```
#|

 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.

|#

;          Proof of the Correctness of the STRTOK Function
```

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

```
#|
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 strtok function in the Berkeley string library.

char *
strtok(s, delim)
        register char *s;
        register const char *delim;
{
        register const char *spanp;
        register int c, sc;
        char *tok;
        static char *last;
```

```
        if (s == NULL && (s = last) == NULL)
                return (NULL);

        /*
         * Skip (span) leading delimiters (s += strspn(s, delim), sort of).
         */
cont:
        c = *s++;
        for (spanp = delim; (sc = *spanp++) != 0;) {
                if (c == sc)
                        goto cont;
        }

        if (c == 0) {           /* no non-delimiter characters */
                last = NULL;
                return (NULL);
        }
        tok = s - 1;

        /*
         * Scan token (scan for delimiters: s += strcspn(s, delim), sort of).
         * Note that delim must have one NUL; we stop if we see that, too.
         */
        for (;;) {
                c = *s++;
                spanp = delim;
                do {
                        if ((sc = *spanp++) == c) {
                                if (c == 0)
                                        s = NULL;
                                else
                                        s[-1] = 0;
                                last = s;
                                return (tok);
                        }
                } while (sc != 0);
        }
        /* NOTREACHED */
}
```

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

```
0x2768 <strtok>:        linkw fp,#0
0x276c <strtok+4>:      moveml d2-d3,sp@-
0x2770 <strtok+8>:      moveal fp@(8),a1
0x2774 <strtok+12>:     movel fp@(12),d3
0x2778 <strtok+16>:     tstl a1
0x277a <strtok+18>:     bne 0x278a <strtok+34>
0x277c <strtok+20>:     moveal @#0x20098 <edata>,a1
0x2782 <strtok+26>:     tstl a1
0x2784 <strtok+28>:     bne 0x278a <strtok+34>
0x2786 <strtok+30>:     clrl d0
0x2788 <strtok+32>:     bra 0x27d8 <strtok+112>
0x278a <strtok+34>:     moveb a1@+,d1
0x278c <strtok+36>:     extbl d1
0x278e <strtok+38>:     moveal d3,a0
0x2790 <strtok+40>:     bra 0x2796 <strtok+46>
0x2792 <strtok+42>:     cmpl d1,d0
0x2794 <strtok+44>:     beq 0x278a <strtok+34>
0x2796 <strtok+46>:     moveb a0@+,d0
0x2798 <strtok+48>:     extbl d0
0x279a <strtok+50>:     bne 0x2792 <strtok+42>
0x279c <strtok+52>:     tstl d1
0x279e <strtok+54>:     bne 0x27aa <strtok+66>
0x27a0 <strtok+56>:     clrl @#0x20098 <edata>
0x27a6 <strtok+62>:     clrl d0
0x27a8 <strtok+64>:     bra 0x27d8 <strtok+112>
0x27aa <strtok+66>:     movel a1,d2
0x27ac <strtok+68>:     subl #1,d2
0x27ae <strtok+70>:     moveb a1@+,d1
0x27b0 <strtok+72>:     extbl d1
0x27b2 <strtok+74>:     moveal d3,a0
0x27b4 <strtok+76>:     moveb a0@+,d0
0x27b6 <strtok+78>:     extbl d0
0x27b8 <strtok+80>:     cmpl d0,d1
0x27ba <strtok+82>:     bne 0x27d2 <strtok+106>
0x27bc <strtok+84>:     tstl d1
0x27be <strtok+86>:     bne 0x27c4 <strtok+92>
0x27c0 <strtok+88>:     subal a1,a1
0x27c2 <strtok+90>:     bra 0x27c8 <strtok+96>
0x27c4 <strtok+92>:     clrb a1@(-1)
0x27c8 <strtok+96>:     movel a1,@#0x20098 <edata>
0x27ce <strtok+102>:    movel d2,d0
0x27d0 <strtok+104>:    bra 0x27d8 <strtok+112>
0x27d2 <strtok+106>:    tstl d0
```

```
0x27d4 <strtok+108>:    bne 0x27b4 <strtok+76>
0x27d6 <strtok+110>:    bra 0x27ae <strtok+70>
0x27d8 <strtok+112>:    moveml fp@(-8),d2-d3
0x27de <strtok+118>:    unlk fp
0x27e0 <strtok+120>:    rts

The machine code of the above program is:

<strtok>:      0x4e56  0x0000  0x48e7  0x3000  0x226e  0x0008  0x262e  0x000c
<strtok+16>:   0x4a89  0x660e  0x2279  0x0002  0x0098  0x4a89  0x6604  0x4280
<strtok+32>:   0x604e  0x1219  0x49c1  0x2043  0x6004  0xb081  0x67f4  0x1018
<strtok+48>:   0x49c0  0x66f6  0x4a81  0x660a  0x42b9  0x0002  0x0098  0x4280
<strtok+64>:   0x602e  0x2409  0x5382  0x1219  0x49c1  0x2043  0x1018  0x49c0
<strtok+80>:   0xb280  0x6616  0x4a81  0x6604  0x93c9  0x6004  0x4229  0xffff
<strtok+96>:   0x23c9  0x0002  0x0098  0x2002  0x6006  0x4a80  0x66de  0x60d6
<strtok+112>:  0x4cee  0x000c  0xfff8  0x4e5e  0x4e75

'(78      86      0       0       72      231     48      0
  34      110     0       8       38      46      0       12
  74      137     102     14      34      121     0       2
  0       152     74      137     102     4       66      128
  96      78      18      25      73      193     32      67
  96      4       176     129     103     244     16      24
  73      192     102     246     74      129     102     10
  66      185     0       2       0       152     66      128
  96      46      36      9       83      130     18      25
  73      193     32      67      16      24      73      192
  178     128     102     22      74      129     102     4
  147     201     96      4       66      41      255     255
  35      201     0       2       0       152     32      2
  96      6       74      128     102     222     96      214
  76      238     0       12      255     248     78      94
  78      117)
|#

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

DEFINITION:
STRTOK-CODE
```
=  '(78 86 0 0 72 231 48 0 34 110 0 8 38 46 0 12 74 137
     102 14 34 121 -1 -1 -1 -1 74 137 102 4 66 128 96 78
     18 25 73 193 32 67 96 4 176 129 103 244 16 24 73 192
     102 246 74 129 102 10 66 185 -1 -1 -1 -1 66 128 96
     46 36 9 83 130 18 25 73 193 32 67 16 24 73 192 178
```

```
128 102 22 74 129 102 4 147 201 96 4 66 41 255 255
35 201 -1 -1 -1 -1 32 2 96 6 74 128 102 222 96 214
76 238 0 12 255 248 78 94 78 117)
```

CONSERVATIVE AXIOM: strtok-load
strtok-loadp (s)
= (evenp (STRTOK-ADDR)
∧ (STRTOK-ADDR ∈ **N**)
∧ nat-rangep (STRTOK-ADDR, 32)
∧ (STRTOK-LAST-ADDR ∈ **N**)
∧ nat-rangep (STRTOK-LAST-ADDR, 32)
∧ ram-addrp (STRTOK-LAST-ADDR, mc-mem (s), 4)
∧ rom-addrp (STRTOK-ADDR, mc-mem (s), 122)
∧ mcode-addrp (STRTOK-ADDR, mc-mem (s), STRTOK-CODE)
∧ (pc-read-mem (add (32, STRTOK-ADDR, 22), mc-mem (s), 4)
= STRTOK-LAST-ADDR)
∧ (pc-read-mem (add (32, STRTOK-ADDR, 58), mc-mem (s), 4)
= STRTOK-LAST-ADDR)
∧ (pc-read-mem (add (32, STRTOK-ADDR, 98), mc-mem (s), 4)
= STRTOK-LAST-ADDR))

Simultaneously, we introduce the new function symbols *strtok-loadp*, *strtok-addr*, and *strtok-last-addr*.

THEOREM: stepn-strtok-loadp
strtok-loadp (stepn (s, n)) = strtok-loadp (s)

```
; the computation time of the program.
```

DEFINITION:
strtok-t0 (*i2*, *n2*, *lst2*, *ch*)
= **if** *i2* < *n2*
then if get-nth (*i2*, *lst2*) = 0 **then** 3
elseif get-nth (*i2*, *lst2*) = *ch* **then** 5
else splus (5, strtok-t0 (1 + *i2*, *n2*, *lst2*, *ch*)) **endif**
else 0 **endif**

DEFINITION:
strtok-t1 (*n2*, *lst2*, *ch*) = splus (4, strtok-t0 (0, *n2*, *lst2*, *ch*))

DEFINITION:
strtok-t2 (*i1*, *n1*, *lst1*, *n2*, *lst2*)
= **if** *i1* < *n1*
then if strchr1 (0, *n2*, *lst2*, get-nth (*i1*, *lst1*))
then splus (strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*)),

```
                    strtok-t2 (1 + i1, n1, lst1, n2, lst2))
        else strtok-t1 (n2, lst2, get-nth (i1, lst1)) endif
    else 0 endif

Definition:
strtok-t3 (i2, n2, lst2, ch)
=   if i2 < n2
    then if ch = get-nth (i2, lst2)
        then if ch = 0 then 14
            else 13 endif
        elseif get-nth (i2, lst2) = 0 then 7
        else splus (6, strtok-t3 (1 + i2, n2, lst2, ch)) endif
    else 0 endif

Definition:
strtok-t4 (n2, lst2, ch) = splus (3, strtok-t3 (0, n2, lst2, ch))

Definition:
strtok-t5 (i1, n1, lst1, n2, lst2)
=   if i1 < n1
    then if strchr (0, n2, lst2, get-nth (i1, lst1))
        then strtok-t4 (n2, lst2, get-nth (i1, lst1))
        else splus (strtok-t4 (n2, lst2, get-nth (i1, lst1)),
                    strtok-t5 (1 + i1, n1, lst1, n2, lst2)) endif
    else 0 endif

Definition:
strtok-t6 (i1, n1, lst1, n2, lst2)
=   if get-nth (i1, lst1) = 0 then 8
    else splus (4, strtok-t5 (1 + i1, n1, lst1, n2, lst2)) endif

Definition:
strtok-t (str1, last, n1, lst1, n2, lst2)
=   if nat-to-uint (str1) = 0
    then if nat-to-uint (last) = 0 then 14
        else splus (9,
                    splus (strtok-t2 (0, n1, lst1, n2, lst2),
                           strtok-t6 (strspn (0, n1, lst1, n2, lst2),
                                      n1,
                                      lst1,
                                      n2,
                                      lst2))) endif
    else splus (6,
                splus (strtok-t2 (0, n1, lst1, n2, lst2),
                       strtok-t6 (strspn (0, n1, lst1, n2, lst2),
```

n1,
lst1,
n2,
lst2))) **endif**

```
; two induction hints.
```

DEFINITION:
strtok-induct0 (*s*, *i2**, *i2*, *n2*, *lst2*, *ch*)
= **if** *i2* < *n2*
then if get-nth (*i2*, *lst2*) = `0` **then t**
elseif get-nth (*i2*, *lst2*) = *ch* **then t**
else strtok-induct0 (stepn (*s*, `5`),
add (`32`, *i2**, `1`),
1 + *i2*,
n2,
lst2,
ch) **endif**
else t endif

DEFINITION:
strtok-induct1 (*s*, *i1**, *i1*, *n1*, *lst1*, *n2*, *lst2*)
= **if** *i1* < *n1*
then if strchr1 (`0`, *n2*, *lst2*, get-nth (*i1*, *lst1*))
then strtok-induct1 (stepn (*s*, strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*))),
add (`32`, *i1**, `1`),
1 + *i1*,
n1,
lst1,
n2,
lst2)
else t endif
else t endif

DEFINITION:
strtok-induct2 (*s*, *i2**, *i2*, *n2*, *lst2*, *ch*)
= **if** *i2* < *n2*
then if *ch* = get-nth (*i2*, *lst2*) **then t**
elseif get-nth (*i2*, *lst2*) = `0` **then t**
else strtok-induct2 (stepn (*s*, `6`),
add (`32`, *i2**, `1`),
1 + *i2*,
n2,
lst2,
ch) **endif**
else t endif

```
DEFINITION:
strtok-induct3 (s, i1*, i1, n1, lst1, n2, lst2)
=  if i1 < n1
   then if strchr (0, n2, lst2, get-nth (i1, lst1)) then t
        else strtok-induct3 (stepn (s,
                                    strtok-t4 (n2,
                                               lst2,
                                               get-nth (i1, lst1))),
                             add (32, i1*, 1),
                             1 + i1,
                             n1,
                             lst1,
                             n2,
                             lst2) endif
   else t endif

; the preconditions of the initial state.

DEFINITION:
strtok-statep (s, str1, n1, lst1, str2, n2, lst2)
=  ((mc-status (s) = 'running)
    ∧  strtok-loadp (s)
    ∧  (mc-pc (s) = STRTOK-ADDR)
    ∧  ram-addrp (sub (32, 12, read-sp (s)), mc-mem (s), 24)
    ∧  ram-addrp (str2, mc-mem (s), n2)
    ∧  mem-lst (1, str2, mc-mem (s), n2, lst2)
    ∧  disjoint (str2, n2, sub (32, 12, read-sp (s)), 24)
    ∧  disjoint (STRTOK-LAST-ADDR, 4, sub (32, 12, read-sp (s)), 24)
    ∧  (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))
    ∧  if nat-to-uint (str1) = 0
       then let last be read-mem (STRTOK-LAST-ADDR, mc-mem (s), 4)
            in
            if nat-to-uint (last) = 0 then t
            else ram-addrp (last, mc-mem (s), n1)
                 ∧  mem-lst (1,
                             last,
                             mc-mem (s),
                             n1,
                             lst1)
                 ∧  disjoint (last,
                              n1,
                              sub (32, 12, read-sp (s)),
                              24)
```

```
                ∧   disjoint (last,
                                n1,
                                STRTOK-LAST-ADDR,
                                4) endif endlet
        else ram-addrp (str1, mc-mem (s), n1)
                ∧   mem-lst (1, str1, mc-mem (s), n1, lst1)
                ∧   disjoint (str1, n1, sub (32, 12, read-sp (s)), 24)
                ∧   disjoint (str1, n1, STRTOK-LAST-ADDR, 4) endif
    ∧   (slen (0, n1, lst1) < n1)
    ∧   (slen (0, n2, lst2) < n2)
    ∧   (n1 ∈ N)
    ∧   uint-rangep (n1, 32)
    ∧   (n2 ∈ N)
    ∧   uint-rangep (n2, 32))

; intermediate states.

DEFINITION:
strtok-s0p (s, i1*, i1, str1, n1, lst1, str2, n2, lst2)
=   ((mc-status (s) = 'running)
    ∧   strtok-loadp (s)
    ∧   (mc-pc (s) = add (32, STRTOK-ADDR, 34))
    ∧   ram-addrp (sub (32, 8, 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 (STRTOK-LAST-ADDR, 4, sub (32, 8, read-an (32, 6, s)), 24)
    ∧   disjoint (str1, n1, STRTOK-LAST-ADDR, 4)
    ∧   disjoint (str1, n1, sub (32, 8, read-an (32, 6, s)), 24)
    ∧   disjoint (str2, n2, sub (32, 8, read-an (32, 6, s)), 24)
    ∧   equal* (read-an (32, 1, s), add (32, str1, i1*))
    ∧   (str2 = read-dn (32, 3, s))
    ∧   (i1* ∈ N)
    ∧   nat-rangep (i1*, 32)
    ∧   (i1 = nat-to-uint (i1*))
    ∧   (slen (i1, n1, lst1) < n1)
    ∧   (slen (0, n2, lst2) < n2)
    ∧   (n1 ∈ N)
    ∧   uint-rangep (n1, 32)
    ∧   (n2 ∈ N)
    ∧   uint-rangep (n2, 32))

DEFINITION:
strtok-s1p (s, i1*, i1, str1, n1, lst1, i2*, i2, str2, n2, lst2, ch)
```

= ((mc-status (*s*) = 'running)
∧ strtok-loadp (*s*)
∧ (mc-pc (*s*) = add (32, STRTOK-ADDR, 46))
∧ ram-addrp (sub (32, 8, 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 (STRTOK-LAST-ADDR, 4, sub (32, 8, read-an (32, 6, *s*)), 24)
∧ disjoint (*str1*, *n1*, STRTOK-LAST-ADDR, 4)
∧ disjoint (*str1*, *n1*, sub (32, 8, read-an (32, 6, *s*)), 24)
∧ disjoint (*str2*, *n2*, sub (32, 8, read-an (32, 6, *s*)), 24)
∧ (*str2* = read-dn (32, 3, *s*))
∧ (*ch* = uread-dn (8, 1, *s*))
∧ equal* (read-dn (32, 1, *s*), ext (8, read-dn (8, 1, *s*), 32))
∧ equal* (read-an (32, 1, *s*), add (32, *str1*, add (32, *i1**, 1)))
∧ equal* (read-an (32, 0, *s*), add (32, *str2*, *i2**))
∧ (*i1** ∈ **N**)
∧ nat-rangep (*i1**, 32)
∧ (*i1* = nat-to-uint (*i1**))
∧ (*i2** ∈ **N**)
∧ nat-rangep (*i2**, 32)
∧ (*i2* = nat-to-uint (*i2**))
∧ (slen (0, *n2*, *lst2*) < *n2*)
∧ (slen (*i1*, *n1*, *lst1*) < *n1*)
∧ (slen (*i2*, *n2*, *lst2*) < *n2*)
∧ (*n1* ∈ **N**)
∧ uint-rangep (*n1*, 32)
∧ (*n2* ∈ **N**)
∧ uint-rangep (*n2*, 32))

DEFINITION:
strtok-s2p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
= ((mc-status (*s*) = 'running)
∧ strtok-loadp (*s*)
∧ (mc-pc (*s*) = add (32, STRTOK-ADDR, 52))
∧ ram-addrp (sub (32, 8, 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 (STRTOK-LAST-ADDR, 4, sub (32, 8, read-an (32, 6, *s*)), 24)
∧ disjoint (*str1*, *n1*, STRTOK-LAST-ADDR, 4)
∧ disjoint (*str1*, *n1*, sub (32, 8, read-an (32, 6, *s*)), 24)

$\wedge$ equal* (read-dn (32, 1, s), ext (8, read-dn (8, 1, s), 32))
$\wedge$ equal* (read-an (32, 1, s), add (32, *str1*, add (32, *i1**, 1)))
$\wedge$ (*str2* = read-dn (32, 3, s))
$\wedge$ (*ch* = uread-dn (8, 1, s))
$\wedge$ (*i1** $\in$ **N**)
$\wedge$ nat-rangep (*i1**, 32)
$\wedge$ (*i1* = nat-to-uint (*i1**))
$\wedge$ (slen (*i1*, *n1*, *lst1*) $<$ *n1*)
$\wedge$ (slen (0, *n2*, *lst2*) $<$ *n2*)
$\wedge$ (*n1* $\in$ **N**)
$\wedge$ uint-rangep (*n1*, 32)
$\wedge$ (*n2* $\in$ **N**)
$\wedge$ uint-rangep (*n2*, 32))

DEFINITION:
strtok-s3p (s, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
= ((mc-status (s) = 'running)
$\wedge$ strtok-loadp (s)
$\wedge$ (mc-pc (s) = add (32, STRTOK-ADDR, 70))
$\wedge$ ram-addrp (sub (32, 8, read-an (32, 6, s)), mc-mem (s), 24)
$\wedge$ ram-addrp (*str1*, mc-mem (s), *n1*)
$\wedge$ mem-lst (1, *str1*, mc-mem (s), *n1*, *lst1*)
$\wedge$ ram-addrp (*str2*, mc-mem (s), *n2*)
$\wedge$ mem-lst (1, *str2*, mc-mem (s), *n2*, *lst2*)
$\wedge$ disjoint (STRTOK-LAST-ADDR, 4, sub (32, 8, read-an (32, 6, s)), 24)
$\wedge$ disjoint (*str1*, *n1*, STRTOK-LAST-ADDR, 4)
$\wedge$ disjoint (*str1*, *n1*, sub (32, 8, read-an (32, 6, s)), 24)
$\wedge$ (*str2* = read-dn (32, 3, s))
$\wedge$ (*tok* = read-dn (32, 2, s))
$\wedge$ equal* (read-an (32, 1, s), add (32, *str1*, *i1**))
$\wedge$ (*i1** $\in$ **N**)
$\wedge$ nat-rangep (*i1**, 32)
$\wedge$ (*i1* = nat-to-uint (*i1**))
$\wedge$ (slen (*i1*, *n1*, *lst1*) $<$ *n1*)
$\wedge$ (slen (0, *n2*, *lst2*) $<$ *n2*)
$\wedge$ (*n1* $\in$ **N**)
$\wedge$ uint-rangep (*n1*, 32)
$\wedge$ (*n2* $\in$ **N**)
$\wedge$ uint-rangep (*n2*, 32))

DEFINITION:
strtok-s4p (s, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
= ((mc-status (s) = 'running)
$\wedge$ strtok-loadp (s)

$\wedge$ (mc-pc (s) = add (32, STRTOK-ADDR, 76))
$\wedge$ ram-addrp (sub (32, 8, read-an (32, 6, s)), mc-mem (s), 24)
$\wedge$ ram-addrp ($str1$, mc-mem (s), $n1$)
$\wedge$ mem-lst (1, $str1$, mc-mem (s), $n1$, $lst1$)
$\wedge$ ram-addrp ($str2$, mc-mem (s), $n2$)
$\wedge$ mem-lst (1, $str2$, mc-mem (s), $n2$, $lst2$)
$\wedge$ disjoint (STRTOK-LAST-ADDR, 4, sub (32, 8, read-an (32, 6, s)), 24)
$\wedge$ disjoint ($str1$, $n1$, STRTOK-LAST-ADDR, 4)
$\wedge$ disjoint (sub (32, 8, read-an (32, 6, s)), 24, $str1$, $n1$)
$\wedge$ equal* (read-an (32, 1, s), add (32, $str1$, add (32, $i1*$, 1)))
$\wedge$ equal* (read-an (32, 0, s), add (32, $str2$, $i2*$))
$\wedge$ ($str2$ = read-dn (32, 3, s))
$\wedge$ (ch = uread-dn (8, 1, s))
$\wedge$ (tok = read-dn (32, 2, s))
$\wedge$ equal* (read-dn (32, 1, s), ext (8, read-dn (8, 1, s), 32))
$\wedge$ ($i1* \in \mathbf{N}$)
$\wedge$ nat-rangep ($i1*$, 32)
$\wedge$ ($i1$ = nat-to-uint ($i1*$))
$\wedge$ ($i2* \in \mathbf{N}$)
$\wedge$ nat-rangep ($i2*$, 32)
$\wedge$ ($i2$ = nat-to-uint ($i2*$))
$\wedge$ (slen (0, $n2$, $lst2$) < $n2$)
$\wedge$ (slen ($i1$, $n1$, $lst1$) < $n1$)
$\wedge$ (slen ($i2$, $n2$, $lst2$) < $n2$)
$\wedge$ ($n1 \in \mathbf{N}$)
$\wedge$ uint-rangep ($n1$, 32)
$\wedge$ ($n2 \in \mathbf{N}$)
$\wedge$ uint-rangep ($n2$, 32))

```
; from the initial state to exit.  s --> sn, when str1 == NULL & last == NULL.
```

THEOREM: strtok-s-sn
let sn **be** stepn (s, 14)
in
(strtok-statep (s, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$)
$\wedge$ (nat-to-uint ($str1$) = 0)
$\wedge$ (uread-mem (STRTOK-LAST-ADDR, mc-mem (s), 4) = 0))
$\rightarrow$ ((mc-status (sn) = 'running)
$\wedge$ (mc-pc (sn) = rts-addr (s))
$\wedge$ (read-dn (32, 0, sn) = 0)
$\wedge$ (read-mem (STRTOK-LAST-ADDR, mc-mem (sn), 4)
= read-mem (STRTOK-LAST-ADDR, mc-mem (s), 4))
$\wedge$ (read-rn (32, 15, mc-rfile (sn))
= add (32, read-an (32, 7, s), 4))
$\wedge$ (read-rn (32, 14, mc-rfile (sn)) = read-an (32, 6, s))) **endlet**

THEOREM: strtok-s-sn-rfile
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (uread-mem (STRTOK-LAST-ADDR, mc-mem (s), 4) = 0)
 ∧ (*oplen* ≤ 32)
 ∧ d2-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 14)))
 = read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: strtok-s-sn-mem
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (uread-mem (STRTOK-LAST-ADDR, mc-mem (s), 4) = 0)
 ∧ disjoint (x, k, sub (32, 12, read-sp (s)), 24))
→ (read-mem (x, mc-mem (stepn (s, 14)), k) = read-mem (x, mc-mem (s), k))

```
; from the initial state to s0.  s --> s0, when str1 =\= NULL.
```

THEOREM: strtok-s-s0-1
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ (nat-to-uint (*str1*) ≠ 0))
→ strtok-s0p (stepn (s, 6), 0, 0, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)

THEOREM: strtok-s-s0-else-1
let *s0* **be** stepn (s, 6)
in
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) ≠ 0))
→ ((linked-rts-addr (*s0*) = rts-addr (s))
 ∧ (linked-a6 (*s0*) = read-an (32, 6, s))
 ∧ (read-rn (32, 14, mc-rfile (*s0*))
 = sub (32, 4, read-sp (s)))
 ∧ (movem-saved (*s0*, 4, 8, 2)
 = readm-rn (32, '(2 3), mc-rfile (s)))) **endlet**

THEOREM: strtok-s-s0-rfile-1
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) ≠ 0)
 ∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 6)))
 = read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: strtok-s-s0-mem-1
(strtok-statep (s, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) ≠ 0)
 ∧ disjoint (x, k, sub (32, 12, read-sp (s)), 24))
→ (read-mem (x, mc-mem (stepn (s, 6)), k) = read-mem (x, mc-mem (s), k))

```
; from s to s0.  s -> s0, when str1 == NULL and last =\= NULL.
```

THEOREM: strtok-s-s0-2
let *last* **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), 4)
in
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (nat-to-uint (*last*) ≠ 0))
→ strtok-s0p (stepn (*s*, 9), 0, 0, *last*, *n1*, *lst1*, *str2*, *n2*, *lst2*) **endlet**

THEOREM: strtok-s-s0-else-2
let *s0* **be** stepn (*s*, 9),
 last **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), 4)
in
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (nat-to-uint (*last*) ≠ 0))
→ ((linked-rts-addr (*s0*) = rts-addr (*s*))
  ∧ (linked-a6 (*s0*) = read-an (32, 6, *s*))
  ∧ (read-rn (32, 14, mc-rfile (*s0*))
    = sub (32, 4, read-sp (*s*)))
  ∧ (movem-saved (*s0*, 4, 8, 2)
    = readm-rn (32, '(2 3), mc-rfile (*s*)))) **endlet**

THEOREM: strtok-s-s0-rfile-2
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (uread-mem (STRTOK-LAST-ADDR, mc-mem (*s*), 4) ≠ 0)
 ∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 9)))
  = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: strtok-s-s0-mem-2
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ (nat-to-uint (*str1*) = 0)
 ∧ (uread-mem (STRTOK-LAST-ADDR, mc-mem (*s*), 4) ≠ 0)
 ∧ disjoint (*x*, *k*, sub (32, 12, read-sp (*s*)), 24))
→ (read-mem (*x*, mc-mem (stepn (*s*, 9)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from s0 to s1.  s0 --> s1.
; s0 --> s1.
```

THEOREM: strtok-s0-s1
strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ strtok-s1p (stepn (*s*, 4),

*i1**,
i1,
str1,
n1,
lst1,
0,
0,
str2,
n2,
lst2,
get-nth (*i1*, *lst1*))

THEOREM: strtok-s0-s1-else
let *s1* **be** stepn (*s*, 4)
in
strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((linked-rts-addr (*s1*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s1*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s1*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s1*, 4, 16, 4) = movem-saved (*s*, 4, 16, 4))
∧ (read-mem (*x*, mc-mem (*s1*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s0-s1-rfile
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 4)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

```
; loop: s1 --> s1.
; base case: s1 --> s2, when lst2[i] == 0.
```

THEOREM: strtok-s1-s2-base
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
∧ (get-nth (*i2*, *lst2*) = 0))
→ (strtok-s2p (stepn (*s*, 3), *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
∧ (read-rn (32, 14, mc-rfile (stepn (*s*, 3))) = read-an (32, 6, *s*))
∧ (linked-a6 (stepn (*s*, 3)) = linked-a6 (*s*))
∧ (linked-rts-addr (stepn (*s*, 3)) = linked-rts-addr (*s*))
∧ (read-mem (*x*, mc-mem (stepn (*s*, 3)), *k*)
= read-mem (*x*, mc-mem (*s*), *k*)))

THEOREM: strtok-s1-s2-rfile-base
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
∧ (get-nth (*i2*, *lst2*) = 0)
∧ d4-7a2-5p (*rn*))

$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 3)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

```
; base case: s1 --> s0.
```

THEOREM: strtok-s1-s0-base
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (*ch* = get-nth (*i1*, *lst1*))
$\wedge$ (get-nth (*i2*, *lst2*) $\neq$ 0)
$\wedge$ (get-nth (*i2*, *lst2*) = *ch*))
$\rightarrow$ (strtok-s0p (stepn (*s*, 5),
add (32, *i1**, 1),
1 + *i1*,
str1,
n1,
lst1,
str2,
n2,
lst2)
$\wedge$ (read-rn (32, 14, mc-rfile (stepn (*s*, 5)))
= read-rn (32, 14, mc-rfile (*s*)))
$\wedge$ (linked-a6 (stepn (*s*, 5)) = linked-a6 (*s*))
$\wedge$ (linked-rts-addr (stepn (*s*, 5)) = linked-rts-addr (*s*))
$\wedge$ (read-mem (*x*, mc-mem (stepn (*s*, 5)), *k*)
= read-mem (*x*, mc-mem (*s*), *k*)))

THEOREM: strtok-s1-s0-rfile-base
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (get-nth (*i2*, *lst2*) $\neq$ 0)
$\wedge$ (get-nth (*i2*, *lst2*) = *ch*)
$\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 5)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

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

THEOREM: strtok-s1-s1
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (get-nth (*i2*, *lst2*) $\neq$ 0)
$\wedge$ (get-nth (*i2*, *lst2*) $\neq$ *ch*))
$\rightarrow$ (strtok-s1p (stepn (*s*, 5),
*i1**,
i1,
str1,
n1,

lst1,
add (32, *i2**, 1),
1 + *i2*,
str2,
n2,
lst2,
ch)
∧ (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*))
∧ (rn-saved (stepn (*s*, 5)) = rn-saved (*s*))
∧ (read-mem (*x*, mc-mem (stepn (*s*, 5)), *k*)
= read-mem (*x*, mc-mem (*s*), *k*)))

THEOREM: strtok-s1-s1-rfile
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
∧ (get-nth (*i2*, *lst2*) ≠ 0)
∧ (get-nth (*i2*, *lst2*) ≠ *ch*)
∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 5)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

```
; put together:  s1 --> s0, when (strchr1 i2 n2 lst2 ch).
```

THEOREM: strtok-s1p-info
strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
→ ((*i2* < *n2*) = **t**)

THEOREM: strtok-s1-s0
let *s0* **be** stepn (*s*, strtok-t0 (*i2*, *n2*, *lst2*, *ch*))
in
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
∧ (*ch* = get-nth (*i1*, *lst1*))
∧ strchr1 (*i2*, *n2*, *lst2*, *ch*))
→ (strtok-s0p (*s0*, add (32, *i1**, 1), 1 + *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ (read-rn (32, 14, mc-rfile (*s0*)) = read-an (32, 6, *s*))
∧ (linked-a6 (*s0*) = linked-a6 (*s*))
∧ (linked-rts-addr (*s0*) = linked-rts-addr (*s*))
∧ (movem-saved (*s0*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s0*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s1-s0-rfile
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
∧ strchr1 (*i2*, *n2*, *lst2*, *ch*)

$\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, strtok-t0 (*i2*, *n2*, *lst2*, *ch*))))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

```
; put together:  s1 --> s2, when (strchr1 i2 n2 lst2 ch).
```

THEOREM: strtok-s1-s2
let *s2* **be** stepn (*s*, strtok-t0 (*i2*, *n2*, *lst2*, *ch*))
in
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ ($\neg$ strchr1 (*i2*, *n2*, *lst2*, *ch*)))
$\rightarrow$ (strtok-s2p (*s2*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (read-rn (32, 14, mc-rfile (*s2*)) = read-an (32, 6, *s*))
$\wedge$ (linked-a6 (*s2*) = linked-a6 (*s*))
$\wedge$ (linked-rts-addr (*s2*) = linked-rts-addr (*s*))
$\wedge$ (movem-saved (*s2*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
$\wedge$ (read-mem (*x*, mc-mem (*s2*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s1-s2-rfile
(strtok-s1p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ ($\neg$ strchr1 (*i2*, *n2*, *lst2*, *ch*))
$\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, strtok-t0 (*i2*, *n2*, *lst2*, *ch*))))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

EVENT: Disable strtok-s1p-info.

```
; from s0 to s2:  s0 --> s2.
; base case: s0 --> s2.
```

THEOREM: strtok-s0-s2-base
let *s2* **be** stepn (*s*, strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
$\wedge$ ($\neg$ strchr1 (0, *n2*, *lst2*, get-nth (*i1*, *lst1*))))
$\rightarrow$ (strtok-s2p (*s2*,
*i1**,
i1,
str1,
n1,
lst1,
str2,
n2,
lst2,

get-nth (*i1*, *lst1*))
∧ (linked-rts-addr (*s2*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s2*) = linked-a6 (*s*))
∧ (read-rn (32, 14, mc-rfile (*s2*))
= read-rn (32, 14, mc-rfile (*s*)))
∧ (movem-saved (*s2*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s2*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s0-s2-rfile-base
let *s2* **be** stepn (*s*, strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ (¬ strchr1 (0, *n2*, *lst2*, get-nth (*i1*, *lst1*)))
∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (*s2*))
= read-rn (*oplen*, *rn*, mc-rfile (*s*))) **endlet**

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

THEOREM: strtok-s0-s0
let *s0* **be** stepn (*s*, strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ strchr1 (0, *n2*, *lst2*, get-nth (*i1*, *lst1*)))
→ (strtok-s0p (*s0*, add (32, *i1**, 1), 1 + *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ (linked-rts-addr (*s0*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s0*) = linked-a6 (*s*))
∧ (read-rn (32, 14, mc-rfile (*s0*))
= read-rn (32, 14, mc-rfile (*s*)))
∧ (movem-saved (*s0*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s0*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s0-s0-rfile
let *s0* **be** stepn (*s*, strtok-t1 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ strchr1 (0, *n2*, *lst2*, get-nth (*i1*, *lst1*))
∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (*s0*))
= read-rn (*oplen*, *rn*, mc-rfile (*s*))) **endlet**

```
; put together: s0 --> s2.
```

THEOREM: strtok-s0p-info
strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((*i1* ∈ **N**) ∧ ((*i1* < *n1*) = **t**))

THEOREM: strtok-s0-s2
let *s2* **be** stepn (*s*, strtok-t2 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ strtok-s2p (*s2*,
strspn* (*i1**, *i1*, *n1*, *lst1*, *n2*, *lst2*),
strspn (*i1*, *n1*, *lst1*, *n2*, *lst2*),
str1,
n1,
lst1,
str2,
n2,
lst2,
get-nth (strspn (*i1*, *n1*, *lst1*, *n2*, *lst2*), *lst1*)) **endlet**

EVENT: Disable strtok-s0p-info.

THEOREM: strtok-s0-s2-else
let *s2* **be** stepn (*s*, strtok-t2 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((linked-rts-addr (*s2*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s2*) = linked-a6 (*s*))
∧ (read-rn (32, 14, mc-rfile (*s2*))
= read-rn (32, 14, mc-rfile (*s*)))
∧ (movem-saved (*s2*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s2*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s0-s2-rfile
let *s2* **be** stepn (*s*, strtok-t2 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-s0p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*) ∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (*s2*))
= read-rn (*oplen*, *rn*, mc-rfile (*s*))) **endlet**

```
; from s2 to exit:  s2 --> sn.
```

THEOREM: strtok-s2-sn-1
let *sn* **be** stepn (*s*, 8)
in
(strtok-s2p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*) ∧ (*ch* = 0))
→ ((mc-status (*sn*) = 'running)
∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
∧ (read-dn (32, 0, *sn*) = 0)

$\wedge$ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), 4) = 0)
$\wedge$ mem-lst (1, *str1*, mc-mem (*sn*), *n1*, *lst1*)
$\wedge$ (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: strtok-s2-sn-rfile-1
(strtok-s2p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (*ch* = 0)
$\wedge$ d2-7a2-5p (*rn*)
$\wedge$ (*oplen* $\leq$ 32))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 8)))
= **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
else get-vlst (*oplen*, 0, *rn*, '(2 3), movem-saved (*s*, 4, 8, 2)) **endif**)

THEOREM: strtok-s2-sn-mem-1
(strtok-s2p (*s*, *i**, *i*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (*ch* = 0)
$\wedge$ disjoint (*x*, *k*, STRTOK-LAST-ADDR, 4))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s*, 8)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from s2 to s3:  s2 --> s3.
```

THEOREM: strtok-s2-s3
(strtok-s2p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (*ch* = get-nth (*i1*, *lst1*))
$\wedge$ (*ch* $\neq$ 0))
$\rightarrow$ strtok-s3p (stepn (*s*, 4),
add (32, *i1**, 1),
1 + *i1*,
str1,
n1,
lst1,
str2,
n2,
lst2,
add (32, *str1*, *i1**))

THEOREM: strtok-s2-s3-else
let *s3* **be** stepn (*s*, 4)
in
(strtok-s2p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*) $\wedge$ (*ch* $\neq$ 0))
$\rightarrow$ ((linked-rts-addr (*s3*) = linked-rts-addr (*s*))
$\wedge$ (linked-a6 (*s3*) = linked-a6 (*s*))
$\wedge$ (read-rn (*oplen*, 14, mc-rfile (*s3*))

$=$ read-rn ($oplen$, 14, mc-rfile (s)))
$\wedge$ (movem-saved ($s3$, 4, 8, 2) = movem-saved (s, 4, 8, 2))
$\wedge$ (read-mem (x, mc-mem ($s3$), k) = read-mem (x, mc-mem (s), k))) **endlet**

THEOREM: strtok-s2-s3-rfile
(strtok-s2p (s, $i1*$, $i1$, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, ch)
$\wedge$ ($ch \neq$ 0)
$\wedge$ d4-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 4)))
$=$ read-rn ($oplen$, rn, mc-rfile (s)))

```
; from s3 to s4:  s3 --> s4.
```

THEOREM: strtok-s3-s4
strtok-s3p (s, $i1*$, $i1$, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, tok)
$\rightarrow$ strtok-s4p (stepn (s, 3),
$i1*$,
$i1$,
$str1$,
$n1$,
$lst1$,
0,
0,
$str2$,
$n2$,
$lst2$,
get-nth ($i1$, $lst1$),
tok)

THEOREM: strtok-s3-s4-else
let $s4$ **be** stepn (s, 3)
in
strtok-s3p (s, $i1*$, $i1$, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, tok)
$\rightarrow$ ((linked-rts-addr ($s4$) = linked-rts-addr (s))
$\wedge$ (linked-a6 ($s4$) = linked-a6 (s))
$\wedge$ (read-rn ($oplen$, 14, mc-rfile ($s4$))
$=$ read-rn ($oplen$, 14, mc-rfile (s)))
$\wedge$ (movem-saved ($s4$, 4, 16, 4) = movem-saved (s, 4, 16, 4))
$\wedge$ (read-mem (x, mc-mem ($s4$), k) = read-mem (x, mc-mem (s), k))) **endlet**

THEOREM: strtok-s3-s4-rfile
(strtok-s3p (s, $i1*$, $i1$, $str1$, $n1$, $lst1$, $str2$, $n2$, $lst2$, tok) $\wedge$ d4-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 3)))
$=$ read-rn ($oplen$, rn, mc-rfile (s)))

```
; from s4 to exit: s4 --> sn.
; case 1.
```

THEOREM: strtok-s4-sn-1
let *sn* **be** stepn (*s*, 13)
in
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
 ∧ (get-nth (*i2*, *lst2*) = *ch*)
 ∧ (*ch* ≠ 0))
→ ((mc-status (*sn*) = 'running)
 ∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
 ∧ (read-dn (32, 0, *sn*) = *tok*)
 ∧ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), 4)
 = add (32, *str1*, add (32, *i1**, 1)))
 ∧ mem-lst (1, *str1*, mc-mem (*sn*), *n1*, put-nth (0, *i1*, *lst1*))
 ∧ (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: strtok-s4-sn-rfile-1
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
 ∧ (get-nth (*i2*, *lst2*) = *ch*)
 ∧ (*ch* ≠ 0)
 ∧ d2-7a2-5p (*rn*)
 ∧ (*oplen* ≤ 32))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 13)))
 = **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
 else get-vlst (*oplen*, 0, *rn*, '(2 3), movem-saved (*s*, 4, 8, 2)) **endif**)

THEOREM: strtok-s4-sn-mem-1
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
 ∧ (get-nth (*i2*, *lst2*) = *ch*)
 ∧ (*ch* ≠ 0)
 ∧ disjoint (*x*, *k*, STRTOK-LAST-ADDR, 4)
 ∧ disjoint (*x*, *k*, *str1*, *n1*))
→ (read-mem (*x*, mc-mem (stepn (*s*, 13)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; case 2.
```

THEOREM: strtok-s4-sn-2
let *sn* **be** stepn (*s*, 14)
in
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
 ∧ (get-nth (*i2*, *lst2*) = *ch*)
 ∧ (*ch* = 0))

→ ((mc-status (*sn*) = `'running`)
∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
∧ (read-dn (`32`, `0`, *sn*) = *tok*)
∧ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), `4`) = `0`)
∧ mem-lst (`1`, *str1*, mc-mem (*sn*), *n1*, *lst1*)
∧ (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: strtok-s4-sn-rfile-2
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
∧ (get-nth (*i2*, *lst2*) = *ch*)
∧ (*ch* = `0`)
∧ d2-7a2-5p (*rn*)
∧ (*oplen* ≤ `32`))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, `14`)))
= **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
else get-vlst (*oplen*, `0`, *rn*, `'(2 3)`, movem-saved (*s*, `4`, `8`, `2`)) **endif**)

THEOREM: strtok-s4-sn-mem-2
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
∧ (get-nth (*i2*, *lst2*) = *ch*)
∧ (*ch* = `0`)
∧ disjoint (*x*, *k*, STRTOK-LAST-ADDR, `4`)
∧ disjoint (*x*, *k*, *str1*, *n1*))
→ (read-mem (*x*, mc-mem (stepn (*s*, `14`)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from s4 to s3:  s4 --> s3.
```

THEOREM: strtok-s4-s3-base
let *s3* **be** stepn (*s*, `7`)
in
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
∧ (*ch* = get-nth (*i1*, *lst1*))
∧ (get-nth (*i2*, *lst2*) ≠ *ch*)
∧ (get-nth (*i2*, *lst2*) = `0`))
→ (strtok-s3p (*s3*,
add (`32`, *i1**, `1`),
1 + *i1*,
str1,
n1,
lst1,
str2,
n2,
lst2,

tok)
∧ (linked-rts-addr (*s3*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s3*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s3*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s3*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s3*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s4-s3-rfile-base
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
∧ (get-nth (*i2*, *lst2*) ≠ *ch*)
∧ (get-nth (*i2*, *lst2*) = 0)
∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 7)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

```
; from s4 to s4:  s4 --> s4.
```

THEOREM: strtok-s4-s4
let *s4* **be** stepn (*s*, 6)
in
(strtok-s4p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
∧ (get-nth (*i2*, *lst2*) ≠ *ch*)
∧ (get-nth (*i2*, *lst2*) ≠ 0))
→ (strtok-s4p (*s4*,
*i1**,
i1,
str1,
n1,
lst1,
add (32, *i2**, 1),
1 + *i2*,
str2,
n2,
lst2,
ch,
tok)
∧ (linked-rts-addr (*s4*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s4*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s4*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s4*, 4, 8, 2) = movem-saved (*s*, 4, 8, 2))
∧ (read-mem (*x*, mc-mem (*s4*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s4-s4-rfile

```
(strtok-s4p (s, i1*, i1, str1, n1, lst1, i2*, i2, str2, n2, lst2, ch, tok)
 ∧   (get-nth (i2, lst2) ≠ ch)
 ∧   (get-nth (i2, lst2) ≠ 0)
 ∧   d4-7a2-5p (rn))
→   (read-rn (oplen, rn, mc-rfile (stepn (s, 6)))
      =   read-rn (oplen, rn, mc-rfile (s)))

; put together: s4 --> sn.

THEOREM: strtok-s4-sn
let sn be stepn (s, strtok-t3 (i2, n2, lst2, ch))
in
(strtok-s4p (s, i1*, i1, str1, n1, lst1, i2*, i2, str2, n2, lst2, ch, tok)
 ∧   strchr (i2, n2, lst2, ch))
→   ((mc-status (sn) = 'running)
      ∧   (mc-pc (sn) = linked-rts-addr (s))
      ∧   (read-dn (32, 0, sn) = tok)
      ∧   (read-mem (STRTOK-LAST-ADDR, mc-mem (sn), 4)
            =   if ch = 0 then 0
                else add (32, str1, add (32, i1*, 1)) endif)
      ∧   mem-lst (1, str1, mc-mem (sn), n1, strtok-lst0 (i1, lst1, ch))
      ∧   (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: strtok-s4-sn-rfile
let sn be stepn (s, strtok-t3 (i2, n2, lst2, ch))
in
(strtok-s4p (s, i*, i, str1, n1, lst1, i2*, i2, str2, n2, lst2, ch, tok)
 ∧   strchr (i2, n2, lst2, ch)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
      =   if d4-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
          else get-vlst (oplen,
                         0,
                         rn,
                         '(2 3),
                         movem-saved (s, 4, 8, 2)) endif) endlet

THEOREM: strtok-s4-sn-mem
let sn be stepn (s, strtok-t3 (i2, n2, lst2, ch))
in
(strtok-s4p (s, i*, i, str1, n1, lst1, i2*, i2, str2, n2, lst2, ch, tok)
 ∧   strchr (i2, n2, lst2, ch)
```

$\wedge$ disjoint (x, k, STRTOK-LAST-ADDR, 4)
$\wedge$ disjoint (x, k, *str1*, *n1*))
$\rightarrow$ (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) **endlet**

```
; put together: s4 --> s3.
```

THEOREM: strtok-s4p-info
strtok-s4p (s, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
$\rightarrow$ ((*i2* < *n2*) = **t**)

THEOREM: strtok-s4-s3
let *s3* **be** stepn (s, strtok-t3 (*i2*, *n2*, *lst2*, *ch*))
in
(strtok-s4p (s, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
$\wedge$ (*ch* = get-nth (*i1*, *lst1*))
$\wedge$ ($\neg$ strchr (*i2*, *n2*, *lst2*, *ch*)))
$\rightarrow$ (strtok-s3p (*s3*,
add (32, *i1**, 1),
1 + *i1*,
str1,
n1,
lst1,
str2,
n2,
lst2,
tok)
$\wedge$ (linked-rts-addr (*s3*) = linked-rts-addr (s))
$\wedge$ (linked-a6 (*s3*) = linked-a6 (s))
$\wedge$ (read-rn (*oplen*, 14, mc-rfile (*s3*))
= read-rn (*oplen*, 14, mc-rfile (s)))
$\wedge$ (movem-saved (*s3*, 4, 8, 2) = movem-saved (s, 4, 8, 2))
$\wedge$ (read-mem (x, mc-mem (*s3*), k) = read-mem (x, mc-mem (s), k))) **endlet**

EVENT: Disable strtok-s4p-info.

THEOREM: strtok-s4-s3-rfile
let *s3* **be** stepn (s, strtok-t3 (*i2*, *n2*, *lst2*, *ch*))
in
(strtok-s4p (s, *i1**, *i1*, *str1*, *n1*, *lst1*, *i2**, *i2*, *str2*, *n2*, *lst2*, *ch*, *tok*)
$\wedge$ ($\neg$ strchr (*i2*, *n2*, *lst2*, *ch*))
$\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (*s3*))
= read-rn (*oplen*, *rn*, mc-rfile (s))) **endlet**

```
; from s3 to exit:  s3 --> sn.
```

THEOREM: strtok-s3-sn-base
let *ch* **be** get-nth (*i1*, *lst1*)
in
let *sn* **be** stepn (*s*, strtok-t4 (*n2*, *lst2*, *ch*))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
 ∧ strchr (`0`, *n2*, *lst2*, *ch*))
→ ((mc-status (*sn*) = `'running`)
 ∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
 ∧ (read-dn (`32`, `0`, *sn*) = *tok*)
 ∧ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), `4`)
 = **if** *ch* = `0` **then** `0`
 else add (`32`, *str1*, add (`32`, *i1**, `1`)) **endif**)
 ∧ mem-lst (`1`,
 str1,
 mc-mem (*sn*),
 n1,
 strtok-lst0 (*i1*, *lst1*, *ch*))
 ∧ (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 endlet**

THEOREM: strtok-s3-sn-rfile-base
let *sn* **be** stepn (*s*, strtok-t4 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
 ∧ strchr (`0`, *n2*, *lst2*, get-nth (*i1*, *lst1*))
 ∧ d2-7a2-5p (*rn*)
 ∧ (*oplen* ≤ `32`))
→ (read-rn (*oplen*, *rn*, mc-rfile (*sn*))
 = **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
 else get-vlst (*oplen*,
 `0`,
 rn,
 `'(2 3)`,
 movem-saved (*s*, `4`, `8`, `2`)) **endif**) **endlet**

THEOREM: strtok-s3-sn-mem-base
let *sn* **be** stepn (*s*, strtok-t4 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
 ∧ strchr (`0`, *n2*, *lst2*, get-nth (*i1*, *lst1*))
 ∧ disjoint (*x*, *k*, STRTOK-LAST-ADDR, `4`)
 ∧ disjoint (*x*, *k*, *str1*, *n1*))
→ (read-mem (*x*, mc-mem (*sn*), *k*) = read-mem (*x*, mc-mem (*s*), *k*)) **endlet**

```
; from s3 to s3:  s3 --> s3.
```

THEOREM: strtok-s3-s3
let *s3* **be** stepn (*s*, strtok-t4 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
 ∧ (¬ strchr (0, *n2*, *lst2*, get-nth (*i1*, *lst1*))))
→ (strtok-s3p (*s3*,
add (32, *i1**, 1),
1 + *i1*,
str1,
n1,
lst1,
str2,
n2,
lst2,
tok)
∧ (linked-rts-addr (*s3*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s3*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s3*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s3*, 4, 16, 4) = movem-saved (*s*, 4, 16, 4))
∧ (read-mem (*x*, mc-mem (*s3*), *k*) = read-mem (*x*, mc-mem (*s*), *k*))) **endlet**

THEOREM: strtok-s3-s3-rfile
let *s3* **be** stepn (*s*, strtok-t4 (*n2*, *lst2*, get-nth (*i1*, *lst1*)))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
 ∧ (¬ strchr (0, *n2*, *lst2*, get-nth (*i1*, *lst1*)))
 ∧ d4-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (*s3*))
= read-rn (*oplen*, *rn*, mc-rfile (*s*))) **endlet**

```
; put together: s3 --> sn.
```

THEOREM: strtok-s3p-info
strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
→ ((*i1* ∈ **N**) ∧ ((*i1* < *n1*) = **t**))

THEOREM: strtok-s3-sn
let *sn* **be** stepn (*s*, strtok-t5 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
→ ((mc-status (*sn*) = 'running)
∧ (mc-pc (*sn*) = linked-rts-addr (*s*))

$\wedge$ (read-dn (32, 0, *sn*) = *tok*)
$\wedge$ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), 4)
= strtok-last0 (*str1*, *i1**, *i1*, *n1*, *lst1*, *n2*, *lst2*))
$\wedge$ mem-lst (1,
str1,
mc-mem (*sn*),
n1,
strtok-lst1 (*i1*, *n1*, *lst1*, *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: strtok-s3-sn-rfile
let *sn* **be** stepn (*s*, strtok-t5 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
$\wedge$ d2-7a2-5p (*rn*)
$\wedge$ (*oplen* $\leq$ 32))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (*sn*))
= **if** d4-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (*s*))
else get-vlst (*oplen*,
0,
rn,
'(2 3),
movem-saved (*s*, 4, 8, 2)) **endif**) **endlet**

THEOREM: strtok-s3-sn-mem
let *sn* **be** stepn (*s*, strtok-t5 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-s3p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *tok*)
$\wedge$ disjoint (*x*, *k*, STRTOK-LAST-ADDR, 4)
$\wedge$ disjoint (*x*, *k*, *str1*, *n1*))
$\rightarrow$ (read-mem (*x*, mc-mem (*sn*), *k*) = read-mem (*x*, mc-mem (*s*), *k*)) **endlet**

EVENT: Disable strtok-s3p-info.

```
; from s2 to exit:  s2 --> sn.
```

THEOREM: strtok-s2-sn
let *sn* **be** stepn (*s*, strtok-t6 (*i1*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-s2p (*s*, *i1**, *i1*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*, *ch*)
$\wedge$ (get-nth (*i1*, *lst1*) = *ch*))
$\rightarrow$ ((mc-status (*sn*) = 'running)

```
    ∧   (mc-pc (sn) = linked-rts-addr (s))
    ∧   (read-dn (32, 0, sn)
          =   if ch = 0 then 0
              else add (32, str1, i1*) endif)
    ∧   (read-mem (STRTOK-LAST-ADDR, mc-mem (sn), 4)
          =   strtok-last1 (str1, i1*, i1, n1, lst1, n2, lst2))
    ∧   mem-lst (1,
                 str1,
                 mc-mem (sn),
                 n1,
                 strtok-lst2 (i1, n1, 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: strtok-s2-sn-rfile

```
let ch  be get-nth (i1, lst1),
    sn  be stepn (s, strtok-t6 (i1, n1, lst1, n2, lst2))
in
(strtok-s2p (s, i1*, i1, str1, n1, lst1, str2, n2, lst2, ch)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
      =   if d4-7a2-5p (rn)  then read-rn (oplen, rn, mc-rfile (s))
          else get-vlst (oplen,
                         0,
                         rn,
                         '(2 3),
                         movem-saved (s, 4, 8, 2)) endif) endlet
```

THEOREM: strtok-s2-sn-mem

```
let ch  be get-nth (i1, lst1),
    sn  be stepn (s, strtok-t6 (i1, n1, lst1, n2, lst2))
in
(strtok-s2p (s, i1*, i1, str1, n1, lst1, str2, n2, lst2, ch)
 ∧   disjoint (x, k, STRTOK-LAST-ADDR, 4)
 ∧   disjoint (x, k, str1, n1))
→   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

```
; the correctness of strtok.
```

THEOREM: strtok-correctness

```
let last  be read-mem (STRTOK-LAST-ADDR, mc-mem (s), 4)
in
let sn  be stepn (s, strtok-t (str1, last, n1, lst1, n2, lst2))
```

in
strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
→ ((mc-status (*sn*) = `'running`)
∧ (mc-pc (*sn*) = rts-addr (*s*))
∧ (read-dn (`32`, `0`, *sn*)
= strtok-tok (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
∧ (read-mem (STRTOK-LAST-ADDR, mc-mem (*sn*), `4`)
= strtok-last (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
∧ (read-rn (`32`, `14`, mc-rfile (*sn*))
= read-rn (`32`, `14`, mc-rfile (*s*)))
∧ (read-rn (`32`, `15`, mc-rfile (*sn*))
= add (`32`, read-sp (*s*), `4`))) **endlet endlet**

THEOREM: strtok-lst-1
let *last* **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), `4`)
in
let *sn* **be** stepn (*s*, strtok-t (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ (nat-to-uint (*str1*) ≠ `0`))
→ mem-lst (`1`,
str1,
mc-mem (*sn*),
n1,
strtok-lst (*n1*, *lst1*, *n2*, *lst2*)) **endlet endlet**

THEOREM: strtok-lst-2
let *last* **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), `4`)
in
let *sn* **be** stepn (*s*, strtok-t (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
∧ (nat-to-uint (*str1*) = `0`)
∧ (nat-to-uint (*last*) ≠ `0`))
→ mem-lst (`1`,
last,
mc-mem (*sn*),
n1,
strtok-lst (*n1*, *lst1*, *n2*, *lst2*)) **endlet endlet**

THEOREM: strtok-rfile
let *last* **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), `4`)
in
let *sn* **be** stepn (*s*, strtok-t (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
in

(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ d2-7a2-5p (*rn*)
 ∧ (*oplen* ≤ 32))
→ (read-rn (*oplen*, *rn*, mc-rfile (*sn*))
 = read-rn (*oplen*, *rn*, mc-rfile (*s*))) **endlet endlet**

THEOREM: strtok-mem
let *last* **be** read-mem (STRTOK-LAST-ADDR, mc-mem (*s*), 4)
in
let *sn* **be** stepn (*s*, strtok-t (*str1*, *last*, *n1*, *lst1*, *n2*, *lst2*))
in
(strtok-statep (*s*, *str1*, *n1*, *lst1*, *str2*, *n2*, *lst2*)
 ∧ disjoint (*x*, *k*, sub (32, 12, read-sp (*s*)), 24)
 ∧ disjoint (*x*, *k*, STRTOK-LAST-ADDR, 4)
 ∧ **if** nat-to-uint (*str1*) = 0 **then** disjoint (*x*, *k*, *last*, *n1*)
 else disjoint (*x*, *k*, *str1*, *n1*) **endif**)
→ (read-mem (*x*, mc-mem (*sn*), *k*) = read-mem (*x*, mc-mem (*s*), *k*)) **endlet endlet**

EVENT: Disable strtok-t.

```
; strspn* -> strspn, strcspn* --> strcspn.
```

THEOREM: strspn*-strspn
(strspn (*i*, *n1*, *lst1*, *n2*, *lst2*)
 ∧ (*i* = nat-to-uint (*i**))
 ∧ nat-rangep (*i**, 32)
 ∧ uint-rangep (*n1*, 32))
→ (nat-to-uint (strspn* (*i**, *i*, *n1*, *lst1*, *n2*, *lst2*))
 = strspn (*i*, *n1*, *lst1*, *n2*, *lst2*))

THEOREM: strcspn*-strcspn
(strcspn (*i*, *n1*, *lst1*, *n2*, *lst2*)
 ∧ (*i* = nat-to-uint (*i**))
 ∧ nat-rangep (*i**, 32)
 ∧ uint-rangep (*n1*, 32))
→ (nat-to-uint (strcspn* (*i**, *i*, *n1*, *lst1*, *n2*, *lst2*))
 = strcspn (*i*, *n1*, *lst1*, *n2*, *lst2*))

```
; some properties of strtok.
; see the file cstring.events.
```

Index

add, 5, 7–12, 16, 17, 19, 21, 23–32

d2-7a2-5p, 13, 21, 23, 24, 26, 28, 30, 31, 33
d4-7a2-5p, 13–31
disjoint, 8–14, 21, 23, 24, 27, 28, 30, 31, 33

equal*, 9–12
evenp, 5
ext, 10–12

get-nth, 5–8, 15–31
get-vlst, 21, 23, 24, 26, 28, 30, 31

linked-a6, 13–31
linked-rts-addr, 13–29, 31

mc-mem, 5, 8–33
mc-pc, 8–12, 20, 23, 24, 26, 28, 29, 31, 32
mc-rfile, 12–33
mc-status, 8–12, 20, 23, 24, 26, 28–30, 32
mcode-addrp, 5
mem-lst, 8–12, 21, 23, 24, 26, 28, 30–32
movem-saved, 13–15, 17–31

nat-rangep, 5, 9–12, 33
nat-to-uint, 6, 8–14, 32, 33

pc-read-mem, 5
put-nth, 23

ram-addrp, 5, 8–12
read-an, 9–15, 17, 18, 21, 23, 24, 26, 28, 30, 31
read-dn, 9–12, 20, 23, 24, 26, 28, 30–32
read-mem, 8, 12–33
read-rn, 12–33
read-sp, 8, 9, 13, 14, 32, 33
readm-rn, 13, 14
rn-saved, 17
rom-addrp, 5
rts-addr, 12–14, 32

slen, 9–12
splus, 5–7
stepn, 5, 7, 8, 12–33
stepn-strtok-loadp, 5
strchr, 6, 8, 26–29
strchr1, 5, 7, 17–19
strcspn, 33
strcspn*, 33
strcspn*-strcspn, 33
strspn, 6, 20, 33
strspn*, 20, 33
strspn*-strspn, 33
strtok-addr, 5, 8–12
strtok-code, 4, 5
strtok-correctness, 31
strtok-induct0, 7
strtok-induct1, 7
strtok-induct2, 7
strtok-induct3, 8
strtok-last, 32
strtok-last-addr, 5, 8–14, 21, 23, 24, 26–28, 30–33
strtok-last0, 30
strtok-last1, 31
strtok-load, 5
strtok-loadp, 5, 8–11
strtok-lst, 32
strtok-lst-1, 32
strtok-lst-2, 32
strtok-lst0, 26, 28
strtok-lst1, 30
strtok-lst2, 31
strtok-mem, 33
strtok-rfile, 32
strtok-s-s0-1, 13

strtok-s-s0-2, 14
strtok-s-s0-else-1, 13
strtok-s-s0-else-2, 14
strtok-s-s0-mem-1, 13
strtok-s-s0-mem-2, 14
strtok-s-s0-rfile-1, 13
strtok-s-s0-rfile-2, 14
strtok-s-sn, 12
strtok-s-sn-mem, 13
strtok-s-sn-rfile, 13
strtok-s0-s0, 19
strtok-s0-s0-rfile, 19
strtok-s0-s1, 14
strtok-s0-s1-else, 15
strtok-s0-s1-rfile, 15
strtok-s0-s2, 20
strtok-s0-s2-base, 18
strtok-s0-s2-else, 20
strtok-s0-s2-rfile, 20
strtok-s0-s2-rfile-base, 19
strtok-s0p, 9, 13–20
strtok-s0p-info, 19
strtok-s1-s0, 17
strtok-s1-s0-base, 16
strtok-s1-s0-rfile, 17
strtok-s1-s0-rfile-base, 16
strtok-s1-s1, 16
strtok-s1-s1-rfile, 17
strtok-s1-s2, 18
strtok-s1-s2-base, 15
strtok-s1-s2-rfile, 18
strtok-s1-s2-rfile-base, 15
strtok-s1p, 9, 15–18
strtok-s1p-info, 17
strtok-s2-s3, 21
strtok-s2-s3-else, 21
strtok-s2-s3-rfile, 22
strtok-s2-sn, 30
strtok-s2-sn-1, 20
strtok-s2-sn-mem, 31
strtok-s2-sn-mem-1, 21
strtok-s2-sn-rfile, 31
strtok-s2-sn-rfile-1, 21
strtok-s2p, 10, 15, 18–22, 30, 31
strtok-s3-s3, 29
strtok-s3-s3-rfile, 29
strtok-s3-s4, 22
strtok-s3-s4-else, 22
strtok-s3-s4-rfile, 22
strtok-s3-sn, 29
strtok-s3-sn-base, 28
strtok-s3-sn-mem, 30
strtok-s3-sn-mem-base, 28
strtok-s3-sn-rfile, 30
strtok-s3-sn-rfile-base, 28
strtok-s3p, 11, 21, 22, 25, 27–30
strtok-s3p-info, 29
strtok-s4-s3, 27
strtok-s4-s3-base, 24
strtok-s4-s3-rfile, 27
strtok-s4-s3-rfile-base, 25
strtok-s4-s4, 25
strtok-s4-s4-rfile, 26
strtok-s4-sn, 26
strtok-s4-sn-1, 23
strtok-s4-sn-2, 23
strtok-s4-sn-mem, 26
strtok-s4-sn-mem-1, 23
strtok-s4-sn-mem-2, 24
strtok-s4-sn-rfile, 26
strtok-s4-sn-rfile-1, 23
strtok-s4-sn-rfile-2, 24
strtok-s4p, 11, 22–27
strtok-s4p-info, 27
strtok-statep, 8, 12–14, 32, 33
strtok-t, 6, 31–33
strtok-t0, 5, 17, 18
strtok-t1, 5–7, 18, 19
strtok-t2, 5, 6, 20
strtok-t3, 6, 26, 27
strtok-t4, 6, 8, 28, 29
strtok-t5, 6, 29, 30
strtok-t6, 6, 7, 30, 31
strtok-tok, 32
sub, 8–14, 33

uint-rangep, 9–12, 33
uread-dn, 10–12

uread-mem, 12–14