Gem #122: Команды для точек прерывания - Часть 2

Автор: Jerome Guitton, AdaCore

Краткое содержание: В предыдущем Gem #121 описывались простые примеры применения команд для точек прерывания с целью проверки некоторых параметров в ходе выполнения. В данном Gem #122 продолжается обсуждение этой темы с примерами, демонстрирующими преимущества API Python.

Давайте начнём…

Ранее мы рассматривали пример программы «банк», который управлял только одним счетом. Теперь предположим, что этот пример был расширен для поддержки нескольких счетов и транзакций между этими счетами:

```
package Accounts is   type Customer is (Alice, Bob);
   procedure Open_Account (C : Customer; Cash : Integer);
   procedure Pay (From, To : Customer; Cash : Integer);
end Accounts;
```

Чтобы проверить, что никакие счета не открываются дважды и что транзакции происходят только на открытых счетах, нам нужно вести учет счетов, которые уже были открыты. Когда у нас была только одна учетная запись, состояние было простой логической переменной; теперь это список счетов, который сложнее выразить только с помощью вспомогательных переменных. Интерфейс Python GDB может использоваться для обхода этого ограничения.

В качестве первого шага, давайте опишем всю программу. Оно содержит основную процедуру, которая открывает две учетные записи и планирует набор транзакций между ними:

```
with Accounts; use Accounts;
procedure P is
      begin
         Open_Account (Alice, 50);
         Open_Account (Bob, 40);
         Pay (From => Bob,   To => Alice, Cash => 45);
         Pay (From => Alice, To => Bob,   Cash => 20);
         Pay (From => Bob,   To => Alice, Cash => 45);
         Pay (From => Alice, To => Bob,   Cash => 20);
      end P;
```

Что касается тела пакета учетных записей (`Accounts`), это не представляет никаких трудностей. Как ни странно, в своей разработке она скрывает транзакцию:

```
package body Accounts is

   type Account is limited record
      Balance : Integer;
   end record;

   Bank : array (Customer) of Account;

   procedure Open_Account (C : Customer; Cash : Integer) is
   begin
      Bank (C).Balance := Cash;
   end Open_Account;

   procedure Pay (From, To : Customer; Cash : Integer) is
   begin
      Bank (To).Balance := Bank (To).Balance + Cash;
      Bank (From).Balance := Bank (From).Balance - Cash;
   end Pay;
```

```
   --  At elaboration time, a suspicious operation is scheduled;
   --  Alice secretly pays a bribe to Bob:

begin
   Pay (From => Alice, To => Bob, Cash => 20);
end Accounts;
```

Теперь мы реализуем в Python два Хука-ловушки (hooks), которые должны выполняться всякий раз, когда запланирована операция над учетной записью. Создайте новый файл с именем hooks.py со следующим кодом:

```
current_customers = []def open_account_hook():
    global current_customers
    c = str(gdb.parse_and_eval('c'))
    if c in current_customers:
        print "error: account initialized twice"
    else:
        print "(info) %s opens an account" % c
        current_customers.append(c)
        gdb.execute("continue")def pay_hook():
    global current_customers
    f = str(gdb.parse_and_eval('from'))
    t = str(gdb.parse_and_eval('to'))
    if not f in current_customers:
        print "error: %s tries to pay before opening an account" % f
    elif not t in current_customers:
        print "error: %s cannot be paid before opening an account" % t
    else:
        cash = str(gdb.parse_and_eval('cash'))
        print "(info) %s gives %s $%s" % (f, t, cash)
        gdb.execute("continue")
```

Это определяет три сущности: глобальный список, в котором записываются счета, которые уже были открыты, и две ловушки, которые должны быть выполнены при вызове `Open_Account` и `Pay`.

Оба получают значение параметров, используя функцию `gdb.parse_and_eval` (определенную в Python API), чтобы проверить, что операция допустима. Документация этой функции, а также документация любых сущностей Python, могут быть доступны из GDB с помощью командной строки Python:

```
(gdb) python help(gdb.parse_and_eval)
 Help on built-in function parse_and_eval in module
gdb:parse_and_eval(...)
    parse_and_eval (String) -> Value.
    Parse String as an expression, evaluate it, and return the result as a Value.
```

Теперь нам просто нужно зарегистрировать эти команды в GDB с помощью команды source и прикрепить их к соответствующим точкам останова.

```
(gdb) source hooks.py(gdb) break open_account
Breakpoint 1 at 0x4015b4: file accounts.adb, line 11.
(gdb) commands>python open_account_hook()>end
(gdb) break pay
Breakpoint 2 at 0x4015f6: file accounts.adb, line 16.
(gdb) commands>python pay_hook()>end
```

Это обнаруживает, что первая транзакция действительно недействительна, как это происходит до открытия счетов:

```
(gdb) run
Starting program: C:\home\guitton\GIT\GDBuilds\gems\3\p.exe [New Thread 1872.0x638]
```

```
Breakpoint 2, accounts.pay (from=alice, to=bob, cash=20) at accounts.adb:1616
            Bank (To).Balance := Bank (To).Balance + Cash;
error: alice tries to pay before opening an account
```

API Python обеспечивает усовершенствованный интерфейс программирования для GDB, со средствами для того, чтобы просмотреть следы, потоки, символы, и т. д. Для получения дополнительной информации мы приглашаем Вас консультироваться со специализированным разделом по тем функциям в руководстве пользователя GDB.

Связанный со статьёй текст программы

File: test.cmd

```
gnatmake -g p.adb
gdb p --command=scenario.gdb
```

File: scenario.gdb

```
source hooks.py

break open_account
commands
python open_account_hook()
end

break pay
commands
python pay_hook()
end

run
```

File: hooks.py

```
current_customers = []

def open_account_hook():
    global current_customers
    c = str(gdb.parse_and_eval('c'))
    if c in current_customers:
        print "error: account initialized twice"
    else:
        print "(info) %s opens an account" % c
        current_customers.append(c)
        gdb.execute("continue")

def pay_hook():
    global current_customers
    f = str(gdb.parse_and_eval('from'))
    t = str(gdb.parse_and_eval('to'))
    if not f in current_customers:
        print "error: %s tries to pay before opening an account" % f
    elif not t in current_customers:
        print "error: %s cannot be payed before opening an account" % t
    else:
        cash = str(gdb.parse_and_eval('cash'))
        print "(info) %s gives %s %s$" % (f, t, cash)
        gdb.execute("continue")
```

File: p.adb

```
 1. with Accounts; use Accounts;
 2.
```

```
 3. procedure P is
 4. begin
 5.    Open_Account (Alice, 50);
 6.    Open_Account (Bob, 40);
 7.    Pay (From => Bob,   To => Alice, Cash => 45);
 8.    Pay (From => Alice, To => Bob,   Cash => 20);
 9.    Pay (From => Bob,   To => Alice, Cash => 45);
10.    Pay (From => Alice, To => Bob,   Cash => 20);
11. end P;
```

File: accounts.ads

```
1. package Accounts is
2.
3.    type Customer is (Alice, Bob);
4.
5.    procedure Open_Account (C : Customer; Cash : Integer);
6.
7.    procedure Pay (From, To : Customer; Cash : Integer);
8.
9. end Accounts;
```

File: accounts.adb

```
 1. package body Accounts is
 2.
 3.    type Account is limited record
 4.       Balance : Integer;
 5.    end record;
 6.
 7.    Bank : array (Customer) of Account;
 8.
 9.    procedure Open_Account (C : Customer; Cash : Integer) is
10.    begin
11.       Bank (C).Balance := Cash;
12.    end Open_Account;
13.
14.    procedure Pay (From, To : Customer; Cash : Integer) is
15.    begin
16.       Bank (To).Balance := Bank (To).Balance + Cash;
17.       Bank (From).Balance := Bank (From).Balance - Cash;
18.    end Pay;
19.
20.    --  At elaboration time, a suspicious operation is scheduled;
21.    --  Alice secretly pays a bribe to Bob:
22.
23. begin
24.    Pay (From => Alice, To => Bob, Cash => 20);
25. end Accounts;
```

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Jerome присоединился к AdaCore в 2002 после завершения его исследований в Ecole Nationale des Télécommunations (Париж, Франция), во время которого он уже работал с компанией на одном из ее многих reseach проектов, а именно, PolyORB. Его энтузиазм остался непотускневшим в течение

этих шести лет, и он работал над множеством проектов, а также экспертным знанием получения в отладчиках и перекрестных технологиях. Он недавно приложил усилия к порту GNAT Pro в различных перекрестным платформам (в частности ОС VxWorks 6, ELinOS).

Last Updated: 11/24/2017
Posted on: 3/12/2012

Обсуждение…