Subject: MCR-10051, fix lisp (STATUS DATE)

Author: Eric Swenson

Date: July 30, 2018

Introduction

The lisp special form `status` returns various values, depending on the first argument to the form. One such argument, `date`, causes `status` to return the current date. Thus, (`status date`) returns the current date as a LIST of three FIXNUM values. For example, today, on July 28, 2018, it returns:

```
(status date)
(166 7 34)
```

As it can be seen, the values are returned in the current radix, which defaults to base 8. The values, interpreted in base 10 are:

```
(118. 7. 28.)
```

The year, therefore, is being returned as an offset from 1900. This MCR proposes to change the return value of (`status date`) from a year, modulo 1900 to the 4-digit year as we would expect it in the post-Y2K world. Thus, the year would be returned to as 2018 decimal, or:

```
(3742 7 34)
```

Implications

Changing `(status date)` to return a 4-digit year will impact any callers of this special form. Each of these will have to be inspected and updated to do the right thing in the face of 4-digit years. Fortunately, there appear to be only 3 uses of this form:

- `bound_emacs_full_.s.archive::e_macops_.lisp`
- `bound_lisp_compiler_.s.archive::lcp_cg_.lisp`
- `bound_lisp_compiler_.s.archive::lcp_semant_.lisp`

The reference to `(status date)` in e_macops_.lisp is in the definition of the emacs (internal) `date` function. This function returns a string like “7/28/20” with the current definition of (status date). This function, too, will need to be reworked, since it is obviously wrong now and will continue to be wrong with a 4-digit date.

The implementation of `(status date)` is in:

- `bound_lisp_intrprtr_.1.s.archive::lisp_status_fns_.pl1`

Proposed Changes

lisp_status_fns_.pl1

The code for `(status date)`, in lisp_status_fns_.pl1 is this:

```
saction(9):             /* (status date) */

        call gclock;

        /* cons up a list of y, m, d */

        year = year - 1900;         /* since MACLISP returns two digit year */

cons3n:
        addr(stack -> temp(3)) -> fixnum_fmt.type_info,
        addr(stack -> temp(2)) -> fixnum_fmt.type_info,
        addr(stack -> temp(1)) -> fixnum_fmt.type_info = fixnum_type;
        addr(stack -> temp(1)) -> fixedb = year;
        addr(stack -> temp(2)) -> fixedb = mon;
        addr(stack -> temp(3)) -> fixedb = day;

cons3x:   stack -> temp(4) = nil;
        stack_ptr = addr(stack -> temp(5));
        call lisp_special_fns_$cons;
        call lisp_special_fns_$cons;
        call lisp_special_fns_$cons;
        return;
```

The fix, therefore is easy—remove the line that decrements 1900 from the year. The function `gclock` simply calls `decode_clock_value_` with the current value of `clock_()`.

e_macops_.lisp

The current definition of the date function (only use of `(status date)`) is:

```
(defun date ()                                  ;general utility BSG 10/31/79
       (let ((statdate (mapcar 'decimal-rep (status date))))
            (catenate (cadr statdate)
                      "/"
                      (caddr statdate)
                      "/"
                        (maknam (exploden (remainder (read-from-string (car statdate))
100))))))
```

This function becomes simplified to:

```
(defun date ()                                          ;general utility BSG 10/31/79
       (let ((statdate (mapcar 'decimal-rep (status date))))
            (catenate (cadr statdate)
                      "/"
                      (caddr statdate)
                      "/"
                      (car statdate))))
```

---

lcp_cg_.lisp

Uses `(status date)` in a context where whatever values are returned are emitted to the lisp compiler image as a record of the compiler history. It shouldn't matter whether the 2-digit or 4-digit year is returned. Thus, this file will be left unchanged.

---

lcp_semant_.lisp

(status date) is used in this compiler source to establish the compiler version number. It creates such a version number with this expression:

```
(setq compiler-version (catenate "Multics LISP Compiler, Version " compiler-revision ", "
       (cv-date- (status date))))
```

The function `cv-date-` is defined thus:

```
(defun cv-date- (x)
    (do ((ml '(January February March April May June July August September
               October November December)(cdr ml))
         (yr (+ 1900. (car x)))
         (mx 1 (1+ mx))
         (dy (caddr x)))
        ((= (cadr x) mx)
         ((lambda (base *nopoint)
                  (catenate (car ml) " " (maknam (exploden dy)) ", "
                            (maknam (exploden yr))))
          10. t))))
```

It needs to have the expression `(yr (+ 1900. (car x)))` changed to `(yr (car x))`.

Those appear to be the only callers of `(status date)`.

Version Number Incrementing

Because the `(status date)` behavior has changed in the lisp interpreter, the version number of the interpreter will be increment from the current value of 3 to 4. This value is currently represented (in Multics Maclisp, but not all other versions of Maclisp) as a FIXNUM. Thus, character suffixes or version numbers containing "." are not allowed.

Because the lisp compiler has changed (lcp_semant_.lisp), albeit in a minor way, and because the lisp compiler version number is prompted for during the generation of the lisp saved

environment (compiler.sv.lisp), it will be incremented. This value is a string and currently has the value 2.13c. With this change, the version will be incremented to 2.13d.

Because the behavior of the emacs `(date)` function has changed, and because the emacs command `make-wall-chart` now emits a 4-digit year, the version number of emacs will be incremented as well. It is currently 12.9. It will increment to version 12.10

Testing of the Change

In order to test these changes, the lisp interpreter has to be rebuilt, as does the lisp compiler and emacs. The lisp compiler is actually saved as a saved lisp image, as is emacs. So in both cases, new saved images must be created.

Once these artifacts are in place, the function of (status date) in the interpreter will be tested to make sure it returns proper values. The compiler will be used to compile itself and emacs. Emacs will be tested to make sure callers of the emacs `(date)` function perform as expected and that it appears to work, in general.

Implications for Users

It is, of course possible, that there exist user lisp programs not part of the Multics standard installation, that rely on `(status date)`'s returning 2-digit years. The release notes for the next Multics release that includes this fix should let users know that they should check their Lisp code for uses of `(status date)` and make appropriate changes. These release notes should provide some hints as to what to look for and what to do to accommodate the new behavior.

Note that any start_up.lisp files in users' home directories that reference (status date) and (status lispversion) may need adjusting. This is highly unlikely.

Similarly, any start_up.emacs files in users' home directories that reference `(status date)`, `(status lispversion)`, or `(date)` will have to be updated to handle 4-digit years. Again, the use of any of these in start_up.emacs files is very unlikely.

Bug Reference

- Reference URL of Multics Change Ticket: http://multics-trac.swenson.org/ticket/132.

Documentation

The release notes of the next Multics release will document the new lisp and emacs behavior. A warning will be included to check for user or site lisp/emacs code that might reference the changed functions.

Version History

Date	Revision	Author	Comment
2018-07-28	1.0	Eric Swenson	Initial version of MCR.
2018-07-29	1.1	Eric Swenson	Added Implications for Users section.
2018-07-30	1.2	Eric Swenson	Updated to include Version Increment section.