This time we've decided to blow up (raise an exception) if the renice fails–there's no place for us to return an error otherwise, and it's probably the right thing to do.

STORE this, value

This method will be triggered every time the tied variable is set (assigned). Beyond its self reference, it also expects one (and only one) argument–the new value the user is trying to assign. Don't worry about returning a value from STORE – the semantic of assignment returning the assigned value is implemented with FETCH.

```
sub STORE {
    my $self = shift;
    confess "wrong type" unless ref $self;
    my $new_nicety = shift;
    croak "usage error" if @_;

    if ($new_nicety < PRIO_MIN) {
        carp sprintf
          "WARNING: priority %d less than minimum system priority %d",
              $new_nicety, PRIO_MIN if $^W;
        $new_nicety = PRIO_MIN;
    }

    if ($new_nicety > PRIO_MAX) {
        carp sprintf
          "WARNING: priority %d greater than maximum system priority %d",
              $new_nicety, PRIO_MAX if $^W;
        $new_nicety = PRIO_MAX;
    }

    unless (defined setpriority(PRIO_PROCESS, $$self, $new_nicety)) {
        confess "setpriority failed: $!";
    }
}
```

UNTIE this

This method will be triggered when the `untie` occurs. This can be useful if the class needs to know when no further calls will be made. (Except DESTROY of course.) See The `untie` Gotcha below for more details.

DESTROY this

This method will be triggered when the tied variable needs to be destructed. As with other object classes, such a method is seldom necessary, because Perl deallocates its moribund object's memory for you automatically–this isn't C++, you know. We'll use a DESTROY method here for debugging purposes only.

```
sub DESTROY {
    my $self = shift;
    confess "wrong type" unless ref $self;
    carp "[ Nice::DESTROY pid $$self ]" if $Nice::DEBUG;
}
```

That's about all there is to it. Actually, it's more than all there is to it, because we've done a few nice things here for the sake of completeness, robustness, and general aesthetics. Simpler TIESCALAR classes are certainly possible.