This is a nearly-public-domain reimplementation of the V8 regexp(3) package. It gives C programs the ability to use egrep-style regular expressions, and does it in a much cleaner fashion than the analogous routines in SysV.

Barring a couple of small items in the BUGS list, this implementation is believed 100% compatible with V8. It should even be binary-compatible, sort of, since the only fields in a "struct regexp" that other people have any business touching are declared in exactly the same way at the same location in the struct (the beginning).

This implementation is *NOT* AT&T/Bell code, and is not derived from licensed software.   Even though U of T is a V8 licensee.   This software is based on a V8 manual page sent to me by Dennis Ritchie (the manual page enclosed here is a complete rewrite and hence is not covered by AT&T copyright).   The software was nearly complete at the time of arrival of our V8 tape. I haven't even looked at V8 yet, although a friend elsewhere at U of T has been kind enough to run a few test programs using the V8 regexp(3) to resolve a few fine points.   I admit to some familiarity with regular-expression implementations of the past, but the only one that this code traces any ancestry to is the one published in Kernighan & Plauger (from which this one draws ideas but not code).

Simplistically: put this stuff into a source directory, copy regexp.h into /usr/include, inspect Makefile for compilation options that need changing to suit your local environment, and then do "make r".   This compiles the regexp(3) functions, compiles a test program, and runs a large set of regression tests.   If there are no complaints, then put regexp.o, regsub.o, and regerror.o into your C library, and regexp.3 into your manual-pages directory.

Note that if you don't put regexp.h into /usr/include *before* compiling, you'll have to add "-I." to CFLAGS before compiling.

The files are:

Makefile	instructions to make everything
regexp.3	manual page
regexp.h	header file, for /usr/include
regexp.c	source for regcomp() and regexec()
regsub.c	source for regsub()

regerror.c	source for default regerror()
regmagic.h	internal header file
try.c	source for test program
timer.c	source for timing program
tests	test list for try and timer

This implementation uses nondeterministic automata rather than the deterministic ones found in some other implementations, which makes it simpler, smaller, and faster at compiling regular expressions, but slower at executing them. In theory, anyway. This implementation does employ some special-case optimizations to make the simpler cases (which do make up the bulk of regular expressions actually used) run quickly. In general, if you want blazing speed you're in the wrong place. Replacing the insides of egrep with this stuff is probably a mistake; if you want your own egrep you're going to have to do a lot more work. But if you want to use regular expressions a little bit in something else, you're in luck. Note that many existing text editors use nondeterministic regular-expression implementations, so you're in good company.

This stuff should be pretty portable, given appropriate option settings. If your chars have less than 8 bits, you're going to have to change the internal representation of the automaton, although knowledge of the details of this is fairly localized. There are no "reserved" char values except for NUL, and no special significance is attached to the top bit of chars. The string(3) functions are used a fair bit, on the grounds that they are probably faster than coding the operations in line. Some attempts at code tuning have been made, but this is invariably a bit machine-specific.