

```

1: # Monitor: an example of tie(); stores scalar, array, hash values as
2: # usual, but prints out a record of all reads and writes to each
3: # variable, useful in debugging to find out when a variable suddenly got
4: # a weird value
5:
6: # *****
7:
8: # testmon.pl, to test the modules in Monitor
9:
10: use Monitor::Scalar;
11: use Monitor::Array;
12: use Monitor::Hash;
13:
14: $x;
15: @y;
16:
17: tie $x, 'Monitor::Scalar', \0, 'x';
18: tie @y, 'Monitor::Array', [0,0], 'y';
19: tie %z, 'Monitor::Hash', {}, 'z';
20:
21: $x = 5;
22: print $x, "\n"; # prints 5
23: $y[0] = $x;
24: $y[1] = 12;
25: $y[2] = 13;
26: $z{'abc'} = 28;
27: @y = (1,2,8); # causes execution error, due to lack of CLEAR()
28:
29: # *****
30:
31: package Monitor::Scalar; # file Scalar.pm in directory Monitor
32:
33: # arguments: package name, reference to initial value, name of tied
34: # variable (without $)
35: sub TIESCALAR {
36:     my ($pkg, $rval, $name) = @_;
37:     # $obj will be a reference to an anonymous array, consisting of the
38:     # name of the tied variable and the (dereferenced) value
39:     my $obj = [$name, $$rval];
40:     bless $obj, $pkg;
41:     return $obj;
42: }
43:
44: # argument will just be the variable name
45: sub FETCH {
46:     my ($obj) = @_;
47:     my $val = $obj->[1];
48:     print STDERR 'Read $', $obj->[0], " ... $val \n";
49:     return $val;
50: }
51:
52: # arguments will be the variable name and the value to be written
53: sub STORE {
54:     # need parentheses when 'my' is applied to more than one variable
55:     my ($obj, $val) = @_;
56:     print STDERR 'Wrote $', $obj->[0], " ... $val \n";
57:     $obj->[1] = $val;
58: }
59:
60: 1;
61:
62: # *****
63:
64: package Monitor::Array; # file Array.pm in directory Monitor
65:
66: # arguments: package name, reference to initial value of array, name of
67: # tied variable (without @)
68: sub TIEARRAY {

```

```

69:     my ($pkg, $rarray, $name) = @_;
70:     # $rarray is a reference to an array, so @$rarray is the array, so
71:     # [@$rarray] is a reference to an anonymous array having the same
72:     # elements as @$rarray (recall that, e.g., [2,5] is a reference to an
73:     # anonymous array consisting of 2 and 5)
74:     my $obj = [$name, [@$rarray]];
75:     bless $obj, $pkg;
76:     return $obj;
77: }
78:
79: # argument will be the variable name and an array index
80: sub FETCH {
81:     my ($obj, $index) = @_;
82:     my $val = $obj->[1]->[$index];
83:     print STDERR 'Read $', $obj->[0], "{$index} ... $val\n";
84:     return $val;
85: }
86:
87: # argument will be the variable name, an array index and the value to be
88: # stored in that element of the array
89: sub STORE {
90:     my ($obj, $index, $val) = @_;
91:     print STDERR 'Wrote $', $obj->[0], "{$index} ... $val\n";
92:     $obj->[1]->[$index] = $val;
93: }
94:
95: # to allow full array operations, would also need CLEAR() (called by
96: # interpreter when array is "cleared," i.e. the array name is pointed to
97: # a different array than before), PUSH() (called when push() is called),
98: # etc.; see Perl documentation for full list
99:
100: 1;
101:
102: # *****
103:
104: package Monitor::Hash; # file Hash.pm in directory Monitor
105:
106: # arguments: package name, reference to initial value of hash, name of
107: # tied variable (without %)
108: sub TIEHASH {
109:     my ($pkg, $rhash, $name) = @_;
110:     # see comments in TIEARRAY() in Monitor::Array
111:     my $obj = [$name, {%$rhash}];
112:     return (bless $obj, $pkg);
113: }
114:
115: # argument will be the variable name and a hash key
116: sub FETCH {
117:     my ($obj, $index) = @_;
118:     my $val = $obj->[1]->{$index};
119:     print STDERR 'Read $', $obj->[0], "{$index} ... $val\n";
120:     return $val;
121: }
122:
123: # argument will be the variable name, a hash key and its new value
124: sub STORE {
125:     my ($obj, $index, $val) = @_;
126:     print STDERR 'Wrote $', $obj->[0], "{$index} ... $val\n";
127:     $obj->[1]->{$index} = $val;
128:     return $val;
129: }
130:
131: 1;
132:

```