Debugging (pdb und Integration mit gdb)

Proseminar: Python im Hochleistungsrechnen

Yuliia Lysa

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2019-20-05

Gliederung

Debugger

- Debugger - Werkzeug zur Fehlerdiagnose
- Daten-Analyse, Ablaufsteuerung vom Programm, Abfangen von Exceptions
- Modifikation des Zustands
- ptrace()-Systemaufruf

Implementation

- pdb - Modul in der Python-Standardbibliothek
- als Klasse Pdb realisiert
- class pdb.Pdb(completekey='tab', stdin=None, stdout=None, skip=None, nosigint=False, readrc=True)
  - durch die Module cmd, bdb und readline implementiert

Aufruf des Debuggers

- mit set.trace()

```
1 import pdb; pdb.set_trace()
```

Quelle: [2]

- mit der breakpoint()-Methode
- per Kommandozeile

```
1 python3 -m pdb myscript.py
```

Quelle: [2]

- Post-Mortem-Debugging

```
1 pdb.post_mortem()
```

Quelle: [2]

Befehlssyntax

- vordefinierte Abkürzungen für die Befehle
- Space oder Tab - Trennung von Parametern
- ;; - Trennung von Befehlen auf einer Zeile
- [ ] - optionale Parameter
- | - alternative Angaben
- Enter-Taste - Wiederholung vom Befehl
- ! - Deklaration von Statements

```
1 (Pdb) !next = 5
```

Haltepunkte setzen

- b(reak) [([filename:]lineno | function) [, condition]]

```
1 (Pdb) b util:5
2  Breakpoint 1 at /code/util.py:5
3  (Pdb) b util.get_path
4  Breakpoint 2 at /code/util.py:1
5  (Pdb) b
6  Num Type Disp Enb Where
7  1 breakpoint keep yes at /code/util.py:1
```

Quelle: [3]

Haltepunkte setzen

- enable [bpnumber [bpnumber ...]]
- disable [bpnumber [bpnumber ...]]
- tbreak [([filename:]lineno | function) [, condition]]
- cl(ear) [filename:lineno | bpnumber [bpnumber ...]]

```
1  (Pdb) disable 1
2  Disabled breakpoint 1 at /code/util.py:1
3  (Pdb) b
4  Num Type         Disp Enb   Where
5  1   breakpoint   keep no    at /code/util.py:1
6  (Pdb) enable 1
7  Enabled breakpoint 1 at /code/util.py:1
8  (Pdb) b
9  Num Type         Disp Enb   Where
10 1   breakpoint   keep yes   at /code/util.py:1
11 (Pdb)
```

Quelle: [3]

Variablen inspizieren

- p expression
- pp expression
- display [expression]
- undisplay [expression]

```
1 (Pdb) p filename
2  ' ./example2.py '
3  (Pdb) p head , tail
4 ( ' . ' , ' example2.py ' )
5  (Pdb) p ' filename: ' + filename
6  ' filename: ./example2.py '
```

Quelle: [3]

Code schrittweise durchlaufen

- n(ext)
- s(tep)
- c(ont(inue))
- unt(il) [lineno]

```
1 (Pdb) n
2  > /code/example3.py(15)<module >()
3  -> print(f ' path = {filename_path} ' )
4  (Pdb) s
5  --Call --6 > /code/example3.py(6)get_path()
6  -> def get_path(filename):'
```

Quelle: [3]

Code schrittweise durchlaufen

- l(ist) [first[, last]] oder ll | longlist

```
(Pdb) l .
                return head


filename = __file__
 import pdb; pdb.set_trace()
-> filename_path = get_path(filename)
print(f'path = {filename_path}')
[EOF]
(Pdb)
```

Quelle: [3]

Stacktrace

- w(here)
- u(p) [count]
- d(own) [count]

```
(Pdb) w
 /code/example5.py(12)<module >()
 -> filename_path = get_file_info(filename)
 /code/example5.py(7) get_file_info()
 -> file_path = fileutil.get_path(full_fname)
 > /code/fileutil.py(5)get_path()
```

Quelle: [3]

Aliase einrichten

- alias [name [command]]
- unalias name

```
1 (Pdb) pp locals().keys()
2  dict_keys([ ' output ' , ' n ' ])
3  (Pdb) alias pl pp locals().keys()
4  (Pdb) pl
5  dict_keys([ ' output ' , ' n ' ])
6  (Pdb) alias
7  pl = pp locals().keys()
8  (Pdb) alias pl
9  pl = pp locals().keys()
```

Quelle: [4]

Debugging von einem Programm

```
1 def compute(x, y):
2     return x + y
3 def main():
4     print("This is my program")
5         import pdb; pdb.set_trace()
6     x = compute(2, 3)
7     print(x)
8 if __name__ == '__main__':
9 main()
```

Quelle: [9]

Debugging von einem Programm

```
$ python3.6 simple.py
 This is my program
 > /location/to/simple.py(7)main()
 -> x = compute(2, 3)
 (Pdb) n
 > /location/to/simple.py(8)main()
 -> print(x)
 (Pdb) p x
 5
 (Pdb) repr(x)
 ' 5 '
 (Pdb)
```

Quelle: [9]

GNU-Debugger

- ein Linux-Debugger
- unterstützt verschiedene Sprachen, insbesondere C und C++
- Erweiterung von gdb mithilfe von Python möglich
- Python wird auf der Interpreter-Ebene debuggt

Vorteile von GDB

- Anhängen an einen laufenden Prozess bzw. Debuggen von hängenden Prozessen
- Behandlung von Segfaults
- Core-Datei-Analyse beim Programmabsturz

Installation

- Installation von GDB

```
1 # apt-get install gdb
```

Quelle: [8]

- Installation von Debugger-Symbolen

```
1 # apt-get install python-dbg
```

Quelle: [8]

Aufruf des Debuggers

- vom Debugger aus

```
1 $ gdb python
2  ...
3  (gdb) run <programname >.py <arguments >
```

Quelle: [6]

- an einen laufenden Prozess anhängen

```
1 $ gdb python <pid of running process >
```

Quelle: [6]

Python-spezifische Befehle

- py-list

```
(gdb) py-list
      if options.profile:
      options.profile = False
      profile_me()
      return

      u = UI()
      if not u.quit:
          try:
              gtk.main()
          except KeyboardInterrupt:
          # properly quit on a keyboard
              interrupt...
```

Python-spezifische Befehle

- py-bt

```
(gdb) py-bt
 (output snipped)
 # 68 Frame 0xaa4560 , for file
     ↪ Lib/test/regrtest.py, line 1548, in
     ↪ <module > ()
      main()
```

- py-locals

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

Python-spezifische Befehle

- py-up, py-down

```
(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
```

Python-spezifische Befehle

- py-print

```
(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 '
```

- Link zur GDB-Dokumentation:
  http://sourceware.org/gdb/current/onlinedocs/gdb/

Python-Debugger

- ipdb - IPython-Debugger mit Befehlszeilenergänzung und Syntax-Highlighting
- PyCharm - IDE mit grafischem Interface
- PuDB - terminalbasierter Debugger, Tastatursteuerung
- wdb - Web-Debugger

Zusammenfassung

- pdb
  - Teil der Standardbibliothek
  - erlaubt das Inspizieren von Daten und Stacktrace, bedingte Haltepunkte, Auflistung vom Quellcode
- GDB
  - Standard Linux-Debugger mit Unterstützung von mehreren Programmiersprachen
  - wird bei Bugs eingesetzt, die der interne Debugger nicht effizient behandeln kann

pdb-Interface

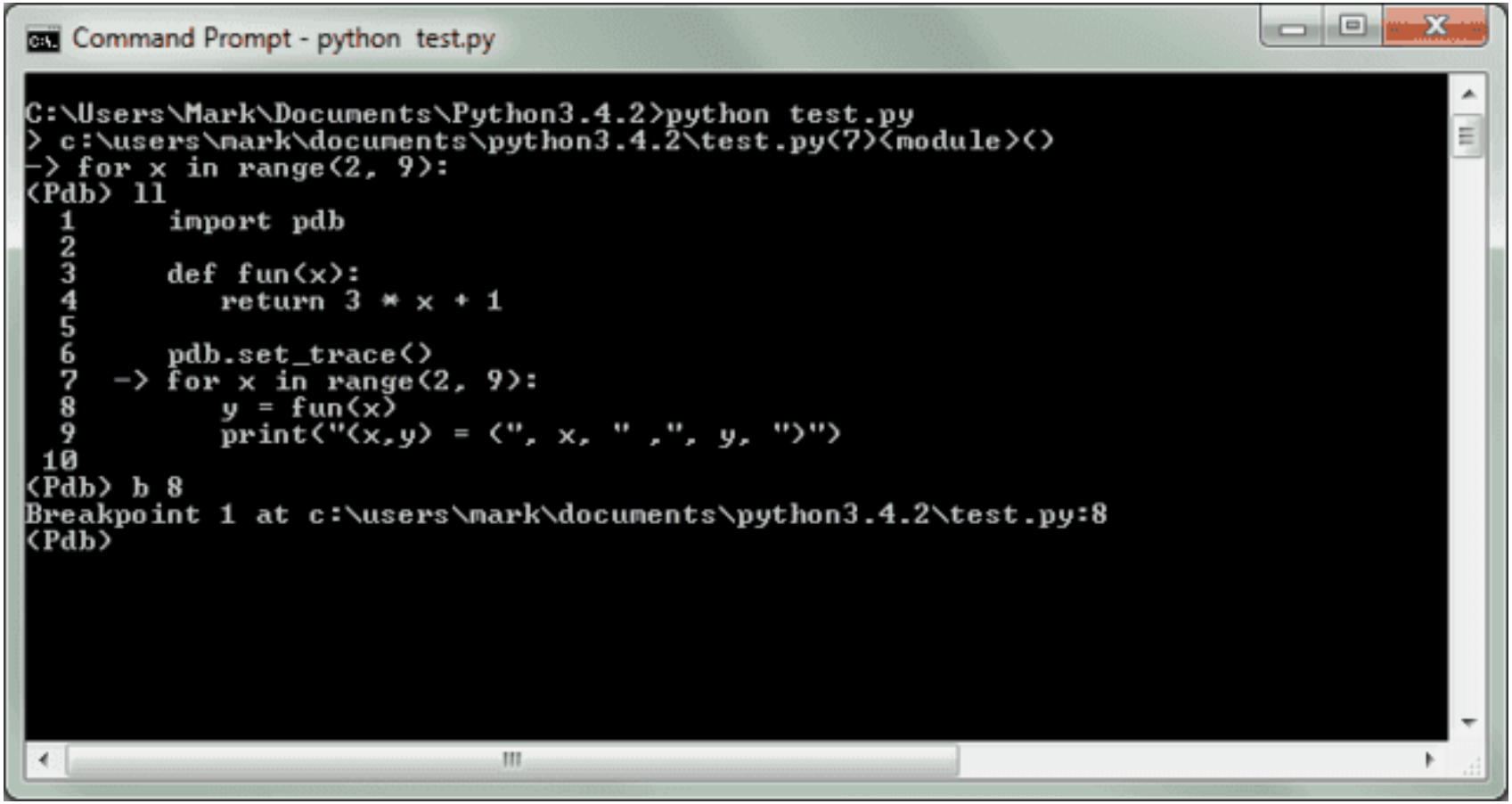

Abbildung: Quelle: https://www.physicsforums.com/insights/simple-python-debugging-pdb-part-2/

GDB-Interface

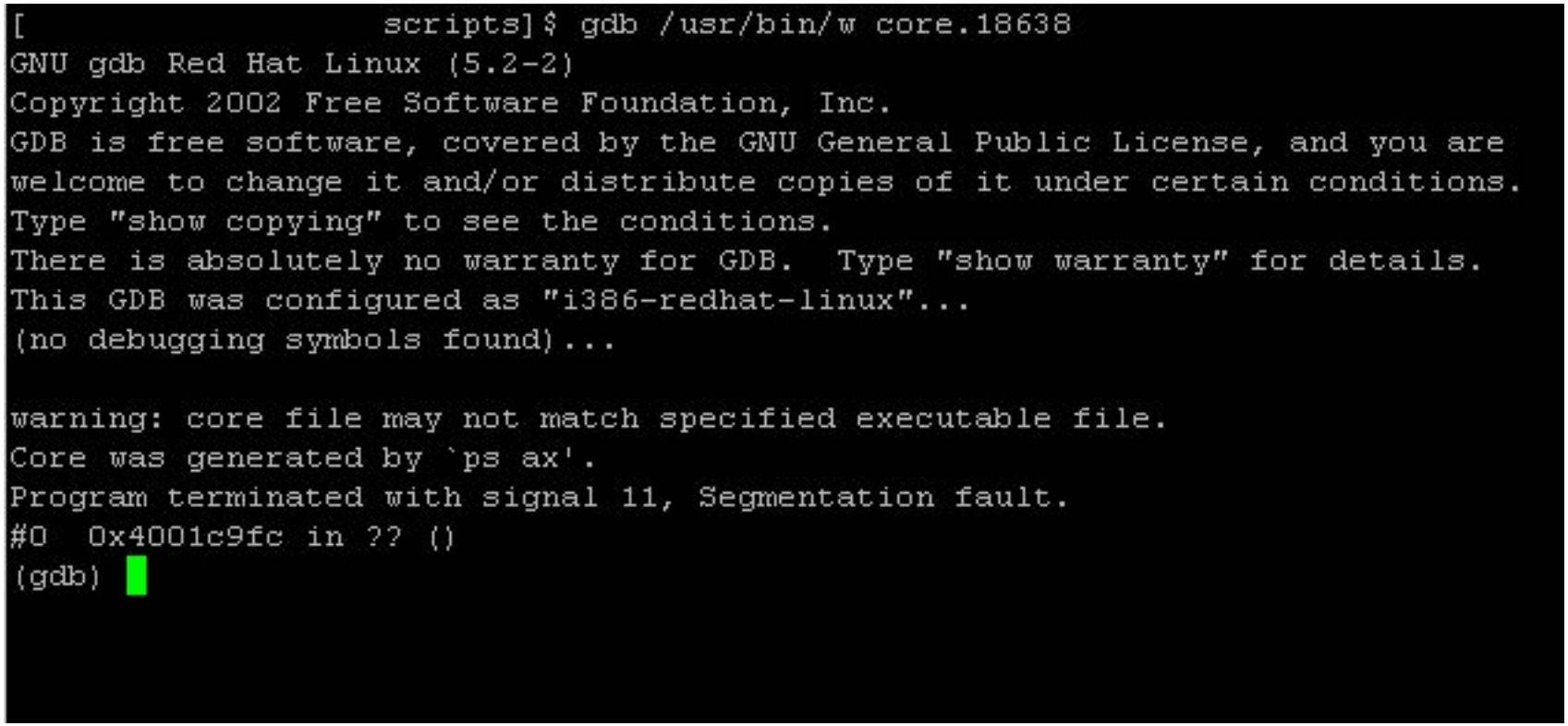

Abbildung: Quelle: http://bloggerdigest.blogspot.com/2006/09/gnu-gdb-core-dump-debugging.html

Literatur

[1] Kristian Rother. Pro Python Best Practices - Debugging, Testing and Maintenance. 2017. Letzter Zugriff: 15.05.2019
[2] Python Software Foundation: "pdb — The Python Debugger"https://docs.python.org/3/library/pdb.html, Letzter Zugriff 15.05.2019
[3] Nathan Jennings: "Python Debugging With Pdb"https://realpython.com/python-debugging-pdb/, Letzter Zugriff: 15.05.2019
[4] Doug Hellmann: "pdb — Interactive Debugger"https://pymotw.com/3/pdb/, Letzter Zugriff: 15.05.2019

Literatur

[5] Python Software Foundation: "gdb Support"https://devguide.python.org/gdb/, Letzter Zugriff: 15.05.2019
[6] Wiki.python. "Debugging-WithGdb"https://wiki.python.org/moin/DebuggingWithGdb, Letzter Zugriff: 15.05.2019
[7] David Malcolm. "Features/EasierPythonDebug-ging"https://fedoraproject.org/wiki/Features/EasierPythonDebugging, Letzter Zugriff: 15.05.2019
[8] Roman Podoliaka. "Debugging of CPython processes with gdb"https://www.podoliaka.org/2016/04/10/debugging-cpython-gdb/, Letzter Zugriff: 15.05.2019

Literatur

[9] Elliot Forbes. "Debugging with the Python Debugger -PDB"https://tutorialedge.net/python/debugging-with-pdb-python/, Letzter Zugriff: 15.05.2019

[10] Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Ptrace, Letzter Zugriff: 15.05.2019

[11] Stephan Augsten. "Was ist ein Debugger?"https://www.dev-insider.de/was-ist-ein-debugger-a-730931/, Letzter Zugriff: 15.05.2019