss_interp_prim_$exec_primitive|3673 (bound_gcos_tss_|21127)
234|5460    gtss_interp_prim_$gtss_interp_prim_|2371 (bound_gcos_tss_|17625)
234|4320    gcos_tss$gtss|1625 (bound_gcos_tss_|7563)
234|3360    command_processor_$command_processor_|2101 (bound_multics_bce_|2101)
234|2700    abbrev$abbrev_processor|6105 (bound_command_loop_|14103)
234|2400    listen_$listen_|554 (bound_command_loop_|34522)
234|2000    initialize_process_$initialize_process_|704 (bound_process_init_|704)
r 06:59 0.490 17 level 2
```

Using probe to examine the arguments passed to com_err_ shows use of an invalid ioa_ control string in the com_err_ error message string. The invalid control is highlighted in red text.

```
probe
Condition conversion raised at formline_|53565 (level 17).

use level 15            [com_err_ stack is 2 levels below formline_ stack frame]
args
ARG   1 @ 234|13006 = 8589679543
ARG   2 @ 234|12330 = "gtss_mcfc_$open  "
ARG   3 @ 234|14036 = "UNEXPECTED LOCK ERROR? (^1 ^i)"
ARG   4 @ 234|13074 = "chain"
ARG   5 @ 234|12420 = 509
..pel 8589679543
8589679543 = No write permission on entry.
```

Examining code in gtss_mcfc_$open finds this call to com_err_ in tests reporting a failure to lock some GTSS database associated with opening files. The failing call to com_err_ is highlighted. Note that earlier checks call com_err_ using a ^a control for the en substitution argument; the failing check uses the ^1 invalid control for this same substitution argument.

```
/* Could not lock. */
               if c = error_table_$lock_wait_time_exceeded then do;
                    if db_mcfc then
                         call com_err_ (c, ENTRY, "(^a ^i)", en, e);
                    return ("1"b);
               end;
               if c = error_table_$locked_by_this_process then do;
                    call com_err_ (c, ENTRY,
                         "BUG? Will not proceed (^a ^i).", en, e);
                    signal cond (gtss_fail);
                    return ("1"b);
               end;
               call com_err_ (c, ENTRY,
                    "UNEXPECTED LOCK ERROR? (^1 ^i)", en, e);
               signal cond (gtss_fail);
               return ("1"b);
```

Bugs

The conversion issue described above may be corrected by changing the ioa_ control to ^a.

```
mbuild:  cmp gtss_mcfc_.pl1

----------  gtss_mcfc_.pl1

compare_ascii >ldd>unb>source>bound_gcos_tss_.2.s.archive::gtss_mcfc_.pl1 ==

A799                          "UNEXPECTED LOCK ERROR? (^1 ^i)", en, e);
Changed by B to:
B799                          "UNEXPECTED LOCK ERROR? (^a ^i)", en, e);

Comparison finished: 1 difference, 2 lines.
```

A quick check of other modules in bound_gcos_tss_.s.**.archive found one other instance for this invalid ioa_ control string. This instance in gtss_mcfc_delete.pl1 will also be corrected.

After both instances of invalid ioa_ controls are corrected, a second issue sometimes occurs. The gtss **save** command can report a bad area format error (or sometimes another error). For example:

```
gtss
GTSS 4JS3 (4.0)
*system basic
*10 print "Multics Rulez, GCOS Droolz"
*run

Multics Rulez, GCOS Droolz

ready
*save MRGD

Error:  bad_area_format condition by gtss_mcfc_$open|1240

r 08:26 0.928 22 level 2
```

This problem was concealed by the conversion condition issue, but still needs to be corrected to make gtss operate correctly.

Investigation of this second problem discovered a root cause for this area format condition, and for the "UNEXPECTED LOCK ERROR?" described in the first issue. The data files referenced in both cases are templates initialized by gtss_mcfc_init_.pl1:

```
>unb>GTSS.MCFC.(FILES NAMES CALLERS_1 CALLERS_2 CALLERS_3 CALLERS_4)
```

Pointers for these template files are stored as internal static data of the gtss_ext_$mcfc, an entry point in an external gtss_ext_ object segment generated by create_data_seg.

gtss_mcfc_init_.pl1 stores pointers to these 6 template segments in an array of pointers declared as:

```
 99  dcl  mcfc_ptr (6)ptr aligned
              based(addr(gtss_ext_$mcfc.files_ptr));
```

where gtss_ext_$mcfc is declared in gtss_ext_.incl.pl1 as:

```
109  dcl 1 gtss_ext_$mcfc          aligned ext,
110        3 multics_lock_id       bit(36),
111        3 wait_time             fixed bin,
112        3 files_ptr             ptr,
113        3 names_ptr             ptr,
114        3 callers_ptr           (0:3)ptr;
```

The first reference to the mcfc_ptr array snaps a link to the external entry point gtss_ext_$mcfc plus a positive offset to the files_ptr element. The files_ptr and names_ptr scalars, and callers_ptr array are overlaid by this mcfc_ptr declaration.

Referencing an external data structure with a non-zero offset triggers a bug in the dps8m simulator code that handles such instruction references when they are interrupted in mid-instruction by a linkage fault (designated an “f2” or “Fault Tag 2” fault in the Multics Processor Manual, Order Number AL39). That bug is described in the dps8m Issue #25 (https://gitlab.com/dps8m/dps8m/-/issues/25). It is the underlying trigger for all the issues described in this MCR.

The simulator bug may be avoided by pre-snapping the link to the external data segment before assigning values to any of the mcfc_ptrs. That avoidance will be included in this MCR.

Correction of the simulator bug is outside of the scope of this MCR, or of any change to the Multics operating system. It must be corrected in the code simulating the Multics processor.

Proposed Changes

The changes proposed to repair the issues/bugs above appear in the source modules shown in the following build script:

```
Description:
 1) Multics Ticket #124 describes a conversion condition encountered while
    issuing a gtss save command.  This condition is caused by an improper
    ioa_ control string in module: gtss_mcfc_.pl1
 2) A similar issue exists in a related module: gtss_mcfc_delete.pl1
 3) After correcting these two problems, the gtss save command may still
    fail due to improper initialization of the gtss_ext_$mcfc data structure
    by module:  gtss_mcfc_init_.pl1
     - This improper initialization is caused by a flaw in the current
       dps8m simulator for Multics hardware.  The flaw may be bypassed by
       pre-snapping this module's reference to gtss_ext_$mcfc.multics_lock_id
       (first element of the structure) before accessing remaining elements.
     - See dps8m issue:  https://gitlab.com/dps8m/dps8m/-/issues/25

Installation_directory:  >udd>m>gd>w>gtss;

 Build_script:     gtss.mb;

 Bound_obj:        bound_gcos_tss_     IN: unb   UPDATE;
   source_arch:    bound_gcos_tss_.2.s.archive  UPDATE;
     source:       gtss_mcfc_.pl1               REPLACE  compiler: pl1 -ot;
     source:       gtss_mcfc_delete.pl1         REPLACE  compiler: pl1 -ot;
     source:       gtss_mcfc_init_.pl1          REPLACE  compiler: pl1 -ot;
```

The changes are summarized in the following comparison of original source modules with repaired version of those modules.

```
mbuild:  compare

----------  gtss_mcfc_.pl1

compare_ascii >ldd>unb>source>bound_gcos_tss_.2.s.archive::gtss_mcfc_.pl1
==

A799                          "UNEXPECTED LOCK ERROR? (^1 ^i)", en, e);
Changed by B to:
B799                          "UNEXPECTED LOCK ERROR? (^a ^i)", en, e);

Comparison finished: 1 difference, 2 lines.

----------  gtss_mcfc_delete.pl1

compare_ascii
>ldd>unb>source>bound_gcos_tss_.2.s.archive::gtss_mcfc_delete.pl1 ==

A333                          , "UNEXPECTED LOCK ERROR? (^1 ^i)"
Changed by B to:
B333                          , "UNEXPECTED LOCK ERROR? (^a ^i)"

Comparison finished: 1 difference, 2 lines.

----------  gtss_mcfc_init_.pl1

compare_ascii
>ldd>unb>source>bound_gcos_tss_.2.s.archive::gtss_mcfc_init_.pl1 ==

Inserted in B:
B42  dcl  snap_link bit(36) aligned;
                  /* Snap link to access gtss_ext_$mcfc internal static.  */
B43    snap_link = gtss_ext_$mcfc.multics_lock_id;
                  /*  - The mcfc_ptr array just below begins setting that */
B44               /*    static data.  dps8m simulator bug has problems in */
B45               /*    link-snap if assigning to a non-zero offset from  */
B46               /*    the known link address.  After link is snapped,   */
B47               /*    these problems do not occur.  For more info, see: */
B48               /*     https://gitlab.com/dps8m/dps8m/-/issues/25       */
Preceding:
A42    do i = 1 to hbound (mcfc_ptr, 1);

Comparison finished: 1 difference, 7 lines.
```

Testing

The changes in this MCR were tested first as individual repairs to particular source modules; then as changed components in a rebuilt bound_gcos_tss_ object segment; and finally after complete installation of the repair into the >unb library. Such system integration-level testing is needed to ensure that all of the repairs interact correctly with other files comprising the GTSS execution environment.

Eric Swenson assisted with this system integration testing as part of his investigation into correction of the dps8m simulator bug underlying which triggered the GTSS issues described in this MCR.

Documentation

These repairs made no interface changes, and therefore require no documentation changes.

Version History

Date	Revision	Author	Comment
2021-05-27	1.0	Gary Dixon	Initial version of this MCR.