

SCHEME's **trace** function

scheme

Scheme saved on Sunday November 21, 1993 at 9:15:23 PM

Release 7.3.1

Microcode 11.146

Runtime 14.166

```
(define fact (lambda (n)
  (if (eq? n 0) 1 (* n (fact (- n 1))))))
```

;Value: fact

```
1 ]=> (fact 0)
```

;Value: 1

```
1 ]=> (fact 8)
```

;Value: 40320

```
1 ]=> (fact -3)
```

;Aborting!: maximum recursion depth exceeded

```
(trace fact)
```

;No value

```
1 ]=> (fact 2)
```

```
[Entering #[compound-procedure 3 fact]
  Args: 2]
```

```
[Entering #[compound-procedure 3 fact]
  Args: 1]
```

```
[Entering #[compound-procedure 3 fact]
  Args: 0]
```

```
[1
  <== #[compound-procedure 3 fact]
  Args: 0]
```

```
[1
  <== #[compound-procedure 3 fact]
  Args: 1]
```

```
[2
  <== #[compound-procedure 3 fact]
  Args: 2]
```

```
;Value: 2
```

Now let's define our own "if-then-else" function

```
(define ite (lambda (bool thenst elsest)
  (cond (bool thenst)
        (#t elsest) ) ))
```

```
;Value: ite
```

```
1 ]=> (ite (< 7 2) 'then 'else)
```

```
;Value: else
```

```
(ite (> 7 2) 'then 'else)
```

```
;Value: then
```

So now redefine fact using ite:

```
(define fact (lambda (n)
  (ite (eq? n 0) 1 (* n (fact (- n 1))))))
```

```
(fact 1)
```

;Aborting!: maximum recursion depth exceeded

WHAT HAPPENED?

```
(define fact (lambda (n)
  (write-string "n equals ") (write n) (newline)
  (ite (eq? n 0) 1 (* n (fact (- n 1))))))
```

```
1 ]=> (fact 2)
```

```
n equals 2
```

```
n equals 1
```

```
n equals 0
```

```
n equals -1
```

```
n equals -2
```

```
n equals -3
```

```
n equals -4
```

```
n equals -5
```