Oh, you `re no (fun _ → more)

Ah well, a dromedary has one hump and a caml has reference cells, buffers, and a garbage collector.

<関数型言語と金融について—席噺... | [OCaml]My tuareg mode conf>

2010-03-09

caml_{enter,leave}_blocking_section

OCaml

What are caml_{enter,leave}_blocking_section? They are not documented in the OCaml reference manual, but are very important if you tweak C code in multi-thread OCaml environment:

Jacque Garrigue wrote:

> With posix threads (or windows threads), every caml thread is mapped
> to a posix thread, but there is a global mutex which any caml thread
> must obtain before running. This makes sure for instance that memory
> allocation and GC work properly.
> So no more than one caml thread may run simultaneously, and you don't
> gain from multiple CPUs.
> However, contrary to vmthreads, this restriction only

プロフィール

camlspotter

Yes in only three years. Er, I tell a lie, four, be fair, five. I've been caml programming for just the 14 years.

> applies while
> executing caml code. If you call some C function, you may choose to
> first release the global lock
> (caml_enter_blocking_section), letting
> other caml threads work while you are on the C side. Don't forget to
> call lock again (caml_leave_blocking_section) when returning, or you
> will crash very soon.
>
> *http://groups.google.com/group/fa.caml/browse_thread/thread/lnk=gst&q=caml_enter_blocking_section#3671a04944223be*

Sounds odd (a blocking section unblocks other threads!), but,

- caml_enter_blocking_section : release the global lock in OCaml runtime.
- caml_leave_blocking_section : lock it again.
- In the section, some other threads may work simultaneously with the current thread.

Yes, it is true, but **be careful**, when your C code accesses OCaml values (alloc, reading pointers, etc) before entering the section. Once after the lock is released, some other OCaml threads are executed, and there is a chance of GC. This makes your OCaml related pointer values no longer reliable!!

Actually Francois Rouaix has pointed it out long ago:

> When you want to be able to switch threads while in C-code, or handle
> signals.
> However, my understanding is that the code in the section must not
> access anything in the Caml heap.
> On Jan 16, 2006, at 7:33 AM, Bauer, Christoph wrote:

ブラリ 関数 関数型 言語

最新タイトル

[OCaml] OCurl or ocaml-curl 0.5.3 has a bug around set_postfields

関数型言語を独学で勉強している学生です への答

[OCaml]1モジュール1データ型主義

[OCaml] OCaml 開発環境について ~ コンパイラに付属しない非公式ツールたち

[OCaml] Meta_conv による OCamlデータ型 と 樹状データ の相互変換自動生成

[OCaml]OCaml 開発環境について ~ OCaml コンパイラソース付属ツール

星のキャミバ様 Adventure Calendar 第366夜: 再入国

[OMake] Ubuntu で OMake をソースからまんまビルドすると OMake の -P が動かないよ

星のキャミバ様 Adventure: F1 シンガポールGP はじまる!

[シンガポール] 星のキャミバ様 Adventure: 滞在8ヶ月 + 図書館古本放出市に行きました

最近のコメント

2013-01-21 ytqwerty

2010-12-12 camlspotter

2010-01-06 t

2010-07-10 cocoatomo

2010-07-10 cocoatomo

最近のトラックバック

2013-01-17 pocketberserkerの爆走

> Hi,
>
> when do I have to call the functions
> caml_enter_blocking_section ()
> and
> caml_leave_blocking_section ()
> in my C-stub code?
>
> Thanks,
>
> Christoph Bauer
> Dipl. Inf.

http://groups.google.com/group/fa.caml/browse_thread/threa
lnk=gst&q=caml_enter_blocking_section#3ab73788b5f142a

Actually we can see such an example of caml_{enter,leave}_blocking_sections in byterun/sys.c

```
CAMLprim value caml_sys_open(value path, value vf
{
  CAMLparam3(path, vflags, vperm);
  int fd, flags, perm;
  char * p;

  p = caml_stat_alloc(caml_string_length(path) +
  strcpy(p, String_val(path));
  flags = caml_convert_flag_list(vflags, sys_open
  perm = Int_val(vperm);
  /* open on a named FIFO can block (PR#1533) */
  caml_enter_blocking_section();
  fd = open(p, flags, perm);
  caml_leave_blocking_section();
  caml_stat_free(p);
  if (fd == -1) caml_sys_error(path);
#if defined(F_SETFD) && defined(FD_CLOEXEC)
  fcntl(fd, F_SETFD, FD_CLOEXEC);
#endif
  CAMLreturn(Val_long(fd));
}
```

Here, the open syscall is in the section, so that it may not

- 関数型言語を独学で勉強している学生です ...

2012-12-04 pocaristの日記 - 自宅の開発環境について

2012-12-16 deruiの日記 - omake-mode.el を Mac で動くようにしてみた

2011-10-15 deruiの日記 - js_of_ocamlの導入とenchant.jsの実行

2010-08-08 armbrust の日記 - fakecygpty と irb

ページビュー

437981

block the other thread; open is a lengthyoperation in certain circumstances. It uses the file path given from the OCaml(path), but since path is in the OCaml heap, we cannot use it safely. The code escapes the contents of path to p, a C-malloc'ed memory before entering the section. Beware, this code demonstrates even CAMLparam'ed pointers may become unreliable due to the GC once we enter the section!!

There may be some other informative posts found in caml-list:

http://groups.google.com/group/fa.caml/search?group=fa.caml&q=caml_enter_blocking_section&qt_g=Se

Permalink | コメント(0) | トラックバック(0) | 14:07

コメントを書く

なまえ　メール(非公開)

URL

画像認証

画像内の文字列を入力して下さい

投稿

トラックバック - http://d.hatena.ne.jp/camlspotter/20100309/1268111257

idトラックバック

トーフサロン - OMake 基礎文法最速マスター

YAMAGUCHI::weblog - Objective Caml 入門 手習い（2章）

リンク元

29 http://twitter.com/camlspotter

13 http://reader.livedoor.com/reader/

8 http://jun.furuse.info/

7 http://d.hatena.ne.jp/Ehren/20091102/1257179244

6 http://www.google.com/reader/view/

4 http://d.hatena.ne.jp/hayamiz/20081203/1228296644

4 http://www.google.co.jp/search?hl=ja&client=firefox-a&rls=org.mozilla:ja-JP-mac:official&q=ocaml+"toplevel"+"UTF-8"&btnG=検索&lr=lang_ja&aq=f&oq=

4 http://www.google.co.jp/search?hl=ja&safe=off&client=firefox-a&hs=kmr&rls=org.mozilla:ja-JP-mac:official&q=ocaml++gc&btnG=検索&lr=lang_ja&aq=f&oq=

3 http://niha.tumblr.com/page/2

3 http://www.google.co.jp/reader/view/

<関数型言語と金融について一席噺... | [OCaml]My tuareg mode conf>

Connection: close