REPRINTED FROM

COLLEGIATE MICROCOMPUTER

PAINTING POLES AND DEKKER'S ALGORITHM

CURTIS COOPER AND LILLIAN J. COOPER

AUGUST AUTUMN 1983 VOLUME I NUMBER 3

COLLEGIATE MICROCOMPUTER
ROSE-HULMAN INSTITUTE OF TECHNOLOGY
TERRE HAUTE IN 47803 USA

PAINTING POLES AND DEKKER'S ALGORITHM

CURTIS COOPER
AND
LILLIAN J. COOPER

Addresses: Department of Mathematical Sciences, Central Missouri State University, Warrensburg MO 64093 USA.

Bendix Corporation, 2000 East 95th St. Kansas City MO 64131 USA.

ABSTRACT: In this article, a problem involving two processes and a shared critical resource is presented. Dekker's algorithm will be used to solve this problem and then the solution to this problem will be simulated on a microcomputer.

KEY WORDS: critical resource, Dekker's algorithm.

In computer systems, resources must be shared in order to be utilized efficiently. This sharing becomes particularly important when the resource allows only one user at a time. Magnetic tape drives and card readers are examples of critical resources. A compiler which is shared by several processes is another example of a critical resource. Also, shared variables that are changed by a number of processes are critical resources. These and other examples of critical resources can be found in [2] and [5]. Therefore, if two processes wish to share the use of a critical resource, they must synchronize their operations so that at most one of them controls the resource at a given time.

Dekker, a Dutch mathematician, devised an algorithm to allow two processes to share a critical resource. Later on, Dijkstra [1] and Knuth [4] generalized the algorithm to allow n processes to share a critical resource.

In this article, a problem involving two processes and a shared critical resource is presented. Dekker's algorithm will be used to solve this problem and then the solution to this problem will be simulated on a microcomputer.

THE POLE PAINTING PROBLEM

There are two painters who spend their lives painting three poles. Painter One alternates painting sections of Pole One and Pole Two with red paint and Painter Two alternates painting sections of Pole Two and Pole Three with green paint. Each pole is painted, a section at a time, from the bottom to the top of the pole. After a pole is completely painted, the next painter on the pole wipes off all the paint on the pole and the pole is painted again. A ladder is required to paint a pole, but only three ladders are provided, one ladder for each pole. Thus Painter One and Painter Two must share the ladder of Pole Two. The problem is to simulate the two painters' activities to allow mutual exclusion (only one painter is allowed on ladder two at a given time) and avoid indefinite postponement (the request by a painter for ladder two is continually denied).

The following algorithm, due to Dekker, solves the Pole Painting Problem. In the algorithm, each task a painter can perform is given a number. The solution uses the variables c1, c2, and turn. Turn is set to point to the other painter when a painter finishes painting a section of Pole Two. The variables c1 and c2 are used to indicate if Painter One or Painter Two are ready to paint Pole Two. c1 = 1 means Painter One is ready to paint Pole Two and c1 = 0 means Painter One is performing some other task. Similarly, c2 = 1 means Painter Two is ready to paint Pole Two and c2 = 0 means Painter Two is performing some other task. Initially, c1 = 0; c2 = 0; and turn = 1.

PAINTER ONE

```
1   c1 = 1;
2   do while (c2 = 1);
3     if turn = 2
        then do;
4         c1 = 0;
5         do while (turn = 2);
        end;
6         c1 = 1;
          end;
      end;
7   climb up Pole Two
8   paint red strip
9   climb down Pole Two
10  c1 = 0;
11  turn = 2;
12  run to Pole One
13  climb up Pole One
14  paint red strip
15  climb down Pole One
16  run to Pole Two
```

PAINTER TWO

```
1   c2 = 1;
2   do while (c1 = 1);
3     if turn  = 1
        then do;
4         c2 = 0;
5         do while (turn = 1);
            end;
6         c2 = 1;
          end;
      end;
7   climb up Pole Two
8   paint green strip
9   climb down Pole Two
10  c2 = 0;
11  turn = 1;
12  run to Pole Three
13  climb up Pole Three
14  paint green strip
15  climb down Pole Three
16  run to Pole Two
```

Painter One can paint Pole Two if c2 = 0. This can happen if Painter Two is painting Pole Three or if turn = 1 and Painter Two is in its inner loop. Painter Two will stay in the inner loop until Painter One sets turn to 2. If Painter Two tries to paint Pole Two while Painter One is painting Pole Two; if turn = 2, Painter Two will remain in its outer loop while if turn = 1, Painter Two will stay in its inner loop. Therefore the algorithm implements the sharing of the ladder on Pole Two as a critical resource and handles each painter's task as its smallest unit.

Many authors have analyzed Dekker's algorithm. Tsichritzis [5] and Holt [3] both discuss Dekker's algorithm. The solution to the Pole Painting Problem using Dekker's algorithm can also be studied using the Applesoft BASIC program that follows.

PROGRAM NOTES AND ALGORITHM IMPLEMENTATION

The main program asks you to input the relative speeds of Painter One and Painter Two. The input "p1, p2" means that Painter One will perform p1 tasks during each time unit and that Painter Two will perform p2 tasks every time unit. The main program simulates the two painters performing their tasks by executing n = p1 + p2 tasks per time unit and assuming each task takes 1/n time units. The painter who is performing the ith task during a time unit depends on the two lists,

1/p1, 2/p1, ..., p1/p1

and

1/p2, 2/p2, ..., p2/p2.

Let V be the n-tuple

(1/p1,2/p1,...,p1/p1,1/p2,2/p2,...,p2/p2),

and W be the n-tuple such that the ith component of W is the position of the ith smallest component of V, and in case two components have the same value, the position which is smaller precedes the larger position. For example, with input "3, 4", p1 = 3, p2 = 4, and n = 7.

Thus V is the 7-tuple,

(1/3, 2/3, 3/3, 1/4, 2/4, 3/4, 4/4),

and W is the 7-tuple,

(4, 1, 5, 2, 6, 3, 7).

T is then defined as the n-tuple where the ith component of T is 2 if the ith component of W is greater than p1. Otherwise the ith component of T is 1. Thus T, in the above example, is the 7-tuple,

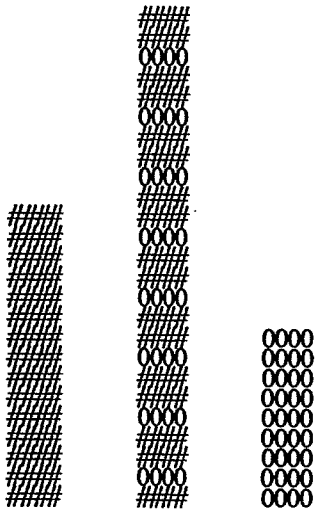
(2, 1, 2, 1, 2, 1, 2).

The ith task during a time unit is executed by the painter whose number is the ith component of the vector T.

Two sections of the main program simulate Painter One and Painter Two. Separate subroutines are written to climb up or down a pole, paint a section of a pole, and run back and forth between poles.

The following is a sample run of the program.

SAMPLE RUN



KEY ##### GREEN 0000 RED

p1 = 4 and p2 = 1.

p1 = GREEN p2 = RED

Painter One does 4 tasks for every 1 task done by Painter Two.

With the graphics capabilities of the Apple-II Plus, a solution to the Pole Painting Problem using Dekker's algorithm can be simulated on a microcomputer. The simulation is fun and exciting and can be a tool in studying both process synchronization problems and their solutions.

REFERENCES

1. Dijkstra, E. W. 1965. Solution to a problem in concurrent programming control. Comm. of ACM. 8: 569.

2. Habermann, A. N. 1976. Introduction to Operating System Design. Chicago: Science Research Associates, Inc.

3. Holt, R. C., G. S. Graham, E. D. Lazowska, and M. A. Scott. 1978. Structured Concurrent Programming with Operating Systems Applications. Reading: Addison-Wesley Publishing Company.

4. Knuth D. 1966. Additional comments on a problem in concurrent programming control. Comm. of ACM. 9: 321-322.

5. Tsichritzis, D. C., and P. A. Bernstein. 1974. Operating Systems. New York: Academic Press.

BIOGRAPHICAL SKETCHES

Lillian J. Cooper received her BS in Mathematics from Appalachian State University in 1976 and her MS in Applied Mathematics from Iowa State University in 1978. She has experience as a programmer with IBM and a teacher at Central Missouri State University. Currently she is a programmer for the Bendix Corporation.

Curtis Cooper received his BA in Mathematics from Culver Stockton College in 1974, his MS and PhD in Mathematics from Iowa State University in 1976 and 1978 respectively. He currently holds a teaching position in the Department of Mathematical Sciences at Central Missouri State University.

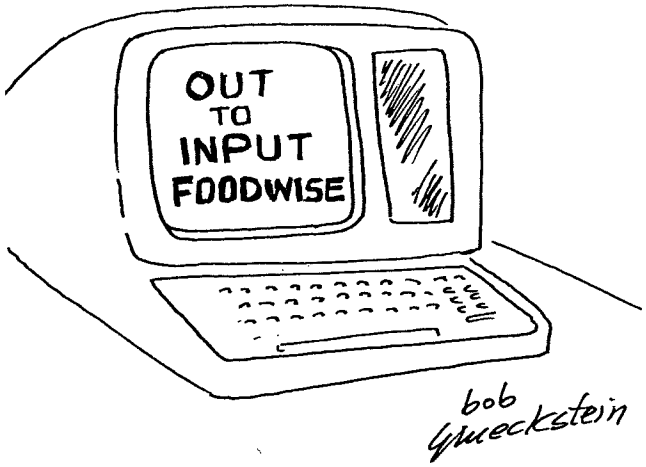
PROGRAM LISTING

```
100 DIM T(1000),X(3),Y(2),P(3,40
   ),R(1000),S(1000)
150 HOME
200 PRINT "THE POLE PAINTING PRO
   BLEM."
250 PRINT ""
300 PRINT "THERE ARE TWO PAINTER
   S WHO SPEND THEIR"
350 PRINT "LIVES PAINTING THREE
   POLES."
400 PRINT "PAINTER ONE ALTERNATE
   S PAINTING POLE"
450 PRINT "ONE AND POLE TWO WITH
    RED PAINT AND"
500 PRINT "PAINTER TWO ALTERNATE
   S PAINTING POLE"
550 PRINT "TWO AND POLE THREE WI
   TH GREEN PAINT."
600 PRINT ""
650 PRINT "TO CONTINUE, HIT ANY
   KEY."
700 GET A$
750 HOME
800 PRINT "EACH POLE IS PAINTED,
    A SECTION AT A"
850 PRINT "TIME, FROM THE BOTTOM
    TO THE TOP OF"
900 PRINT "THE POLE. AFTER A PO
   LE IS COMPLETELY"
950 PRINT "PAINTED, THE NEXT PAI
   NTER ON THE POLE"
1000 PRINT "WIPES OFF ALL THE PA
   INT ON THE POLE"
1050 PRINT "AND THE POLE IS PAIN
   TED AGAIN. A"
1100 PRINT "LADDER IS REQUIRED T
   O PAINT A POLE,"
1150 PRINT "BUT ONLY THREE LADDE
   RS ARE PROVIDED,"
1200 PRINT "ONE LADDER FOR EACH
   POLE. THUS"
1250 PRINT "PAINTER ONE AND PAIN
   TER TWO MUST"
1300 PRINT "SHARE THE LADDER OF
   POLE TWO."
1350 PRINT ""
1400 PRINT "TO CONTINUE, HIT ANY
    KEY."
1450 GET A$
1500 HOME
1550 PRINT "THE PROBLEM IS TO SI
   MULATE THE TWO"
1600 PRINT "PAINTERS TO ALLOW MU
   TUAL EXCLUSION"
1650 PRINT "(ONLY ONE PAINTER IS
   ALLOWED ON"
1700 PRINT "LADDER TWO AT A GIVE
   N TIME) AND"
1750 PRINT "AVOID INDEFINITE POS
   TPONEMENT"
1800 PRINT "(THE REQUEST BY A PA
   INTER FOR"
1850 PRINT "LADDER TWO IS CONTIN
   UALLY DENIED)."
1900 PRINT ""
1950 PRINT "TO CONTINUE, HIT ANY
    KEY."
2000 GET A$
2050 HOME
2100 PRINT "THE FOLLOWING ALGORI
   THM, DUE TO"
2150 PRINT "DEKKER, SOLVES THE P
   OLE PAINTING"
2200 PRINT "PROBLEM."
2250 PRINT ""
2300 PRINT "TO BEGIN THE SIMULAT
   ION,"
2350 PRINT "INPUT THE RATIO REPR
   ESENTING THE"
2400 PRINT "RELATIVE SPEEDS OF P
   AINTER ONE AND"
2450 PRINT "PAINTER TWO AS P1,P2
   "
2500 PRINT ""
2550 PRINT "TO END THE SIMULATIO
   N, HIT CONTROL-C"
2600 INPUT P1,P2
2650 REM
2700 REM
2750 REM  VARIABLE LIST
2800 REM
2850 REM  P1 AND P2
2900 REM  REPRESENT THE
2950 REM  RELATIVE SPEEDS
3000 REM  OF PAINTER ONE
3050 REM  AND PAINTER TWO.
3100 REM  N=P1+P2 IS THE
3150 REM  NUMBER OF TASKS
3200 REM  PERFORMED BY BOTH
3250 REM  PAINTERS IN A
3300 REM  TIME UNIT.
3350 REM  THE QTH TASK, Q=1,
3400 REM  2,3,... , EXECUTED
3450 REM  BY THE MAIN
3500 REM  PROGRAM IS THE
3550 REM  NEXT TASK
3600 REM  TO BE EXECUTED BY
3650 REM  PAINTER ONE OR
```

```
3700 REM  PAINTER TWO,
3750 REM  DEPENDING ON THE
3800 REM  VALUE OF
3850 REM  T(MOD(Q,N)+1).
3900 REM  A AND B DENOTE THE
3950 REM  NEXT TASK
4000 REM  TO BE EXECUTED BY
4050 REM  PAINTER ONE AND
4100 REM  PAINTER TWO,
4150 REM  RESPECTIVELY.
4200 REM  Y(1) AND Y(2)
4250 REM  REPRESENT THE TWO
4300 REM  COLORS, RED AND
4350 REM  GREEN RESPECTIVELY.
4400 REM  X(1), X(2), AND X(3)
4450 REM  DENOTE HOW MANY
4500 REM  SECTIONS OF POLE ONE,
4550 REM  TWO, OR THREE HAVE
4600 REM  BEEN PAINTED. P(I,J)
4650 REM  DENOTES THE COLOR ON
4700 REM  THE ITH POLE AND IN
4750 REM  THE JTH SECTION. Z
4800 REM  REPRESENTS THE CURRENT

4850 REM  POLE. FINALLY, C1,
4900 REM  C2, AND TN ARE USED
4950 REM  TO SHARE THE LADDER
5000 REM  OF POLE TWO.
5050 REM
5100 REM
5150 REM  THIS SECTION
5200 REM  CALCULATES THE
5250 REM  T ARRAY
5300 REM
5350 REM
5400 N = P1 + P2
5450  FOR I = 1 TO P1
5500 R(I) = I / P1
5550  NEXT I
5600  FOR I = 1 TO P2
5650 S(I) = I / P2
5700  NEXT I
5750 J = 1:K = 1:I = 1
5800  IF I > N THEN 6400
5850  IF R(J) < = S(K) THEN 6050

5900 T(I) = 2
5950 K = K + 1:I = I + 1
6000  GOTO 5800
6050 T(I) = 1:I = I + 1
6100  IF R(J) = 1 THEN 6250
6150 J = J + 1
6200  GOTO 5800

6250  FOR L = I TO N
6300 T(L) = 2
6350  NEXT L
6400 N1 = 30
6450 A = 1:B = 1
6500 X(1) = 0:X(2) = 0:X(3) = 0
6550 Y(1) = 1:Y(2) = 8
6600  GR
6650  COLOR= 15: PLOT 18,35: PLOT
   22,35
6700  REM
6750  REM
6800  REM  THE MAIN PROGRAM
6850  REM  SIMULATES THE
6900  REM  TWO PAINTERS
6950  REM
7000  REM
7050 C1 = 0:C2 = 0:TN = 1
7100 Q = 1
7150 I = Q -  INT (Q / N) * N + 1

7200  ON (T(I) - 1) * 16 + (2 - T
   (I)) * A + (T(I) - 1) * B GOTO
   7600,7650,7750,7850,7900,800
   0,8200,8550,8850,9050,9100,9
   300,9600,9950,10250,10600,11
   150,11200,11300,11400,11450,
   11550,11750,12100,12400,1260
   0,12650,12850,13150,13500,13
   800,14150
7250 Q = Q + 1
7300  GOTO 7150
7350  REM
7400  REM
7450  REM  PAINTER ONE
7500  REM
7550  REM
7600 C1 = 1:A = 2: GOTO 7250
7650  IF C2 < > 1 THEN 10750
7700 A = 3: GOTO 7250
7750  IF TN < > 2 THEN 10800
7800 A = 4: GOTO 7250
7850 C1 = 0:A = 5: GOTO 7250
7900  IF TN < > 2 THEN 10850
7950 A = 5: GOTO 7250
8000 C1 = 1:A = 2: GOTO 7250
8050  REM
8100  REM  CLIMB UP POLE TWO
8150  REM
8200 Z = 2:W = 18
8250 X1 = 34:Y1 = 36 - X(Z):Z1 =
   - 1
8300  GOSUB 14700
8350 A = 8: GOTO 7250
```

```
11000 REM  PAINTER TWO
11050 REM
11150 C2 = 1:B = 2: GOTO 7250
11200  IF C1 < > 1 THEN 14300
11250 B = 3: GOTO 7250
11300  IF TN < > 1 THEN 14350
11350 B = 4: GOTO 7250
11400 C2 = 0:B = 5: GOTO 7250
11450  IF TN < > 1 THEN 14400
11500 B = 5: GOTO 7250
11550 C2 = 1:B = 2: GOTO 7250
11600  REM
11650  REM  CLIMB UP POLE TWO
11700  REM
11750 Z = 2:W = 22
11800 X1 = 34:Y1 = 36 - X(Z):Z1 =
    - 1
11850  GOSUB 14700
11900 B = 8: GOTO 7250
11950  REM
12000  REM  PAINT GREEN STRIP
12050  REM
12100 Z = 2:W = 22
12150  GOSUB 15300
12200 B = 9: GOTO 7250
12250  REM
12300  REM  CLIMB DOWN POLE TWO
12350  REM
12400 Z = 2:W = 22
12450 X1 = 37 - X(Z):Y1 = 34:Z1 =
    1
12500  GOSUB 14700
12550 B = 10: GOTO 7250
12600 C2 = 0:B = 11: GOTO 7250
12650 TN = 1:B = 12: GOTO 7250
12700  REM
12750  REM  RUN TO POLE THREE
12800  REM
12850 X1 = 22:Y1 = 28:Z1 = 1
12900  GOSUB 16900
12950 B = 13: GOTO 7250
13000  REM
13050  REM  CLIMB UP POLE THREE
13100  REM
13150 Z = 3:W = 28
13200 X1 = 34:Y1 = 36 - X(Z):Z1 =
    - 1
13250  GOSUB 14700
13300 B = 14: GOTO 7250
13350  REM
13400  REM  PAINT GREEN STRIP
13450  REM
13500 Z = 3:W = 28
13550  GOSUB 15300
13600 B = 15: GOTO 7250
13650  REM
13700  REM  CLIMB DOWN POLE THREE

13750  REM
13800 Z = 3:W = 28
13850 X1 = 37 - X(Z):Y1 = 34:Z1 =
    1
13900  GOSUB 14700
13950 B = 16: GOTO 7250
14000  REM
14050  REM  RUN TO POLE TWO
14100  REM
14150 X1 = 28:Y1 = 22:Z1 =  - 1
14200  GOSUB 16900
14250 B = 1: GOTO 7250
14300 B = 7: GOTO 7250
14350 B = 2: GOTO 7250
14400 B = 6: GOTO 7250
14450  REM
14500  REM  THIS SUBROUTINE
14550  REM  CLIMBS UP OR
14600  REM  DOWN A POLE
14650  REM
14700  COLOR= 0: PLOT W,X1 - Z1
14750  FOR J = X1 TO Y1 STEP Z1
14800  COLOR= 15: PLOT W,J
14850  COLOR= 0: PLOT W,J
14900  NEXT J
14950  COLOR= 15: PLOT W,Y1 + Z1
15000  RETURN
15050  REM
15100  REM  THIS SUBROUTINE
15150  REM  PAINTS A COLOR
15200  REM  STRIP
15250  REM
15300  COLOR= 0: PLOT W,35 - X(Z)

15350 X(Z) = X(Z) + 1
15400  IF X(Z) = N1 THEN 16000
15450  COLOR= Y(T(I))
15500  PLOT 10 * Z - 1,36 - X(Z):
    PLOT 10 * Z,36 - X(Z): PLOT
    10 * Z + 1,36 - X(Z)
15550 P(Z,X(Z)) = Y(T(I))
15600  COLOR= 15: PLOT W,36 - X(Z
    )
15650  RETURN
15700  REM
15750  REM  THIS SUBROUTINE
15800  REM  WIPES OFF ALL
15850  REM  THE PAINT ON
15900  REM  A POLE
15950  REM
```

```
16000 X(Z) = 0: COLOR= 0
16050 N2 = N1 - 1
16100  FOR J = N2 TO 1 STEP  - 1
16150  PLOT 10 * Z - 1,36 - J: PLOT
    10 * Z,36 - J: PLOT 10 * Z +
    1,36 - J
16200  NEXT J
16250  IF T(I) = 1 THEN A = 10
16300  IF T(I) = 2 THEN B = 10
16350  IF T(I) = 1 THEN 16500
16400  COLOR= 15: PLOT 22,35
16450  RETURN
16500  COLOR= 15: PLOT 18,35
16550  RETURN
16600  REM
16650  REM  THIS SUBROUTINE
16700  REM  RUNS FROM ONE
16750  REM  POLE TO
16800  REM  ANOTHER POLE
16850  REM
16900  FOR J = X1 TO Y1 STEP Z1
16950  COLOR= 15: PLOT J,35: COLOR=
    0: PLOT J,35
17000  NEXT J
17050  COLOR= 15: PLOT Y1,35
17100  RETURN
```

"WE FIGURE IF HE CAN TYPE OUT A NOVEL, HE CAN WRITE A PROGRAM!"