This is a test of making searchable PDF files when using Type 3 bitmap fonts with the plain TEX, dvips, ps2pdf toolchain.

I've mocked up what I think should happen in `addencodings.pl`. To use it, create a dvi file, process it with `dvips -V1`, stream the PostScript result through addencodings.pl, and convert to PDF with ps2pdf. The script fakedvips.pl does all of this.

Here is some sample text from testfont.

On November 14, 1885, Senator & Mrs. Leland Stanford called together at their San Francisco mansion the 24 prominent men who had been chosen as the first trustees of The Leland Stanford Junior University. They handed to the board the Founding Grant of the University, which they had executed three days before. This document—with various amendments, legislative acts, and court decrees—remains as the University's charter. In bold, sweeping language it stipulates that the objectives of the University are "to qualify students for personal success and direct usefulness in life; and to promote the publick welfare by exercising an influence in behalf of humanity and civilization, teaching the blessings of liberty regulated by law, and inculcating love and reverence for the great principles of government as derived from the inalienable rights of man to life, liberty, and the pursuit of happiness." ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés? (¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890 JUMPS!)

It's unlikely much math will copy/paste legibly, but we can try some simple things and see what happens.

$$a^2 + b^2 = c^2$$

$$F_n = \left\lfloor \frac{\phi^n}{\sqrt{5}} \right\rfloor$$

Font size and Kern vs space tests. Notice the steadily increasing distance between the adjacent 'b' and 'o'; at some point PDF viewers will decide the horizontal space is large enough to treat as a word separator and not a kern. This should probably be somewhere around half the size of a true word space. Also, check that the font size when pasting-with-formatting is roughly correct.

5 point font (cmr5): 0.1 bp: bo. 0.125 bp: bo. 0.15 bp: bo. 0.2 bp: bo. 0.25 bp: bo. 0.3 bp: bo. 0.4 bp: bo. 0.5 bp: bo. 0.6 bp: bo. 0.8 bp: bo. 1 bp: bo. 1.25 bp: bo. 1.5 bp: bo. 2 bp: bo. 2.5 bp: bo. 3 bp: bo. 4 bp: bo. 5 bp: bo. 6 bp: bo. 8 bp: bo. 10 bp: bo. 12.55 bp: bo. 15 bp: bo. 20 bp: bo. 25 bp: bo. 30 bp: bo.

17 point font (cmr10): 0.1 bp: bo. 0.125 bp: bo. 0.15 bp: bo. 0.2 bp: bo. 0.25 bp: bo. 0.3 bp: bo. 0.4 bp: bo. 0.5 bp: bo. 0.6 bp: bo. 0.8 bp: bo. 1 bp: bo. 1.25 bp: bo. 1.5 bp: bo. 2 bp: bo. 2.5 bp: bo. 3 bp: bo. 4 bp: bo. 5 bp: bo. 6 bp: bo. 8 bp: bo. 10 bp: bo. 12.55 bp: bo. 15 bp: bo. 20 bp: bo. 25 bp: bo. 30 bp: bo.

This is some additional sample text from a random paper in two column format.

In 1979, in the first edition of his "Notes on the Magic Cube," David Singmaster asserted that every position of the Rubik's cube can be solved in twenty moves or less. It would be more than thirty years until this was shown to be true, with many distinct individual contributions. This is a chronological story of the insights and breakthroughs that culminated in the proof of Singmaster's assertion.

The Rubik's Cube is a great example of Martin Gardner's philosophy of teaching math through puzzles. Right at this moment there are thousands of teenagers who understand the mathematical concepts of commutators, conjugation, permutation parity, groups, and nested subgroups solely through their exposure to the cube.

Has the cube been solved? Over 3,000 individuals can solve it in less than 20 seconds, many with only one hand. Robots have been built to solve it; one can do it in less than four seconds. Underwater solving is commonplace, and over 1,000 individuals have been able to solve it blindfolded. Solution books have been sold by the millions, and larger versions of the cube, up to 7x7x7, are popular. Certainly we understand this puzzle at this point.

But much remains unknown in the mathematics of the cube. The fundamental problem, determining just how scrambled a cube can be, is intuitively interesting. When using a human algorithm to solve the cube (one that is intended to be used manually, rather than in a computer search), it is fairly clear that the solution we construct uses too many moves. Just like a short proof is valued over a long one, a short solution for a given position is more interesting than a long one. It is hard to avoid asking just how short a solution for a particular position can be, and then further, what positions require the longest solution sequences, and how long those sequences must be.

Every speed-solver uses, and every "how to solve the cube" book contains, an algorithm for solving the cube. These algorithms all depend on observation of specific portions of the cube state, followed by specific move sequences based on that observation. Typical algorithms are short, and can be expressed with a few pages of notes. Some advanced algorithms have extensive tables and move sequences with perhaps a few hundred alternatives. But in general these algorithms lead to move sequences that are far from optimal; typical algorithms may require fifty to eighty moves or more. In general, algorithms that end up requiring fewer moves, and thus can be executed more rapidly, are more complex, requiring longer tables and more alternatives. The limit of this is known as God's Algorithm—for every position, a move sequence is given (or calculated) that is of minimal possible length (optimal). God's Number is just the length of the longest sequence that is optimal—the length of the longest sequence that will ever be returned by God's Algorithm.

With modern computers, God's Algorithm can be implemented by a computer program that uses search to find optimal solutions. With modern desktop computers, typical positions are solved optimally in minutes; seconds, with machines with a significant amount of memory. But having access to God's Algorithm does not immediately provide us with God's Number; we still need to figure out what position is the hardest to solve, and prove it is hardest to solve. This was finally accomplished (for the half-turn metric) using about 35 CPU years at Google in 2010—the final result was twenty moves.

The history of progress towards God's Number can be divided into three eras separated by long periods of calm. The first era was intense but short, corresponding to the initial cube craze from 1979 through 1982. The second era, from 1989 to 1998, was initiated by availability of 16-bit home computers, leading to new ideas and algorithms. Finally, the third era, from 2003 to 2010, was initiated by the growing popularity of the web, which lead to a renewed interest in speed cubing and computer cubing.

For the most part, progress was characterized primarily by competition and cooperation among a small group of people. As is common with recreational mathematics, most work was unfunded, done with spare time and spare cycles, driven mostly by curiosity and mere challenge.