```
#|

 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 STRCPY 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 strcpy function in the Berkeley string library.

/* copy char from[] to to[] */
char *
strcpy(to, from)
        register char *to, *from;
{
        char *save = to;

        for (; *to = *from; ++from, ++to);
        return(save);
```

```
}

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

0x2558 <strcpy>:        linkw fp,#0
0x255c <strcpy+4>:      moveal fp@(8),a0
0x2560 <strcpy+8>:      moveal fp@(12),a1
0x2564 <strcpy+12>:     movel a0,d1
0x2566 <strcpy+14>:     bra 0x256c <strcpy+20>
0x2568 <strcpy+16>:     addqw #1,a1
0x256a <strcpy+18>:     addqw #1,a0
0x256c <strcpy+20>:     moveb a1@,d0
0x256e <strcpy+22>:     moveb d0,a0@
0x2570 <strcpy+24>:     bne 0x2568 <strcpy+16>
0x2572 <strcpy+26>:     movel d1,d0
0x2574 <strcpy+28>:     unlk fp
0x2576 <strcpy+30>:     rts

The machine code of the above program is:

<strcpy>:      0x4e56  0x0000  0x206e  0x0008  0x226e  0x000c  0x2208  0x6004
<strcpy+16>:   0x5249  0x5248  0x1011  0x1080  0x66f6  0x2001  0x4e5e  0x4e75

'(78      86      0       0       32      110     0       8
  34      110     0       12      34      8       96      4
  82      73      82      72      16      17      16      128
  102     246     32      1       78      94      78      117)
|#
; in the logic, the above program is defined by (strcpy-code).
```

DEFINITION:
STRCPY-CODE
= `'(78 86 0 0 32 110 0 8 34 110 0 12 34 8 96 4 82 73 82 72 16 17 16 128 102 246 32 1 78 94 78 117)`

`; the computation time of the program.`

DEFINITION:
strcpy1-t (i, *n2*, *lst2*)
= **if** $i < n2$
then if get-nth (i, *lst2*) = NULL **then** 6
else splus (5, strcpy1-t ($1 + i$, *n2*, *lst2*)) **endif**
else 0 **endif**

```
; the computation time for the program (strcpy-code).
```

DEFINITION: strcpy-t (*n2*, *lst2*) = splus (5, strcpy1-t (0, *n2*, *lst2*))

```
; an induction hint for the loop.
```

DEFINITION:
strcpy-induct (*s*, *i**, *i*, *lst1*, *n2*, *lst2*)
= **if** *i* < *n2*
then if get-nth (*i*, *lst2*) = NULL **then t**
else strcpy-induct (stepn (*s*, 5),
add (32, *i**, 1),
1 + *i*,
put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
n2,
lst2) **endif**
else t endif

```
; the pre-conditions of the initial state.
```

DEFINITION:
strcpy-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
= ((mc-status (*s*) = 'running)
∧ evenp (mc-pc (*s*))
∧ rom-addrp (mc-pc (*s*), mc-mem (*s*), 32)
∧ mcode-addrp (mc-pc (*s*), mc-mem (*s*), STRCPY-CODE)
∧ ram-addrp (sub (32, 4, read-sp (*s*)), mc-mem (*s*), 16)
∧ 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-sp (*s*)), 16, *str1*, *n1*)
∧ disjoint (sub (32, 4, read-sp (*s*)), 16, *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))
∧ (slen (0, *n2*, *lst2*) < *n2*)
∧ (*n2* ≤ *n1*)
∧ (*n1* ∈ **N**)
∧ (*n2* ∈ **N**)
∧ uint-rangep (*n1*, 32))

```
; an intermediate state.
```

DEFINITION:

strcpy-s0p (s, i^*, i, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
= ((mc-status (s) = 'running)
∧ evenp (mc-pc (s))
∧ rom-addrp (sub (32, 20, mc-pc (s)), mc-mem (s), 32)
∧ mcode-addrp (sub (32, 20, mc-pc (s)), mc-mem (s), STRCPY-CODE)
∧ ram-addrp (read-an (32, 6, s), mc-mem (s), 16)
∧ 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 (read-an (32, 6, s), 16, *str1*, *n1*)
∧ disjoint (read-an (32, 6, s), 16, *str2*, *n2*)
∧ disjoint (*str1*, *n1*, *str2*, *n2*)
∧ (*str1* = read-dn (32, 1, s))
∧ equal* (read-an (32, 0, s), add (32, *str1*, i^*))
∧ equal* (read-an (32, 1, s), add (32, *str2*, i^*))
∧ (slen (i, *n2*, *lst2*) < *n2*)
∧ (*n2* ≤ *n1*)
∧ (i < *n2*)
∧ (i^* ∈ **N**)
∧ nat-rangep (i^*, 32)
∧ (i = nat-to-uint (i^*))
∧ (*n1* ∈ **N**)
∧ (*n2* ∈ **N**)
∧ uint-rangep (*n1*, 32))

```
; from the initial state s to s0: s --> s0.
```

THEOREM: strcpy-s-s0
strcpy-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ strcpy-s0p (stepn (s, 5), 0, 0, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)

THEOREM: strcpy-s-s0-else
strcpy-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((linked-rts-addr (stepn (s, 5)) = rts-addr (s))
∧ (linked-a6 (stepn (s, 5)) = read-an (32, 6, s))
∧ (read-rn (32, 14, mc-rfile (stepn (s, 5)))
= sub (32, 4, read-sp (s))))

THEOREM: strcpy-s-s0-rfile
(strcpy-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ d2-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 5)))
= read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: strcpy-s-s0-mem

(strcpy-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ disjoint (*x*, *k*, sub (32, 4, read-sp (*s*)), 16))
→ (read-mem (*x*, mc-mem (stepn (*s*, 5)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from s0 to exit (base case), from s0 to s0 (induction case).
; base case: s0 --> exit.
```

THEOREM: strcpy-s0-sn-base
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ (get-nth (*i*, *lst2*) = 0))
→ ((mc-status (stepn (*s*, 6)) = 'running)
 ∧ (mc-pc (stepn (*s*, 6)) = linked-rts-addr (*s*))
 ∧ (read-rn (32, 0, mc-rfile (stepn (*s*, 6))) = *str1*)
 ∧ mem-lst (1, *str1*, mc-mem (stepn (*s*, 6)), *n1*, put-nth (0, *i*, *lst1*))
 ∧ (read-rn (32, 14, mc-rfile (stepn (*s*, 6))) = linked-a6 (*s*))
 ∧ (read-rn (32, 15, mc-rfile (stepn (*s*, 6)))
  = add (32, read-an (32, 6, *s*), 8)))

THEOREM: strcpy-s0-sn-rfile-base
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ d2-7a2-5p (*rn*)
 ∧ (get-nth (*i*, *lst2*) = 0))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 6)))
  = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: strcpy-s0-sn-mem-base
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ disjoint (*x*, *k*, *str1*, *n1*)
 ∧ (get-nth (*i*, *lst2*) = 0))
→ (read-mem (*x*, mc-mem (stepn (*s*, 6)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; induction case: s0 --> s0.
```

THEOREM: strcpy-s0-s0
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ (get-nth (*i*, *lst2*) ≠ 0))
→ (strcpy-s0p (stepn (*s*, 5),
  add (32, *i**, 1),
  1 + *i*,
  str1,
  n1,
  put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
  str2,
  n2,
  lst2)
 ∧ (read-rn (32, 14, mc-rfile (stepn (*s*, 5)))
  = read-rn (32, 14, mc-rfile (*s*)))
 ∧ (linked-a6 (stepn (*s*, 5)) = linked-a6 (*s*))
 ∧ (linked-rts-addr (stepn (*s*, 5)) = linked-rts-addr (*s*)))

THEOREM: strcpy-s0-s0-rfile
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ d2-7a2-5p (*rn*)
 ∧ (get-nth (*i*, *lst2*) ≠ 0))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 5)))
 = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: strcpy-s0-s0-mem
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ disjoint (*x*, *k*, *str1*, *n1*)
 ∧ (get-nth (*i*, *lst2*) ≠ 0))
→ (read-mem (*x*, mc-mem (stepn (*s*, 5)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; put together (s0 --> exit).
```

THEOREM: strcpy-s0p-info
strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) → ((*i* < *n2*) = **t**)

THEOREM: strcpy-s0-sn
let *sn* **be** stepn (*s*, strcpy1-t (*i*, *n2*, *lst2*))
in
strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((mc-status (*sn*) = 'running)
 ∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
 ∧ (read-dn (32, 0, *sn*) = *str1*)
 ∧ mem-lst (1, *str1*, mc-mem (*sn*), *n1*, strcpy (*i*, *lst1*, *n2*, *lst2*))
 ∧ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (*s*))
 ∧ (read-rn (32, 15, mc-rfile (*sn*))
 = add (32, read-an (32, 6, *s*), 8))) **endlet**

THEOREM: strcpy-s0-sn-rfile
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ d2-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, strcpy1-t (*i*, *n2*, *lst2*))))
 = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: strcpy-s0-sn-mem
(strcpy-s0p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ disjoint (*x*, *k*, *str1*, *n1*))
→ (read-mem (*x*, mc-mem (stepn (*s*, strcpy1-t (*i*, *n2*, *lst2*))), *k*)
 = read-mem (*x*, mc-mem (*s*), *k*))

EVENT: Disable strcpy-s0p-info.

```
; the correctness of the strcpy program.
```

THEOREM: strcpy-correctness
let *sn* **be** stepn (*s*, strcpy-t (*n2*, *lst2*))
in
strcpy-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((mc-status (*sn*) = `'running`)
∧ (mc-pc (*sn*) = rts-addr (*s*))
∧ (read-rn (`32`, `14`, mc-rfile (*sn*))
= read-rn (`32`, `14`, mc-rfile (*s*)))
∧ (read-rn (`32`, `15`, mc-rfile (*sn*))
= add (`32`, read-an (`32`, `7`, *s*), `4`))
∧ (d2-7a2-5p (*rn*)
→ (read-rn (*oplen*, *rn*, mc-rfile (*sn*))
= read-rn (*oplen*, *rn*, mc-rfile (*s*))))
∧ ((disjoint (*x*, *k*, *str1*, *n1*)
∧ disjoint (*x*, *k*, sub (`32`, `4`, read-sp (*s*)), `16`))
→ (read-mem (*x*, mc-mem (*sn*), *k*)
= read-mem (*x*, mc-mem (*s*), *k*)))
∧ (read-dn (`32`, `0`, *sn*) = *str1*)
∧ mem-lst (`1`, *str1*, mc-mem (*sn*), *n1*, strcpy (`0`, *lst1*, *n2*, *lst2*))) **endlet**

EVENT: Disable strcpy-t.

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

Index