Now Part of Accenture

Drill down into the log writer inner working and communication to foreground processes.

Frits Hoogland

```
This is the font size used for showing screen output. Be sure this is readable for you.
```

This is the font used to accentuate text/console output. Make sure this is readable for you too!

$(whoami)

- Frits Hoogland
- Working with Oracle products since 1996

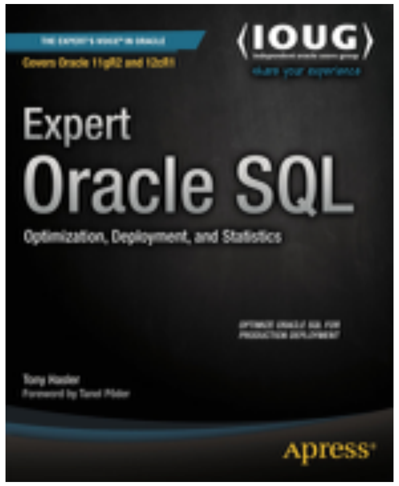
- Blog: http://fritshoogland.wordpress.com
- Twitter: @fritshoogland
- Email: frits.hoogland@enkitec.com
- Oracle ACE Director
- OakTable Member

Books

IOUG
SECOND EDITION
Expert
Oracle
Exadata
Martin Bach, Karl Arao, Andy Colvin, Frits Hoogland, Kerry Osborne, Randy Johnson, and Tanel Poder
Apress

Author, together with Martin Bach, Karl Arao and Andy Colvin.

Technical reviewer:



What is this presentation about?

```
SQL> commit;
Commit complete.
```

What happens between the foreground and the LGWR on commit in polling mode?

Specifically: how do these communicate.

What is this presentation about?

It’s not not the destination but the journey that matters.

Summary of the poem “Ithaca” by Constantine Cavafy.

Warning

- Looking at the **inner working** of Oracle.
  - A lot of this is **undocumented**.
  - This means looking at (in) the Oracle executable and o/s resources.
- The techniques used are **no** methods for daily administration tasks.
  - Rather techniques to be used in **specialistic edge cases**.
- *Using these techniques wrong can have severe consequences (instance down, corruption)!*

Prerequisites

Final scare away warning!

- Basic understanding of how the processes of an Oracle database work and communicate.
- Basic understanding of C coding and basic flow of execution.
- Understanding of the logic of redo and redo concepts for the foreground and log writer processes.
- *This is not a high level overview. This is a microscopic look into the inner working.*

Test system

- The tests and investigation is done in a VM:
  - Host: Mac OSX 10.11.3 / VMWare Fusion 7.1.3.
  - VM: Oracle Linux x86_64 7u2 (3.10.0-123.el7.x86_64).
  - Oracle database 12.1.0.2.

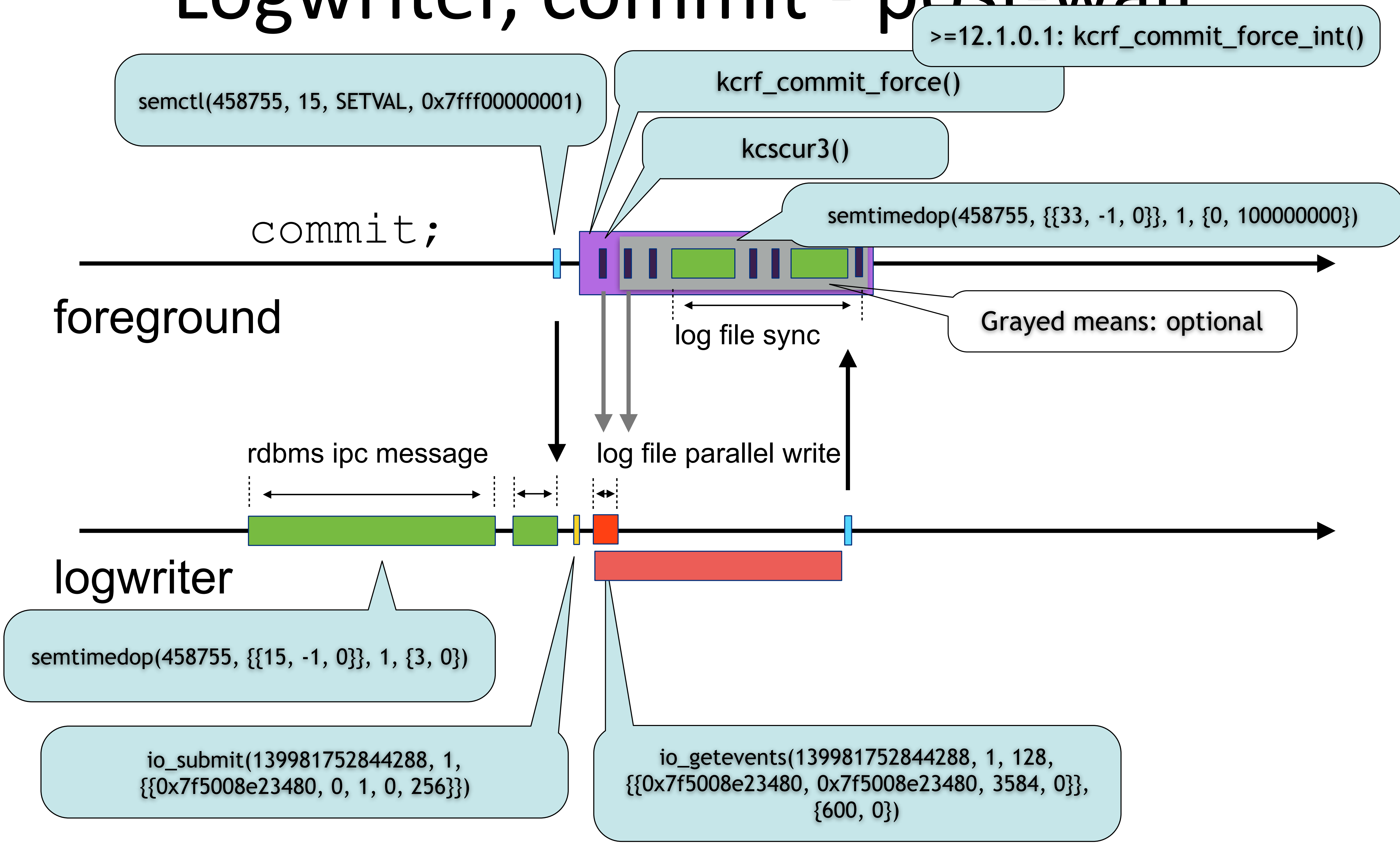
- The tests in this presentation are done with the default settings for:
  - COMMIT_LOGGING (immediate), COMMIT_WAIT (wait) and COMMIT_WRITE*.

Recap

- Following is a summary from my 'profiling the logwriter and database writer' presentation.

- There are two methods for a foreground process to understand its redo has been written:
  - Post/wait
  - Polling

Logwriter, commit - post-wait



>=12.1.0.1: kcrf_commit_force_int()

semctl(458755, 15, SETVAL, 0x7fff00000001)

kcrf_commit_force()

kcscur3()

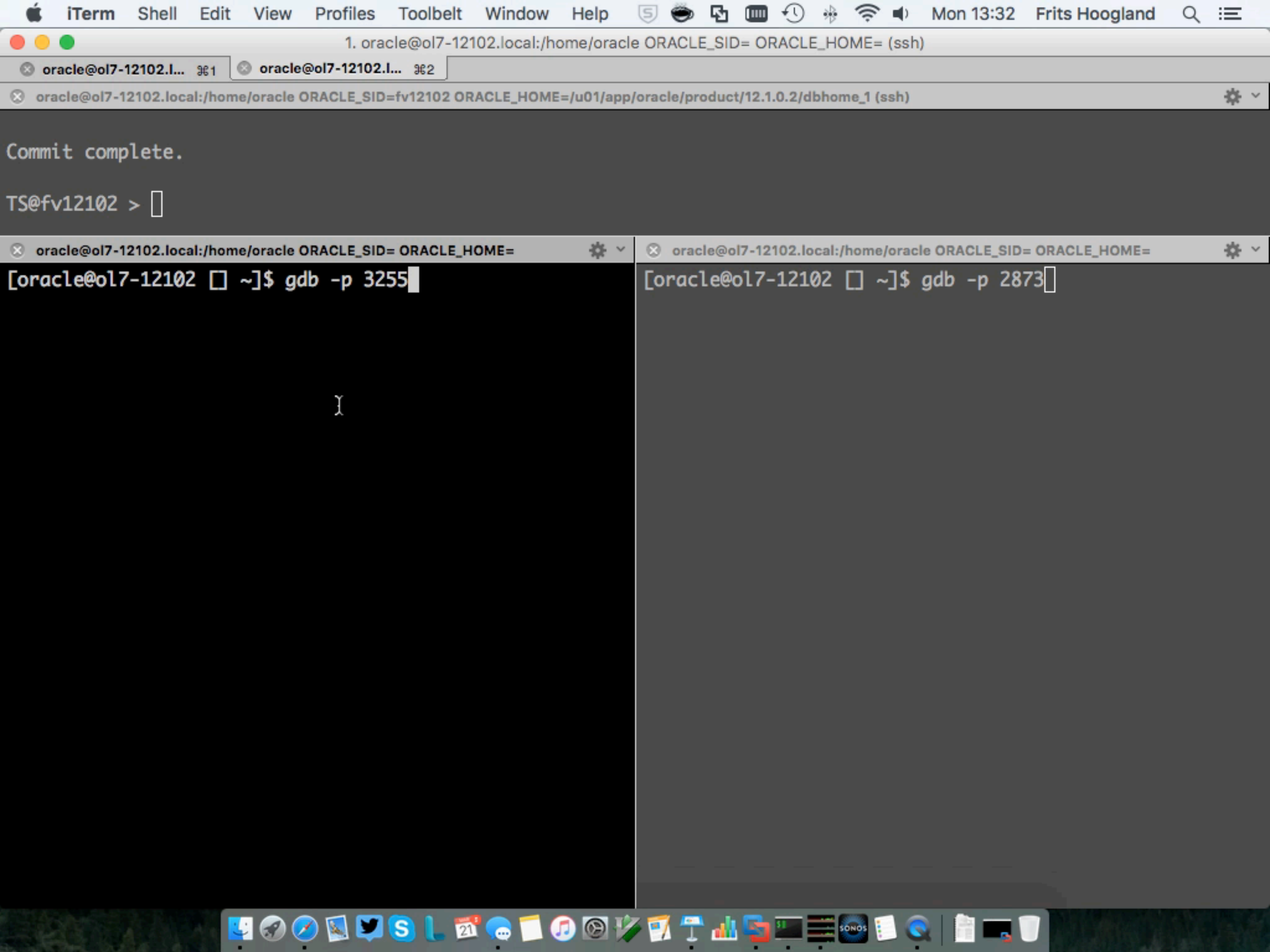
semtimedop(458755, {{33, -1, 0}}, 1, {0, 100000000})

Grayed means: optional

semtimedop(458755, {{15, -1, 0}}, 1, {3, 0})

io_submit(139981752844288, 1, {{0x7f5008e23480, 0, 1, 0, 256}})

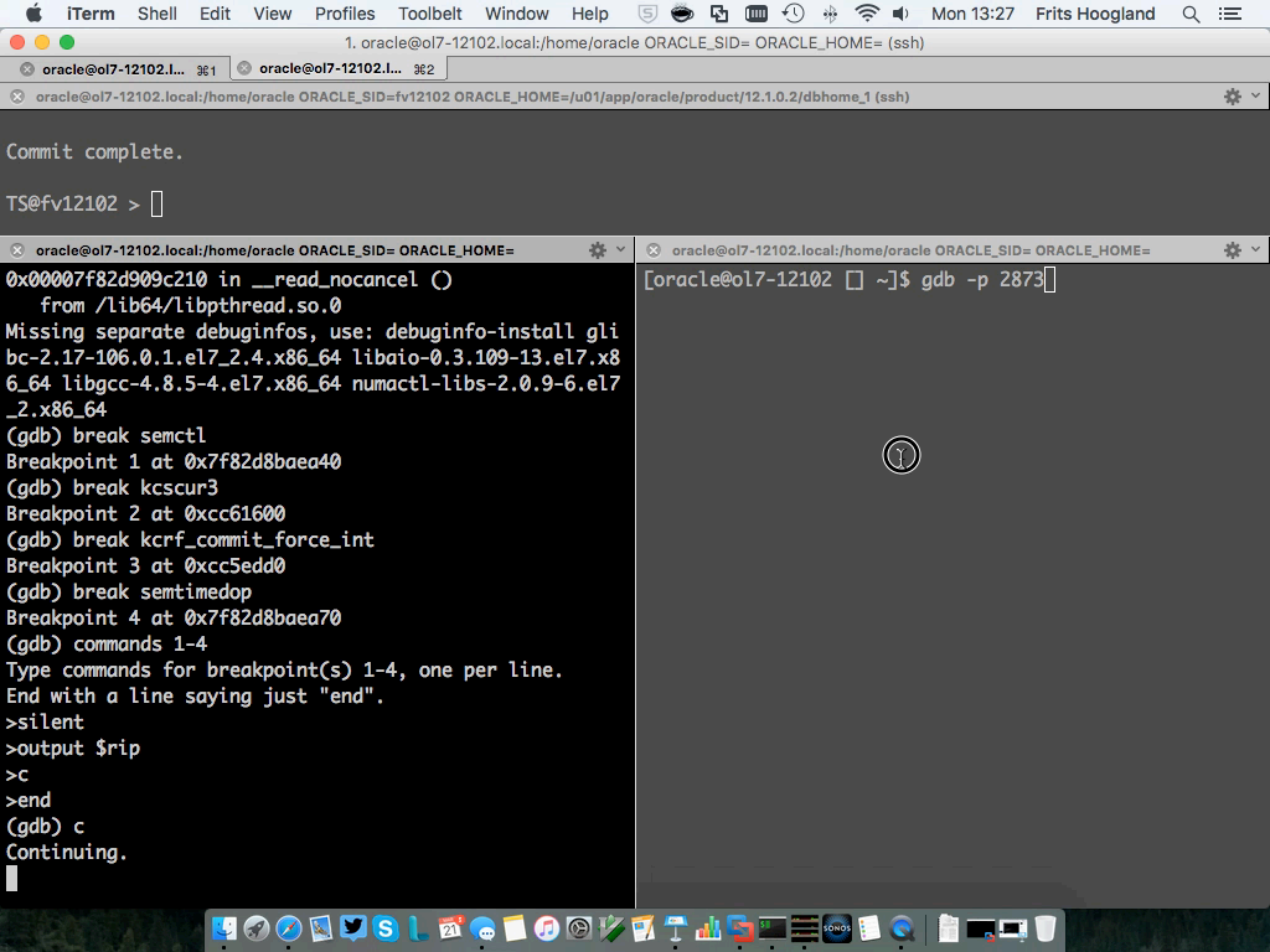
io_getevents(139981752844288, 1, 128, {{0x7f5008e23480, 0x7f5008e23480, 3584, 0}}, {600, 0})


iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 13:32 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
Commit complete.

TS@fv12102 >
```

```
[oracle@ol7-12102 □ ~]$ gdb -p 3255
```

```
[oracle@ol7-12102 □ ~]$ gdb -p 2873
```


iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 13:27 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

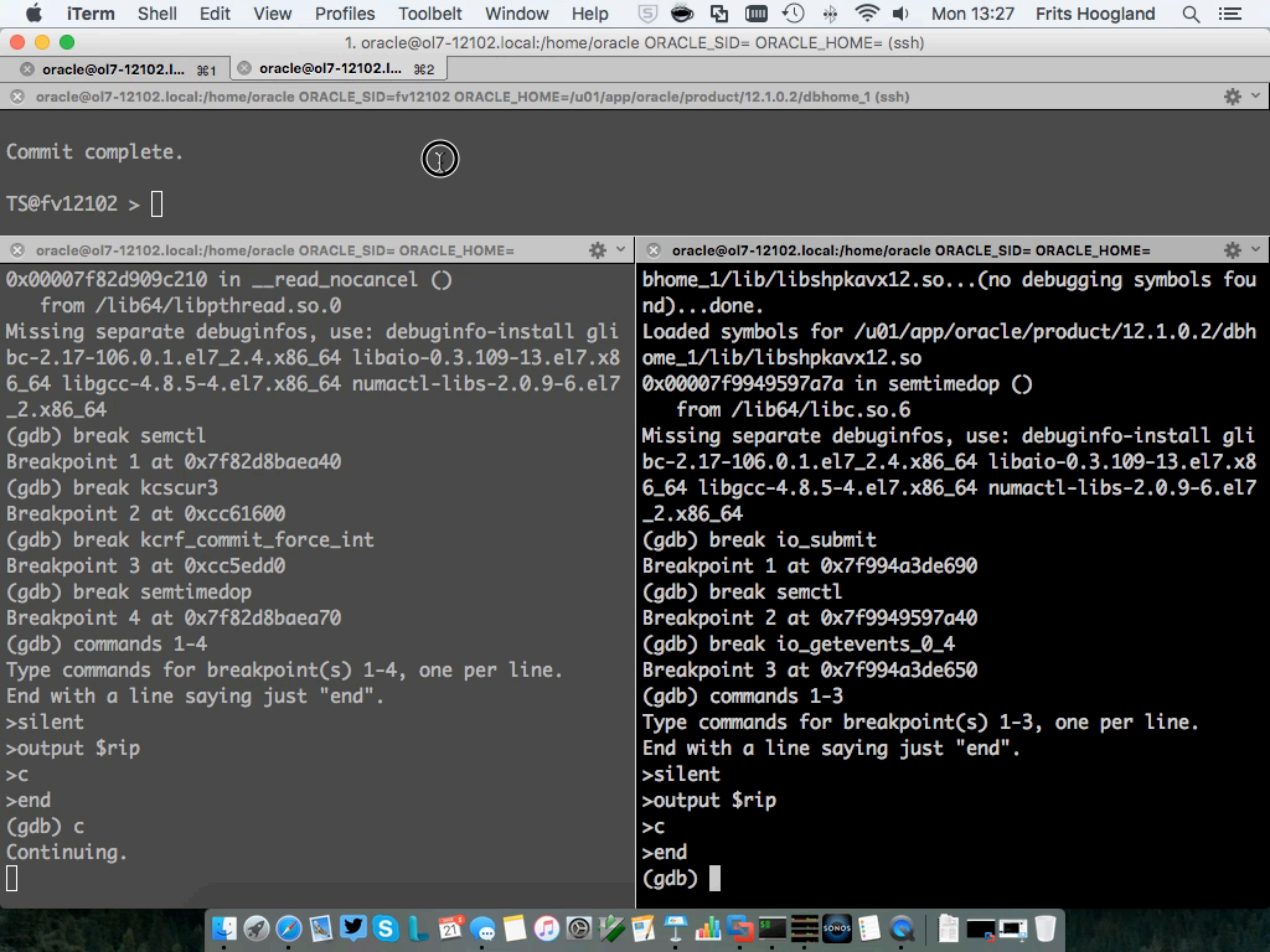
```
Commit complete.

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
0x00007f82d909c210 in __read_nocancel ()
   from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7f82d8baea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break semtimedop
Breakpoint 4 at 0x7f82d8baea70
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 ~]$ gdb -p 2873
```



oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

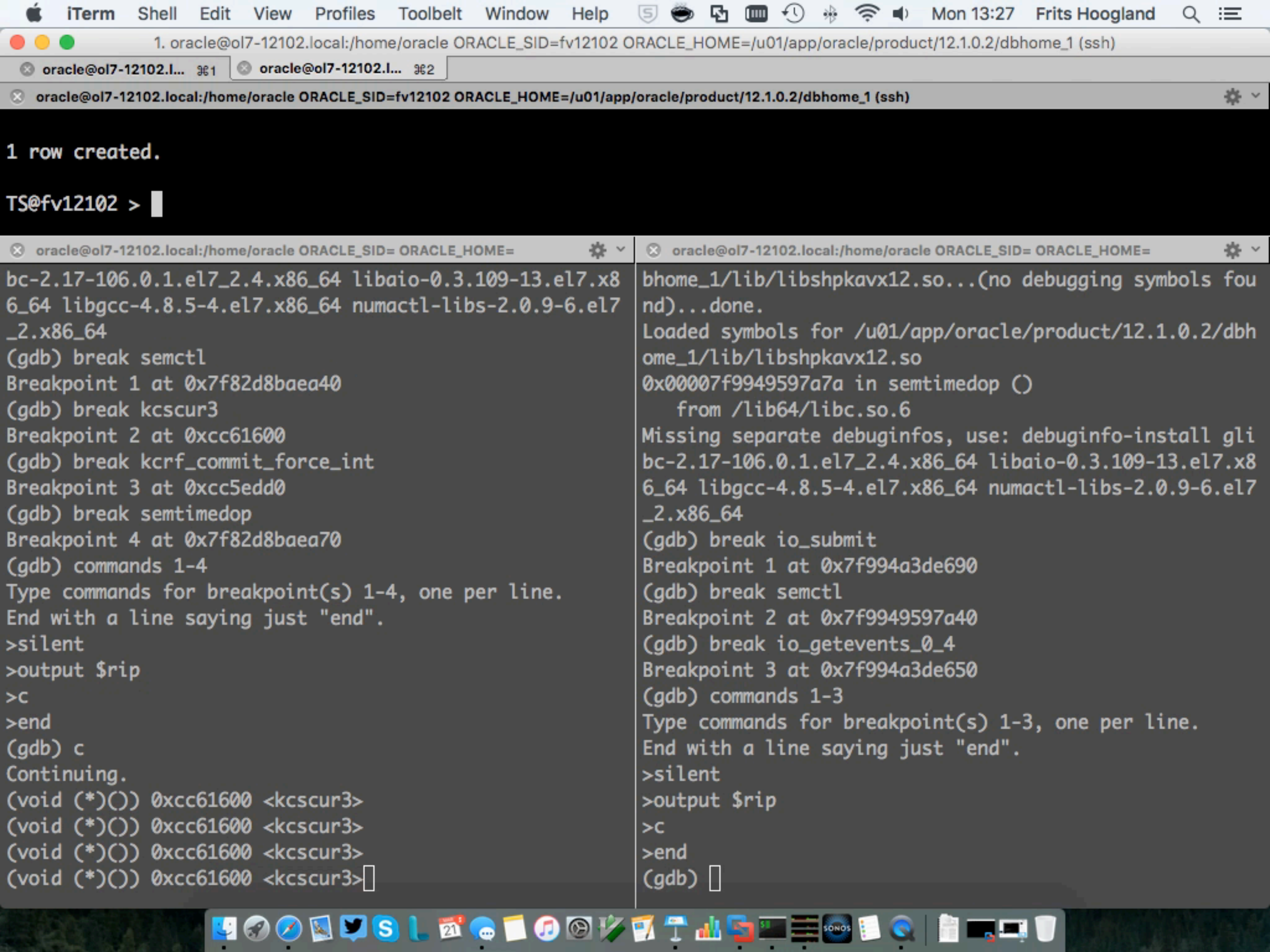
```

Commit complete.

TS@fv12102 > 
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
0x00007f82d909c210 in __read_nocancel ()
   from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7f82d8baea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break semtimedop
Breakpoint 4 at 0x7f82d8baea70
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.

```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

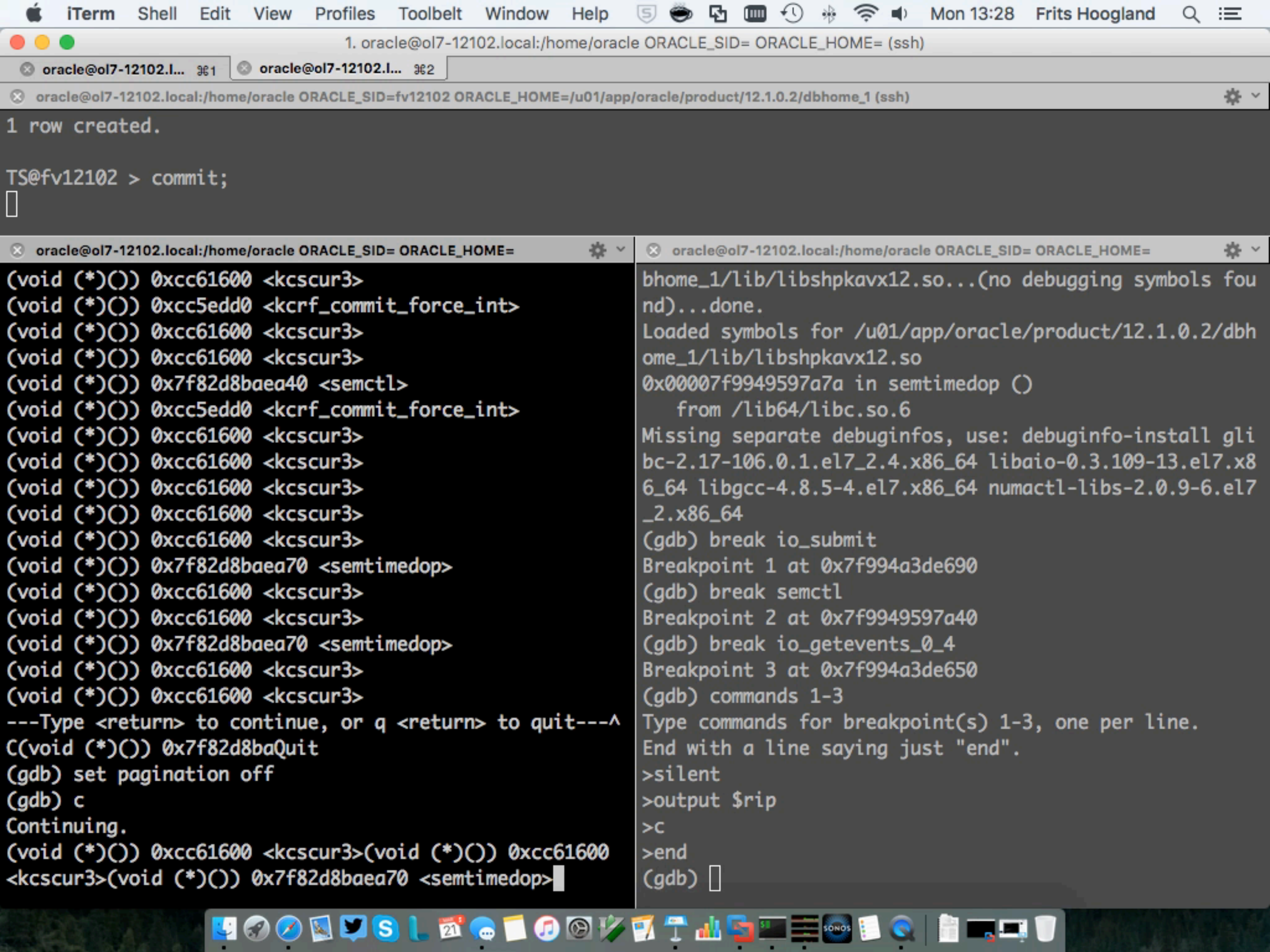
```
bhome_1/lib/libshpkavx12.so...(no debugging symbols fou
nd)...done.
Loaded symbols for /u01/app/oracle/product/12.1.0.2/dbh
ome_1/lib/libshpkavx12.so
0x00007f9949597a7a in semtimedop ()
   from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break io_submit
Breakpoint 1 at 0x7f994a3de690
(gdb) break semctl
Breakpoint 2 at 0x7f9949597a40
(gdb) break io_getevents_0_4
Breakpoint 3 at 0x7f994a3de650
(gdb) commands 1-3
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) 
```


iTerm
Shell
Edit
View
Profiles
Toolbelt
Window
Help
Mon 13:27
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
1 row created.

TS@fv12102 >
```

```
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7f82d8baea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break semtimedop
Breakpoint 4 at 0x7f82d8baea70
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
```

```
bhome_1/lib/libshpkavx12.so...(no debugging symbols fou
nd)...done.
Loaded symbols for /u01/app/oracle/product/12.1.0.2/dbh
ome_1/lib/libshpkavx12.so
0x00007f9949597a7a in semtimedop ()
   from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break io_submit
Breakpoint 1 at 0x7f994a3de690
(gdb) break semctl
Breakpoint 2 at 0x7f9949597a40
(gdb) break io_getevents_0_4
Breakpoint 3 at 0x7f994a3de650
(gdb) commands 1-3
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb)
```



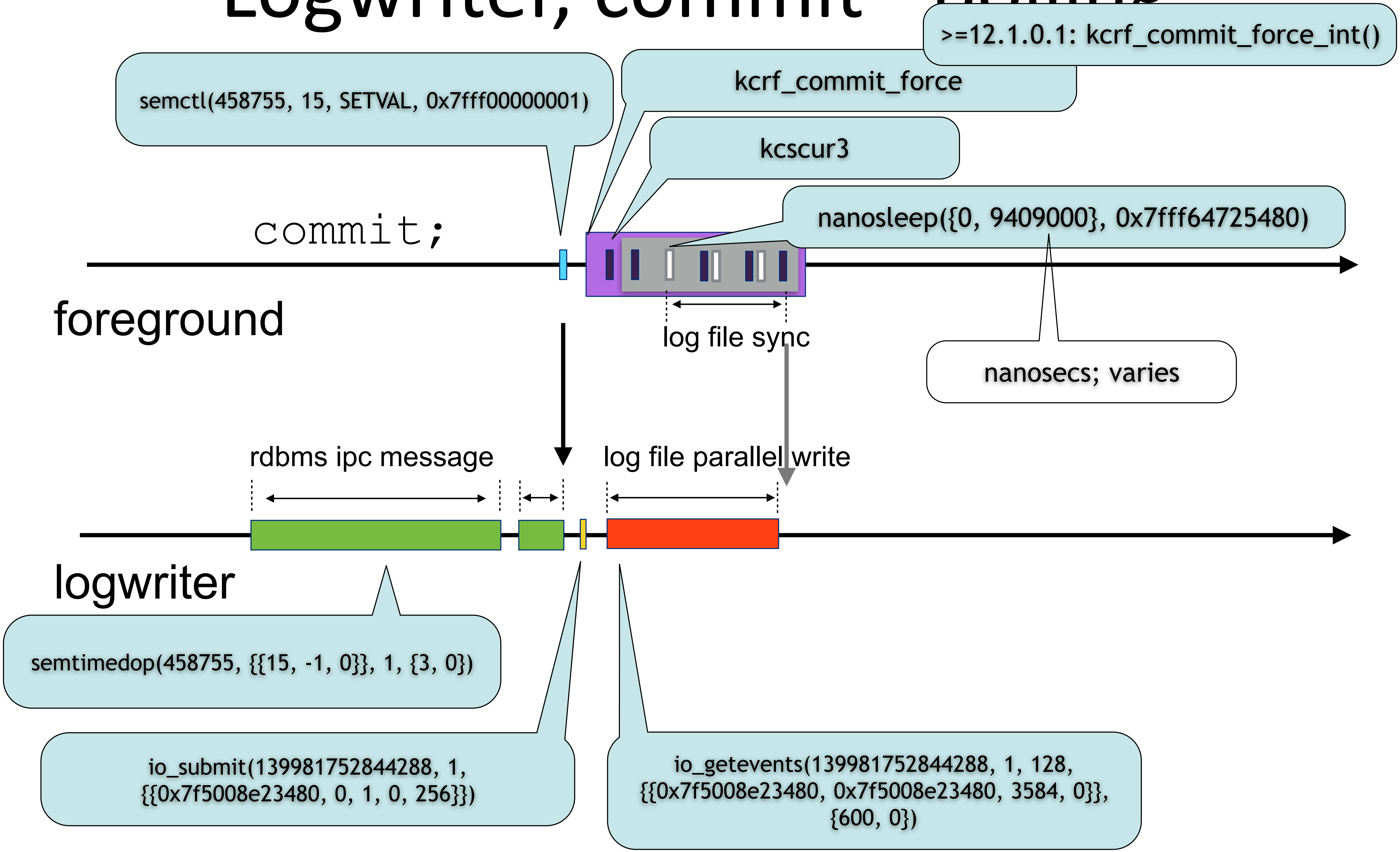
iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 13:28 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
1 row created.

TS@fv12102 > commit;
```

```
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc5edd0 <kcrf_commit_force_int>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0x7f82d8baea40 <semctl>
(void (*)()) 0xcc5edd0 <kcrf_commit_force_int>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0x7f82d8baea70 <semtimedop>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0x7f82d8baea70 <semtimedop>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
---Type <return> to continue, or q <return> to quit---^
C(void (*)()) 0x7f82d8baQuit
(gdb) set pagination off
(gdb) c
Continuing.
(void (*)()) 0xcc61600 <kcscur3>(void (*)()) 0xcc61600
<kcscur3>(void (*)()) 0x7f82d8baea70 <semtimedop>
```

```
bhome_1/lib/libshpkavx12.so...(no debugging symbols fou
nd)...done.
Loaded symbols for /u01/app/oracle/product/12.1.0.2/dbh
ome_1/lib/libshpkavx12.so
0x00007f9949597a7a in semtimedop ()
   from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break io_submit
Breakpoint 1 at 0x7f994a3de690
(gdb) break semctl
Breakpoint 2 at 0x7f9949597a40
(gdb) break io_getevents_0_4
Breakpoint 3 at 0x7f994a3de650
(gdb) commands 1-3
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb)
```

Logwriter, commit - polling



>=12.1.0.1: kcrf_commit_force_int()

semctl(458755, 15, SETVAL, 0x7fff00000001)

kcrf_commit_force

kcscur3

commit;

nanosleep({0, 9409000}, 0x7fff64725480)

foreground

log file sync

nanosecs; varies

rdbms ipc message

log file parallel write

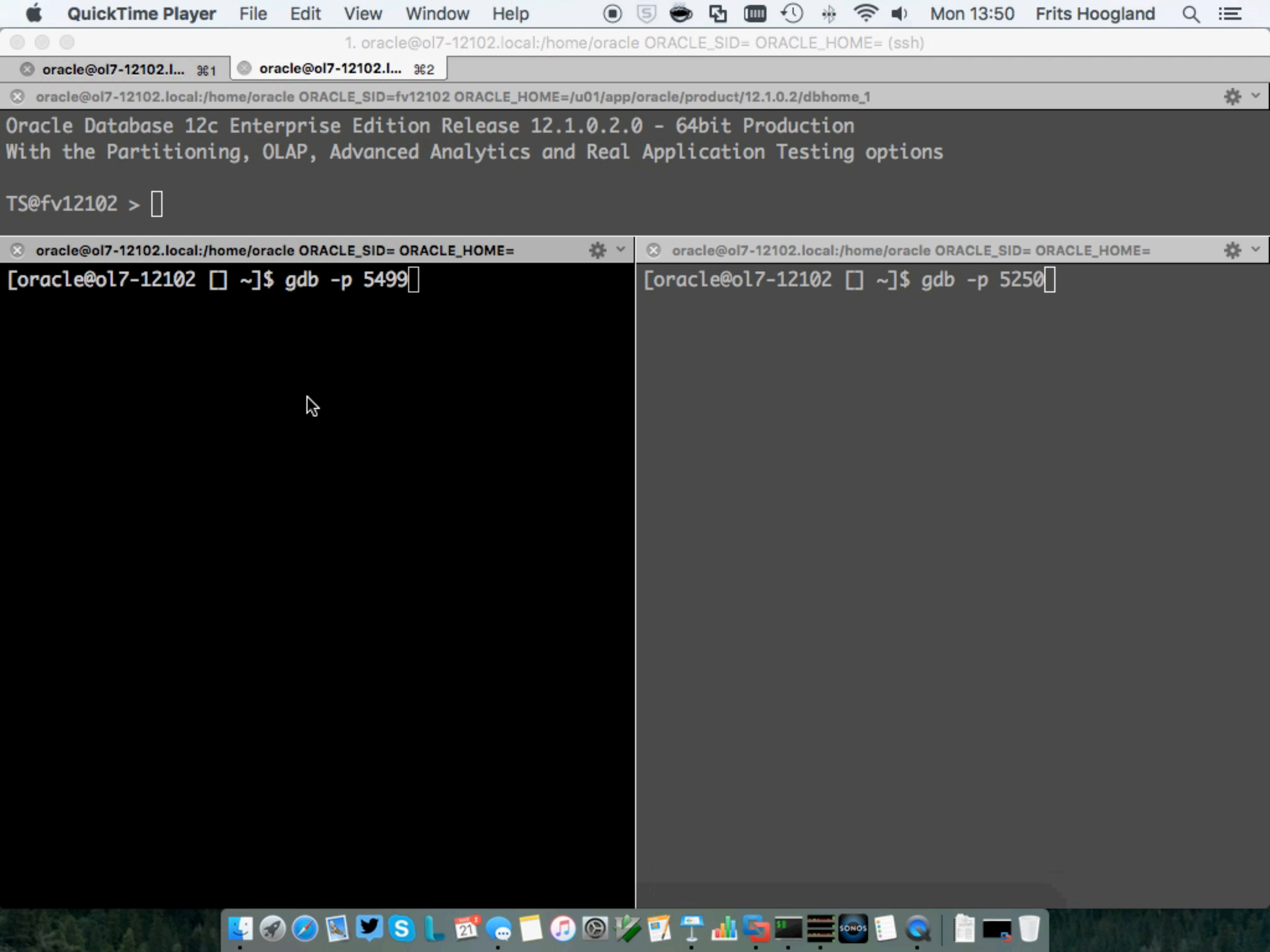
logwriter

semtimedop(458755, {{15, -1, 0}}, 1, {3, 0})

io_submit(139981752844288, 1, {{0x7f5008e23480, 0, 1, 0, 256}})

io_getevents(139981752844288, 1, 128, {{0x7f5008e23480, 0x7f5008e23480, 3584, 0}}, {600, 0})

enkitec



oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

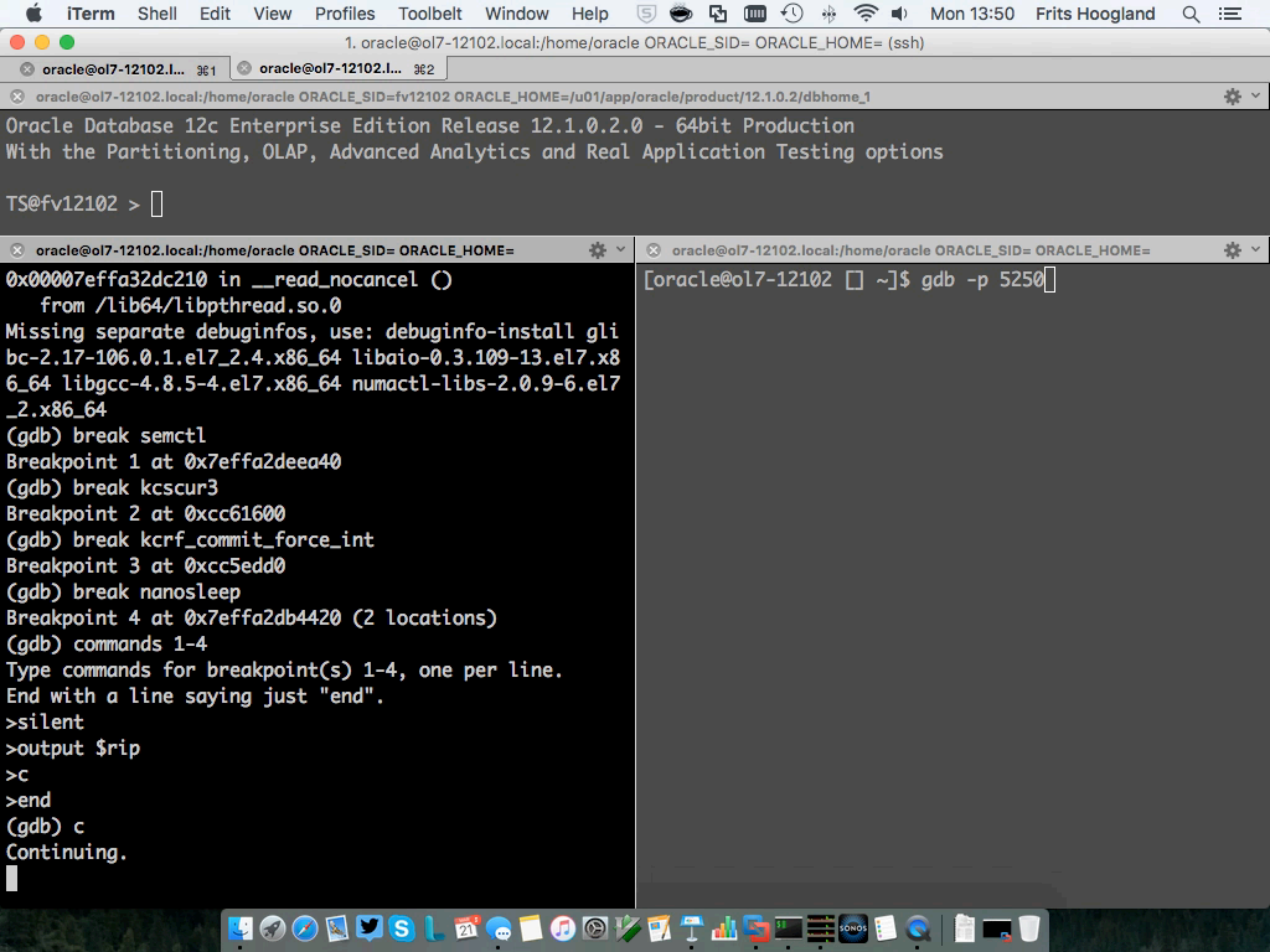
```
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 □ ~]$ gdb -p 5499
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 □ ~]$ gdb -p 5250
```



1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)

oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

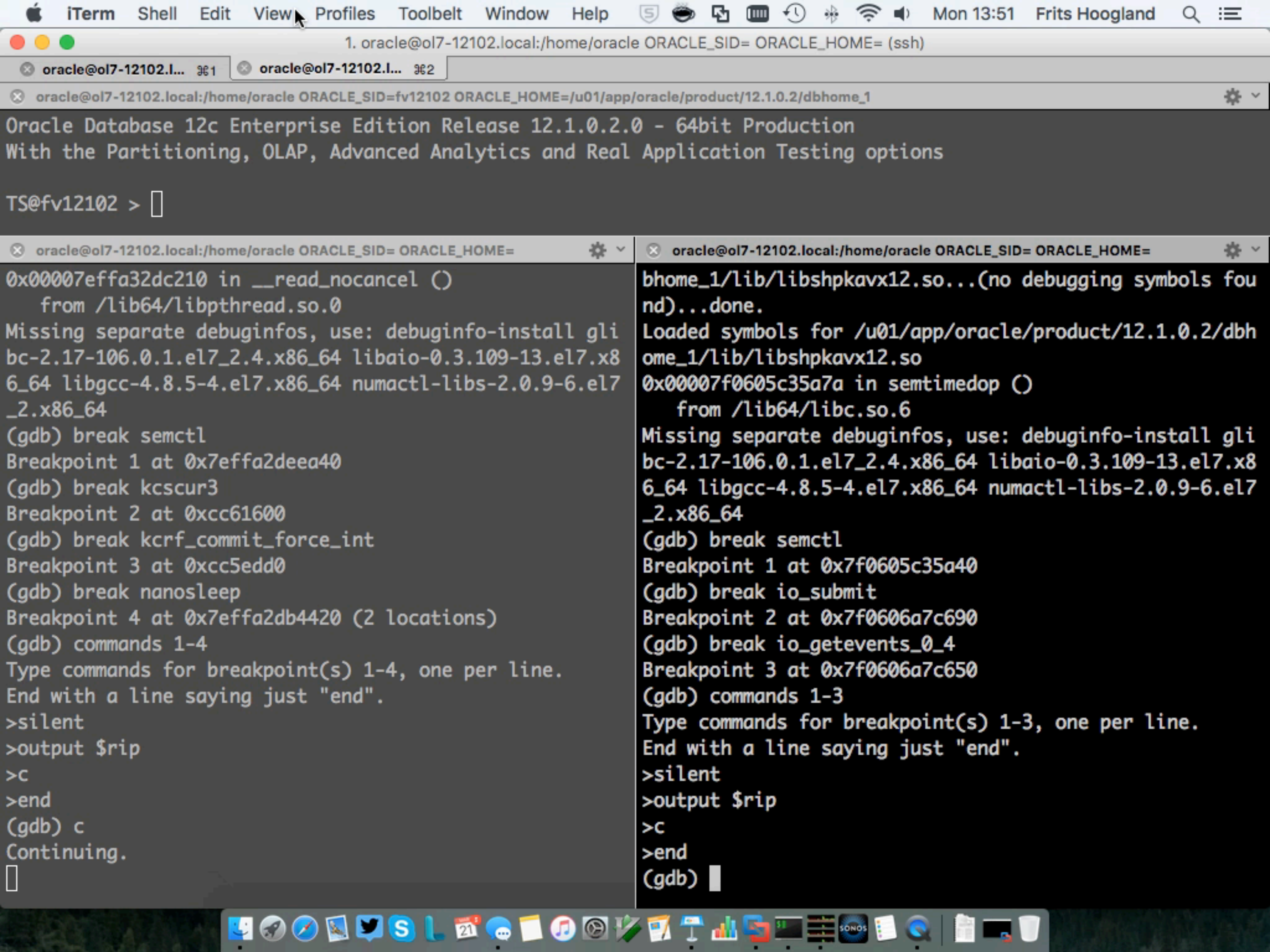
```
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
0x00007effa32dc210 in __read_nocancel ()
   from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7effa2deea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break nanosleep
Breakpoint 4 at 0x7effa2db4420 (2 locations)
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 □ ~]$ gdb -p 5250
```



oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

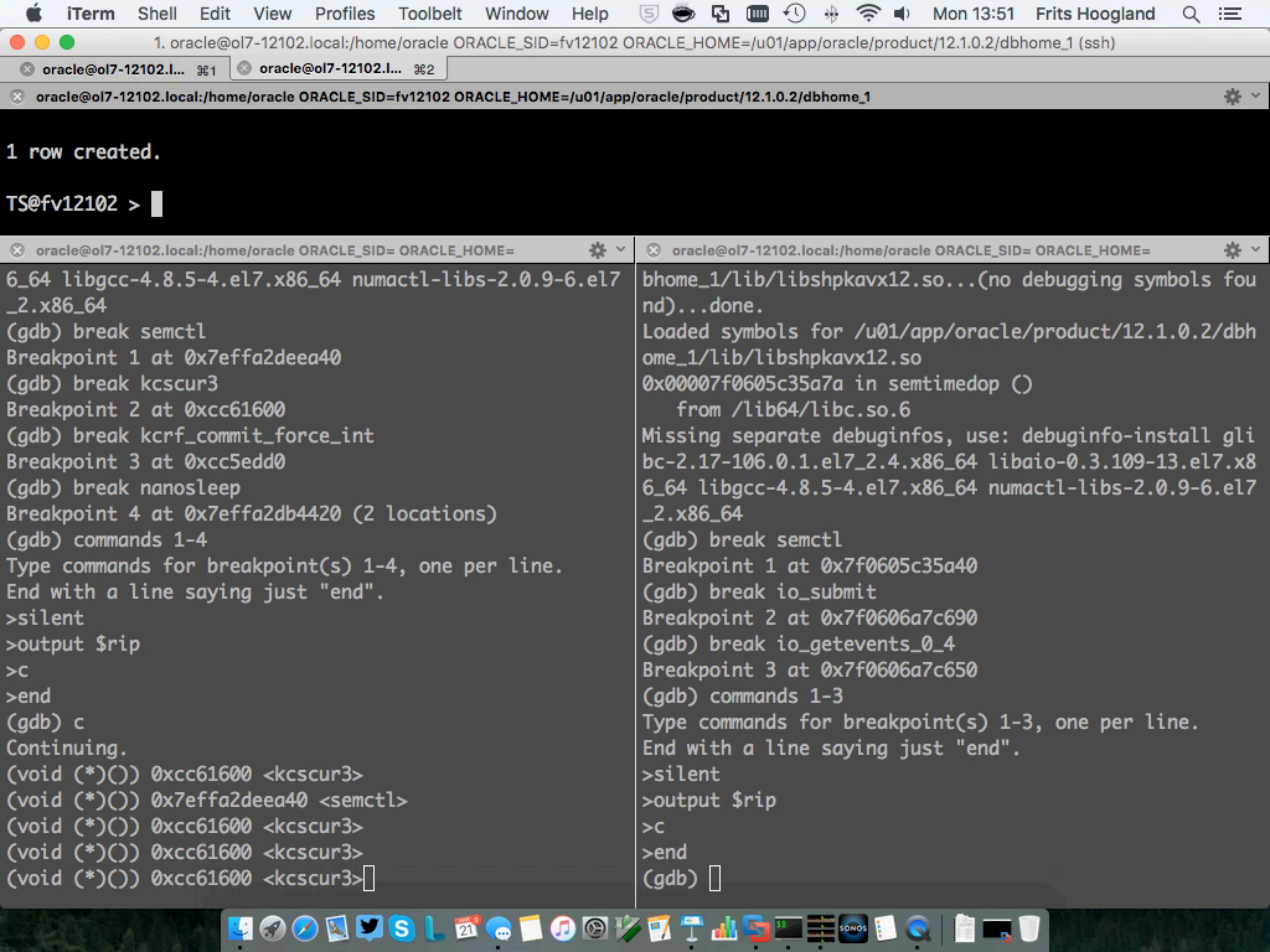
```
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
0x00007effa32dc210 in __read_nocancel ()
   from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7effa2deea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break nanosleep
Breakpoint 4 at 0x7effa2db4420 (2 locations)
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
bhome_1/lib/libshpkavx12.so...(no debugging symbols fou
nd)...done.
Loaded symbols for /u01/app/oracle/product/12.1.0.2/dbh
ome_1/lib/libshpkavx12.so
0x00007f0605c35a7a in semtimedop ()
   from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7f0605c35a40
(gdb) break io_submit
Breakpoint 2 at 0x7f0606a7c690
(gdb) break io_getevents_0_4
Breakpoint 3 at 0x7f0606a7c650
(gdb) commands 1-3
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb)
```


iTerm
Shell
Edit
View
Profiles
Toolbelt
Window
Help
Mon 13:51
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```

1 row created.

TS@fv12102 >
```

```
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7effa2deea40
(gdb) break kcscur3
Breakpoint 2 at 0xcc61600
(gdb) break kcrf_commit_force_int
Breakpoint 3 at 0xcc5edd0
(gdb) break nanosleep
Breakpoint 4 at 0x7effa2db4420 (2 locations)
(gdb) commands 1-4
Type commands for breakpoint(s) 1-4, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb) c
Continuing.
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0x7effa2deea40 <semctl>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
(void (*)()) 0xcc61600 <kcscur3>
```

```
bhome_1/lib/libshpkavx12.so...(no debugging symbols fou
nd)...done.
Loaded symbols for /u01/app/oracle/product/12.1.0.2/dbh
ome_1/lib/libshpkavx12.so
0x00007f0605c35a7a in semtimedop ()
   from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install gli
bc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x8
6_64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7
_2.x86_64
(gdb) break semctl
Breakpoint 1 at 0x7f0605c35a40
(gdb) break io_submit
Breakpoint 2 at 0x7f0606a7c690
(gdb) break io_getevents_0_4
Breakpoint 3 at 0x7f0606a7c650
(gdb) commands 1-3
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>output $rip
>c
>end
(gdb)
```

Logwriter polling mode

- There does not seem to be any notification from the LGWR to the foreground process.
  - Which must mean ‘polling’ means the foreground process polls something to see if its log buffer contents are written.

Logwriter polling mode

- previously, I discovered some functions* being called in the foreground process after semctl():
  - kcrf_commit_force() / kcrf_commit_force_int() as the main routine in responsible for making sure the redo contents are written by the LGWR.
  - nanosleep() as a way of suspending execution for a FIXED period of time.
  - kcscur3() as a function that does "something", probably scanning the commit SCN?
    - Also used p/w, edge case consistent w/assumption.

Logwriter polling mode

- Based on the observed pattern:
  - semctl (to signal the LGWR)
  - kcrf_commit_force_int (main check loop)
  - kcscur3 (supposed LGWR progress checking)
  - nanosleep (spend a calculated time off CPU)
- I assumed that kcscur3 is checking the commit SCN.
- Jonathan Lewis theorised that all the FG needed to do was keep track of the write status of its blocks in the public redo buffer.

- So the question is:

  How does a FG process in polling mode determine that its public log buffer contents are written to disk?

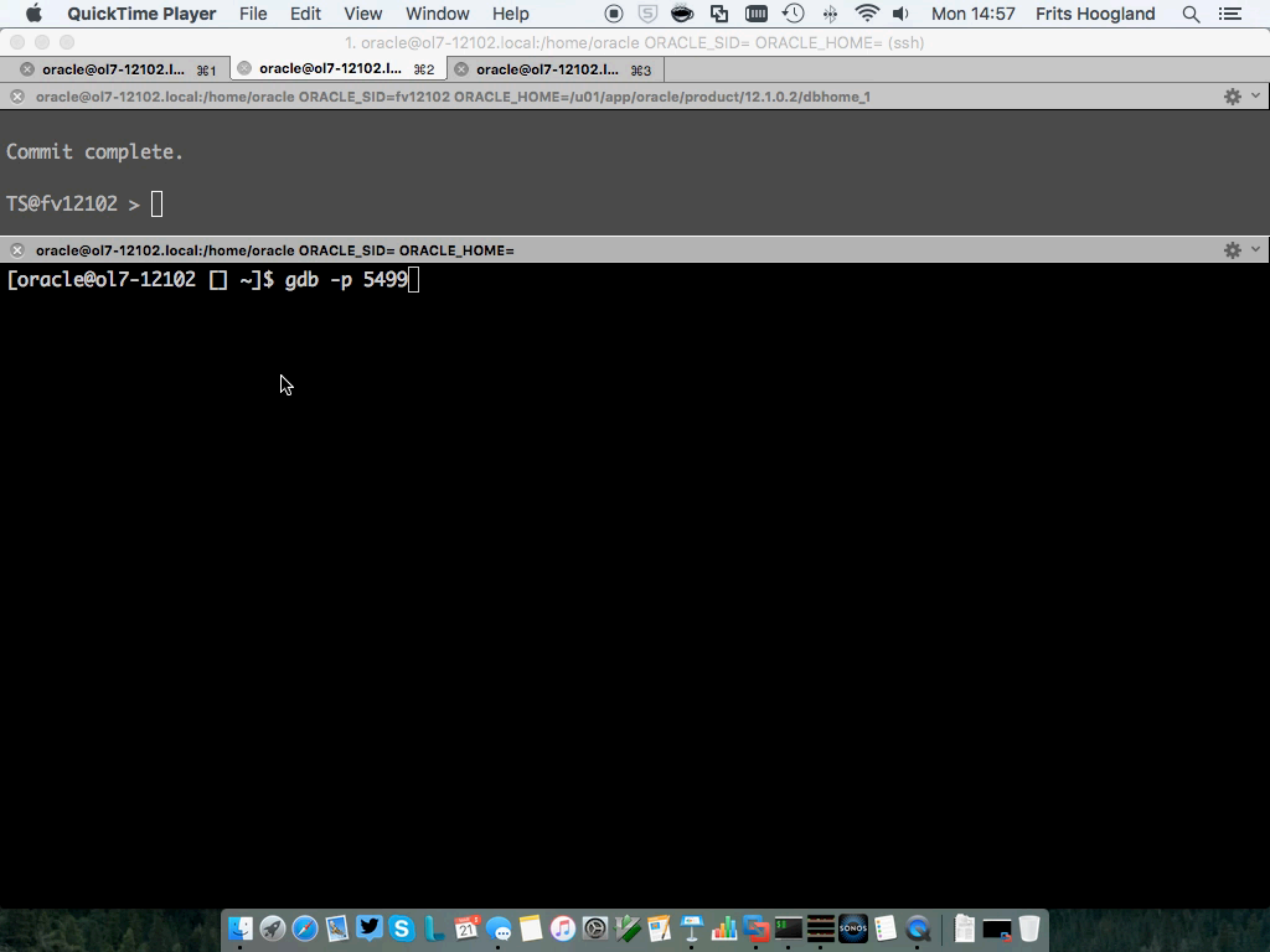
- What information do we have?

  - kcscur3()
  - …nothing else

- We don’t have source code nor debug information from the Oracle executable.
- We can fetch the function arguments:
  - Linux X86_64 follows the AMD64 ABI
  - Which means function arguments are passed via CPU registers:
    - RDI, RSI, RDX, RCX, R8, R9
  - We **do not** know the number of arguments.

kcscur3

- Let's profile the foreground session, and print the arguments of kcscur3 function.



QuickTime Player
File
Edit
View
Window
Help
Mon 14:57
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.l... ⌘3
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

```
Commit complete.

TS@fv12102 >
```

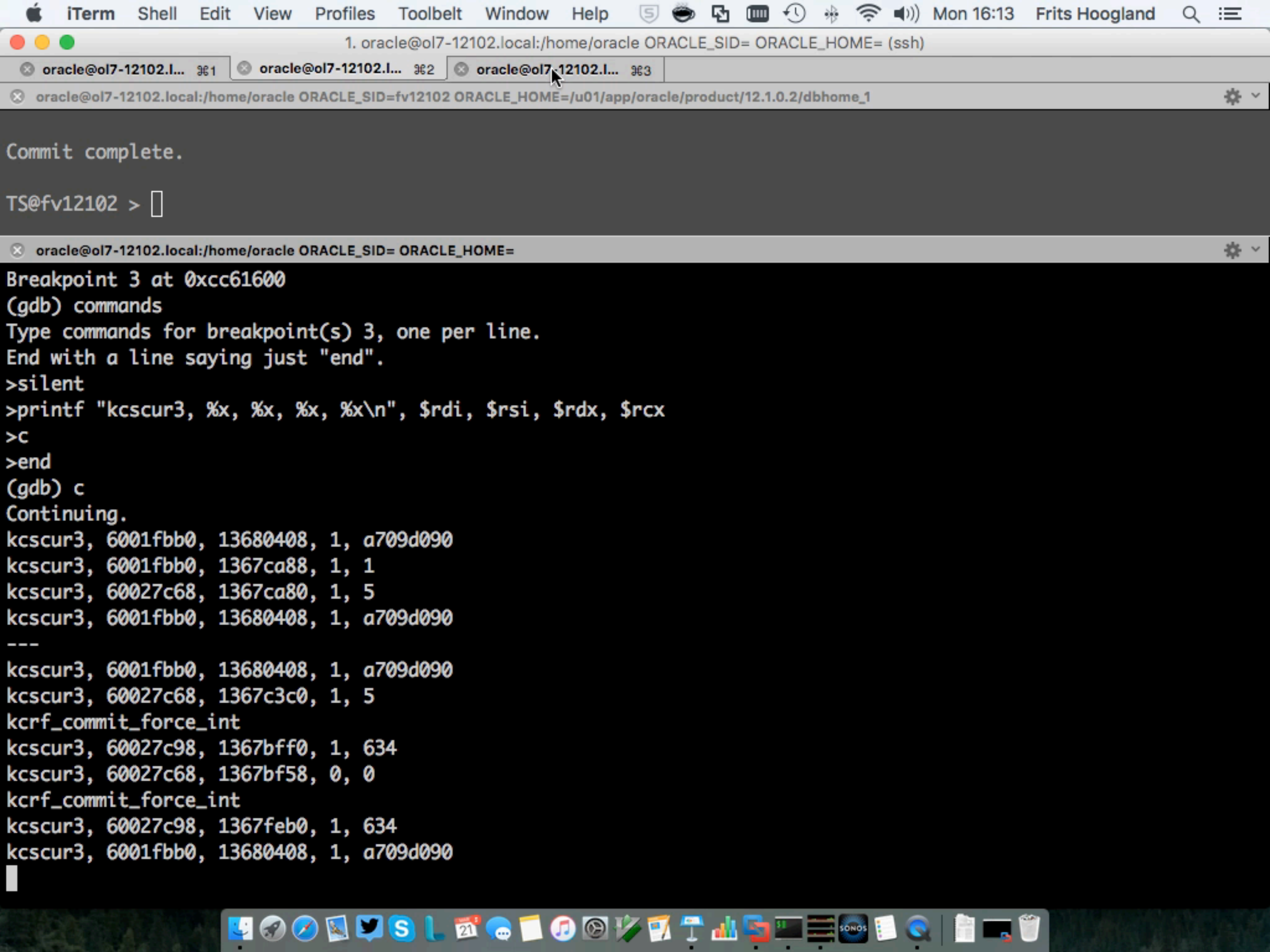
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 □ ~]$ gdb -p 5499
```

kcscur3

- This shows an oddity:
  - The foreground immediately continues.
    - Nanosleep doesn't get called.
    - This means the 'log file sync' wait is omitted too!

- Let's slow down the LGWR!
  - In order to do that, I'll add a sleep of 10ms to the IO reap (=io_getevents call) of the log writer process.


iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 16:13 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.l... ⌘3
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

```
Commit complete.

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
Breakpoint 3 at 0xcc61600
(gdb) commands
Type commands for breakpoint(s) 3, one per line.
End with a line saying just "end".
>silent
>printf "kcscur3, %x, %x, %x, %x\n", $rdi, $rsi, $rdx, $rcx
>c
>end
(gdb) c
Continuing.
kcscur3, 6001fbb0, 13680408, 1, a709d090
kcscur3, 6001fbb0, 1367ca88, 1, 1
kcscur3, 60027c68, 1367ca80, 1, 5
kcscur3, 6001fbb0, 13680408, 1, a709d090
---
kcscur3, 6001fbb0, 13680408, 1, a709d090
kcscur3, 60027c68, 1367c3c0, 1, 5
kcrf_commit_force_int
kcscur3, 60027c98, 1367bff0, 1, 634
kcscur3, 60027c68, 1367bf58, 0, 0
kcrf_commit_force_int
kcscur3, 60027c98, 1367feb0, 1, 634
kcscur3, 6001fbb0, 13680408, 1, a709d090
```

kcscur3

- Partial output of breaks, on commit:

```
Breakpoint 4, 0x00007ffc796a1a40 in semctl () from /lib64/libc.so.6

Breakpoint 2, 0x000000000cc5edd0 in kcrf_commit_force_int ()
kcscur3 60027c98, 7bfe4430, 1, 634
kcscur3 60027c68, 7bfe4398, 0, 0

Breakpoint 1, 0x00007ffc79b8f940 in nanosleep () from /lib64/libpthread.so.0
kcscur3 60027c98, 7bfe4430, 1, 94c7f210
kcscur3 60027c68, 7bfe4398, 0, 0

Breakpoint 1, 0x00007ffc79b   940 in nanosleep () from /lib64/libpthread.so.0
kcscur3 60027c98, 7bfe4430, 1, 94c
kcscur3 6001fbb0, 7bfe4988, 1, 7d950090
```

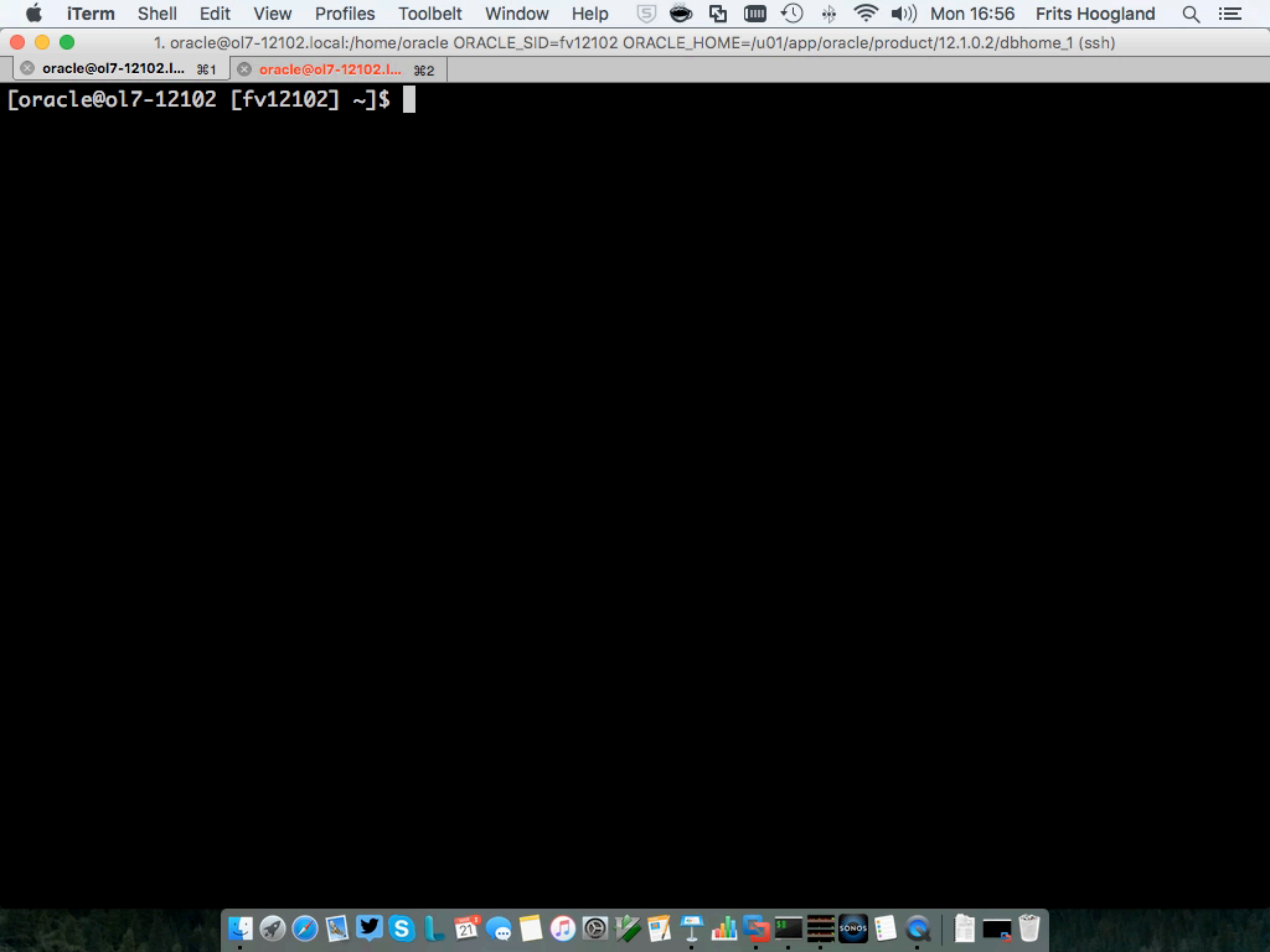
There are 3 addresses as the first argument to kcscur3(): **0x60027c98**, **0x60027c68** and **0x6001fbb0**.

kcscur3

- What are these addresses?
- Let’s see if these are shared memory addresses:

kcscur3

- The shared memory area’s of an Oracle database are placed in a couple of shared memory segments.
- These shared memory segments addresses can be dumped with:
  - oradebug ipc


iTerm
Shell
Edit
View
Profiles
Toolbelt
Window
Help
Mon 16:56
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2

```
[oracle@ol7-12102 [fv12102] ~]$
```

kcscur3

```
Handle:              0x117fdea0 `/u01/app/oracle/product/12.1.0.2/dbhome_1fv12102'
 Dump of unix-generic realm handle `/u01/app/oracle/product/12.1.0.2/
dbhome_1fv12102', flags = 00000000
  key 3512777704 actual_key 3512777704 num_areas 4 num_subareas 4
  primary shmid: 753667 primary sanum 3 version 3
  deferred alloc: FALSE (0) def_post_create: FALSE (0) exp_memlock: 1002M
 Area #0 `Fixed Size' containing Subareas 2-2
  Total size 00000000002cbe70 Minimum Subarea size 00000000
   Area  Subarea    Shmid     Segment Addr     Stable Addr      Actual Addr
      0        2   655360 0x00000060000000 0x00000060000000 0x00000060000000
               Subarea size     Segment size   Req_Protect  Cur_protect
                          00000000002cc000 00000000002cc000 default       readwrite
 Area #1 `Variable Size' containing Subareas 0-0
  Total size 0000000036000000 Minimum Subarea size 00400000
   Area  Subarea    Shmid     Segment Addr     Stable Addr      Actual Addr
      1        0   688129 0x00000060400000 0x00000060400000 0x00000060400000
               Subarea size     Segment size   Req_Protect  Cur_protect
                          0000000036000000 0000000036000000 default       readwrite
```

Fixed SGA

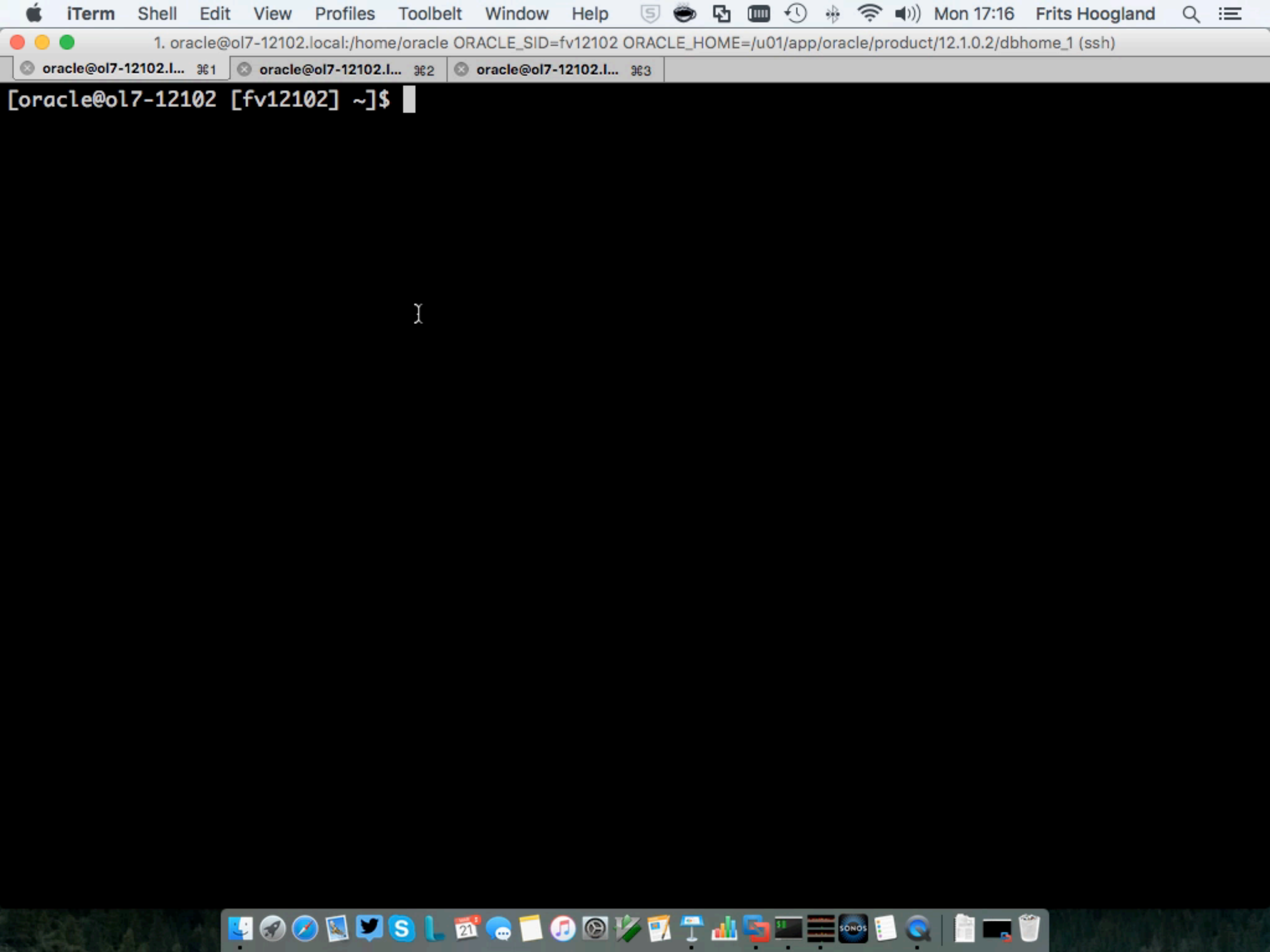
- The fixed SGA variables are visible in x$ksmfsv
  - The fixed SGA contains more than SGA variables, like latches*.

```
SQL> select ksmfsnam, ksmfsadr, ksmfssiz from x$ksmfsv
  2  where to_number(‘60027c98’,’XXXXXXXX’)
  3  between to_number(ksmfsadr,’XXXXXXXXXXXXXXXX’)
  4  and to_number(ksmfsadr,’XXXXXXXXXXXXXXXX’)+ksmfssiz-1;
KSMFSNAM                       KSMFSADR           KSMFSSIZ
------------------------------ ---------------- --------
kcrfsg_                        0000000060027C30     1608
-- and 60027c68:
KSMFSNAM                       KSMFSADR           KSMFSSIZ
------------------------------ ---------------- --------
kcrfsg_                        0000000060027C30     1608
-- and 6001fbb0:
KSMFSNAM                       KSMFSADR           KSMFSSIZ
------------------------------ ---------------- --------
kcsgscn_                       000000006001FBB0       48
```

- The two addresses 0x60027c98 & 0x60027c68
- Point to a fixed SGA variable called ‘kcrfsg’
  - This variable starts at 0x60027c30
  - This likely is a c ‘struct’, which resembles a table.

- And the address 0x6001fbb0
- Points to a fixed SGA variable called ‘kcsgscn’

- Okay, one step at a time…
- What else can we see?
- What values do these memory locations contain?



```
[oracle@ol7-12102 [fv12102] ~]$
```

struct kcrfsg & kcsgscn

- gdb x command: examine, /d=decimal

```
(gdb) x/d 0x60027c98
0x60027c98:   537122
(gdb) x/d 0x60027c68
0x60027c68:   537124
(gdb) x/d 0x6001fbb0
0x6001fbb0:   537126
```

- What are these numbers??

```
SQL> select current_scn from v$database;

CURRENT_SCN
-----------
     537136
```

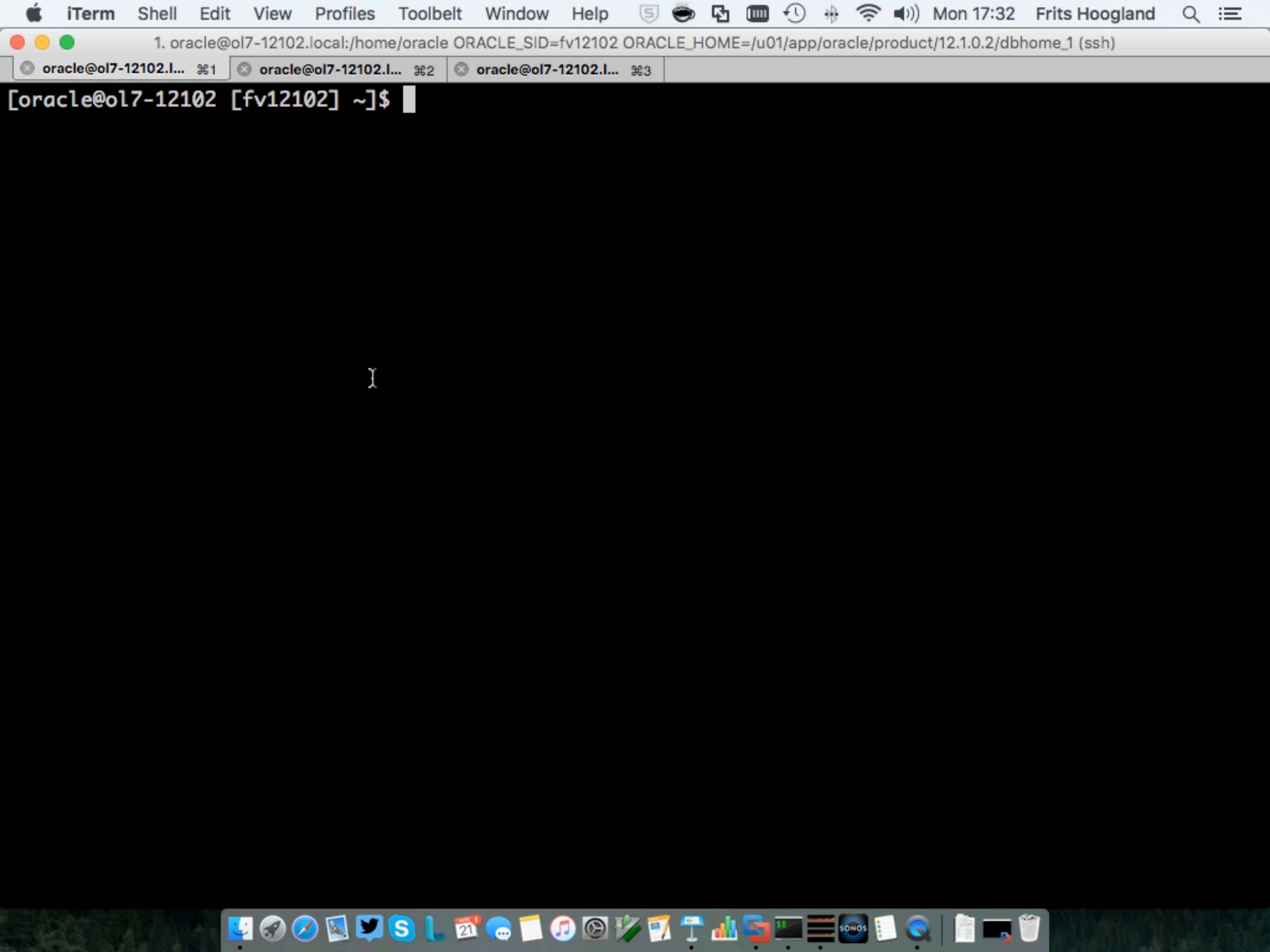
- That’s too close to be a coincidence!
  - It looks like these all contain SCNs!
  - Another small step taken.

Fixed SGA variable kcsgscn

- KCSGSCN (alias address 0x6001fbb0)
  - KCS - probably Kernel Cache Service
  - G - global? group?
  - SCN - probably SCN; System Change Number

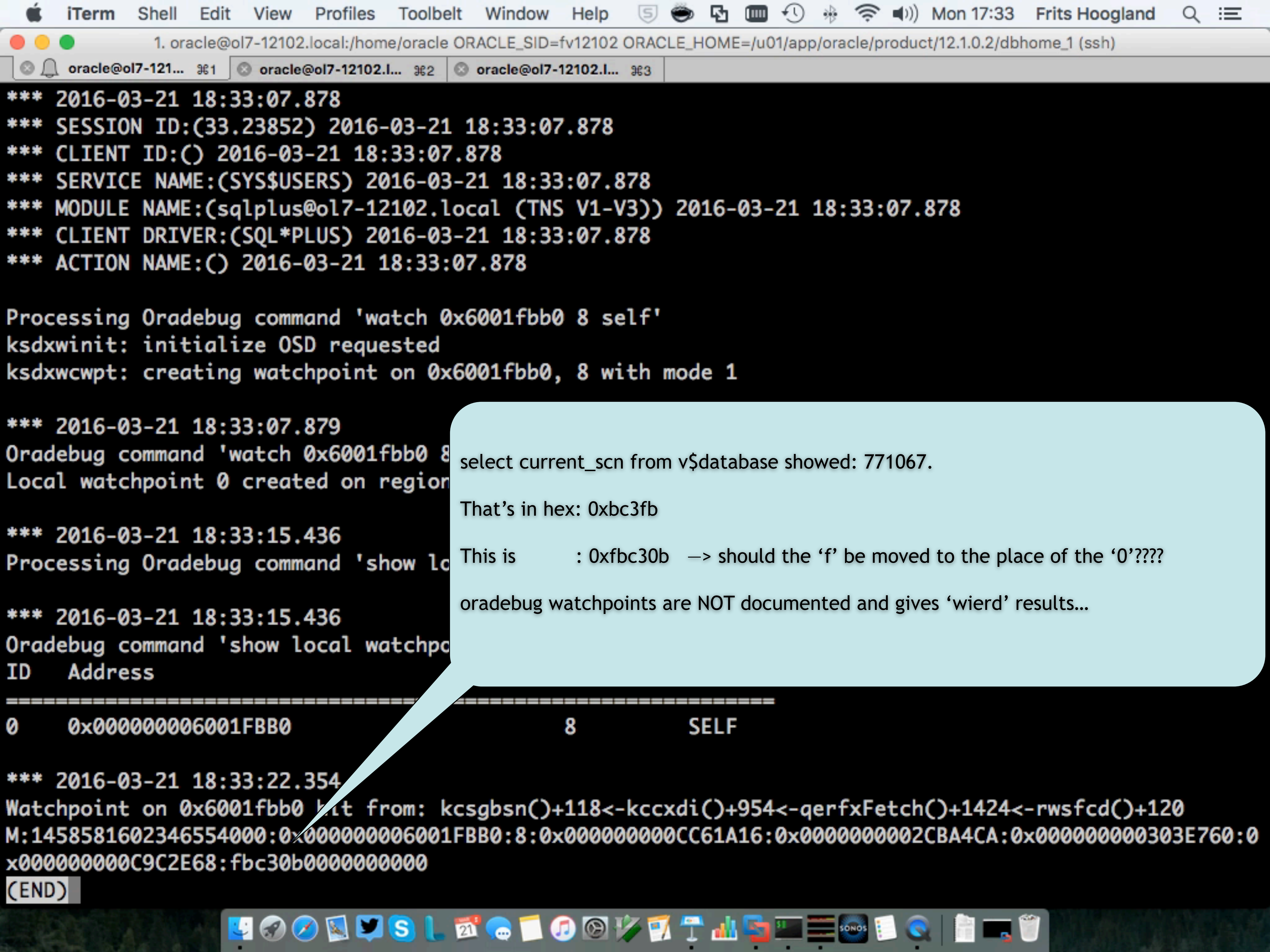
- A way of detecting usage of kcsgscn is using a **watchpoint.**

oradebug watchpoint



1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

```
[oracle@ol7-12102 [fv12102] ~]$
```


iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 17:33 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-121... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.l... ⌘3

```
*** 2016-03-21 18:33:07.878
*** SESSION ID:(33.23852) 2016-03-21 18:33:07.878
*** CLIENT ID:() 2016-03-21 18:33:07.878
*** SERVICE NAME:(SYS$USERS) 2016-03-21 18:33:07.878
*** MODULE NAME:(sqlplus@ol7-12102.local (TNS V1-V3)) 2016-03-21 18:33:07.878
*** CLIENT DRIVER:(SQL*PLUS) 2016-03-21 18:33:07.878
*** ACTION NAME:() 2016-03-21 18:33:07.878

Processing Oradebug command 'watch 0x6001fbb0 8 self'
ksdxwinit: initialize OSD requested
ksdxwcwpt: creating watchpoint on 0x6001fbb0, 8 with mode 1

*** 2016-03-21 18:33:07.879
Oradebug command 'watch 0x6001fbb0 8
Local watchpoint 0 created on region

*** 2016-03-21 18:33:15.436
Processing Oradebug command 'show lo

*** 2016-03-21 18:33:15.436
Oradebug command 'show local watchpo
ID   Address
==============================================================
0    0x000000006001FBB0                        8         SELF

*** 2016-03-21 18:33:22.354
Watchpoint on 0x6001fbb0 hit from: kcsgbsn()+118<-kccxdi()+954<-qerfxFetch()+1424<-rwsfcd()+120
M:1458581602346554000:0x000000006001FBB0:8:0x000000000CC61A16:0x0000000002CBA4CA:0x000000000303E760:0
x000000000C9C2E68:fbc30b0000000000
(END)
```

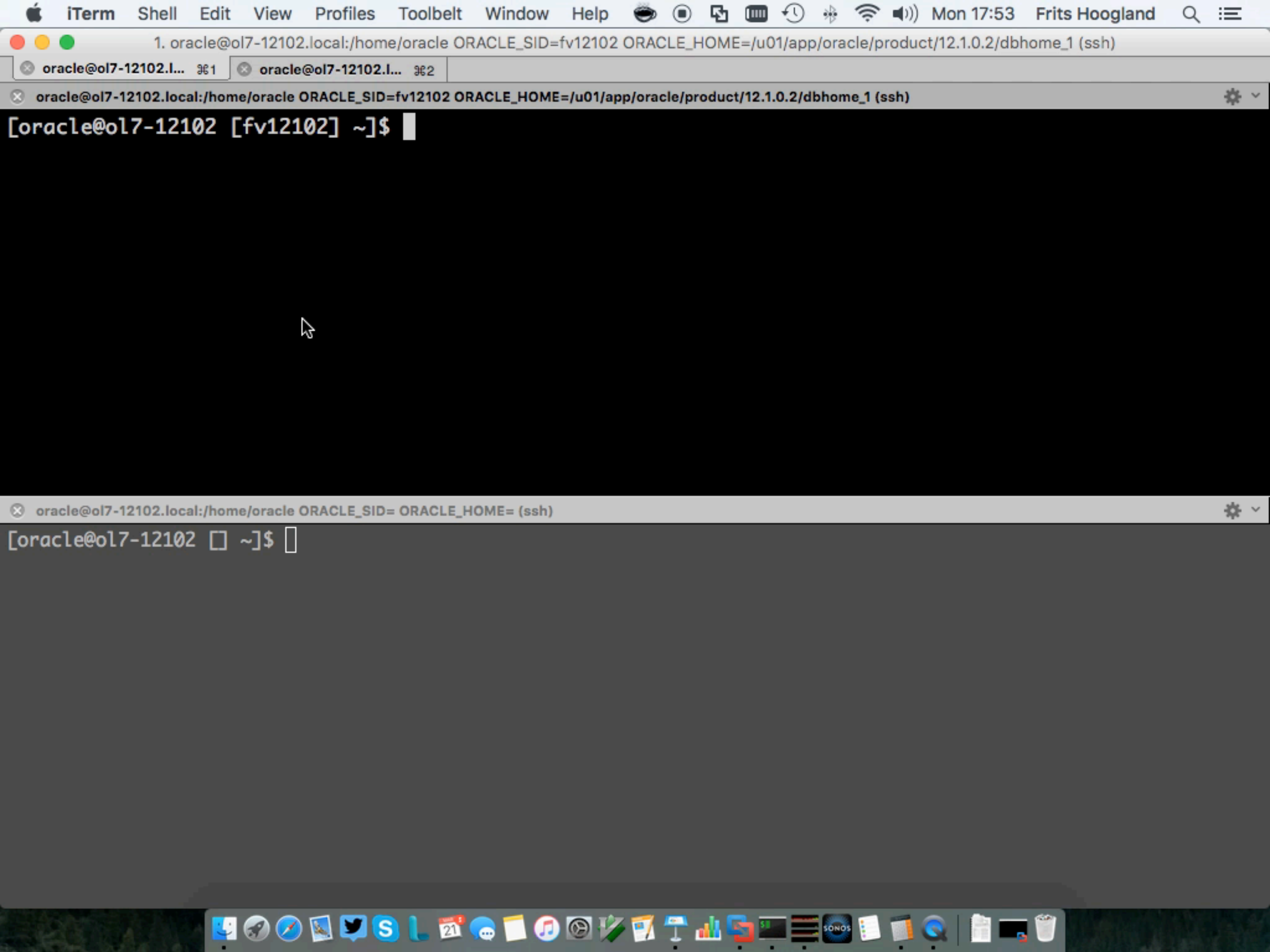
select current_scn from v$database showed: 771067.

That's in hex: 0xbc3fb

This is : 0xfbc30b —> should the 'f' be moved to the place of the '0'????

oradebug watchpoints are NOT documented and gives 'wierd' results…

gdb watchpoint



iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 17:53 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)

```
[oracle@ol7-12102 [fv12102] ~]$
```

```
[oracle@ol7-12102 □ ~]$
```

Fixed SGA variable kcsgscn

- The watchpoint shows current_scn using this variable

- In the previous gdb watchpoint we saw the function kcsgbsn() accessing it.
  - kcsgbsn = kernel cache service get batched SCN

- **kcsgscn contains the current SCN for the instance.**

struct kcrfsg

- Let’s see of there are x$ views that resemble the struct name:

```
SQL> select name from v$fixed_table where upper(name) like upper(‘%kcrf%’);
NAME
------------------------------
X$KCRFWS
X$KCRFSTRAND
X$KCRFDEBUG
X$KCRFX
```

struct kcrfsg

```
SQL> select addr from x$kcrfx;
no rows selected
SQL> select addr from x$kcrfdebug;
ADDR
----------------
0000000060028828
SQL> select addr from x$kcrfstrand;
ADDR
----------------
00007F68F1F195C0
00007F68F1F195C0
SQL> select addr from x$kcrfws;
ADDR
----------------
0000000060027C38
```

Close, but past kcrfsg (0x60027c30) and 0x60027c98 and 0x60027c68.

These are PGA memory addresses (the high ones structurally are). And this makes sense with X$KCRFSTRAND, which probably has to do with *private* redo strands.

Bingo! 8 bytes past kcrfsg (0x60027c30) and before 0x60027c98 and 0x60027c68.

X$KCRFWS

- So we got a fixed SGA variable called kcrfsg_
- Which is (quite probably) a struct called kcrfsg
- Which is externalised by X$KCRFWS

- Is X$KCRFWS used in a ‘dynamic performance view’, alias a V$ view?

X$KCRFWS

```
SQL> select view_name from v$fixed_view_definition
  2    where lower(view_definition) like '%kcrfws%';

VIEW_NAME
------------------------------
GV$XSTREAM_CAPTURE
```

- Streams??
  - Xstream is the next generation streams used by OGG.
  - Actually, it makes sense that streams/OGG have a strong dependency on redo write details!

X$KCRFWS

- Streams??
  - Searching the internet I found sites mentioning X $KCRFWS is related to streams.
    - Yes, a view related to streams uses it.
  - *Don't trust the internet until you have verified!*

- X$KCRFWS is all about redo.
  - My *guess* is X$KCRFWS actually means:
    - "Kernel Cache Redo File Write Status".

X$KCRFWS

Okay, what do we know?
- We got a view called X$KCRFWS that describes redo writing.
- It contains one record (in my case).
- The starting address is in the ADDR field.
- Which field(s) are 0x60027c68 and 0x60027c98?

```
Name
--------------------------------------------
ADDR
INDX                                               NUMBER
INST_ID                                            NUMBER
CON_ID                                             NUMBER
NEXT_BLK                                           NUMBER
LAST_BLK                                           NUMBER
ON_DISK_SCN_BAS                                    NUMBER
ON_DISK_SCN_WRP                                    NUMBER
ON_DISK_PING_SCN_BAS                                    R
ON_DISK_PING_SCN_WRP                               NUMBER
LAST_WRITE_SCN_BAS                                 NUMBER
LAST_WRITE_SCN_WRP                                      R
LWN_SCN_BAS                                        NUMBER
LWN_SCN_WRP                                        NUMBER
LAST_WRITE_SCN                                     NUMBER
LAST_WRITE_SCN_TIME                                DATE
REAL_REDO_SCN_BAS                                  NUMBER
REAL_REDO_SCN_WRP                                  NUMBER
REAL_WRITE_TIME                                    DATE
```

0x60027c38

0x60027c68?

0x60027c98?

X$KCRFWS

- I first thought using the “magic offset table” would be an easy way:

```
SQL> select c.kqfconam, c.kqfcooff
  2  from x$kqfco c, x$kqfta t
  3  where t.indx = c.kqfcotab
  4  and t.kqftanam='X$KCRFWS'
  5  order by c.kqfcooff;
```

```
KQFCONAM                           KQFCOOFF
------------------------------ ----------
ADDR                                    0
INDX                                    0
REAL_REDO_SCN_WRP                       0
REAL_REDO_SCN_BAS                       0
LWN_SCN_WRP                             0
LWN_SCN_BAS                             0
ON_DISK_PING_SCN_WRP                    0
INST_ID                                 0
CON_ID                                  0
NEXT_BLK                                0
ON_DISK_SCN_BAS                         0
ON_DISK_SCN_WRP                         0
ON_DISK_PING_SCN_BAS                    0
LAST_BLK                                4
LAST_WRITE_SCN_TIME                    12
LAST_WRITE_SCN_BAS                    144
LAST_WRITE_SCN                        144
LAST_WRITE_SCN_WRP                    148
REAL_WRITE_TIME                       328
```

0x60027c68-0x60027c38= 48
0x60027c98-0x60027c38= 96

Both offsets are not in the offset table…

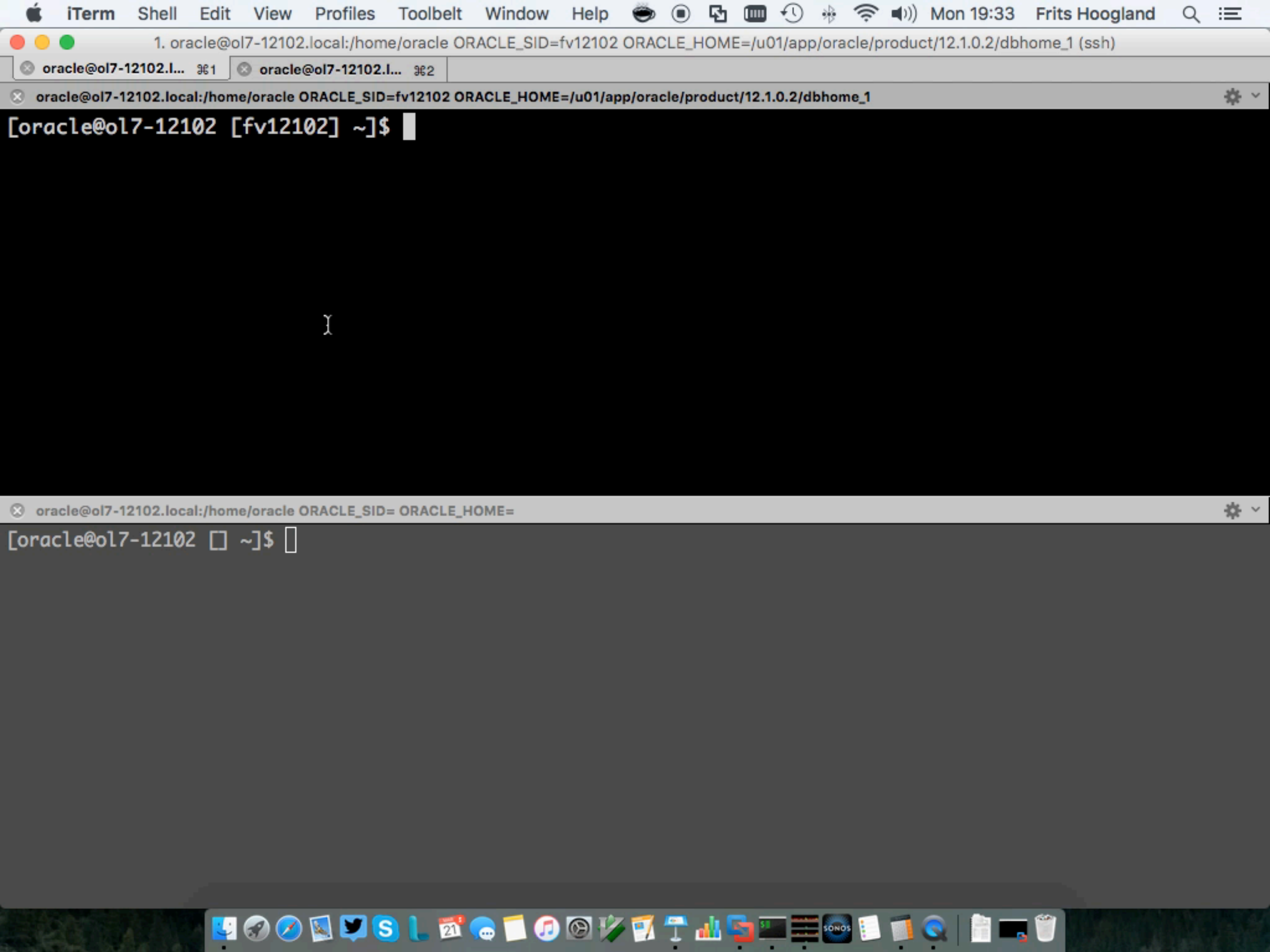
X$KCRFWS

- Then it needs a more “hardcore” approach…

- For this a **watchpoint** can be used too.
  - A watchpoint breaks execution if the specified address is read, written or both.

X$KCRFWS

- Now for the trick to find the field that belong to 0x60027c68 and 0x60027c98:
  - Put a *read* watchpoint on the addresses.
  - Query X$KCRFWS field by field until it hits the watchpoint.



1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

```
[oracle@ol7-12102 [fv12102] ~]$ 
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

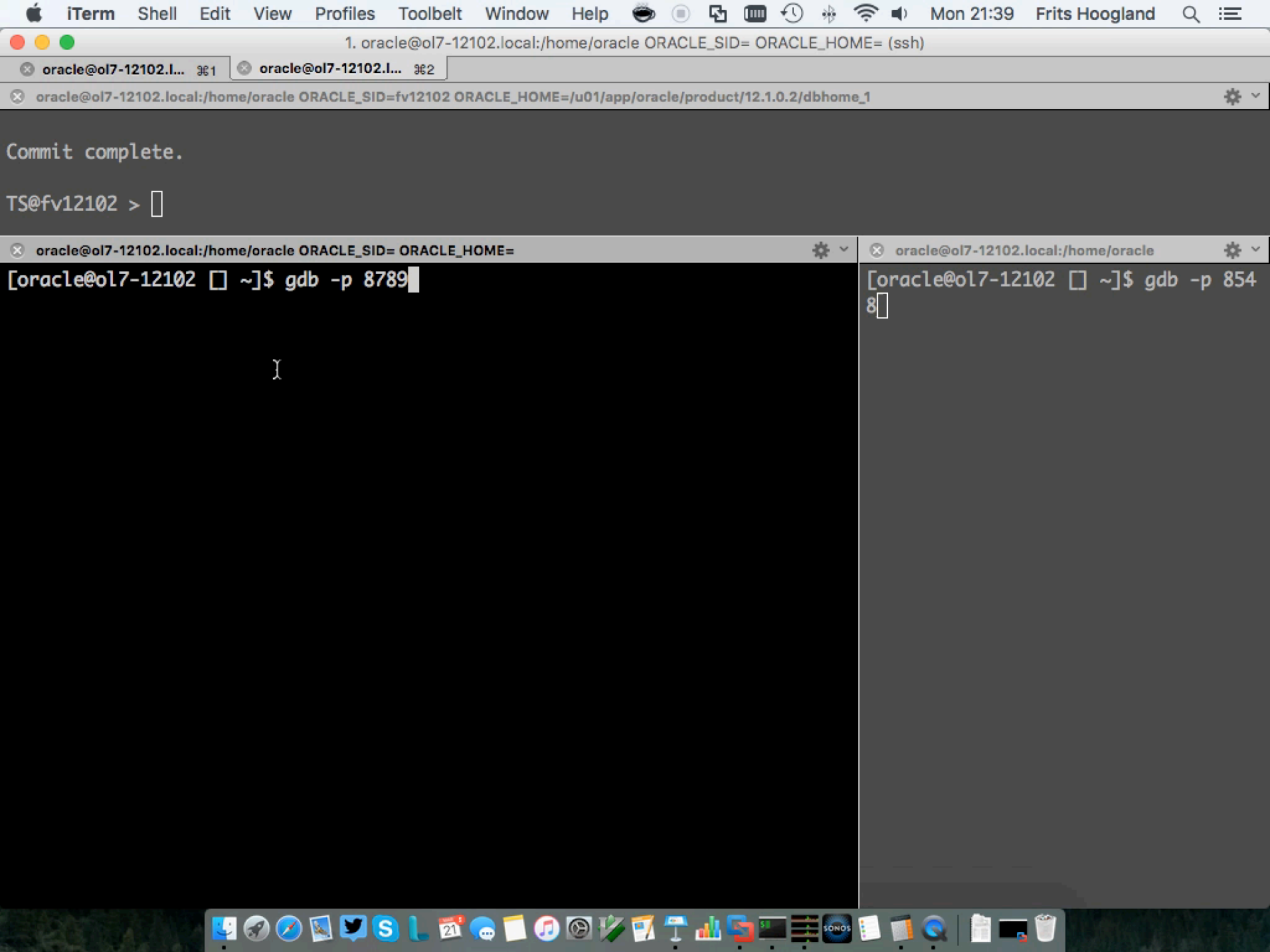
```
[oracle@ol7-12102 □ ~]$ 
```

X$KCRFWS

- Address 0x60027c68:
  - LWN_SCN_BAS and LWN_SCN_WRP
    - LWN: Log Write Number; a group of redo blocks to be written by the LGWR is appointed a number called LWN.
    - LWN SCN: The potential maximum SCN in the current LWN.
- Address 0x60027c98:
  - ON_DISK_SCN_BAS and ON_DISK_SCN_WRP
    - On disk SCN: the highest SCN that the database can be recovered to with written redo.

kcscur3

- Back to the original investigation.
  - The first argument of kcscur3 is actually a variety of SCN numbers:
    - 0x60027c68: LWN SCN
    - 0x60027c98: On disk SCN
    - 0x6001fbb0: global (current) SCN
  - So: the function kcrf_commit_force(_int) checks different SCN values using kcscur3 during commit.
- Let's look at what is happening during commit again:

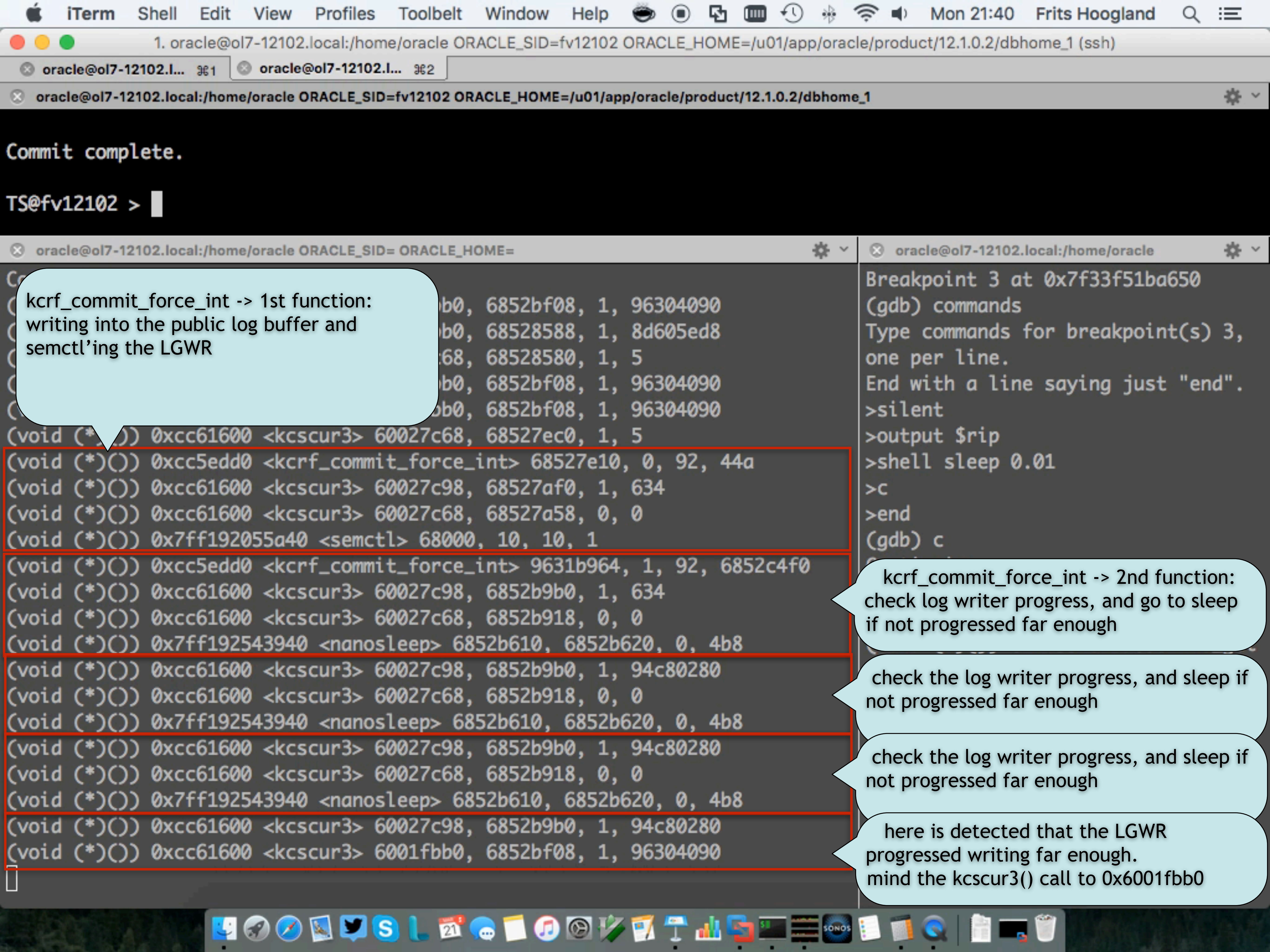


iTerm
Shell
Edit
View
Profiles
Toolbelt
Window
Help
Mon 21:39
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=
oracle@ol7-12102.local:/home/oracle

```
Commit complete.

TS@fv12102 >
```

```
[oracle@ol7-12102 ~]$ gdb -p 8789
```

```
[oracle@ol7-12102 ~]$ gdb -p 854
8
```



1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1

```
Commit complete.

TS@fv12102 >
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
Co
(                                          b0, 6852bf08, 1, 96304090
(                                          b0, 68528588, 1, 8d605ed8
(                                          68, 68528580, 1, 5
(                                          b0, 6852bf08, 1, 96304090
(                                          b0, 6852bf08, 1, 96304090
(void (*  )) 0xcc61600 <kcscur3> 60027c68, 68527ec0, 1, 5
(void (*)()) 0xcc5edd0 <kcrf_commit_force_int> 68527e10, 0, 92, 44a
(void (*)()) 0xcc61600 <kcscur3> 60027c98, 68527af0, 1, 634
(void (*)()) 0xcc61600 <kcscur3> 60027c68, 68527a58, 0, 0
(void (*)()) 0x7ff192055a40 <semctl> 68000, 10, 10, 1
(void (*)()) 0xcc5edd0 <kcrf_commit_force_int> 9631b964, 1, 92, 6852c4f0
(void (*)()) 0xcc61600 <kcscur3> 60027c98, 6852b9b0, 1, 634
(void (*)()) 0xcc61600 <kcscur3> 60027c68, 6852b918, 0, 0
(void (*)()) 0x7ff192543940 <nanosleep> 6852b610, 6852b620, 0, 4b8
(void (*)()) 0xcc61600 <kcscur3> 60027c98, 6852b9b0, 1, 94c80280
(void (*)()) 0xcc61600 <kcscur3> 60027c68, 6852b918, 0, 0
(void (*)()) 0x7ff192543940 <nanosleep> 6852b610, 6852b620, 0, 4b8
(void (*)()) 0xcc61600 <kcscur3> 60027c98, 6852b9b0, 1, 94c80280
(void (*)()) 0xcc61600 <kcscur3> 60027c68, 6852b918, 0, 0
(void (*)()) 0x7ff192543940 <nanosleep> 6852b610, 6852b620, 0, 4b8
(void (*)()) 0xcc61600 <kcscur3> 60027c98, 6852b9b0, 1, 94c80280
(void (*)()) 0xcc61600 <kcscur3> 6001fbb0, 6852bf08, 1, 96304090
```

oracle@ol7-12102.local:/home/oracle

```
Breakpoint 3 at 0x7f33f51ba650
(gdb) commands
Type commands for breakpoint(s) 3, one per line.
End with a line saying just "end".
>silent
>output $rip
>shell sleep 0.01
>c
>end
(gdb) c
```

kcrf_commit_force_int -> 1st function: writing into the public log buffer and semctl'ing the LGWR

kcrf_commit_force_int -> 2nd function: check log writer progress, and go to sleep if not progressed far enough

check the log writer progress, and sleep if not progressed far enough

check the log writer progress, and sleep if not progressed far enough

here is detected that the LGWR progressed writing far enough.
mind the kcscur3() call to 0x6001fbb0

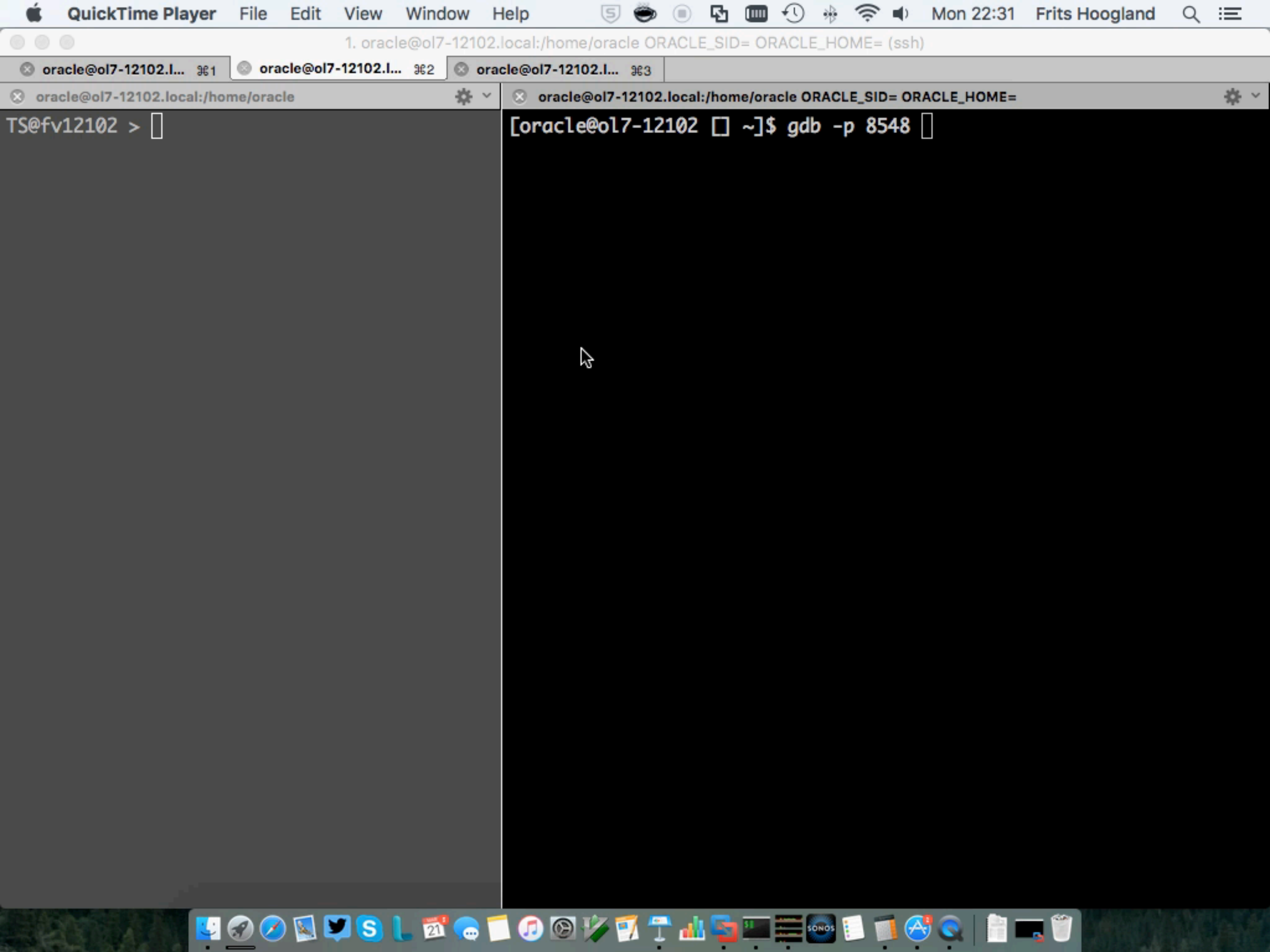
Yet another step…

- So, we now know that the committing process checks the on-disk and LWN SCN.
  - I think the on-disk SCN is used by the foreground process to check for redo write progress in polling mode (and post/wait in certain cases).
- Obviously, another process must change the on-disk SCN.
  - That process is quite likely the logwriter.
  - Or the LGWR slaves, which I disabled for the sake of clarity.

Yet another step…

- So, I suspect the log writer:
  - Writes the log buffer.
  - Then updates the on-disk SCN to indicate write progress.

- To understand what the LGWR does we can:
  - Put a read/write watchpoint on
    - 0x60027c68 (LWN SCN)
    - ox60027c98 (on disk SCN)
  - To see what the LGWR is doing.



QuickTime Player
File
Edit
View
Window
Help
Mon 22:31
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.l... ⌘3
oracle@ol7-12102.local:/home/oracle
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
TS@fv12102 >
```

```
[oracle@ol7-12102 □ ~]$ gdb -p 8548
```

```
watch *0x60027c68
watch *0x60027c98
commands 1-2
c
end
break kcsnew3
break kcscur3
break kcsadj3
break io_submit
break io_getevents_0_4
commands 3-7
silent
output $rip
printf “\t%x, %x, %x, %x\n”, $rdi, $rsi, $rdx, $rcx
c
end
break semtimedop
silent
printf “semtimedop\n”
c
end
```

- Log writer wake-up functions without write:

```
(void (*)()) 0xcc61670 <kcsnew3> 6001fbb0, a9060900, 60027c68, 3b37e000
Hardware watchpoint 2: *0x60027c68

Old value = 780329
New value = 780330
0x00000000cc61754 in kcsnew3 ()
(void (*)()) 0xcc61600 <kcscur3> 60027c98, a906073c, 1, 0
(void (*)()) 0xcc61600 <kcscur3> 60027c68, a9060708, 1, 0
(void (*)()) 0x2d5aed0 <kcsadj3> 60027c98, a9060708, 0, 0
Hardware watchpoint 1: *0x60027c98

Old value = 780329
New value = 780330
0x000000002d5af53 in kcsadj3 ()
(void (*)()) 0xcc61600 <kcscur3> 6001fbb0, a9060270, 1, 79291280
```

Every wake-up of the log writer increases the LWN SCN using the kcsnew3 function.

Also, the on-disk SCN is increased.

Please mind there is nothing written in the online redo log!

- Log writer wake-up functions with write:

```
(void (*)()) 0xcc61670 <kcsnew3> 6001fbb0, a9060aa0, 60027c68, 3b37e000
Hardware watchpoint 2: *0x60027c68

Old value = 780330
New value = 780350
0x00000000cc61754 in kcsnew3 ()
(void (*)()) 0x2d5aed0 <kcsadj3> 60027d50, a  60860, 0, 0
(void (*)()) 0x7f4a75e0d690 <io_submit> 7929f00   1, a9059360, 8f24bbe8
(void (*)()) 0x7f4a75e0d650 <io_getevents> 7929f   0, 1, 80, a905f1e8
(void (*)()) 0xcc61600 <kcscur3> 60027c98, a906083   1, 0
(void (*)()) 0x2d5aed0 <kcsadj3> 60027c98, 92fac724,   060834, a906082c
Hardware watchpoint 1: *0x60027c98

Old value = 780330
New value = 780350
0x0000000002d5af53 in kcsadj3 ()
(void (*)()) 0xcc61600 <kcscur3> 6001fbb0, a9060270, 1, 79291280
```

Look! There is another kcsadj3 call. This call has 0x60027d50 as first argument.

0x60027d50 is REAL_REDO_SCN_(BAS|WRP)

The LWN and on-disk SCNs are increased when the log writer writes too, as expected.

Probably not all SCNs need writing.

- So, what I think is happening is:
  - A FG session commits and notes commit SCN.
  - FG semctl's LGWR to write*.
  - Then checks on-disk SCN if SCN increased beyond its commit SCN, then nanosleep().
  - LGWR determines LWN SCN.
  - Writes blocks in the LWN batch.
  - Updates on-disk SCN.
  - FG reads updated on-disk SCN and continues.

- How to validate a FG just checks the on-disk SCN?

```
(void (*)()) 0xcc61670 <kcsnew3> 6001fbb0, a9060aa0, 60027c68, 3b37e000
Hardware watchpoint 2: *0x60027c68

Old value = 780330
New value = 780350
0x000000000cc61754 in kcsnew3 ()
             csadj3> 60027d50, a9060860, 0
             90 <io_submit> 7929f000, 1, a9
               <io_getevents> 792
             cscur3> 60027c98, a906083c, 1,
(void (*)()) 0x2d5aed0 <kcsadj3> 60027c98, 92fac724, a90
Hardware watchpoint 1: *0x60027c98

Old value = 780330
New value = 780350
0x000000002d5af53 in kcsadj3 ()
(void (*)()) 0xcc61600 <kcscur3> 6001fbb0, a9060270, 1, 79291280
```

So what if we make the LGWR stop here?

The FG should keep on waiting.

While the LGWR is suspended at this point, the FG waits.

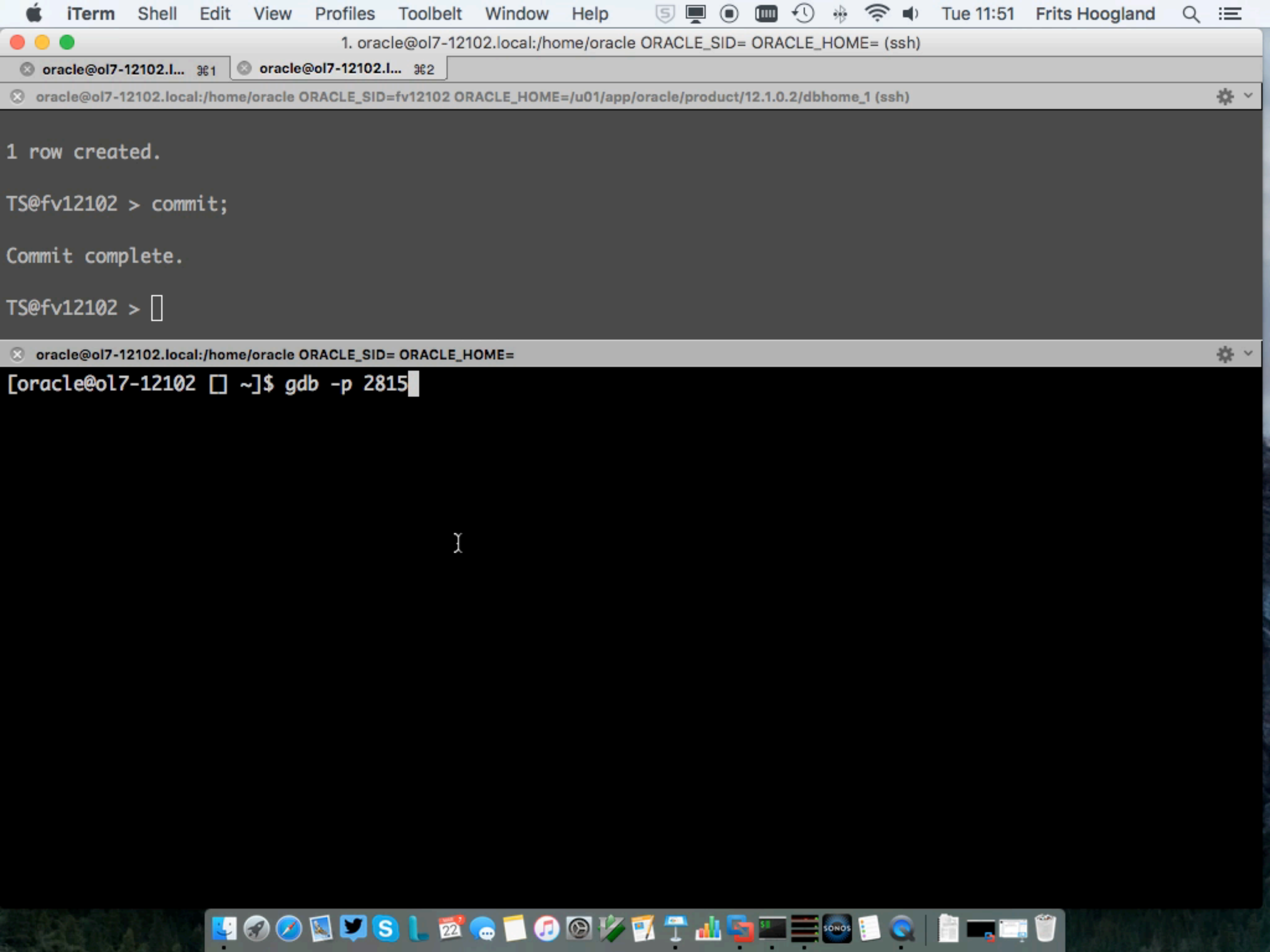
We can prove the FG only waits for the on-disk SCN, if the FG continues when we:
- Keep the LGWR suspended, but
- MANUALLY update the on-disk SCN

At this point the on-disk SCN is updated. This supposedly makes the FG to continue if the SCN is high enough.

WARNING

- The following techniques are for experimenting and investigation ONLY.

- **Doing this on a real, live database could cause corruption or loss of the entire database!**



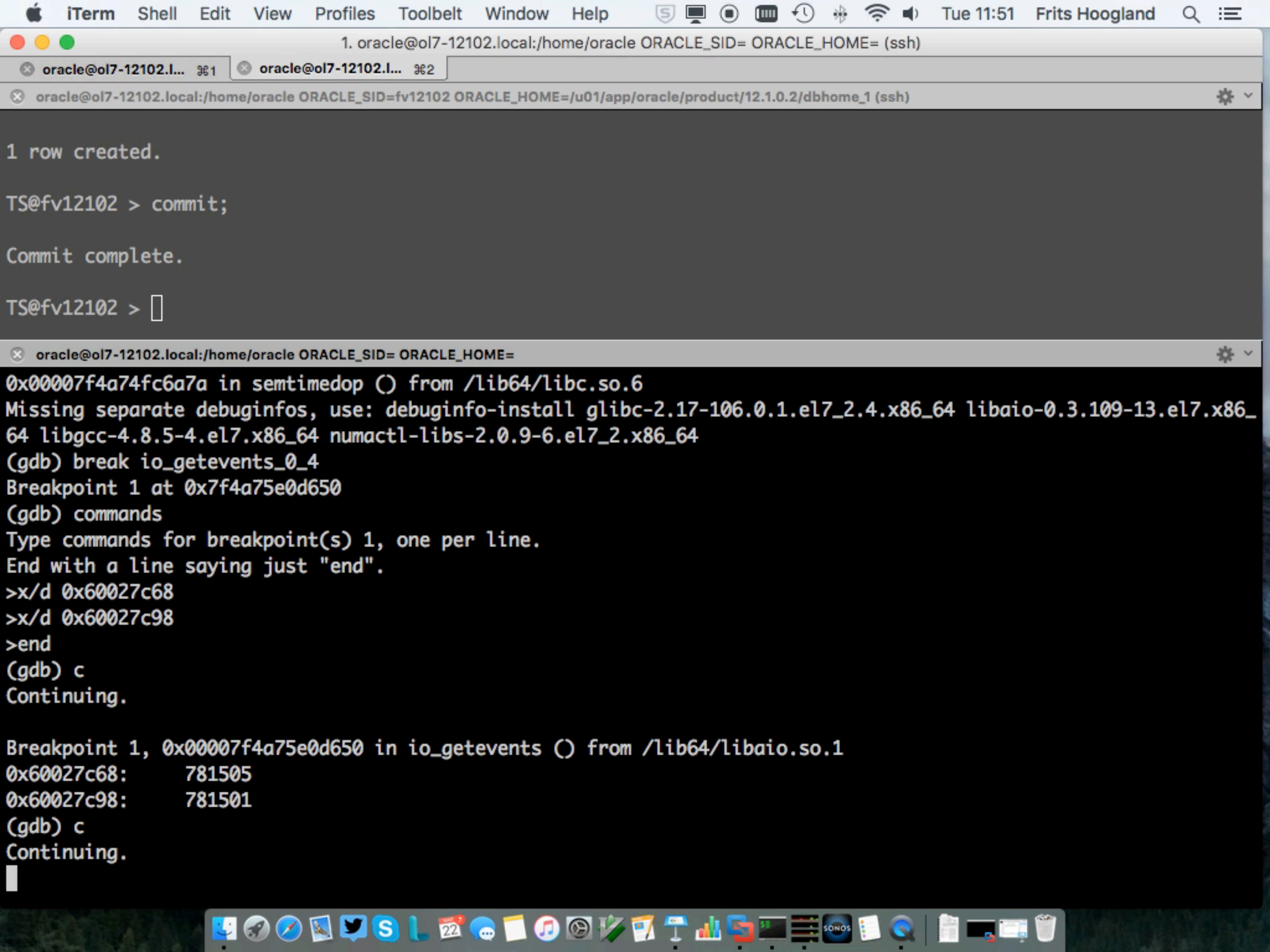
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)

oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

```
1 row created.

TS@fv12102 > commit;

Commit complete.

TS@fv12102 >
```

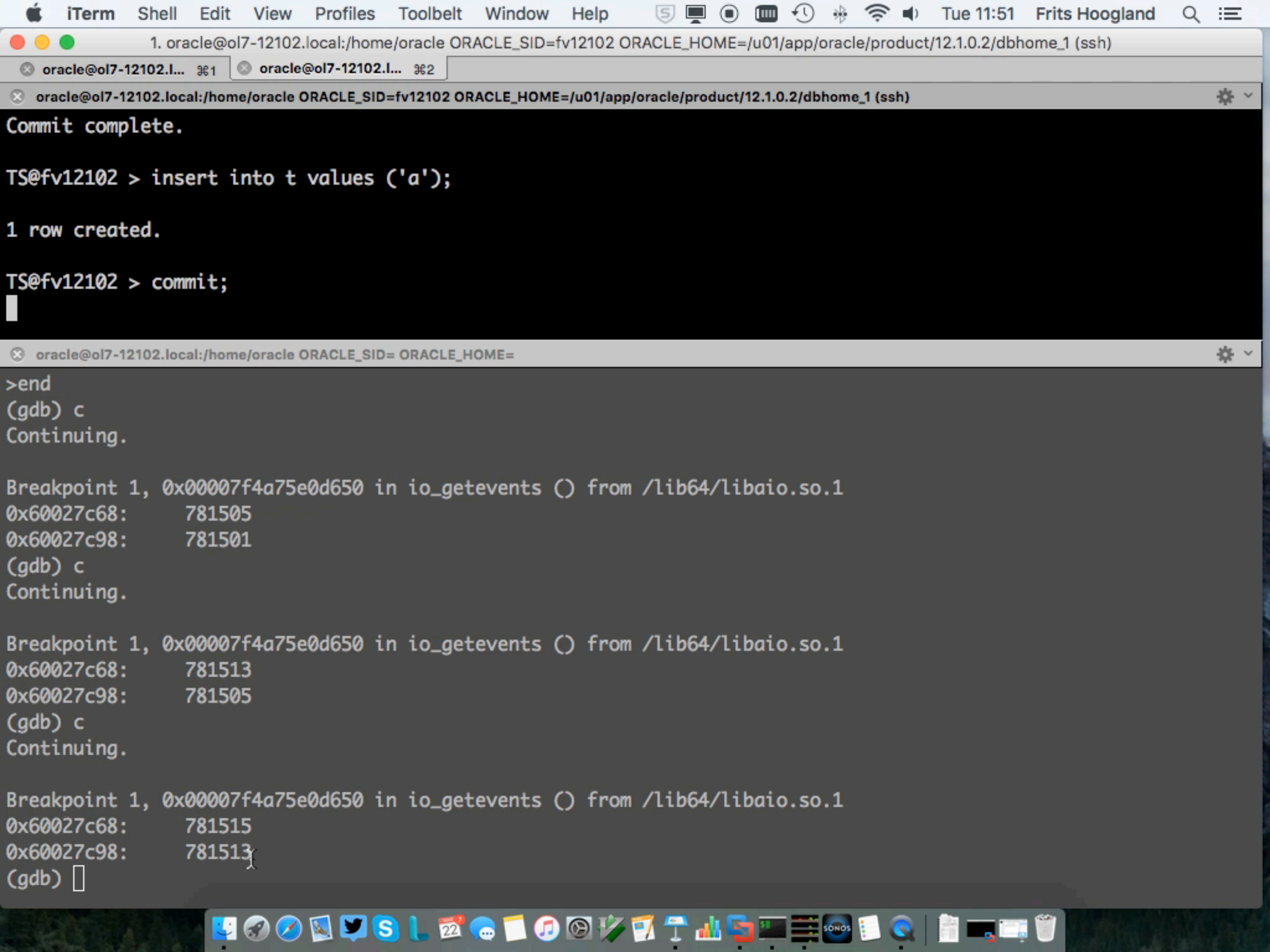
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
[oracle@ol7-12102 □ ~]$ gdb -p 2815
```


iTerm Shell Edit View Profiles Toolbelt Window Help
Tue 11:51 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
1 row created.

TS@fv12102 > commit;

Commit complete.

TS@fv12102 > 
```

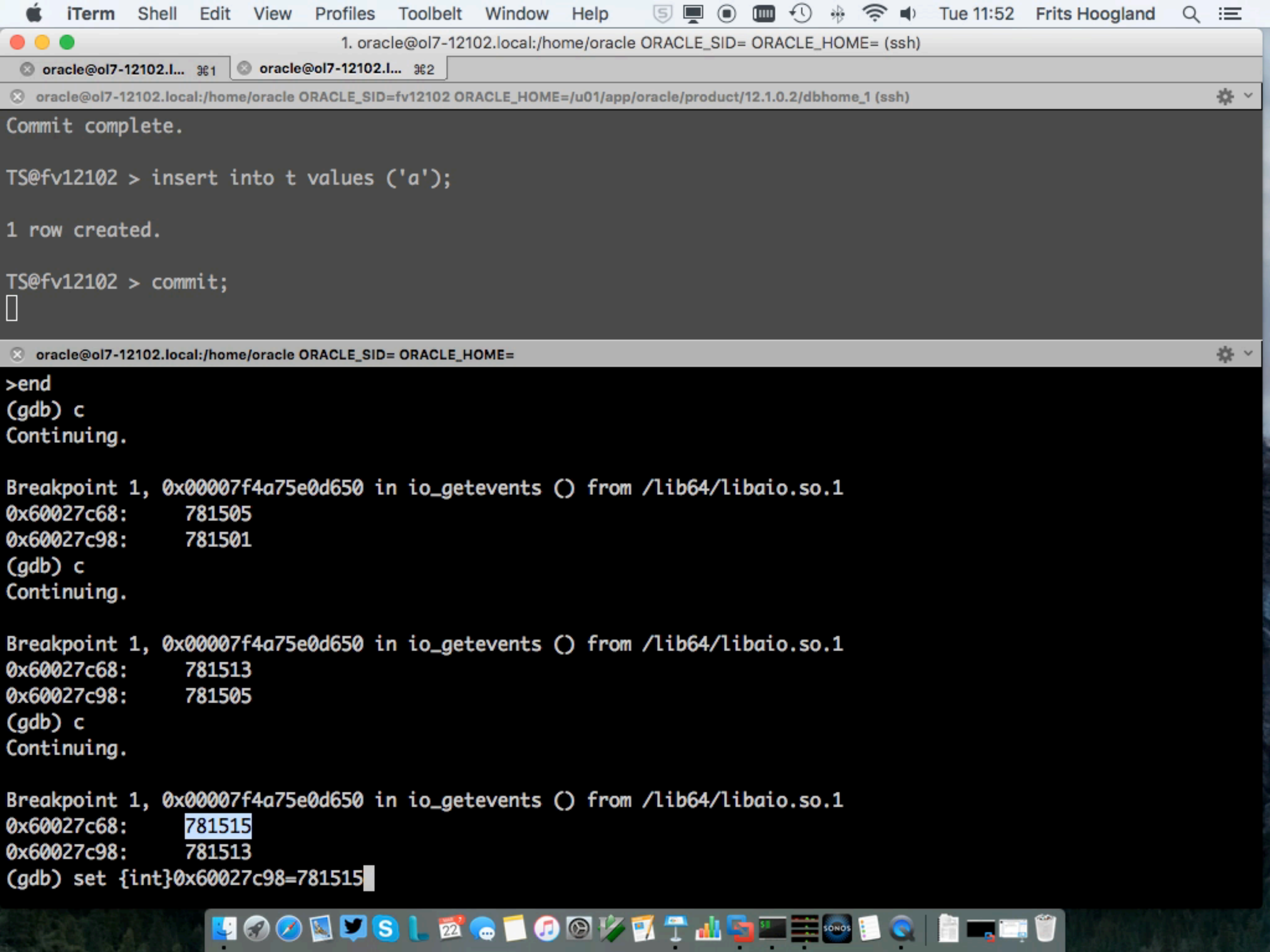
```
0x00007f4a74fc6a7a in semtimedop () from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install glibc-2.17-106.0.1.el7_2.4.x86_64 libaio-0.3.109-13.el7.x86_
64 libgcc-4.8.5-4.el7.x86_64 numactl-libs-2.0.9-6.el7_2.x86_64
(gdb) break io_getevents_0_4
Breakpoint 1 at 0x7f4a75e0d650
(gdb) commands
Type commands for breakpoint(s) 1, one per line.
End with a line saying just "end".
>x/d 0x60027c68
>x/d 0x60027c98
>end
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781505
0x60027c98:     781501
(gdb) c
Continuing.
```


iTerm
Shell
Edit
View
Profiles
Toolbelt
Window
Help
Tue 11:51
Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

```
Commit complete.

TS@fv12102 > insert into t values ('a');

1 row created.

TS@fv12102 > commit;
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
>end
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781505
0x60027c98:     781501
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781513
0x60027c98:     781505
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781515
0x60027c98:     781513
(gdb) 
```


iTerm Shell Edit View Profiles Toolbelt Window Help
Tue 11:52 Frits Hoogland
1. oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME= (ssh)
oracle@ol7-12102.l... ⌘1
oracle@ol7-12102.l... ⌘2
oracle@ol7-12102.local:/home/oracle ORACLE_SID=fv12102 ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1 (ssh)

```
Commit complete.

TS@fv12102 > insert into t values ('a');

1 row created.

TS@fv12102 > commit;
```

oracle@ol7-12102.local:/home/oracle ORACLE_SID= ORACLE_HOME=

```
>end
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781505
0x60027c98:     781501
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781513
0x60027c98:     781505
(gdb) c
Continuing.

Breakpoint 1, 0x00007f4a75e0d650 in io_getevents () from /lib64/libaio.so.1
0x60027c68:     781515
0x60027c98:     781513
(gdb) set {int}0x60027c98=781515
```

Conclusion

- A foreground session has two commit modes:
  - Post/wait, the traditional way.
  - Polling, the new method.

- A foreground session notifies the LGWR by executing a ‘semctl’ call.
  - No notification necessary if LGWR already progressed beyond FG commit SCN.

Conclusion

- There are a couple of SCN values the database keeps in the fixed SGA:
  - kcbgscn, global/current SCN, 0x6001fbb0
  - kcrfsg, LWN SCN, 0x60027c68
  - kcrfsg, on-disk SCN, 0x60027c98
  - kcrfsg, real redo SCN, 0x60027d50

- The kcscur3 function seems to be the function to read these variables.

Conclusion

- The foreground process uses the kcrf_commit_force(_int) function to:
  - Flush its redo data into the public logbuffer.
  - Check log writer progress via the on-disk SCN.

- This of course is in polling mode.
  - With post/wait, the on-disk SCN is checked too!

Conclusion

- The log writer has a certain cycle every 3 sec:
  - Read current SCN and LWN SCN.
  - Update LWN SCN.
  - If needed: update real redo SCN and write out public log buffer.
  - Update on-disk SCN.

- It seems the SCN set as LWN SCN at the beginning of the cycle, is equal to the on-disk SCN.

Conclusion

- LWN and on-disk SCNs progress even if there is no redo written from log buffer to disk.
- The SCN of the latest redo truly written to disk is in the real redo SCN.