50.7.2 A Note about the Examples

Although thread support is considered to be stable, there are still a number of quirks that may startle you when you try out any of the examples below. In a real situation, care should be taken that all threads are finished executing before the program exits. That care has **not** been taken in these examples in the interest of simplicity. Running these examples "as is" will produce error messages, usually caused by the fact that there are still threads running when the program exits. You should not be alarmed by this. Future versions of Perl may fix this problem.

50.7.3 Creating Threads

The *threads* package provides the tools you need to create new threads. Like any other module, you need to tell Perl that you want to use it; `use threads` imports all the pieces you need to create basic threads.

The simplest, most straightforward way to create a thread is with new():

```
use threads;

$thr = threads->new(\&sub1);

sub sub1 {
    print "In the thread\n";
}
```

The new() method takes a reference to a subroutine and creates a new thread, which starts executing in the referenced subroutine. Control then passes both to the subroutine and the caller.

If you need to, your program can pass parameters to the subroutine as part of the thread startup. Just include the list of parameters as part of the `threads::new` call, like this:

```
use threads;

$Param3 = "foo";
$thr = threads->new(\&sub1, "Param 1", "Param 2", $Param3);
$thr = threads->new(\&sub1, @ParamList);
$thr = threads->new(\&sub1, qw(Param1 Param2 Param3));

sub sub1 {
    my @InboundParameters = @_;
    print "In the thread\n";
    print "got parameters >", join("<>", @InboundParameters), "<\n";
}
```

The last example illustrates another feature of threads. You can spawn off several threads using the same subroutine. Each thread executes the same subroutine, but in a separate thread with a separate environment and potentially separate arguments.

`create()` is a synonym for `new()`.

50.7.4 Waiting For A Thread To Exit

Since threads are also subroutines, they can return values. To wait for a thread to exit and extract any values it might return, you can use the join() method:

```
use threads;

$thr = threads->new(\&sub1);
```