使用 GDB 來🄵 C API 擴充功能和 CPython 🄵部偵錯

發🄵 *3.14.0a1*

Guido van Rossum and the Python development team

11 月 11, 2024

Python Software Foundation
Email: docs@python.org

Contents

本文件解釋如何將 Python GDB 擴充功能 `python-gdb.py` 與 GDB 偵錯器一起使用來🄵 CPython 擴充功能和 CPython 直譯器本身偵錯。

在🄵崩潰或死鎖等低階問題幀錯時，低階偵錯器（例如 GDB）對於診斷和修正問題非常有用。預設情🄵下，GDB（或其任何前端）不支援特定於 CPython 直譯器的高階資訊。

`python-gdb.py` 擴充功能將 CPython 直譯器資訊新增至 GDB。此擴充有助於🄵省 (introspect) 目前執行的 Python 函式的堆🄵。給定一個由 `PyObject*` 指標表示的 Python 物件，擴充功能會顯示該物件的型🄵和值。

正在開發 CPython 擴充功能或修補用 C 編寫之 CPython 部分的開發人員可以使用本文件來學習如何將 `python-gdb.py` 擴充功能與 GDB 一起使用。

備🄵

本文件假設你熟悉 GDB 和 CPython C API 的基礎知識。它整合了 devguide 和 Python wiki 的指引。

1 先□條件

你需要有：

- GDB 7 或更之後的版本。（對於早期版本的 GDB，請參□ Python 3.11 或更早版本的原始程式碼中的 `Misc/gdbinit`。）
- 對 Python 和你在偵錯的任何擴充功能來□的與 GDB 相容的偵錯資訊。
- `python-gdb.py` 擴充。

該擴充功能是用 Python 建置的，但可能會單獨發布或根本不發布。以下我們將一些常見系統的提示作□範例。請注意，即使□明與你的系統匹配，它們也可能已過時。

1.1 使用從原始碼建置的 Python 進行設定

當你從原始程式碼建立 CPython 時，偵錯資訊應該可用，□且建置應該將 `python-gdb.py` 檔案新增至儲存庫的根目□中。

若要□用支援，你必須將包含 `python-gdb.py` 的目□新增至 GDB 的"auto-load-safe-path"。如果你還□有這樣做，最新版本的 GDB 將列印警告，其中包含有關如何執行此操作的□明。

備□

如果你□有看到適合你的 GDB 版本的□明，請將其放入你的設定檔中（`~/.gdbinit` 或 `~/.config/gdb/gdbinit`）：

```
add-auto-load-safe-path /path/to/cpython
```

你也可以新增多個路徑，要以 : 分隔。

1.2 從 Linux 發行版設定 Python

大多數 Linux 系統在名□ `python-debuginfo`、`python-dbg` 或類似的套件中提供系統 Python 的偵錯資訊。例如：

- Fedora：

```
sudo dnf install gdb
sudo dnf debuginfo-install python3
```

- Ubuntu：

```
sudo apt install gdb python3-dbg
```

在最近的幾個 Linux 系統上，GDB 可以使用 *debuginfod* 自動下載偵錯符號。但是這不會安裝 `python-gdb.py` 擴充功能；你通常需要另外安裝偵錯資訊套件。

2 使用偵錯建置與使用開發模式

□了更輕松地進行偵錯，你可能需要：

- 使用 Python 的偵錯建置。（從原始碼建置時，請使用 `configure --with-pydebug`。在 Linux 發行版上，安裝□執行諸如 `python-debug` 或 `python-dbg` 之類的套件（如果可用））。
- 使用 runtime 開發模式 (`-X dev`)。

兩者都□用額外的斷言 (assertion) □停用一些最佳化。有時這會隱藏你試圖想尋找的錯誤，但在大多數情□下，它們會使過程變得更容易。

3 使用 python-gdb 擴充功能

載入擴充功能後，它提供兩個主要功能：Python 值的漂亮列印器和附加命令。

3.1 漂亮列印器

這是🄵用此擴充功能時 GDB 回溯 (backtrace) 的樣子（有被截斷）：

```
#0  0x000000000041a6b1 in PyObject_Malloc (nbytes=Cannot access memory at address␣
↪0x7fffff7fefe8
) at Objects/obmalloc.c:748
#1  0x000000000041b7c0 in _PyObject_DebugMallocApi (id=111 'o', nbytes=24) at Objects/
↪obmalloc.c:1445
#2  0x000000000041b717 in _PyObject_DebugMalloc (nbytes=24) at Objects/obmalloc.c:1412
#3  0x000000000044060a in _PyUnicode_New (length=11) at Objects/unicodeobject.c:346
#4  0x00000000004466aa in PyUnicodeUCS2_DecodeUTF8Stateful (s=0x5c2b8d "__lltrace__",␣
↪size=11, errors=0x0, consumed=
    0x0) at Objects/unicodeobject.c:2531
#5  0x0000000000446647 in PyUnicodeUCS2_DecodeUTF8 (s=0x5c2b8d "__lltrace__", size=11,␣
↪errors=0x0)
    at Objects/unicodeobject.c:2495
#6  0x000000000440d1b in PyUnicodeUCS2_FromStringAndSize (u=0x5c2b8d "__lltrace__", size=11)
    at Objects/unicodeobject.c:551
#7  0x000000000440d94 in PyUnicodeUCS2_FromString (u=0x5c2b8d "__lltrace__") at Objects/
↪unicodeobject.c:569
#8  0x000000000584abd in PyDict_GetItemString (v=
    {'Yuck': <type at remote 0xad4730>, '__builtins__': <module at remote 0x7ffff7fd5ee8>, '_
↪_file__': 'Lib/test/crashers/nasty_eq_vs_dict.py', '__package__': None, 'y': <Yuck(i=0) at␣
↪remote 0xaacd80>, 'dict': {0: 0, 1: 1, 2: 2, 3: 3}, '__cached__': None, '__name__': '__
↪main__', 'z': <Yuck(i=0) at remote 0xaace60>, '__doc__': None}, key=
    0x5c2b8d "__lltrace__") at Objects/dictobject.c:2171
```

請注意 `PyDict_GetItemString` 的字典引數如何顯示🄵其 `repr()`，而不是不透明的 `PyObject *` 指標。

該擴充功能透過🄵 `PyObject *` 型🄵的值提供自訂列印例程來運作。如果需要存取物件較低階的詳細資訊，請將值轉🄵🄵適當型🄵的指標。例如：

```
(gdb) p globals
$1 = {'__builtins__': <module at remote 0x7ffff7fb1868>, '__name__':
'__main__', 'ctypes': <module at remote 0x7ffff7f14360>, '__doc__': None,
'__package__': None}

(gdb) p *(PyDictObject*)globals
$2 = {ob_refcnt = 3, ob_type = 0x3dbdf85820, ma_fill = 5, ma_used = 5,
ma_mask = 7, ma_table = 0x63d0f8, ma_lookup = 0x3dbdc7ea70
<lookdict_string>, ma_smalltable = {{me_hash = 7065186196740147912,
me_key = '__builtins__', me_value = <module at remote 0x7ffff7fb1868>},
{me_hash = -368181376027291943, me_key = '__name__',
me_value ='__main__'}, {me_hash = 0, me_key = 0x0, me_value = 0x0},
{me_hash = 0, me_key = 0x0, me_value = 0x0},
{me_hash = -9177857982131165996, me_key = 'ctypes',
me_value = <module at remote 0x7ffff7f14360>},
{me_hash = -8518757509529533123, me_key = '__doc__', me_value = None},
{me_hash = 0, me_key = 0x0, me_value = 0x0}, {
  me_hash = 6614918939584953775, me_key = '__package__', me_value = None}}}
```

請注意，漂亮列印器其實🄵不呼叫 `repr()`。對於基本型🄵，他們嘗試緊密匹配其結果。

一個可能令人困惑的地方是，某些型🄵的自訂列印器看起來很像 GDB 標准類型的🄵建列印器。例如，Python `int` (`PyLongObject*`) 的漂亮列印器給出的表示法無法與常規機器層級整數之其一區分：

```
(gdb) p some_machine_integer
$3 = 42

(gdb) p some_python_integer
$4 = 42
```

可以透過轉🄵 (cast) 🄵 `PyLongObject*` 來揭示🄵部結構：

> (gdb) p *(PyLongObject)*some_python_integer $5 = {ob_base = {ob_base = {ob_refcnt = 8, ob_type = 0x3dad39f5e0}, ob_size = 1}, ob_digit = {42}}

使用 `str` 型🄵時也可能會出現類似的困惑，其中的輸出看起來很像對於 `char *` 的 gdb 🄵建列印器：

```
(gdb) p ptr_to_python_str
$6 = '__builtins__'
```

`str` 實例的漂亮列印器預設使用單引號（Python 的 `repr` 對於字串也是如此），而 `char *` 值的標準列印器使用雙引號🄵包含十六進位位址：

```
(gdb) p ptr_to_char_star
$7 = 0x6d72c0 "hello world"
```

同樣，可以透過轉🄵🄵 `PyUnicodeObject*` 來揭示實作細節：

```
(gdb) p *(PyUnicodeObject*)$6
$8 = {ob_base = {ob_refcnt = 33, ob_type = 0x3dad3a95a0}, length = 12,
str = 0x7ffff2128500, hash = 7065186196740147912, state = 1, defenc = 0x0}
```

3.2 py-list

該擴充功能新增了一個 `py-list` 命令，該命令列出了所選執行緒中當前 frame 的 Python 原始程式碼（如果有）。當前的列會標有">"：

```
(gdb) py-list
 901        if options.profile:
 902            options.profile = False
 903            profile_me()
 904            return
 905
>906        u = UI()
 907        if not u.quit:
 908            try:
 909                gtk.main()
 910            except KeyboardInterrupt:
 911                # properly quit on a keyboard interrupt...
```

使用 `py-list START` 列出 Python 原始碼中不同的列號，使用 `py-list START,END` 列出 Python 原始碼中特定範圍的列。

3.3 py-up 和 py-down

`py-up` 和 `py-down` 命令類似於 GDB 的常規 `up` 和 `down` 命令，但嘗試在 CPython frame 層級移動，而不是 C frame。

GDB 🄵不總是能🄵讀取相關的 frame 資訊，這取🄵於編譯 CPython 的最佳化等級。在🄵部，這些指令會尋找正在執行預設 frame 計算 (evaluation) 函式（即 CPython 中🄵圈的核心位元組碼直譯器）的 C frame，🄵尋找相關 `PyFrameObject *` 的值。

它們在執行緒🄵發出（於 C 層級的）frame 編號。

例如：

```
(gdb) py-up
#37 Frame 0x9420b04, for file /usr/lib/python2.6/site-packages/
gnome_sudoku/main.py, line 906, in start_game ()
    u = UI()
(gdb) py-up
#40 Frame 0x948e82c, for file /usr/lib/python2.6/site-packages/
gnome_sudoku/gnome_sudoku.py, line 22, in start_game(main=<module at remote␣
↪0xb771b7f4>)
    main.start_game()
(gdb) py-up
Unable to find an older python frame
```

所以現在我們處於 Python 堆🄵的頂端。

frame 編號與 GDB 標准 `backtrace` 指令顯示的 frame 編號相對應。此指令會跳過不執行 Python 程式碼的 C frame。

回到下面：

```
(gdb) py-down
#37 Frame 0x9420b04, for file /usr/lib/python2.6/site-packages/gnome_sudoku/main.
↪py, line 906, in start_game ()
    u = UI()
(gdb) py-down
#34 (unable to read python frame information)
(gdb) py-down
#23 (unable to read python frame information)
(gdb) py-down
#19 (unable to read python frame information)
(gdb) py-down
#14 Frame 0x99262ac, for file /usr/lib/python2.6/site-packages/gnome_sudoku/game_
↪selector.py, line 201, in run_swallowed_dialog (self=
↪<NewOrSavedGameSelector(new_game_model=<gtk.ListStore at remote 0x98fab44>,␣
↪puzzle=None, saved_games=[{'gsd.auto_fills': 0, 'tracking': {}, 'trackers': {},
↪'notes': [], 'saved_at': 1270084485, 'game': '7 8 0 0 0 0 0 5 6 0 0 9 0 8 0 1 0␣
↪0 0 4 6 0 0 0 0 7 0 6 5 0 0 0 4 7 9 2 0 0 0 9 0 1 0 0 0 3 9 7 6 0 0 0 1 8 0 6 0␣
↪0 0 0 2 8 0 0 0 5 0 4 0 6 0 0 2 1 0 0 0 0 0 0 4 5\n7 8 0 0 0 0 0 5 6 0 0 9 0 8 0␣
↪1 0 0 0 4 6 0 0 0 0 7 0 6 5 1 8 3 4 7 9 2 0 0 0 9 0 1 0 0 0 3 9 7 6 0 0 0 1 8 0␣
↪6 0 0 0 0 2 8 0 0 0 5 0 4 0 6 0 0 2 1 0 0 0 0 0 0 4 5', 'gsd.impossible_hints': 0,
↪ 'timer.__absolute_start_time__': <float at remote 0x984b474>, 'gsd.hints': 0,
↪'timer.active_time': <float at remote 0x984b494>, 'timer.total_time': <float at␣
↪remote 0x984b464>}], dialog=<gtk.Dialog at remote 0x98faaa4>, saved_game_model=
↪<gtk.ListStore at remote 0x98fad24>, sudoku_maker=<SudokuMaker(terminated=False,
↪ played=[], batch_siz...(truncated)
            swallower.run_dialog(self.dialog)
(gdb) py-down
#11 Frame 0x9aead74, for file /usr/lib/python2.6/site-packages/gnome_sudoku/
↪dialog_swallower.py, line 48, in run_dialog (self=<SwappableArea(running=<gtk.
↪Dialog at remote 0x98faaa4>, main_page=0) at remote 0x98fa6e4>, d=<gtk.Dialog␣
↪at remote 0x98faaa4>)
            gtk.main()
(gdb) py-down
#8 (unable to read python frame information)
(gdb) py-down
Unable to find a newer python frame
```

我們現在處於 Python 堆🄵的底部。

請注意，在 Python 3.12 及更高版本中，同一個 C 堆🄵 frame 可用於多個 Python 堆🄵 frame。這意味著 `py-up` 和 `py-down` 可以一次移動多個 Python frame。例如：

```
(gdb) py-up
#6 Frame 0x7ffff7fb62b0, for file /tmp/rec.py, line 5, in recursive_function (n=0)
```

(繼續下一頁)

(繼續上一頁)

```
   time.sleep(5)
#6 Frame 0x7ffff7fb6240, for file /tmp/rec.py, line 7, in recursive_function (n=1)
   recursive_function(n-1)
#6 Frame 0x7ffff7fb61d0, for file /tmp/rec.py, line 7, in recursive_function (n=2)
   recursive_function(n-1)
#6 Frame 0x7ffff7fb6160, for file /tmp/rec.py, line 7, in recursive_function (n=3)
   recursive_function(n-1)
#6 Frame 0x7ffff7fb60f0, for file /tmp/rec.py, line 7, in recursive_function (n=4)
   recursive_function(n-1)
#6 Frame 0x7ffff7fb6080, for file /tmp/rec.py, line 7, in recursive_function (n=5)
   recursive_function(n-1)
#6 Frame 0x7ffff7fb6020, for file /tmp/rec.py, line 9, in <module> ()
   recursive_function(5)
(gdb) py-up
Unable to find an older python frame
```

3.4 py-bt

`py-bt` 指令嘗試顯示目前執行緒的 Python 層級回溯。

例如：

```
(gdb) py-bt
#8 (unable to read python frame information)
#11 Frame 0x9aead74, for file /usr/lib/python2.6/site-packages/gnome_sudoku/
↪dialog_swallower.py, line 48, in run_dialog (self=<SwappableArea(running=<gtk.
↪Dialog at remote 0x98faaa4>, main_page=0) at remote 0x98fa6e4>, d=<gtk.Dialog␣
↪at remote 0x98faaa4>)
            gtk.main()
#14 Frame 0x99262ac, for file /usr/lib/python2.6/site-packages/gnome_sudoku/game_
↪selector.py, line 201, in run_swallowed_dialog (self=
↪<NewOrSavedGameSelector(new_game_model=<gtk.ListStore at remote 0x98fab44>,␣
↪puzzle=None, saved_games=[{'gsd.auto_fills': 0, 'tracking': {}, 'trackers': {},
↪'notes': [], 'saved_at': 1270084485, 'game': '7 8 0 0 0 0 0 5 6 0 0 9 0 8 0 1 0␣
↪0 0 4 6 0 0 0 0 7 0 6 5 0 0 0 4 7 9 2 0 0 0 9 0 1 0 0 0 3 9 7 6 0 0 0 1 8 0 6 0␣
↪0 0 0 2 8 0 0 0 5 0 4 0 6 0 0 2 1 0 0 0 0 0 4 5\n7 8 0 0 0 0 0 5 6 0 0 9 0 8 0␣
↪1 0 0 0 4 6 0 0 0 0 7 0 6 5 1 8 3 4 7 9 2 0 0 0 9 0 1 0 0 0 3 9 7 6 0 0 0 1 8 0␣
↪6 0 0 0 0 2 8 0 0 0 5 0 4 0 6 0 0 2 1 0 0 0 0 0 4 5', 'gsd.impossible_hints': 0,
↪ 'timer.__absolute_start_time__': <float at remote 0x984b474>, 'gsd.hints': 0,
↪'timer.active_time': <float at remote 0x984b494>, 'timer.total_time': <float at␣
↪remote 0x984b464>}], dialog=<gtk.Dialog at remote 0x98faaa4>, saved_game_model=
↪<gtk.ListStore at remote 0x98fad24>, sudoku_maker=<SudokuMaker(terminated=False,
↪ played=[], batch_siz...(truncated)
            swallower.run_dialog(self.dialog)
#19 (unable to read python frame information)
#23 (unable to read python frame information)
#34 (unable to read python frame information)
#37 Frame 0x9420b04, for file /usr/lib/python2.6/site-packages/gnome_sudoku/main.
↪py, line 906, in start_game ()
    u = UI()
#40 Frame 0x948e82c, for file /usr/lib/python2.6/site-packages/gnome_sudoku/gnome_
↪sudoku.py, line 22, in start_game (main=<module at remote 0xb771b7f4>)
    main.start_game()
```

The frame numbers correspond to those displayed by GDB's standard `backtrace` command.

3.5 py-print

py-print 命令查找 Python 名稱🄵嘗試列印它。它在當前執行緒中尋找局部變數，然後是全域變數，最後是🄵建變數：

```
(gdb) py-print self
local 'self' = <SwappableArea(running=<gtk.Dialog at remote 0x98faaa4>,
main_page=0) at remote 0x98fa6e4>
(gdb) py-print __name__
global '__name__' = 'gnome_sudoku.dialog_swallower'
(gdb) py-print len
builtin 'len' = <built-in function len>
(gdb) py-print scarlet_pimpernel
'scarlet_pimpernel' not found
```

如果目前 C frame 對應多個 Python frame，則 py-print 僅考慮第一個。

3.6 py-locals

py-locals 命令尋找所選執行緒中當前 Python frame 🄵的所有 Python 局部變數，🄵列印它們的表示：

```
(gdb) py-locals
self = <SwappableArea(running=<gtk.Dialog at remote 0x98faaa4>,
main_page=0) at remote 0x98fa6e4>
d = <gtk.Dialog at remote 0x98faaa4>
```

如果目前 C frame 對應於多個 Python frame，則會顯示所有這些 frame 的局部變數：

```
(gdb) py-locals
Locals for recursive_function
n = 0
Locals for recursive_function
n = 1
Locals for recursive_function
n = 2
Locals for recursive_function
n = 3
Locals for recursive_function
n = 4
Locals for recursive_function
n = 5
Locals for <module>
```

4 與 GDB 指令一起使用

擴充命令補充了 GDB 的🄵建命令。例如，你可以將 py-bt 顯示的 frame 編號與 frame 命令一同使用來跳到所選執行緒中的特定 frame，如下所示：

```
(gdb) py-bt
(output snipped)
#68 Frame 0xaa4560, for file Lib/test/regrtest.py, line 1548, in <module> ()
        main()
(gdb) frame 68
#68 0x00000000004cd1e6 in PyEval_EvalFrameEx (f=Frame 0xaa4560, for file Lib/test/regrtest.
↪py, line 1548, in <module> (), throwflag=0) at Python/ceval.c:2665
2665                            x = call_function(&sp, oparg);
(gdb) py-list
1543        # Run the tests in a context manager that temporary changes the CWD to a
1544        # temporary and writable directory. If it's not possible to create or
```

(繼續下一頁)

(繼續上一頁)

```
1545        # change the CWD, the original CWD will be used. The original CWD is
1546        # available from test_support.SAVEDCWD.
1547        with test_support.temp_cwd(TESTCWD, quiet=True):
>1548            main()
```

info threads 命令將🄵你提供行程🄵的執行緒串列，你可以使用 thread 命令選擇不同的執行緒：

```
(gdb) info threads
  105 Thread 0x7fffefa18710 (LWP 10260)  sem_wait () at ../nptl/sysdeps/unix/sysv/linux/x86_
↪64/sem_wait.S:86
  104 Thread 0x7fffdf5fe710 (LWP 10259)  sem_wait () at ../nptl/sysdeps/unix/sysv/linux/x86_
↪64/sem_wait.S:86
* 1 Thread 0x7ffff7fe2700 (LWP 10145)  0x00000038e46d73e3 in select () at ../sysdeps/unix/
↪syscall-template.S:82
```

你可以使用 thread apply all COMMAND（或簡短地用 t a a COMMAND）在所有執行緒上執行命令。使用 py-bt 你可以看到每個執行緒在 Python 層級正在做什🄵：

```
(gdb) t a a py-bt

Thread 105 (Thread 0x7fffefa18710 (LWP 10260)):
#5 Frame 0x7fffd00019d0, for file /home/david/coding/python-svn/Lib/threading.py, line 155,␣
↪in _acquire_restore (self=<_RLock(_Verbose__verbose=False, _RLock__owner=140737354016512, _
↪RLock__block=<thread.lock at remote 0x858770>, _RLock__count=1) at remote 0xd7ff40>, count_
↪owner=(1, 140737213728528), count=1, owner=140737213728528)
        self.__block.acquire()
#8 Frame 0x7fffac001640, for file /home/david/coding/python-svn/Lib/threading.py, line 269,␣
↪in wait (self=<_Condition(_Condition__lock=<_RLock(_Verbose__verbose=False, _RLock__
↪owner=140737354016512, _RLock__block=<thread.lock at remote 0x858770>, _RLock__count=1) at␣
↪remote 0xd7ff40>, acquire=<instancemethod at remote 0xd80260>, _is_owned=<instancemethod␣
↪at remote 0xd80160>, _release_save=<instancemethod at remote 0xd803e0>, release=
↪<instancemethod at remote 0xd802e0>, _acquire_restore=<instancemethod at remote 0xd7ee60>,␣
↪_Verbose__verbose=False, _Condition__waiters=[]) at remote 0xd7fd10>, timeout=None, waiter=
↪<thread.lock at remote 0x858a90>, saved_state=(1, 140737213728528))
            self._acquire_restore(saved_state)
#12 Frame 0x7fffb8001a10, for file /home/david/coding/python-svn/Lib/test/lock_tests.py,␣
↪line 348, in f ()
            cond.wait()
#16 Frame 0x7fffb8001c40, for file /home/david/coding/python-svn/Lib/test/lock_tests.py,␣
↪line 37, in task (tid=140737213728528)
                f()

Thread 104 (Thread 0x7fffdf5fe710 (LWP 10259)):
#5 Frame 0x7fffe4001580, for file /home/david/coding/python-svn/Lib/threading.py, line 155,␣
↪in _acquire_restore (self=<_RLock(_Verbose__verbose=False, _RLock__owner=140737354016512, _
↪RLock__block=<thread.lock at remote 0x858770>, _RLock__count=1) at remote 0xd7ff40>, count_
↪owner=(1, 140736940992272), count=1, owner=140736940992272)
        self.__block.acquire()
#8 Frame 0x7fffc8002090, for file /home/david/coding/python-svn/Lib/threading.py, line 269,␣
↪in wait (self=<_Condition(_Condition__lock=<_RLock(_Verbose__verbose=False, _RLock__
↪owner=140737354016512, _RLock__block=<thread.lock at remote 0x858770>, _RLock__count=1) at␣
↪remote 0xd7ff40>, acquire=<instancemethod at remote 0xd80260>, _is_owned=<instancemethod␣
↪at remote 0xd80160>, _release_save=<instancemethod at remote 0xd803e0>, release=
↪<instancemethod at remote 0xd802e0>, _acquire_restore=<instancemethod at remote 0xd7ee60>,␣
↪_Verbose__verbose=False, _Condition__waiters=[]) at remote 0xd7fd10>, timeout=None, waiter=
↪<thread.lock at remote 0x858860>, saved_state=(1, 140736940992272))
            self._acquire_restore(saved_state)
#12 Frame 0x7fffac001c90, for file /home/david/coding/python-svn/Lib/test/lock_tests.py,␣
↪line 348, in f ()
            cond.wait()
#16 Frame 0x7fffac0011c0, for file /home/david/coding/python-svn/Lib/test/lock_tests.py,␣
```

(繼續下一頁)

(繼續上一頁)

```
↪line 37, in task (tid=140736940992272)
                f()

Thread 1 (Thread 0x7ffff7fe2700 (LWP 10145)):
#5 Frame 0xcb5380, for file /home/david/coding/python-svn/Lib/test/lock_tests.py, line 16,␣
↪in _wait ()
    time.sleep(0.01)
#8 Frame 0x7fffd00024a0, for file /home/david/coding/python-svn/Lib/test/lock_tests.py, line␣
↪378, in _check_notify (self=<ConditionTests(_testMethodName='test_notify', _
↪resultForDoCleanups=<TestResult(_original_stdout=<cStringIO.StringO at remote 0xc191e0>,␣
↪skipped=[], _mirrorOutput=False, testsRun=39, buffer=False, _original_stderr=<file at␣
↪remote 0x7ffff7fc6340>, _stdout_buffer=<cStringIO.StringO at remote 0xc9c7f8>, _stderr_
↪buffer=<cStringIO.StringO at remote 0xc9c790>, _moduleSetUpFailed=False,␣
↪expectedFailures=[], errors=[], _previousTestClass=<type at remote 0x928310>,␣
↪unexpectedSuccesses=[], failures=[], shouldStop=False, failfast=False) at remote 0xc185a0>,
↪ _threads=(0,), _cleanups=[], _type_equality_funcs={<type at remote 0x7eba00>:
↪<instancemethod at remote 0xd750e0>, <type at remote 0x7e7820>: <instancemethod at remote␣
↪0xd75160>, <type at remote 0x7e30e0>: <instancemethod at remote 0xd75060>, <type at remote␣
↪0x7e7d20>: <instancemethod at remote 0xd751e0>, <type at remote 0x7f19e0...(truncated)
        _wait()
```