```
#|

 Copyright (C) 1994 by Yuan Yu.  All Rights Reserved.

 This script is hereby placed in the public domain, and therefore unlimited
 editing and redistribution is permitted.

 NO WARRANTY

 Yuan Yu PROVIDES ABSOLUTELY NO WARRANTY.  THE EVENT SCRIPT IS PROVIDED "AS IS"
 WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT
 LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
 SCRIPT IS WITH YOU.  SHOULD THE SCRIPT PROVE DEFECTIVE, YOU ASSUME THE COST OF
 ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

 IN NO EVENT WILL Yuan Yu BE LIABLE TO YOU FOR ANY DAMAGES, ANY LOST PROFITS,
 LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT
 OF THE USE OR INABILITY TO USE THIS SCRIPT (INCLUDING BUT NOT LIMITED TO LOSS
 OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD
 PARTIES), EVEN IF YOU HAVE ADVISED US OF THE POSSIBILITY OF SUCH DAMAGES, OR
 FOR ANY CLAIM BY ANY OTHER PARTY.

|#
```

EVENT: Start with the library "mc20-2" using the compiled version.

```
;          Proof of the Correctness of the STRXFRM Function
#|
This is part of our effort to verify the Berkeley string library.  The
Berkeley string library is widely used as part of the Berkeley Unix OS.

This is the source code of strxfrm function in the Berkeley string library.

/*
 * Transform src, storing the result in dst, such that
 * strcmp() on transformed strings returns what strcoll()
 * on the original untransformed strings would return.
 */
size_t
strxfrm(dst, src, n)
        register char *dst;
        register const char *src;
```

```
        register size_t n;
{
        register size_t r = 0;
        register int c;

        /*
         * Since locales are unimplemented, this is just a copy.
         */
        if (n != 0) {
                while ((c = *src++) != 0) {
                        r++;
                        if (--n == 0) {
                                while (*src++ != 0)
                                        r++;
                                break;
                        }
                        *dst++ = c;
                }
                *dst = 0;
        }
        return (r);
}
```

The MC68020 assembly code of the C function strxfrm on SUN-3 is given as follows. This binary is generated by "gcc -O".

```
0x23a0 <strxfrm>:       linkw fp,#0
0x23a4 <strxfrm+4>:     movel d2,sp@-
0x23a6 <strxfrm+6>:     moveal fp@(8),a1
0x23aa <strxfrm+10>:    moveal fp@(12),a0
0x23ae <strxfrm+14>:    movel fp@(16),d0
0x23b2 <strxfrm+18>:    clrl d1
0x23b4 <strxfrm+20>:    tstl d0
0x23b6 <strxfrm+22>:    beq 0x23d4 <strxfrm+52>
0x23b8 <strxfrm+24>:    bra 0x23cc <strxfrm+44>
0x23ba <strxfrm+26>:    addql #1,d1
0x23bc <strxfrm+28>:    subl #1,d0
0x23be <strxfrm+30>:    bne 0x23ca <strxfrm+42>
0x23c0 <strxfrm+32>:    bra 0x23c4 <strxfrm+36>
0x23c2 <strxfrm+34>:    addql #1,d1
0x23c4 <strxfrm+36>:    tstb a0@+
0x23c6 <strxfrm+38>:    bne 0x23c2 <strxfrm+34>
0x23c8 <strxfrm+40>:    bra 0x23d2 <strxfrm+50>
0x23ca <strxfrm+42>:    moveb d2,a1@+
```

```
0x23cc <strxfrm+44>:    moveb a0@+,d2
0x23ce <strxfrm+46>:    extbl d2
0x23d0 <strxfrm+48>:    bne 0x23ba <strxfrm+26>
0x23d2 <strxfrm+50>:    clrb a1@
0x23d4 <strxfrm+52>:    movel d1,d0
0x23d6 <strxfrm+54>:    movel fp@(-4),d2
0x23da <strxfrm+58>:    unlk fp
0x23dc <strxfrm+60>:    rts

The machine code of the above program is:

<strxfrm>:      0x4e56  0x0000  0x2f02  0x226e  0x0008  0x206e  0x000c  0x202e
<strxfrm+16>:   0x0010  0x4281  0x4a80  0x671c  0x6012  0x5281  0x5380  0x660a
<strxfrm+32>:   0x6002  0x5281  0x4a18  0x66fa  0x6008  0x12c2  0x1418  0x49c2
<strxfrm+48>:   0x66e8  0x4211  0x2001  0x242e  0xfffc  0x4e5e  0x4e75

'(78     86      0       0       47      2       34      110
  0      8       32      110     0       12      32      46
  0      16      66      129     74      128     103     28
  96     18      82      129     83      128     102     10
  96     2       82      129     74      24      102     250
  96     8       18      194     20      24      73      194
  102    232     66      17      32      1       36      46
  255    252     78      94      78      117)
|#

; in the logic, the above program is defined by (strxfrm-code).
```

DEFINITION:
STRXFRM-CODE
= '(78 86 0 0 47 2 34 110 0 8 32 110 0 12 32 46 0 16 66
129 74 128 103 28 96 18 82 129 83 128 102 10 96 2 82
129 74 24 102 250 96 8 18 194 20 24 73 194 102 232
66 17 32 1 36 46 255 252 78 94 78 117)

`; the Berkeley strxfrm returns the following value.  It seems a bug!`

DEFINITION:
strxfrm-n ($n2$, $lst2$, n)
= **if** $n \simeq 0$ **then** 0
else strlen (0, $n2$, $lst2$) **endif**

`; the computation time of the program.`

DEFINITION:

strxfrm-t2 (*j*, *n2*, *lst2*)
= **if** *j* < *n2*
then **if** get-nth (*j*, *lst2*) = 0 **then** 8
else splus (3, strxfrm-t2 (1 + *j*, *n2*, *lst2*)) **endif**
else 0 **endif**

DEFINITION:
strxfrm-t1 (*i*, *n2*, *lst2*) = splus (7, strxfrm-t2 (1 + *i*, *n2*, *lst2*))

DEFINITION:
strxfrm-t0 (*i*, *n2*, *lst2*, *n*)
= **if** get-nth (*i*, *lst2*) = 0 **then** 8
elseif (*n* − 1) = 0 **then** strxfrm-t1 (*i*, *n2*, *lst2*)
else splus (7, strxfrm-t0 (1 + *i*, *n2*, *lst2*, *n* − 1)) **endif**

DEFINITION:
strxfrm-t (*n2*, *lst2*, *n*)
= **if** *n* ≃ 0 **then** 12
else splus (9, strxfrm-t0 (0, *n2*, *lst2*, *n*)) **endif**

```
; two induction hints.
```

DEFINITION:
strxfrm-induct2 (*s*, *j**, *j*, *n2*, *lst2*)
= **if** *j* < *n2*
then **if** get-nth (*j*, *lst2*) = 0 **then** **t**
else strxfrm-induct2 (stepn (*s*, 3),
add (32, *j**, 1),
1 + *j*,
n2,
lst2) **endif**
else **t** **endif**

DEFINITION:
strxfrm-induct1 (*s*, *i**, *i*, *lst1*, *lst2*, *n*)
= **if** get-nth (*i*, *lst2*) = 0 **then** **t**
elseif (*n* − 1) = 0 **then** **t**
else strxfrm-induct1 (stepn (*s*, 7),
add (32, *i**, 1),
1 + *i*,
put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
lst2,
n − 1) **endif**

```
; the preconditions of the initial state.
```

DEFINITION:
strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
= ((mc-status (s) = 'running)
∧ evenp (mc-pc (s))
∧ rom-addrp (mc-pc (s), mc-mem (s), 62)
∧ mcode-addrp (mc-pc (s), mc-mem (s), STRXFRM-CODE)
∧ ram-addrp (sub (32, 8, read-sp (s)), mc-mem (s), 24)
∧ ram-addrp (*str1*, mc-mem (s), *n1*)
∧ mem-lst (1, *str1*, mc-mem (s), *n1*, *lst1*)
∧ ram-addrp (*str2*, mc-mem (s), *n2*)
∧ mem-lst (1, *str2*, mc-mem (s), *n2*, *lst2*)
∧ disjoint (sub (32, 8, read-sp (s)), 24, *str1*, *n1*)
∧ disjoint (sub (32, 8, read-sp (s)), 24, *str2*, *n2*)
∧ disjoint (*str1*, *n1*, *str2*, *n2*)
∧ (*str1* = read-mem (add (32, read-sp (s), 4), mc-mem (s), 4))
∧ (*str2* = read-mem (add (32, read-sp (s), 8), mc-mem (s), 4))
∧ (n = uread-mem (add (32, read-sp (s), 12), mc-mem (s), 4))
∧ (slen (0, *n2*, *lst2*) < *n2*)
∧ (*n2* ≤ *n1*)
∧ (*n1* ∈ **N**)
∧ (*n2* ∈ **N**)
∧ uint-rangep (*n1*, 32)
∧ uint-rangep (*n2*, 32))

```
; an intermediate state s0.
```

DEFINITION:
strxfrm-s0p (s, $i*$, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
= ((mc-status (s) = 'running)
∧ evenp (mc-pc (s))
∧ rom-addrp (sub (32, 44, mc-pc (s)), mc-mem (s), 62)
∧ mcode-addrp (sub (32, 44, mc-pc (s)), mc-mem (s), STRXFRM-CODE)
∧ ram-addrp (sub (32, 4, read-an (32, 6, s)), mc-mem (s), 24)
∧ ram-addrp (*str1*, mc-mem (s), *n1*)
∧ mem-lst (1, *str1*, mc-mem (s), *n1*, *lst1*)
∧ ram-addrp (*str2*, mc-mem (s), *n2*)
∧ mem-lst (1, *str2*, mc-mem (s), *n2*, *lst2*)
∧ disjoint (sub (32, 4, read-an (32, 6, s)), 24, *str1*, *n1*)
∧ disjoint (sub (32, 4, read-an (32, 6, s)), 24, *str2*, *n2*)
∧ disjoint (*str1*, *n1*, *str2*, *n2*)
∧ equal* (read-an (32, 1, s), add (32, *str1*, $i*$))
∧ equal* (read-an (32, 0, s), add (32, *str2*, $i*$))
∧ (n = nat-to-uint (read-dn (32, 0, s)))
∧ ($i*$ = read-dn (32, 1, s))

$\wedge \quad (i = \text{nat-to-uint}\,(i*))$
$\wedge \quad (n \neq 0)$
$\wedge \quad (\text{slen}\,(i,\ n2,\ lst2) < n2)$
$\wedge \quad (n2 \leq n1)$
$\wedge \quad (i < n2)$
$\wedge \quad (n1 \in \mathbf{N})$
$\wedge \quad (n2 \in \mathbf{N})$
$\wedge \quad \text{uint-rangep}\,(n1,\ 32)$
$\wedge \quad \text{uint-rangep}\,(n2,\ 32))$

`; an intermediate state s1.`

DEFINITION:
strxfrm-s1p $(s,\ i*,\ i,\ str1,\ n1,\ lst1,\ j*,\ j,\ str2,\ n2,\ lst2)$
$= \quad ((\text{mc-status}\,(s) = \texttt{'running})$
$\wedge \quad \text{evenp}\,(\text{mc-pc}\,(s))$
$\wedge \quad \text{rom-addrp}\,(\text{sub}\,(32,\ 36,\ \text{mc-pc}\,(s)),\ \text{mc-mem}\,(s),\ 62)$
$\wedge \quad \text{mcode-addrp}\,(\text{sub}\,(32,\ 36,\ \text{mc-pc}\,(s)),\ \text{mc-mem}\,(s),\ \text{STRXFRM-CODE})$
$\wedge \quad \text{ram-addrp}\,(\text{sub}\,(32,\ 4,\ \text{read-an}\,(32,\ 6,\ s)),\ \text{mc-mem}\,(s),\ 24)$
$\wedge \quad \text{ram-addrp}\,(str1,\ \text{mc-mem}\,(s),\ n1)$
$\wedge \quad \text{mem-lst}\,(1,\ str1,\ \text{mc-mem}\,(s),\ n1,\ lst1)$
$\wedge \quad \text{ram-addrp}\,(str2,\ \text{mc-mem}\,(s),\ n2)$
$\wedge \quad \text{mem-lst}\,(1,\ str2,\ \text{mc-mem}\,(s),\ n2,\ lst2)$
$\wedge \quad \text{disjoint}\,(\text{sub}\,(32,\ 4,\ \text{read-an}\,(32,\ 6,\ s)),\ 24,\ str1,\ n1)$
$\wedge \quad \text{disjoint}\,(\text{sub}\,(32,\ 4,\ \text{read-an}\,(32,\ 6,\ s)),\ 24,\ str2,\ n2)$
$\wedge \quad \text{disjoint}\,(str1,\ n1,\ str2,\ n2)$
$\wedge \quad \text{equal*}\,(\text{read-an}\,(32,\ 1,\ s),\ \text{add}\,(32,\ str1,\ i*))$
$\wedge \quad \text{equal*}\,(\text{read-an}\,(32,\ 0,\ s),\ \text{add}\,(32,\ str2,\ j*))$
$\wedge \quad (j* = \text{read-dn}\,(32,\ 1,\ s))$
$\wedge \quad (j = \text{nat-to-uint}\,(j*))$
$\wedge \quad (i < n1)$
$\wedge \quad (\text{slen}\,(j,\ n2,\ lst2) < n2)$
$\wedge \quad (i* \in \mathbf{N})$
$\wedge \quad \text{nat-rangep}\,(i*,\ 32)$
$\wedge \quad (i = \text{nat-to-uint}\,(i*))$
$\wedge \quad (n1 \in \mathbf{N})$
$\wedge \quad (n2 \in \mathbf{N})$
$\wedge \quad \text{uint-rangep}\,(n1,\ 32)$
$\wedge \quad \text{uint-rangep}\,(n2,\ 32))$

`; from the initial state s to exit: s --> sn, when n = 0.`

THEOREM: strxfrm-s-sn
$(\text{strxfrm-statep}\,(s,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \wedge (n \simeq 0))$
$\rightarrow \quad ((\text{mc-status}\,(\text{stepn}\,(s,\ 12)) = \texttt{'running})$

$\wedge$ (mc-pc (stepn (s, 12)) = rts-addr (s))
$\wedge$ mem-lst (1, *str1*, mc-mem (stepn (s, 12)), *n1*, *lst1*)
$\wedge$ (uread-dn (32, 0, stepn (s, 12)) = 0)
$\wedge$ (read-rn (32, 15, mc-rfile (stepn (s, 12)))
= add (32, read-an (32, 7, s), 4))
$\wedge$ (read-rn (32, 14, mc-rfile (stepn (s, 12))) = read-an (32, 6, s)))

THEOREM: strxfrm-s-sn-rfile
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\wedge$ ($n \simeq 0$)
$\wedge$ (*oplen* $\leq$ 32)
$\wedge$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 12)))
= read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: strxfrm-s-sn-mem
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\wedge$ ($n \simeq 0$)
$\wedge$ disjoint (x, k, sub (32, 8, read-sp (s)), 24))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 12)), k) = read-mem (x, mc-mem (s), k))

```
; from the initial state to s0: s --> s0, when n =\= 0.
```

THEOREM: strxfrm-s-s0
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n) $\wedge$ ($n \not\simeq 0$))
$\rightarrow$ strxfrm-s0p (stepn (s, 9), 0, 0, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)

THEOREM: strxfrm-s-s0-else
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n) $\wedge$ ($n \not\simeq 0$))
$\rightarrow$ ((linked-rts-addr (stepn (s, 9)) = rts-addr (s))
$\wedge$ (linked-a6 (stepn (s, 9)) = read-an (32, 6, s))
$\wedge$ (read-rn (32, 14, mc-rfile (stepn (s, 9)))
= sub (32, 4, read-sp (s)))
$\wedge$ (rn-saved (stepn (s, 9)) = read-dn (32, 2, s)))

THEOREM: strxfrm-s-s0-rfile
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n) $\wedge$ ($n \not\simeq 0$) $\wedge$ d3-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 9)))
= read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: strxfrm-s-s0-mem
(strxfrm-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ disjoint (x, k, sub (32, 8, read-sp (s)), 24))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 9)), k) = read-mem (x, mc-mem (s), k))

```
; from s1 to exit: s1 --> sn.  By induction.
; base case:  s1 --> sn, when lst2[i] == 0.
```

THEOREM: strxfrm-s1-sn-base
(strxfrm-s1p (s, $i*$, i, *str1*, *n1*, *lst1*, $j*$, j, *str2*, *n2*, *lst2*)
$\land$ (get-nth (j, *lst2*) $=$ `0`))
$\rightarrow$ ((mc-status (stepn (s, `8`)) $=$ `'running`)
  $\land$ (mc-pc (stepn (s, `8`)) $=$ linked-rts-addr (s))
  $\land$ mem-lst (`1`, *str1*, mc-mem (stepn (s, `8`)), *n1*, put-nth (`0`, i, *lst1*))
  $\land$ (uread-dn (`32`, `0`, stepn (s, `8`)) $=$ j)
  $\land$ (read-rn (`32`, `14`, mc-rfile (stepn (s, `8`))) $=$ linked-a6 (s))
  $\land$ (read-rn (`32`, `15`, mc-rfile (stepn (s, `8`)))
    $=$ add (`32`, read-an (`32`, `6`, s), `8`)))

THEOREM: strxfrm-s1-sn-rfile-base
(strxfrm-s1p (s, $i*$, i, *str1*, *n1*, *lst1*, $j*$, j, *str2*, *n2*, *lst2*)
$\land$ (get-nth (j, *lst2*) $=$ `0`)
$\land$ (*oplen* $\leq$ `32`)
$\land$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, `8`)))
  $=$ **if** d3-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (s))
    else head (rn-saved (s), *oplen*) **endif**)

THEOREM: strxfrm-s1-sn-mem-base
(strxfrm-s1p (s, $i*$, i, *str1*, *n1*, *lst1*, $j*$, j, *str2*, *n2*, *lst2*)
$\land$ (get-nth (j, *lst2*) $=$ `0`)
$\land$ disjoint (x, k, *str1*, *n1*))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, `8`)), k) $=$ read-mem (x, mc-mem (s), k))

```
; induction case:  s1 --> s1.
```

THEOREM: strxfrm-s1-s1
(strxfrm-s1p (s, $i*$, i, *str1*, *n1*, *lst1*, $j*$, j, *str2*, *n2*, *lst2*)
$\land$ (get-nth (j, *lst2*) $\neq$ `0`))
$\rightarrow$ (strxfrm-s1p (stepn (s, `3`),
    $i*$,
    i,
    str1,
    n1,
    lst1,
    add (`32`, $j*$, `1`),
    $1 + j$,
    str2,
    n2,
    lst2)

$\wedge$ (read-rn (32, 14, mc-rfile (stepn (s, 3)))
  = read-rn (32, 14, mc-rfile (s)))
$\wedge$ (linked-a6 (stepn (s, 3)) = linked-a6 (s))
$\wedge$ (linked-rts-addr (stepn (s, 3)) = linked-rts-addr (s))
$\wedge$ (read-mem (x, mc-mem (stepn (s, 3)), k)
  = read-mem (x, mc-mem (s), k))
$\wedge$ (rn-saved (stepn (s, 3)) = rn-saved (s)))

THEOREM: strxfrm-s1-s1-rfile
(strxfrm-s1p (s, $i*$, i, $str1$, $n1$, $lst1$, $j*$, j, $str2$, $n2$, $lst2$)
$\wedge$ (get-nth (j, $lst2$) $\neq$ 0)
$\wedge$ d3-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 3)))
  = read-rn ($oplen$, rn, mc-rfile (s)))

```
; put together: s1 --> sn.
```

THEOREM: strxfrm-s1p-info
strxfrm-s1p (s, $i*$, i, $str1$, $n1$, $lst1$, $j*$, j, $str2$, $n2$, $lst2$) $\rightarrow$ (($j < n2$) = **t**)

THEOREM: strxfrm-s1-sn
let sn **be** stepn (s, strxfrm-t2 (j, $n2$, $lst2$))
in
strxfrm-s1p (s, $i*$, i, $str1$, $n1$, $lst1$, $j*$, j, $str2$, $n2$, $lst2$)
$\rightarrow$ ((mc-status (sn) = `'running`)
  $\wedge$ (mc-pc (sn) = linked-rts-addr (s))
  $\wedge$ mem-lst (1, $str1$, mc-mem (sn), $n1$, put-nth (0, i, $lst1$))
  $\wedge$ (uread-dn (32, 0, sn) = strlen (j, $n2$, $lst2$))
  $\wedge$ (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
  $\wedge$ (read-rn (32, 15, mc-rfile (sn))
    = add (32, read-an (32, 6, s), 8))) **endlet**

THEOREM: strxfrm-s1-sn-rfile
(strxfrm-s1p (s, $i*$, i, $str1$, $n1$, $lst1$, $j*$, j, $str2$, $n2$, $lst2$)
$\wedge$ ($oplen \leq 32$)
$\wedge$ d2-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, strxfrm-t2 (j, $n2$, $lst2$))))
  = **if** d3-7a2-5p (rn) **then** read-rn ($oplen$, rn, mc-rfile (s))
    else head (rn-saved (s), $oplen$) **endif**)

THEOREM: strxfrm-s1-sn-mem
(strxfrm-s1p (s, $i*$, i, $str1$, $n1$, $lst1$, $j*$, j, $str2$, $n2$, $lst2$)
$\wedge$ disjoint (x, k, $str1$, $n1$))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, strxfrm-t2 (j, $n2$, $lst2$))), k)
  = read-mem (x, mc-mem (s), k))

EVENT: Disable strxfrm-s1p-info.

```
; from s0 to exit: s0 --> sn.  By induction.
; base case 1. s0 --> sn, when lst2[i] = 0.
```

THEOREM: strxfrm-s0-sn-base1
let *sn* **be** stepn (s, 8)
in
(strxfrm-s0p (s, $i*$, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\land$ (get-nth (i, *lst2*) = 0))
$\rightarrow$ ((mc-status (*sn*) = 'running)
$\land$ (mc-pc (*sn*) = linked-rts-addr (s))
$\land$ mem-lst (1, *str1*, mc-mem (*sn*), *n1*, put-nth (0, i, *lst1*))
$\land$ (uread-dn (32, 0, *sn*) = i)
$\land$ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (s))
$\land$ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 6, s), 8))) **endlet**

THEOREM: strxfrm-s0-sn-rfile-base1
(strxfrm-s0p (s, $i*$, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\land$ (get-nth (i, *lst2*) = 0)
$\land$ (*oplen* $\leq$ 32)
$\land$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 8)))
= **if** d3-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (s))
else head (rn-saved (s), *oplen*) **endif**)

THEOREM: strxfrm-s0-sn-mem-base1
(strxfrm-s0p (s, $i*$, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\land$ (get-nth (i, *lst2*) = 0)
$\land$ disjoint (x, k, *str1*, *n1*))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 8)), k) = read-mem (x, mc-mem (s), k))

```
; base case 2:  s0 --> s1 --> sn, when lst2[i] =\= 0 and n-1 == 0.
; s0 --> s1.
```

THEOREM: strxfrm-s0-s1
(strxfrm-s0p (s, $i*$, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, n)
$\land$ (get-nth (i, *lst2*) $\neq$ 0)
$\land$ (($n - 1$) = 0))
$\rightarrow$ strxfrm-s1p (stepn (s, 7),
$i*$,
i,
str1,

$n1$,
$lst1$,
add (32, i^*, 1),
$1 + i$,
$str2$,
$n2$,
$lst2$)

THEOREM: strxfrm-s0-s1-else
(strxfrm-s0p (s, i^*, i, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, n)
 $\wedge$ (get-nth (i, $lst2$) $\neq$ 0)
 $\wedge$ (($n - 1$) = 0))
$\rightarrow$ ((linked-rts-addr (stepn (s, 7)) = linked-rts-addr (s))
   $\wedge$ (linked-a6 (stepn (s, 7)) = linked-a6 (s))
   $\wedge$ (read-rn (32, 14, mc-rfile (stepn (s, 7)))
     = read-rn (32, 14, mc-rfile (s)))
   $\wedge$ (rn-saved (stepn (s, 7)) = rn-saved (s)))

THEOREM: strxfrm-s0-s1-rfile
(strxfrm-s0p (s, i^*, i, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, n)
 $\wedge$ (get-nth (i, $lst2$) $\neq$ 0)
 $\wedge$ (($n - 1$) = 0)
 $\wedge$ d3-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 7)))
   = read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: strxfrm-s0-s1-mem
(strxfrm-s0p (s, i^*, i, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, n)
 $\wedge$ (get-nth (i, $lst2$) $\neq$ 0)
 $\wedge$ (($n - 1$) = 0)
 $\wedge$ disjoint (x, k, $str1$, $n1$))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 7)), k) = read-mem (x, mc-mem (s), k))

```
; s0 --> sn.
```

THEOREM: strxfrm-s0-sn-base2
let sn **be** stepn (s, strxfrm-t1 (i, $n2$, $lst2$))
in
(strxfrm-s0p (s, i^*, i, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, n)
 $\wedge$ (get-nth (i, $lst2$) $\neq$ 0)
 $\wedge$ (($n - 1$) = 0))
$\rightarrow$ ((mc-status (sn) = `'running`)
   $\wedge$ (mc-pc (sn) = linked-rts-addr (s))
   $\wedge$ mem-lst (1, $str1$, mc-mem (sn), $n1$, put-nth (0, i, $lst1$))
   $\wedge$ (uread-dn (32, 0, sn) = strlen ($1 + i$, $n2$, $lst2$))

$\wedge$ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (*s*))
$\wedge$ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 6, *s*), 8))) **endlet**

THEOREM: strxfrm-s0-sn-rfile-base2
(strxfrm-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *n*)
$\wedge$ (get-nth (*i*, *lst2*) $\neq$ 0)
$\wedge$ (($n - 1$) = 0)
$\wedge$ (*oplen* $\leq$ 32)
$\wedge$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, strxfrm-t1 (*i*, *n2*, *lst2*))))
= **if** d3-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
else head (rn-saved (*s*), *oplen*) **endif**)

THEOREM: strxfrm-s0-sn-mem-base2
(strxfrm-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *n*)
$\wedge$ (get-nth (*i*, *lst2*) $\neq$ 0)
$\wedge$ (($n - 1$) = 0)
$\wedge$ disjoint (*x*, *k*, *str1*, *n1*))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s*, strxfrm-t1 (*i*, *n2*, *lst2*))), *k*)
= read-mem (*x*, mc-mem (*s*), *k*))

```
; induction case: s0 --> s0, when lst2[i] =\= 0 and n-1 =\= 0.
```

THEOREM: strxfrm-s0-s0
(strxfrm-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *n*)
$\wedge$ (get-nth (*i*, *lst2*) $\neq$ 0)
$\wedge$ (($n - 1$) $\neq$ 0))
$\rightarrow$ (strxfrm-s0p (stepn (*s*, 7),
add (32, *i**, 1),
$1 + i$,
str1,
n1,
put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
str2,
n2,
lst2,
$n - 1$)
$\wedge$ (read-rn (32, 14, mc-rfile (stepn (*s*, 7)))
= read-rn (32, 14, mc-rfile (*s*)))
$\wedge$ (linked-a6 (stepn (*s*, 7)) = linked-a6 (*s*))
$\wedge$ (linked-rts-addr (stepn (*s*, 7)) = linked-rts-addr (*s*))
$\wedge$ (rn-saved (stepn (*s*, 7)) = rn-saved (*s*)))

THEOREM: strxfrm-s0-s0-rfile

$$\begin{array}{l}(\text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \\ \ \wedge\ \ (\text{get-nth}\,(i,\ lst2) \neq \texttt{0}) \\ \ \wedge\ \ ((n-1) \neq \texttt{0}) \\ \ \wedge\ \ \text{d3-7a2-5p}\,(rn)) \\ \rightarrow\ \ (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \texttt{7}))) \\ \qquad = \ \ \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s)))\end{array}$$

THEOREM: strxfrm-s0-s0-mem

$$\begin{array}{l}(\text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \\ \ \wedge\ \ (\text{get-nth}\,(i,\ lst2) \neq \texttt{0}) \\ \ \wedge\ \ ((n-1) \neq \texttt{0}) \\ \ \wedge\ \ \text{disjoint}\,(x,\ k,\ str1,\ n1)) \\ \rightarrow\ \ (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s,\ \texttt{7})),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k))\end{array}$$

```
; put together: s0 --> sn.
```

THEOREM: strxfrm-s0p-info

$$\text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \rightarrow ((i < n2) = \mathbf{t})$$

THEOREM: strxfrm-s0-sn

$$\begin{array}{l}\textbf{let}\ sn\ \ \textbf{be}\ \ \text{stepn}\,(s,\ \text{strxfrm-t0}\,(i,\ n2,\ lst2,\ n)) \\ \textbf{in} \\ \text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \\ \rightarrow\ \ ((\text{mc-status}\,(sn) = \texttt{'running}) \\ \qquad \wedge\ \ (\text{mc-pc}\,(sn) = \text{linked-rts-addr}\,(s)) \\ \qquad \wedge\ \ \text{mem-lst}\,(\texttt{1},\ str1,\ \text{mc-mem}\,(sn),\ n1,\ \text{strxfrm1}\,(i,\ lst1,\ lst2,\ n)) \\ \qquad \wedge\ \ (\text{uread-dn}\,(\texttt{32},\ \texttt{0},\ sn) = \text{strlen}\,(i,\ n2,\ lst2)) \\ \qquad \wedge\ \ (\text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(sn)) = \text{linked-a6}\,(s)) \\ \qquad \wedge\ \ (\text{read-rn}\,(\texttt{32},\ \texttt{15},\ \text{mc-rfile}\,(sn)) \\ \qquad\quad = \ \ \text{add}\,(\texttt{32},\ \text{read-an}\,(\texttt{32},\ \texttt{6},\ s),\ \texttt{8}))) \ \textbf{endlet}\end{array}$$

THEOREM: strxfrm-s0-sn-rfile

$$\begin{array}{l}(\text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \\ \ \wedge\ \ (oplen \leq \texttt{32}) \\ \ \wedge\ \ \text{d2-7a2-5p}\,(rn)) \\ \rightarrow\ \ (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{strxfrm-t0}\,(i,\ n2,\ lst2,\ n)))) \\ \qquad = \ \ \textbf{if}\ \text{d3-7a2-5p}\,(rn)\ \ \textbf{then}\ \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s)) \\ \qquad\quad\ \ \textbf{else}\ \text{head}\,(\text{rn-saved}\,(s),\ oplen)\ \textbf{endif})\end{array}$$

THEOREM: strxfrm-s0-sn-mem

$$\begin{array}{l}(\text{strxfrm-s0p}\,(s,\ i^*,\ i,\ str1,\ n1,\ lst1,\ str2,\ n2,\ lst2,\ n) \wedge \text{disjoint}\,(x,\ k,\ str1,\ n1)) \\ \rightarrow\ \ (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s,\ \text{strxfrm-t0}\,(i,\ n2,\ lst2,\ n))),\ k) \\ \qquad = \ \ \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k))\end{array}$$

EVENT: Disable strxfrm-s0p-info.

```
; the correctness of strxfrm.
```

THEOREM: strxfrm-correctness
let *sn* **be** stepn (s, strxfrm-t ($n2$, $lst2$, n))
in
strxfrm-statep (s, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, n)
$\rightarrow$ ((mc-status (sn) = `'running`)
 $\land$ (mc-pc (sn) = rts-addr (s))
 $\land$ (read-rn (32, 14, mc-rfile (sn))
  = read-rn (32, 14, mc-rfile (s)))
 $\land$ (read-rn (32, 15, mc-rfile (sn))
  = add (32, read-an (32, 7, s), 4))
 $\land$ ((d2-7a2-5p (rn) $\land$ ($oplen \leq 32$))
  $\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (sn))
   = read-rn ($oplen$, rn, mc-rfile (s))))
 $\land$ ((disjoint (x, k, $str1$, $n1$)
  $\land$ disjoint (x, k, sub (32, 8, read-sp (s)), 24))
  $\rightarrow$ (read-mem (x, mc-mem (sn), k)
   = read-mem (x, mc-mem (s), k)))
 $\land$ (uread-dn (32, 0, sn) = strxfrm-n ($n2$, $lst2$, n))
 $\land$ mem-lst (1, $str1$, mc-mem (sn), $n1$, strxfrm ($lst1$, $lst2$, n))) **endlet**

EVENT: Disable strxfrm-t.

```
; some properties of strxfrm.
; see file cstring.events.
```

Index