J2

COLLABORATORS

	TITLE : J2		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 23, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

J2

1.1 J2

```
                    DOCUMENTATION FOR THE J2 PATTERN GENERATOR
          ------------------------------------------
         Jack Boyce (jboyce@tybalt.caltech.edu), 12/91


J2 is a program to generate juggling patterns. All  parameters  and options
are entered on the command line, in order to make the code portable.

There are 4 basic modes, or types of patterns which can be found:
(1) solo asynchronous juggling,
(2) solo synchronous juggling,
(3) two person passing, and
(4) a custom mode.

Mode (1), or 'site swap' mode, is the default. For an  explanation  of  the
notations  used  in  each of these modes (except for (4), described below),
refer to the file notation.doc.

In short, you give the program a  list  of  parameters  and  it  finds  ALL
patterns  which fit the given constraints. Since the number of patterns can
be very large, it is wise to have an idea of what  kinds  of  patterns  you
want when you ask the program to do something.
```

TABLE OF CONTENTS

1.2 TABLE OF CONTENTS

1.3 REQUIRED PARAMETERS

```
                Three parameters are required by the program.  They must be ↩
                   specified
on the command line immediately following 'J2'.  These numbers are,
in order:
             (1)  Number of objects juggled
             (2)  Maximum throw value to use
             (3)  Pattern length

Obviously, as any of these numbers increases the program will  take  longer
to run, since there are more possibilities for the computer to consider. If
the computer is taking too long, press any key to exit the program.

Examples are:
j2 3 5 3     ->  a list of 3 object tricks including '441' and
'531' (here we are using the default 'site swap' mode)
j2 5 7 1     ->  all we get is '5', the cascade. There are no other
valid tricks of length 1.
j2 5 7 5     ->  a list containing many interesting 5 object tricks

In some of these runs several of the patterns printed  will  have  portions
before  and  after, separated by spaces from the patterns themselves. These
are 'excited state' patterns, discussed in notation.doc.  The  throws  to
the  left  are starting sequences, those to the right are ending sequences.
Use these throws to get in and out of  the  trick  from  the  middle  of  a
cascade.  Any  pattern  without  these  throws before and after is a ground
state pattern.

Important: The throws in the starting/ending sequences are  not  parsed  by
the  multiplexing  filter,  so  they may require you to make 2 simultaneous
catches from different places. The '-x' exclude flag doesn't apply  either.
The routine for finding starting/ending sequences was a lot easier to write
without taking these into account. [See below  for  explanations  of  these
terms.]
```

```
Next item:
                OPTIONS
```

1.4 OPTIONS

```
                Flags following the 3 parameters discussed above activate various ↩
                    options.
These can be mixed at will, unless there is a direct conflict (for example,
the -g and -ng flags). These flags and their effects are:

    -s           Find solo synchronous patterns (gets out of
                 default site swap mode).

    -p           Find two person passing patterns.

    -c <file>    Go into custom mode.  The computer reads in a definition
                 file to figure out what kind of patterns you want.  (See
                 the section below for more information.)

    -g           Print only ground state patterns.

    -ng          Print only excited state patterns (no ground).

    -se          Disables  printing of  starting and  ending sequence  for
                 excited state tricks, printing '*'s instead.  For passing
                 and  multiplexing  these  sequences can  become long  and
                 cumbersome.

    -n           Computer counts  the number of  patterns found and prints
                 it at the bottom of the outputted list.

    -no          Like -n, but print the number of patterns only
                 (suppress printing of patterns).

    -write <file>  Writes output to the specified disk file

    -noprint     Disables screen printing

    -noexit      As a default, the program will stop running when the user
                 hits any key.  This flag disables the exit procedure.

    -f           The 'full' flag, which needs some explanation. Any two
                 patterns  with  the  same  starting  and ending sequences
                 (including none, for two ground state  patterns)  can  be
                 adjoined  to get another valid pattern which has the same
                 starting and ending sequences as each of its parents. For
                 example, the 3 object ground state tricks 3, 441, and 531
                 can be stuck together to get 5313441, a valid site  swap.
                 Ordinarily   J2  will  not  display  patterns  which  are
                 composed in this way of two or more smaller  ones.  Using
                 the   -f   flag   overrides   this,   printing  even  the
                 decomposable patterns.

    -simple      Related to the -f flag above. As the  default  J2
                 actually  does  print some decomposable patterns, such as
                 the 3 object trick 45141, a rotated composition of 51 and
```

```
              414.  The  reason for this is that 414 is not listed; its
              ground state rotation 441 is instead.  J2  excludes  only
              those  compositions  which can be formed from two or more
              tricks LISTED by the program. The -simple flag  overrides
              this, removing all compositions. Be aware that some valid
              tricks, such as 45141,  will  not  be  obvious  from  the
              program's  output.  (It  is  not  obvious  that 441, when
              rotated, has the same starting and  ending  sequences  as
              51.)  As a theoretical note, using the -simple flag makes
              the number of tricks for a fixed number  of  objects  and
              max.  throw value a finite number (as opposed to infinite
              for the default and -f cases above).

     -lame    This one is specific to the solo asynchronous mode.
              Patterns  containing  the  throwing  sequence  '11'  are
              eliminated by default, just because I think it's lame. If
              you want to include these, use this flag.

     -x <throw 1> <throw 2> ... The exclude option gets rid of
              those  patterns  containing  the listed throw values (for
              passing this  applies  only  to  self-throws).  Very  low
              throws  (like  '3') are sometimes hard to do in a pattern
              containing other high ones. Excluding several throws also
              speeds  up  the program quite a bit. Example: j2 5 7 5 -f
              -x 0 3 -g

     -i <throw 1> <throw  2>  ...  Each  pattern  listed  must
              contain  at  least  one  of  each of the given self-throw
              values.

     -xp <throw 1> <throw 2> ... Used to exclude passing throw
              values.  Again,  low passes are hard to do, and this lets
              you get rid of them. This flag has  no  effect  when  not
              passing.

     -m <number> Turns  on  multiplexing.  By  default   the
              computer  does  not find multiplexing patterns; this flag
              allows you to. The <number> following  the  flag  is  the
              maximum number of throws you want any hand to make at any
              time. Usually this will be 2, and very rarely  more  than
              3. As <number> increases the program slows down a lot.

     -mf      When I first implemented the  multiplexing  option  I
              noticed  that  most of the patterns found by the computer
              required  the  juggler  to  make  2  or  more   catches
              simultaneously  with the same hand. This is doable if the
              objects come from the same place (a clump of 2), but it's
              really  tough otherwise. Therefore I added a filter which
              gets rid of all multiplexing patterns which  require  the
              simultaneous  catching of objects from 2 different places
              (unless one of the caught  "throws"  was  a  hold).  This
              filter  is the default; if you want to disable it use the
              -mf flag.

     -d <number> The  delay  flag  only  has  an  effect  when
              passing. If you are doing a standard ground state passing
              pattern (such as <3p|3p><3|3> for  6  objects),  you  and
```

```
                 your  partner  can  switch  into any ground statE pattern
                 instantly, with no intermediate throws. However, many  of
                 the  patterns  printed  will  require  you  both to start
                 throwing differently at the same time (you have to  count
                 down   to   the   start   of   the   pattern,  to  ensure
                 synchronization). It is nice to allow for a communication
                 delay,  though,  so that person #2 has time to react when
                 person #1 starts throwing a trick (many  of  the  popular
                 passing  tricks  have this property). This is what the -d
                 delay flag does. The <number> is  the  number  of  throws
                 after  the  beginning of the trick before person #2 needs
                 to throw something different from what he was while doing
                 a  standard  ground  state pattern (like <3p|3p><3|3>). A
                 few  words  need  to  be  said  about  what  comprises  a
                 "standard  passing  pattern".  These  are  those patterns
                 which are composed of ground state patterns of length  1.
                 For  6  objects, for example, we type 'j2 6 4 1 -p -g'and
                 get two ground state patterns  of  length  1:  <3|3>  and
                 <3p|3p>.   Any   combination   of  these  stuck  together
                 qualifies as a "standard  ground  state  pattern";  these
                 include  standard  passing <3p|3p><3|3>, ultimate passing
                 <3p|3p>, and so on. The delay  flag  lists  all  patterns
                 which  provide a communication delay of at least <number>
                 for at least ONE of these "standard passing patterns".

                 As an example, we type 'j2 6 4 3 -p -d 2'  and  the  list
                 includes  the  two patterns: (the guy in the left slot is
                 the one "leading"  the  tricks)  <4|3p><4p|3><3|1>  which
                 assumes  the  people were doing the standard <3p|3p><3|3>
                 before the trick was being done. Note that person #1  has
                 to   begin   when   his   partner  is  throwing  a  pass.
                 <4p|3p><4p|3p><3|1>  which  assumes   the   people   were
                 ultimate  passing  before starting the trick. Some of the
                 patterns will require a 2-count passing  pattern  to  get
                 the  requested communication delay, others a 3-count, and
                 so on. When you use the -d flag just scan  the  list  for
                 the  patterns  relevant  to  your  case. The -d flag also
                 implies a '-g'  flag.  Only  ground  state  patterns  are
                 listed.

     -l <person> For use with the -d delay  flag  above.  This
                 sets  the  person  who  is  the  "leader". The default is
                 person #1, whose throws are printed in the left  position
                 of each < | >.
```

Next item:
EXAMPLES

1.5 EXAMPLES

```
                Running through these with the program should help give you an ↩
                   idea
of what it can do:
```

```
j2 3 5 3                    Simple site swaps
j2 3 5 3 -f                 Decomposable patterns too
j2 3 5 3 -f -g -n           Other flags
j2 3 5 4 -n                 Patterns of length 4
j2 3 5 4 -lame -n           Includes trick '5511' in listing
j2 5 5 3 -m 2 -g            Gatto does '24[54]' in his act
j2 5 7 5 -x 0 -m 2 -g       A LOT of interesting multiplexing
                            tricks
j2 5 6 4 -s                 Some fun 5-ball synchronous patterns
j2 6 4 2 -p -f -g           2 person, 6 object passing patterns,
                            includes standard 2-count, etc.  Most
                            are trivial, a few interesting.

j2 6 3 3 -p -m 2 -d 3       Shows the 'left-hand single' passing
                            trick mentioned in notation.doc.  Note
                            that the '-g' and '-d' options here
                            vastly reduce the number of tricks.

j2 6 3 3 -p -m 2 -d 3 -xp 1 Same as above, but eliminates
                            passed 1s (they're too low and fast).

j2 10 6 2 -p -g -f          10 object passing
j2 9 3 1 -c 3person         Basic 3 person, 9 object patterns

j2 9 4 2 -c 3person -g -xp 1 -no   A lot of patterns here
j2 9 3 3 -c 3person -m 2 -d 3 -x 0 1 -xp 1
                            Finds a bunch of multiplexing tricks
                            for 3 person, 9 object passing.  The
                            filters are crucial in limiting the
                            number of tricks to a reasonable size.
```

Next item:
USING THE CUSTOM MODE

1.6 USING THE CUSTOM MODE

The custom mode allows you to find patterns for just about any ↩ juggling
situation imaginable, beyond the 3 basic built-in modes. If you are doing 3 person passing, or if your passing partner breaks an arm and can only use the other one, or if you meet an alien with 4 arms, or if you want only those site swap multiplexing patterns which require multiple throws from the right hand, any of these cases can be handled.

All you are required to do is set up a file containing the necessary information, and then give the filename to J2 using the -c flag, as in
'j2 9 4 3 -c 3person',
where '3person' is the file which has been set up to define 3 person passing. All of the command line options listed above work in custom mode too, so you can find multiplexing patterns, ground state tricks, and so on. (The exception is the -lame flag, specific to site swaps.)

The definition file contains the particular throwing rhythm that you want to use. Let each hand in the pattern have its own line, and let the time axis be horizontal. Put a '1' at the times that each hand makes a throw. For asynchronous solo juggling, we would have something like:

```
          Right hand:  1 1 1 1 1 1 1 1 1 1 1   . . .
           Left hand:   1 1 1 1 1 1 1 1 1 1 1  . . .     time ->
                           ^   ^
```

A single horizontal space equals one unit of throw value, so an object thrown with a '4' at the first arrow above is thrown again at the second one. For a given site swap trick, you can draw on a diagram like the above an arrow for each throw, from where it is thrown to where it is thrown next. The horizontal spacing on the above diagram is critical, since it determines what throw value is needed to get from one throwing position to another.

For synchronous solo juggling, we have:

```
          Right hand:  1 1 1 1 1 1 1 1 1 1 1   . . .
           Left hand:  1 1 1 1 1 1 1 1 1 1 1   . . .
```

Note the spaces between the 1s in the lower case; if we were to remove these each hand would be throwing twice as often as it did in the upper case, which we don't want. Anyway, each of these throwing rhythms is periodic, so we can just write the repeated unit:

```
          |1 |   for the top,      |1 |   for the bottom
          | 1|                     |1 |
```

It is this repeated unit that you specify in the definition file, along with two other numbers for each hand: The person number associated with the hand (must start with person #1), and the throw value for that hand that is to be considered a hold (this is for the multiplexing filter). The format is, for the two cases above:

```
        1 |1 | 2     and         1 |1 | 2
        1 | 1| 2                 1 |1 | 2
        ^      ^-hold throw
    person # that hand belongs to
```

Subtleties: Each of the 1s in the throwing rhythms above actually signifies that at most one throw can be made from that position. If we want to find multiplexing patterns where only our right hand multiplexes (at most 2 throws at a time), we can write:

```
        1 |2 | 2     or          1 |2 | 2
        1 | 1| 2                 1 |1 | 2
```

in the definition file (the spaces in the rhythm are implied 0s). The only other subtlety has to do with the fact that, for asynchronous solo juggling, both hands never throw at the same time. Therefore we could do any site swap pattern with a single hand, in theory, if we ran it back and forth fast enough (we could no longer treat a 2 as a hold, however, but would have to throw it). We can compress the two hands into one, and the repeated unit for the asynchronous throwing rhythm is now:

```
 1 |1| 2
```

We can separate the two hands when we see a pattern because we know that odd throws cross and even ones don't.

Other examples of definition files are:

```
 1 |1| 2     for 2 person passing (each person only needs one
 "hand",
 2 |1| 2       since each is throwing asynchronously)

 1 |1 | 2
 1 |1 | 2    2 person passing, each person throwing synchronously
 2 |1 | 2      (need a separate line for each hand again)
 2 |1 | 2

 1 |1| 2
 2 |1| 2     3 person passing
 3 |1| 2

 1 |11| 2    Passing between a guy throwing asynchronously and a
 2 |1 | 2    guy with one arm.

 1 |1 1 | 2  Two asynchronous jugglers passing, one juggling half
 1 | 1 1| 2  as fast as the other.  Note that I split the hands
 2 |1   | 4  up again, since the odds cross/evens don't rule
 2 |  1 | 4  doesn't work here anymore.  Also note the '4' hold.

 1 |11 | 1   A guy juggling in a R-R-L-R-R-L throwing rhythm.  For
 1 |  1| 3   the top hand either a 1 or 2 could work as a hold,
             depending on the point in the rhythm.  Choosing 2 would
             give you a different set of multiplexing ptrns.
```

Patterns are printed in custom mode using a notation very similar to that used for the built-in modes. Instructions for different people are separated using < |..| >. If a person makes more than a single throw at a time, the different hand instructions are separated with (,..,). Multiplexed throws are grouped with [/../]. Each individual throw is printed in the following manner:

(1) The throw value, or the number of time units until the object is thrown again. 1 "time unit" is a single character wide in the specified throwing rhythm.
(2) If the throw goes to another person, print a ':' and then the destination person number.
(3) If the destination person has more than one hand, print 'a' to mean the first hand listed in the rhythm file, 'b' the second hand, and so on.

A little experimentation will no doubt help in understanding this. I tried to be as intuitive and simple as possible, but still convey the necessary information.

Next item:
MODIFYING THE PROGRAM

1.7 MODIFYING THE PROGRAM

I have marked places in the code where you can add your own throw and pattern filters for the built-in modes. (The -x exclude flag is a throw filter, the -lame flag is a pattern filter, or actually disables one.) You can also add a throw or pattern filter to use in the custom mode, as well as your own routine to print custom patterns (if you think the default one doesn't do the right job). Add these custom-mode filters and printing routines at the end of the source code, in the section marked.

This documentation was originally written from Jack Boyce. It was converted to j2.guide by Werner Riebesel in 5/95 with no changes in the text except for the added links.