

## Re: Problems passing a reference to a hash between functions

---

*Source:* <http://coding.derkeiler.com/Archive/Perl/comp.lang.perl.misc/2006-01/msg01986.html>

---

- *From:* Uri Guttman <[uri@xxxxxxxxxxxxxxxx](mailto:uri@xxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 27 Jan 2006 18:22:45 -0500
- 

>>>> "M" == Mons <[inthrax@xxxxxxxx](mailto:inthrax@xxxxxxxx)> writes:

M> Agree with previous posters buw want to add a general notices...

M> 1. Ok, it's kinda standart in C to return 0 on succes and -1 on

M> failure.

M> But as noticed Paul Lalli, we're programming Perl ;)

M> so

```
>> if(ProcessTableValues(\%results) < 0)
```

```
>> {
```

```
>> $conn->disconnect;exit(-1);
```

```
>> }
```

M> will look nicer such way:

```
M> unless (ProcessTableValues(...)) {
```

```
M> $conn->disconnect;
```

```
M> exit(-1);
```

```
M> }
```

that is better perl style i agree but the OP has his return value style  
so ingrained i didn't want to comment on it.

M> 2.

```
>> print STDERR "\n", 'Failed to prepare ', $DBI::errstr;
```

M> Be simplier!

```
M> warn 'Failed to prepare ' . $DBI::errstr;
```

be simpler:

```
warn "Failed to prepare $DBI::errstr" ;
```

M> 3. Surely, you don't need to do separate prepare,execute and fetch.

M> Call once

```
M> $conn->selectall_hashref('select tabid, tabname from systables where
```

```
M> tabid < 5', 'tabid')
```

dbi prepare is useful, especially if you use placeholders. on some db  
servers it can actually be much faster since the sql is only compiled  
once.

Re: Problems passing a reference to a hash between functions

M> 4. And finally

M> For such a routines may be better to write them in this way:

```
M> sub some_func ($$) {
```

prototypes are not cool in perl in most cases.

```
M> my ($param1,$param2) = @_;
```

```
M> eval {
```

```
M> # do here all stuff without checking at every step
```

```
M> stuff1();
```

```
M> stuff2() or warn "noncritical IO stuff failed: $!";
```

```
M> stuff2() or die "critical stuff failed";
```

```
M> };
```

but you are checking the last two steps. using eval BLOCK to catch errors is ok when you have deep nesting and you aren't sure where the error should be handled. this style is also called exception handling and there are modules to make it easier to do. but a clean return value design with proper handling can do the job just as effectively.

M> With this guides, your program may be rewritten in about 20 lines.

M> While shorter code is more readable and powerful :)

that is true to a point. code written for the sake of just shortness can be harder to maintain and debug. there is a balance that must be found between concise vs. bloated

uri

---

Uri Guttman ----- uri@xxxxxxxxxxxxxxxxx ----- <http://www.stemsystems.com>

--Perl Consulting, Stem Development, Systems Architecture, Design and Coding--

Search or Offer Perl Jobs ----- <http://jobs.perl.org>

.

---

• **References:**

- ◆ **[Problems passing a reference to a hash between functions](#)**

◇ From: niall . macpherson

- ◆ **[Re: Problems passing a reference to a hash between functions](#)**

◇ From: Mons

- Prev by Date: **[Re: data structure problem](#)**
- Next by Date: **[Re: data structure problem](#)**
- Previous by thread: **[Re: Problems passing a reference to a hash between functions](#)**
- Next by thread: **[Re: Problems passing a reference to a hash between functions](#)**
- Index(es):
  - ◆ **[Date](#)**

Re: Problems passing a reference to a hash between functions

◆ *Thread*