Project Report-- ##Transaction Manager## (Assignment2)

Division of Labor:

Team 31

· Sakshi(sx3702@mavs.uta.edu, 1001993702) (implementation of read, write, omega run)
· Spoorthi Seri(sxs2684@mavs.uta.edu,1002032680) (documentation)
· Neha Thokala(nxt0631@mavs.uta.edu, 1002030631) (implementation of commit, abort)

Overall Status

- Initially, I followed the steps provided to compile and build the code locally. To gain a better understanding of C and C++ programming language we referenced the FAQ section provided by Stroustrop (https://www.stroustrup.com/bs_faq2.html) and other sources online.
- After gaining a better understanding of how the main function runs, we implemented readtx(),writetx(), CommitTx(), and AbortTx() in TxMgr and TxtTx code files.
- Test files were given by sir, and we added new test scenarios to test edge cases.
- After testing locally, copied source files to the UTA omega server.
- The code was compiled and built successfully.
- Test scenarios generated the expected results as log files

Login at omega and create a directory.

```
-mkdir tmproject2
```

After making tmproject2 folder in omega, copy files by running this command in a local terminal

```
- pscp -r /Users/sakshisrivastava/Desktop/uta/Sem3Spring/DBMS/project2/4331-5331_Proj2Spring23_team_31/
sx3702@omega.uta.edu:/home/s/sx/sx3702/tmproject2/
```

SSH to omega and verify.

```
-cd /home/s/sx/sx3702/tmproject2/src
- vi zgt_semaphore.C    // uncomment union semun ZGT_arg
- chmod +x zgt_test
- make clean && make                # compiles the code
- ./zgt_test ../more-test-files/Scenario1.txt        # run test cases.
- vi Scenario5.log
- vi ../more-test-files/Scenario5.txt
```

Where you encountered difficulty

- When building the code locally we faced an error due to the redefinition of semaphore - zgt_semaphore.C:25:7: error: redefinition of 'semun'. The variable was already defined in the <sys/sem.h> header file which caused an error.
- It was difficult to jump into a low-level programming language like C and most of the time was spent on understanding the syntax and semantics. It would help if students were given some references for C to get started with the language. But it was a great learning exercise for us personally.

Overview and file descriptions:

In this project, we were asked to implement one of the lower-level components of DBMS – the Transaction Manager which will be called by higher layers to handle concurrency control. For this purpose, we have implemented Strict Two-Phase Locking (S2PL) protocol where an Exclusive lock is assigned for Write operations and a Shared lock is assigned for Read operations. Hence the Transaction Manager handles Locking and Releasing objects as and when necessary.

The Transaction Manager has been implemented using zgt_tx.c and zgt_tm.c programs.

Transaction manager (zgt_tm.c) manages transactions (zgt_tx) in a multi-threaded environment using the Pthreads library and ensures proper synchronization using mutexes and condition variables.
The code is written in C++ and includes several functions for creating and processing transactions, performing read and write operations, and handling committing or aborting transactions as needed.

Data Structures:

struct param: This structure contains the information required to execute a transaction operation, such as the transaction ID (tid), object number (obno), transaction type (Txtype), and sequence number (count).

<u>Transaction Manager (zgt_tm.c) Functions:</u>

- TxRead: Creates a new thread to perform a read operation in a transaction by initializing the param structure and calling the readtx function within the newly created thread.

- TxWrite: Creates a new thread to perform a write operation in a transaction by initializing the param structure and calling the writetx function within the newly created thread.

- CommitTx: Creates a new thread to commit a transaction by initializing the param structure and calling the committx function within the newly created thread.

- AbortTx: Creates a new thread to abort a transaction by initializing the param structure and calling the aborttx function within the newly created thread.

Transaction (zgt_tx.c) Functions:

- readtx: Receives a struct param as an argument, containing the transaction ID, object number, and count. Calls the process_read_write() function with the transaction ID, object number, count, and a Shared lock mode ('S').

- writetx: Like readtx(), receive a struct param as an argument. Calls the process_read_write() function with the transaction ID, object number, count, and an Exclusive lock mode ('X').

- process_read_write():
  - Starts the operation with the start_operation() function.
  - Sets the lock on the object as an exclusive lock considering the necessary conditions by calling the set_lock() function.
  - Finishes the operation with the finish_operation() function.
  - Exits the thread with pthread_exit(NULL).

- aborttx():
  - Receives a struct param as an argument, containing the transaction ID and count.
  - Starts the operation with the start_operation() function.
  - Calls the do_commit_abort() function for aborting the transaction with status 'A'.
  - Finishes the operation with the finish_operation() function.
  - Exits the thread with pthread_exit(NULL).

- committx():

- Receives a struct param as an argument, containing the transaction ID and count.
- Starts the operation with the start_operation() function.
- Calls the do_commit_abort() function for committing the transaction with status 'C'.
- Finishes the operation with the finish_operation() function.
- Exits the thread with pthread_exit(NULL).

- do_commit_abort():
  - Writes a log record for the transaction (commit or abort).
  - Retrieves the transaction using the transaction ID.
  - If the transaction exists, it releases the locks held by the transaction, and either end or removes the transaction based on the status.
  - If there are transactions waiting on the semaphore, it releases them using zgt_v().

- remove_tx():
  - Removes the transaction from the transaction manager.
  - Scans through the list of transactions and updates the nextr value accordingly.
  - If the transaction does not exist, log an error message.

- set_lock():
  - Sets a lock on an object with a specific lock mode (Shared or Exclusive) for a transaction.
  - Considers conditions such as current transaction ownership, lock modes, and waiting transactions.
  - If successful, returns 0, otherwise, return -1.

Main Difference between commit and abort:

- committx finalizes and saves the changes made during a transaction, making them permanent.
- aborttx discards the changes made during a transaction and restores the system to its state before the transaction started.

Logical errors and how we handled them:

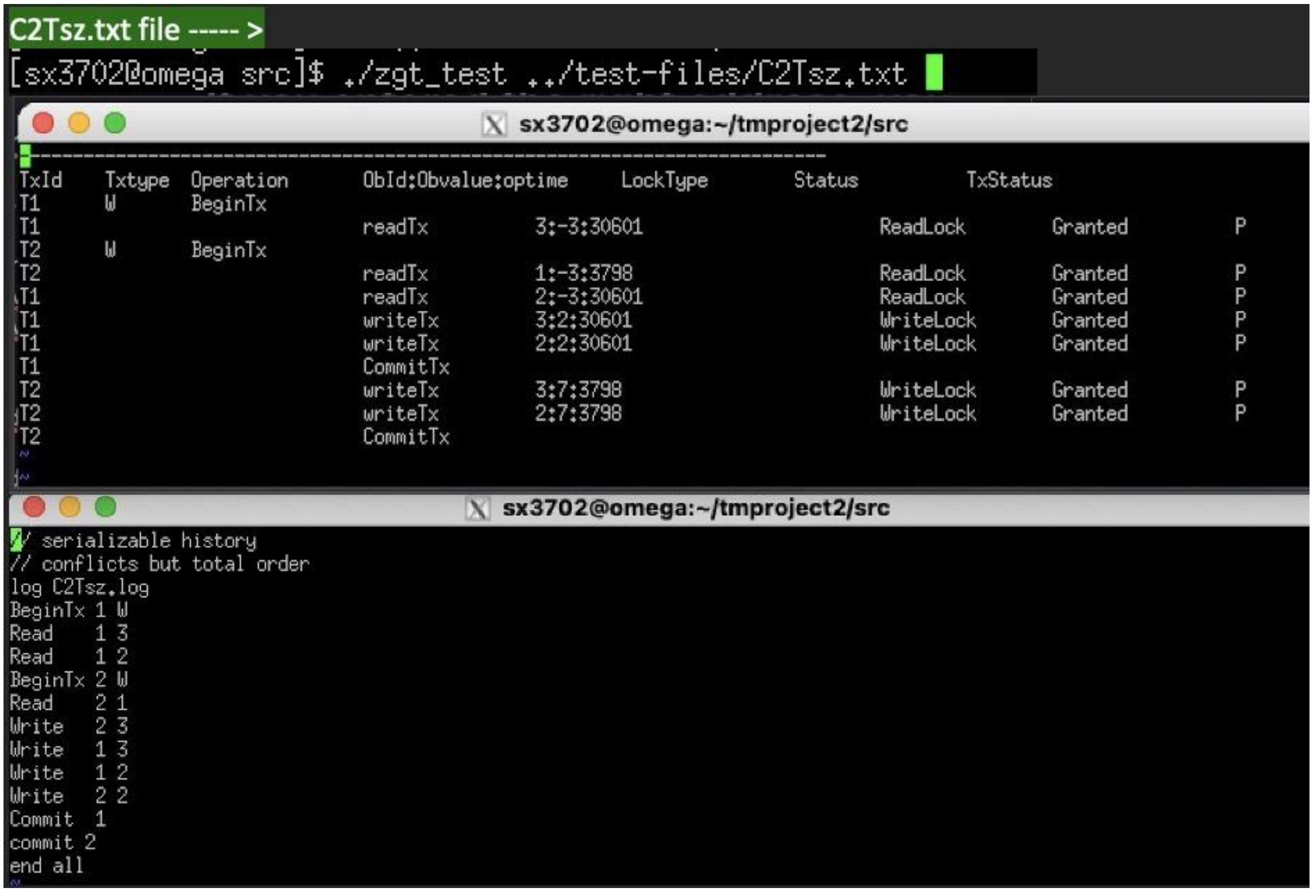
- Deadlock prevention: When a deadlock scenario occurs when two threads try to read/write the same object, the resolution comes by using the semaphore mechanism, which ensures that a transaction waits for a lock only if it can obtain it, preventing deadlocks from occurring.
- Incorrect or non-existent transaction IDs: To avoid a bad (illogical) test scenario, The code checks whether a transaction exists before performing any operation on it (e.g., in the do_commit_abort() function). If a transaction does not exist, it logs an error message and does not proceed with the operation.
- Ensuring lock acquisition: The code ensures that a transaction acquires the appropriate lock (shared or exclusive) before performing read or write operations. This is done in the zgt_tx:set_lock() function, where the lock is acquired if it does not already exist, or if the current transaction already holds the lock.
- Maintaining transaction states: For easier debugging and ease of management, the code keeps track of the state of a transaction, such as active, waiting, or committed. This ensures that operations are performed only when the transaction is in the correct state.
- Proper clean-up after commit or abort: The code makes sure that the locks are released, and the transaction is removed from the transaction manager after a commit or abort operation. This is done in the zgt_tx:free_locks(), zgt_tx:end_tx(), and zgt_tx:remove_tx() functions.
- Ensuring transaction order: The code prevents two operations of the same transaction from following one another by using a sequence number (SEQNUM[tid]). This ensures that the correct operation order is maintained for each transaction.
- Local build errors: Faced some initial local build errors which were resolved after using guards (**reference**). Eventually, we removed them since the error did not show up in the OMEGA server and we were not allowed to touch header files.

Future Improvements

- Deadlock detection and resolution: Enhance the system's deadlock detection and resolution capabilities. Consider implementing a more advanced algorithm, such as the Wait-for-Graph (WFG) or the edge-chasing algorithm, to efficiently detect and resolve deadlocks in the system.
- Distributed transaction support: Extend the system to support distributed transactions across multiple nodes or clusters. Implement a distributed transaction coordinator to manage transactions that span multiple nodes and ensure atomicity, consistency, isolation, and durability (ACID) properties are maintained.
- Object versioning: Introduce object versioning to enable Multi-Version Concurrency Control (MVCC). This can help reduce contention by allowing transactions to work with different versions of objects without needing to lock them.
- Enhanced logging and recovery: Improve the logging and recovery mechanisms by implementing techniques like Write-Ahead Logging (WAL) or the ARIES recovery algorithm. These can help ensure faster and more efficient recovery after a system crash or failure.
- Monitoring and diagnostics: Develop a comprehensive monitoring and diagnostics framework that provides insights into the system's performance, resource usage, and transaction statistics. This can help identify potential bottlenecks, optimize performance, and aid in troubleshooting issues.
- Transaction prioritization: Implement transaction prioritization, allowing certain transactions to be prioritized over others based on criteria like age, resource requirements, or user-defined priorities. This can help improve the responsiveness of the system for critical transactions and reduce the impact of long-running or resource-intensive transactions.
- Security enhancements: Integrate security features such as authentication, authorization, and encryption to protect sensitive data and ensure that only authorized users can perform transactions. Implementing role-based access control (RBAC) can also help manage user permissions more effectively.

Test on omega Screenshots:

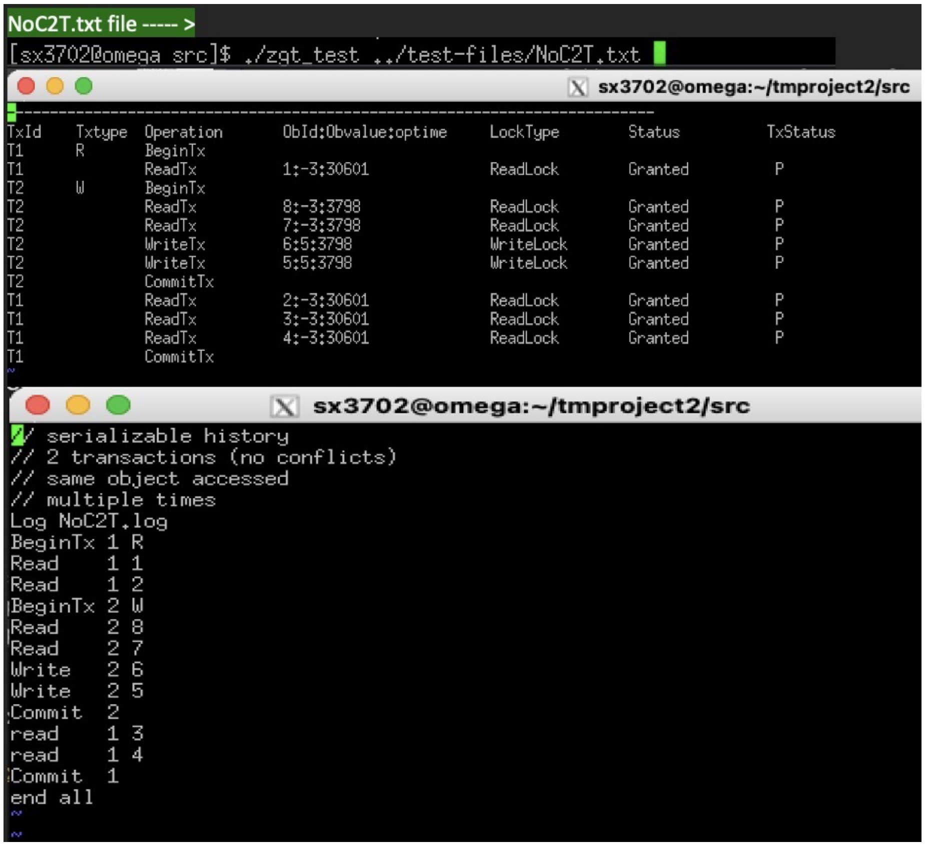
Sirtestfiles-given. --



```
[sx3702@omega src]$ ./zgt_test ../test-files/C2Tsz.txt
```

```
---------------------------------------------------------------------------------
TxId    Txtype  Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1      W       BeginTx
T1                      readTx          3:-3:30601                      ReadLock        Granted         P
T2      W       BeginTx
T2                      readTx          1:-3:3798                       ReadLock        Granted         P
T1                      readTx          2:-3:30601                      ReadLock        Granted         P
T1                      writeTx         3:2:30601                       WriteLock       Granted         P
T1                      writeTx         2:2:30601                       WriteLock       Granted         P
T1                      CommitTx
T2                      writeTx         3:7:3798                        WriteLock       Granted         P
T2                      writeTx         2:7:3798                        WriteLock       Granted         P
T2                      CommitTx
~
~
```

```
// serializable history
// conflicts but total order
log C2Tsz.log
BeginTx 1 W
Read    1 3
Read    1 2
BeginTx 2 W
Read    2 1
Write   2 3
Write   1 3
Write   1 2
Write   2 2
Commit  1
commit 2
end all
```

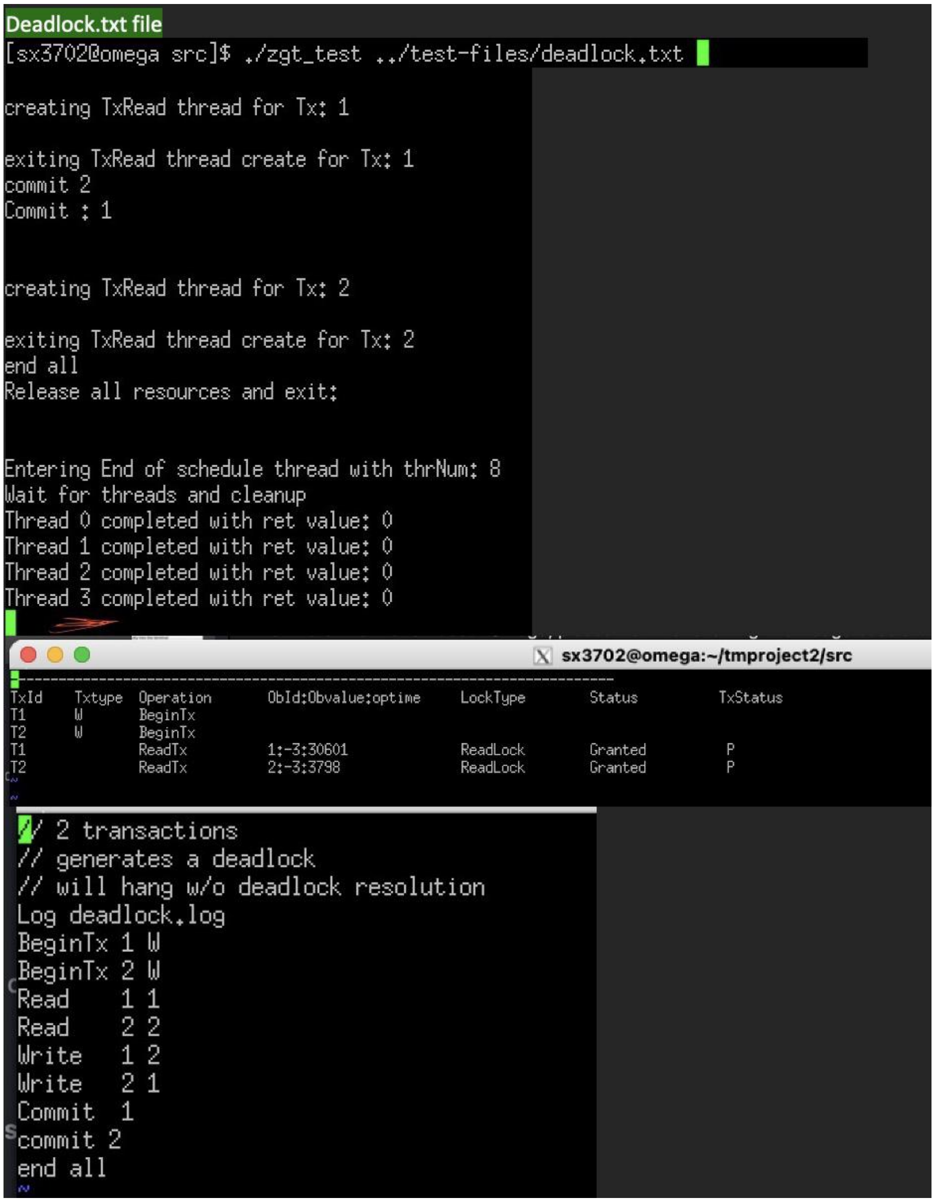


NoC2T.txt file ----- >

```
[sx3702@omega src]$ ./zgt_test ../test-files/NoC2T.txt
```

```
----------------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime      LockType         Status           TxStatus
T1       R        BeginTx
T1                ReadTx           1:-3:30601               ReadLock         Granted           P
T2       W        BeginTx
T2                ReadTx           8:-3:3798                ReadLock         Granted           P
T2                ReadTx           7:-3:3798                ReadLock         Granted           P
T2                WriteTx          6:5:3798                 WriteLock        Granted           P
T2                WriteTx          5:5:3798                 WriteLock        Granted           P
T2                CommitTx
T1                ReadTx           2:-3:30601               ReadLock         Granted           P
T1                ReadTx           3:-3:30601               ReadLock         Granted           P
T1                ReadTx           4:-3:30601               ReadLock         Granted           P
T1                CommitTx
```

```
// serializable history
// 2 transactions (no conflicts)
// same object accessed
// multiple times
Log NoC2T.log
BeginTx 1 R
Read    1 1
Read    1 2
BeginTx 2 W
Read    2 8
Read    2 7
Write   2 6
Write   2 5
Commit  2
read    1 3
read    1 4
Commit  1
end all
```



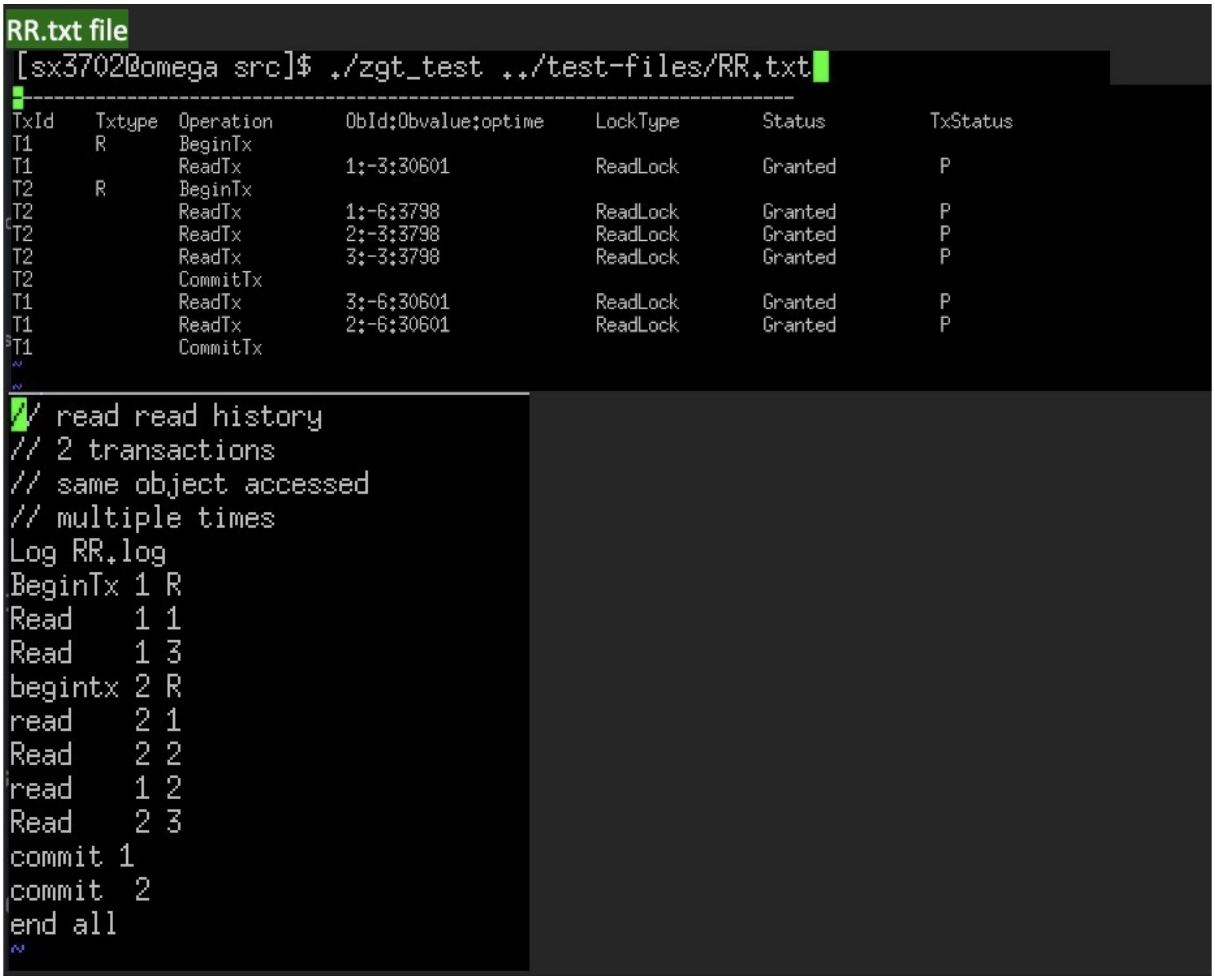
Deadlock.txt file

```
[sx3702@omega src]$ ./zgt_test ../test-files/deadlock.txt

creating TxRead thread for Tx: 1

exiting TxRead thread create for Tx: 1
commit 2
Commit : 1


creating TxRead thread for Tx: 2

exiting TxRead thread create for Tx: 2
end all
Release all resources and exit:


Entering End of schedule thread with thrNum: 8
Wait for threads and cleanup
Thread 0 completed with ret value: 0
Thread 1 completed with ret value: 0
Thread 2 completed with ret value: 0
Thread 3 completed with ret value: 0
```

sx3702@omega:~/tmproject2/src

```
-----------------------------------------------------------------------------------------
TxId    Txtype  Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1      W       BeginTx
T2      W       BeginTx
T1              ReadTx          1:-3:30601              ReadLock        Granted          P
T2              ReadTx          2:-3:3798               ReadLock        Granted          P
~
~
```

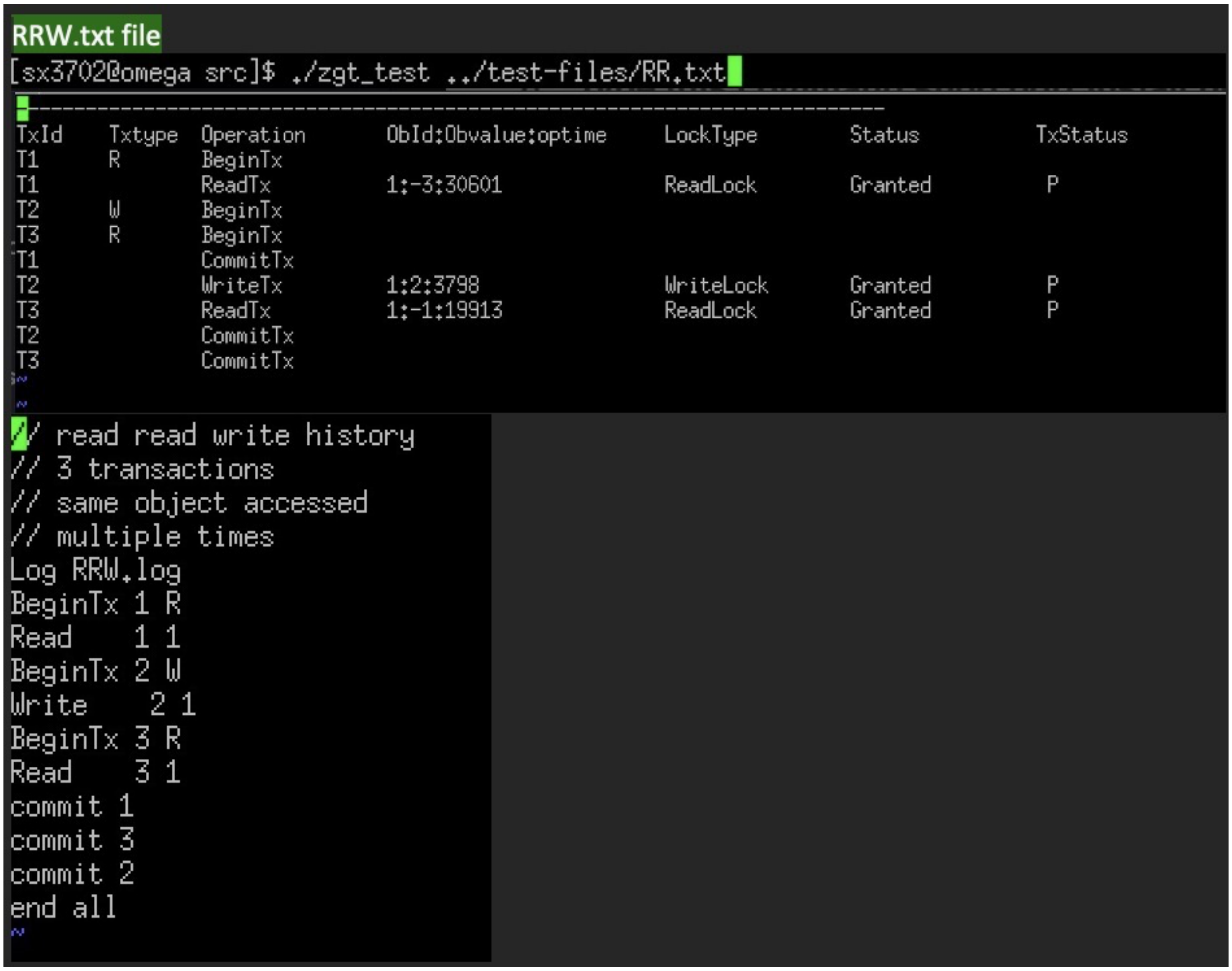
```
// 2 transactions
// generates a deadlock
// will hang w/o deadlock resolution
Log deadlock.log
BeginTx 1 W
BeginTx 2 W
Read    1 1
Read    2 2
Write   1 2
Write   2 1
Commit  1
commit 2
end all
~
```



RR.txt file

```
[sx3702@omega src]$ ./zgt_test ../test-files/RR.txt
-----------------------------------------------------------------------------
TxId    Txtype  Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1      R       BeginTx
T1              ReadTx          1:-3:30601              ReadLock        Granted          P
T2      R       BeginTx
T2              ReadTx          1:-6:3798               ReadLock        Granted          P
T2              ReadTx          2:-3:3798               ReadLock        Granted          P
T2              ReadTx          3:-3:3798               ReadLock        Granted          P
T2              CommitTx
T1              ReadTx          3:-6:30601              ReadLock        Granted          P
T1              ReadTx          2:-6:30601              ReadLock        Granted          P
T1              CommitTx
~
~
```

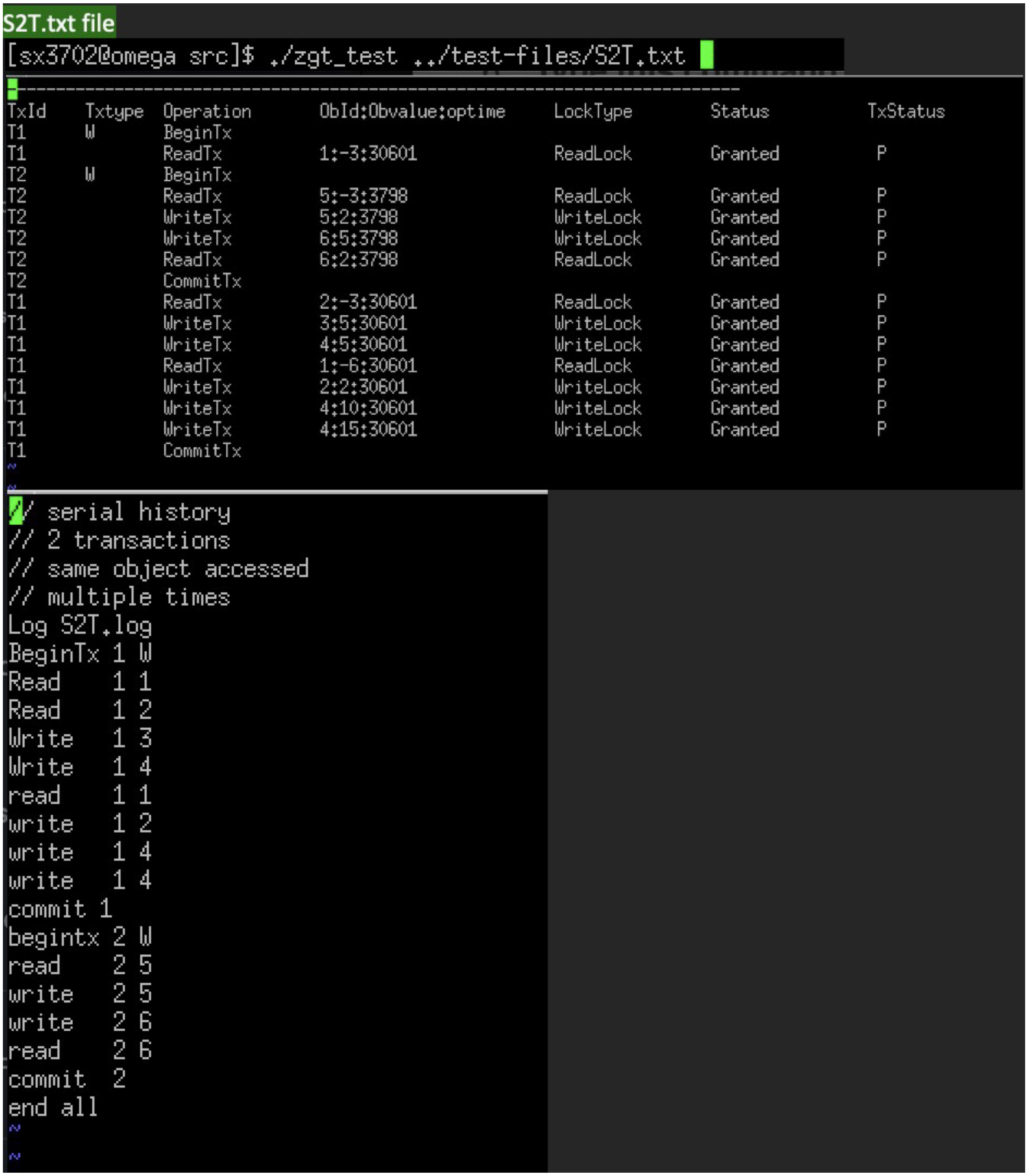
```
// read read history
// 2 transactions
// same object accessed
// multiple times
Log RR.log
BeginTx 1 R
Read    1 1
Read    1 3
begintx 2 R
read    2 1
Read    2 2
read    1 2
Read    2 3
commit 1
commit  2
end all
~
```



RRW.txt file

```
[sx3702@omega src]$ ./zgt_test ../test-files/RR.txt
-------------------------------------------------------------------------
TxId     Txtype   Operation      ObId:Obvalue:optime    LockType       Status         TxStatus
T1       R        BeginTx
T1                ReadTx         1:-3:30601             ReadLock       Granted         P
T2       W        BeginTx
T3       R        BeginTx
T1                CommitTx
T2                WriteTx        1:2:3798               WriteLock      Granted         P
T3                ReadTx         1:-1:19913             ReadLock       Granted         P
T2                CommitTx
T3                CommitTx
~
~
```

```
// read read write history
// 3 transactions
// same object accessed
// multiple times
Log RRW.log
BeginTx 1 R
Read    1 1
BeginTx 2 W
Write    2 1
BeginTx 3 R
Read    3 1
commit 1
commit 3
commit 2
end all
~
```



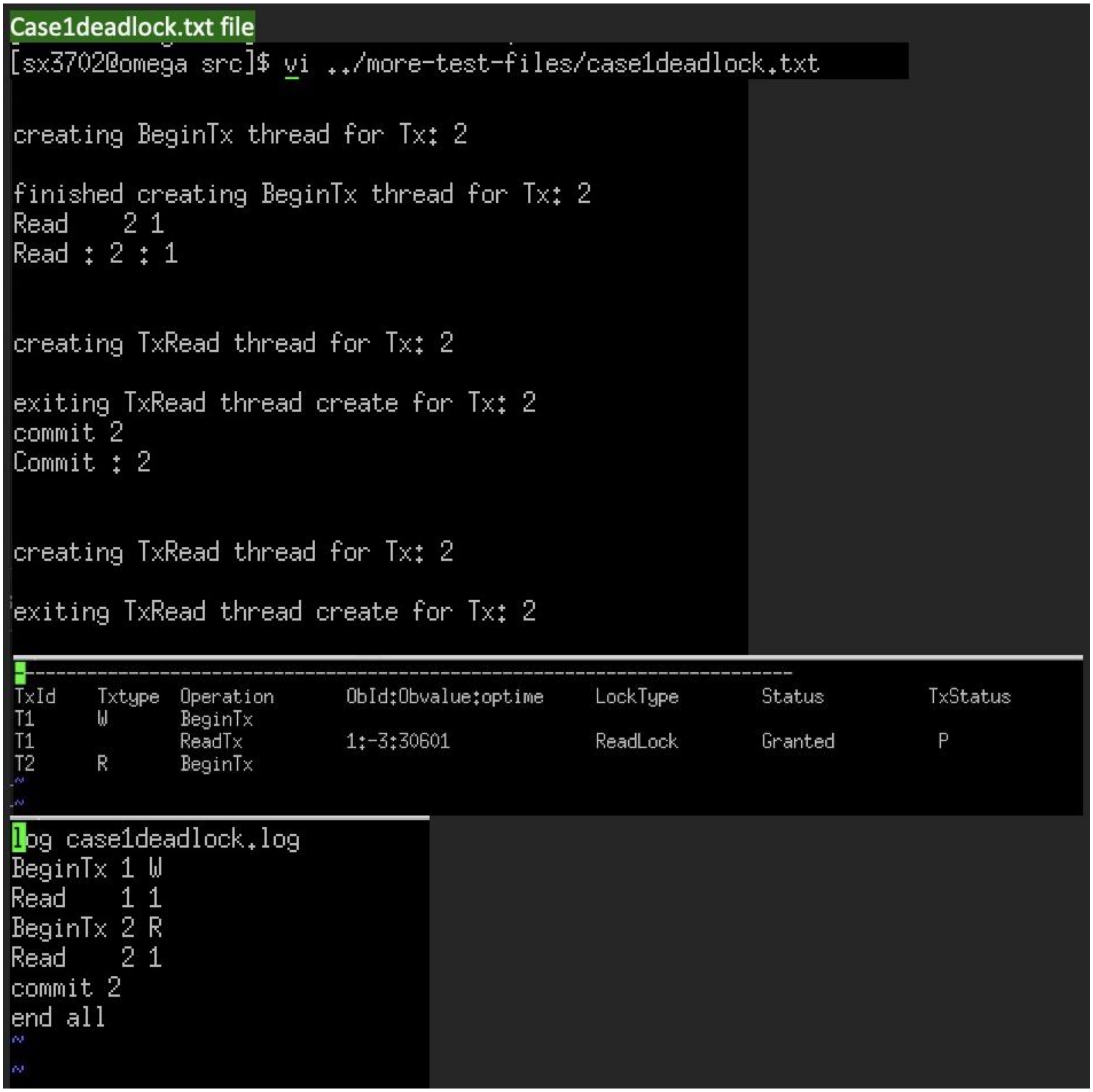
S2T.txt file

```
[sx3702@omega src]$ ./zgt_test ../test-files/S2T.txt
```

```
---------------------------------------------------------------------------
TxId    Txtype   Operation        ObId:Obvalue:optime     LockType        Status          TxStatus
T1      W        BeginTx
T1               ReadTx           1:-3:30601              ReadLock        Granted          P
T2      W        BeginTx
T2               ReadTx           5:-3:3798               ReadLock        Granted          P
T2               WriteTx          5:2:3798                WriteLock       Granted          P
T2               WriteTx          6:5:3798                WriteLock       Granted          P
T2               ReadTx           6:2:3798                ReadLock        Granted          P
T2               CommitTx
T1               ReadTx           2:-3:30601              ReadLock        Granted          P
T1               WriteTx          3:5:30601               WriteLock       Granted          P
T1               WriteTx          4:5:30601               WriteLock       Granted          P
T1               ReadTx           1:-6:30601              ReadLock        Granted          P
T1               WriteTx          2:2:30601               WriteLock       Granted          P
T1               WriteTx          4:10:30601              WriteLock       Granted          P
T1               WriteTx          4:15:30601              WriteLock       Granted          P
T1               CommitTx
~
~
```

```
// serial history
// 2 transactions
// same object accessed
// multiple times
Log S2T.log
BeginTx 1 W
Read    1 1
Read    1 2
Write   1 3
Write   1 4
read    1 1
write   1 2
write   1 4
write   1 4
commit 1
begintx 2 W
read    2 5
write   2 5
write   2 6
read    2 6
commit  2
end all
~
~
```

New test cases added.

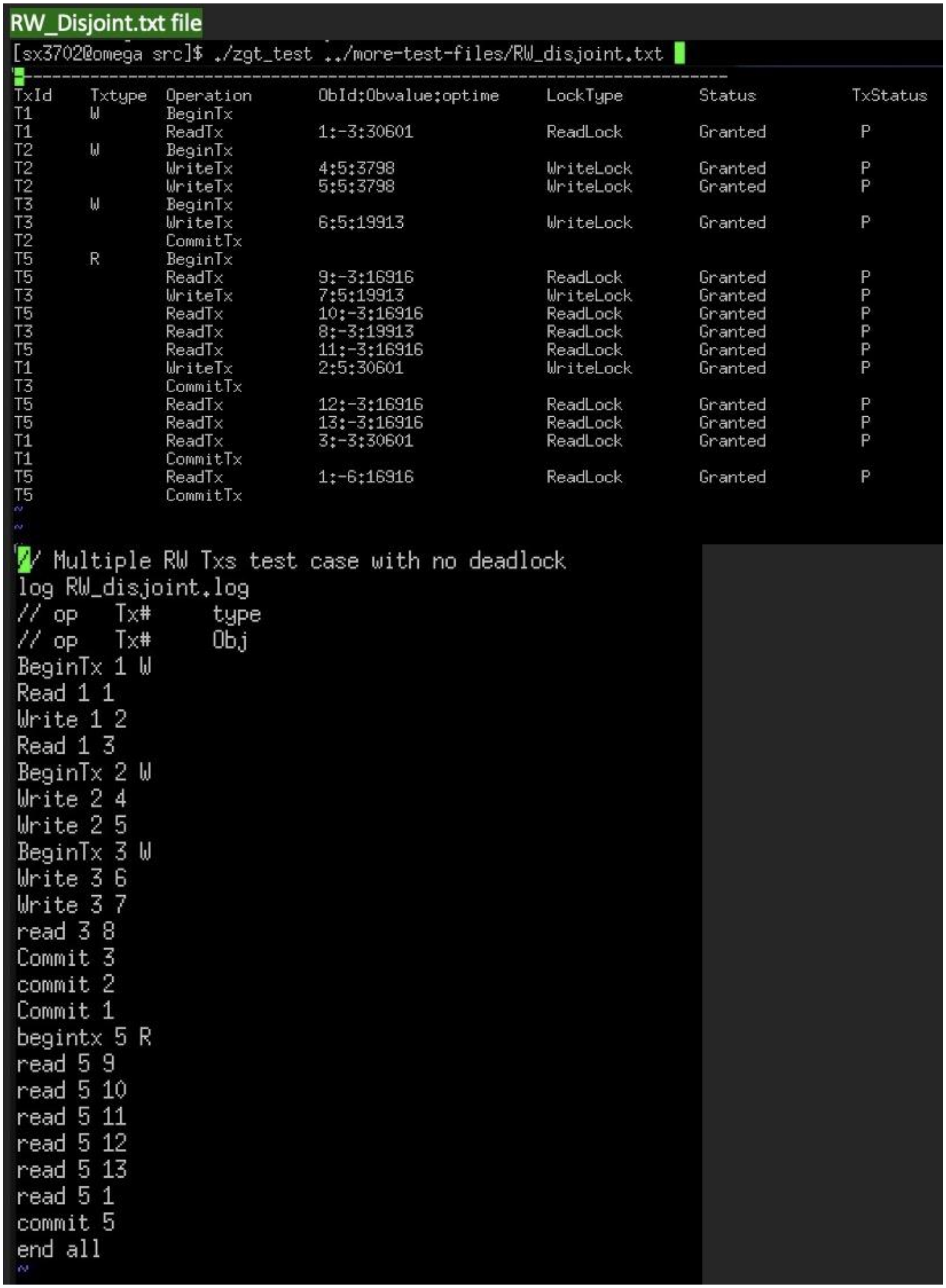


Case1deadlock.txt file

```
[sx3702@omega src]$ vi ../more-test-files/case1deadlock.txt

creating BeginTx thread for Tx: 2

finished creating BeginTx thread for Tx: 2
Read    2 1
Read : 2 : 1


creating TxRead thread for Tx: 2

exiting TxRead thread create for Tx: 2
commit 2
Commit : 2


creating TxRead thread for Tx: 2

exiting TxRead thread create for Tx: 2
```

```
-------------------------------------------------------------------------------------
TxId    Txtype  Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1      W       BeginTx
T1              ReadTx          1:-3:30601              ReadLock        Granted          P
T2      R       BeginTx
~
~
```

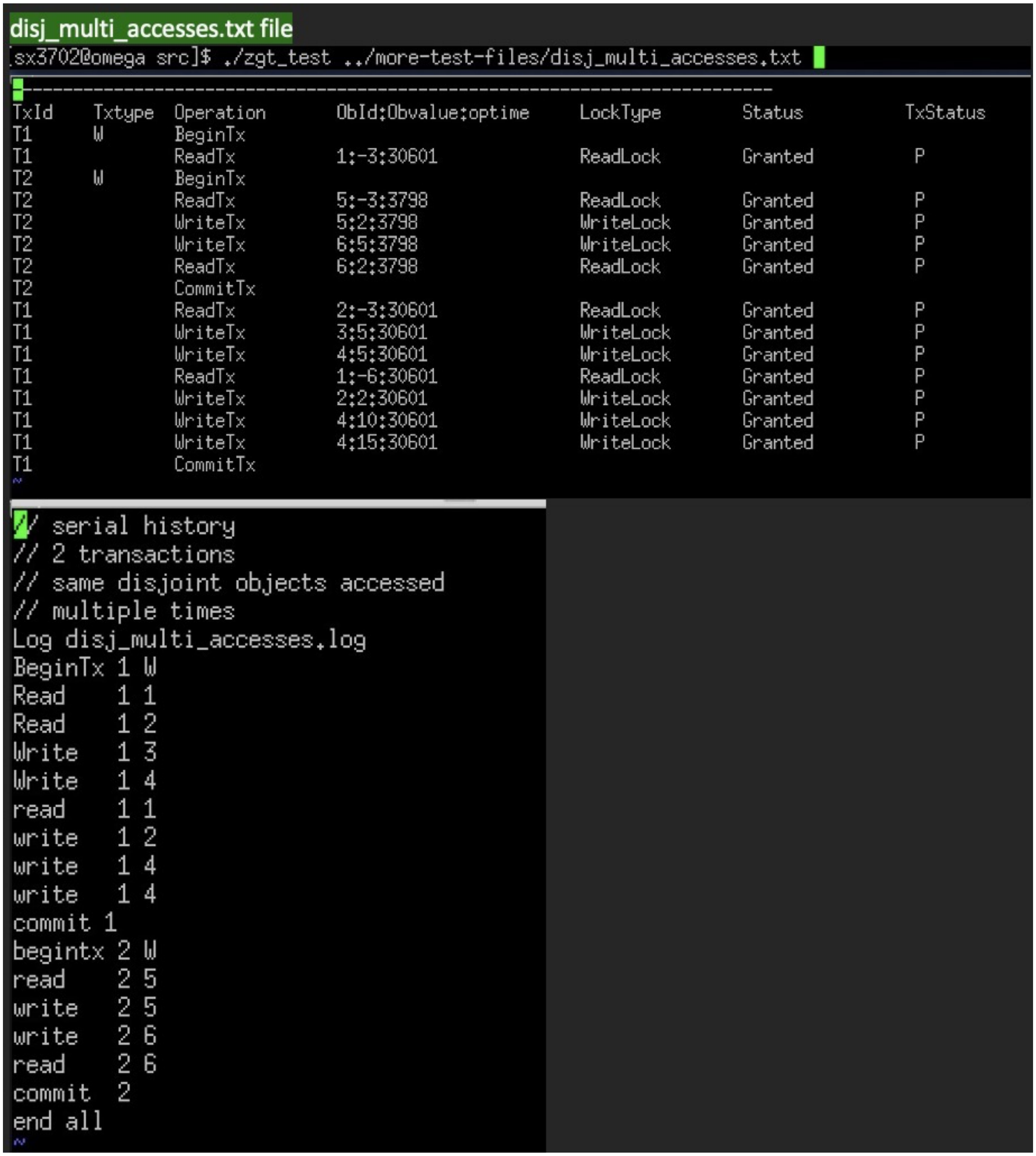
```
log case1deadlock.log
BeginTx 1 W
Read    1 1
BeginTx 2 R
Read    2 1
commit 2
end all
~
~
```



RW_Disjoint.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/RW_disjoint.txt
---------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime      LockType         Status           TxStatus
T1       W        BeginTx
T1                ReadTx           1:-3:30601               ReadLock         Granted           P
T2       W        BeginTx
T2                WriteTx          4:5:3798                 WriteLock        Granted           P
T2                WriteTx          5:5:3798                 WriteLock        Granted           P
T3       W        BeginTx
T3                WriteTx          6:5:19913                WriteLock        Granted           P
T2                CommitTx
T5       R        BeginTx
T5                ReadTx           9:-3:16916               ReadLock         Granted           P
T3                WriteTx          7:5:19913                WriteLock        Granted           P
T5                ReadTx           10:-3:16916              ReadLock         Granted           P
T3                ReadTx           8:-3:19913               ReadLock         Granted           P
T5                ReadTx           11:-3:16916              ReadLock         Granted           P
T1                WriteTx          2:5:30601                WriteLock        Granted           P
T3                CommitTx
T5                ReadTx           12:-3:16916              ReadLock         Granted           P
T5                ReadTx           13:-3:16916              ReadLock         Granted           P
T1                ReadTx           3:-3:30601               ReadLock         Granted           P
T1                CommitTx
T5                ReadTx           1:-6:16916               ReadLock         Granted           P
T5                CommitTx
~
~
```

```
// Multiple RW Txs test case with no deadlock
log RW_disjoint.log
// op   Tx#     type
// op   Tx#     Obj
BeginTx 1 W
Read 1 1
Write 1 2
Read 1 3
BeginTx 2 W
Write 2 4
Write 2 5
BeginTx 3 W
Write 3 6
Write 3 7
read 3 8
Commit 3
commit 2
Commit 1
begintx 5 R
read 5 9
read 5 10
read 5 11
read 5 12
read 5 13
read 5 1
commit 5
end all
~
```

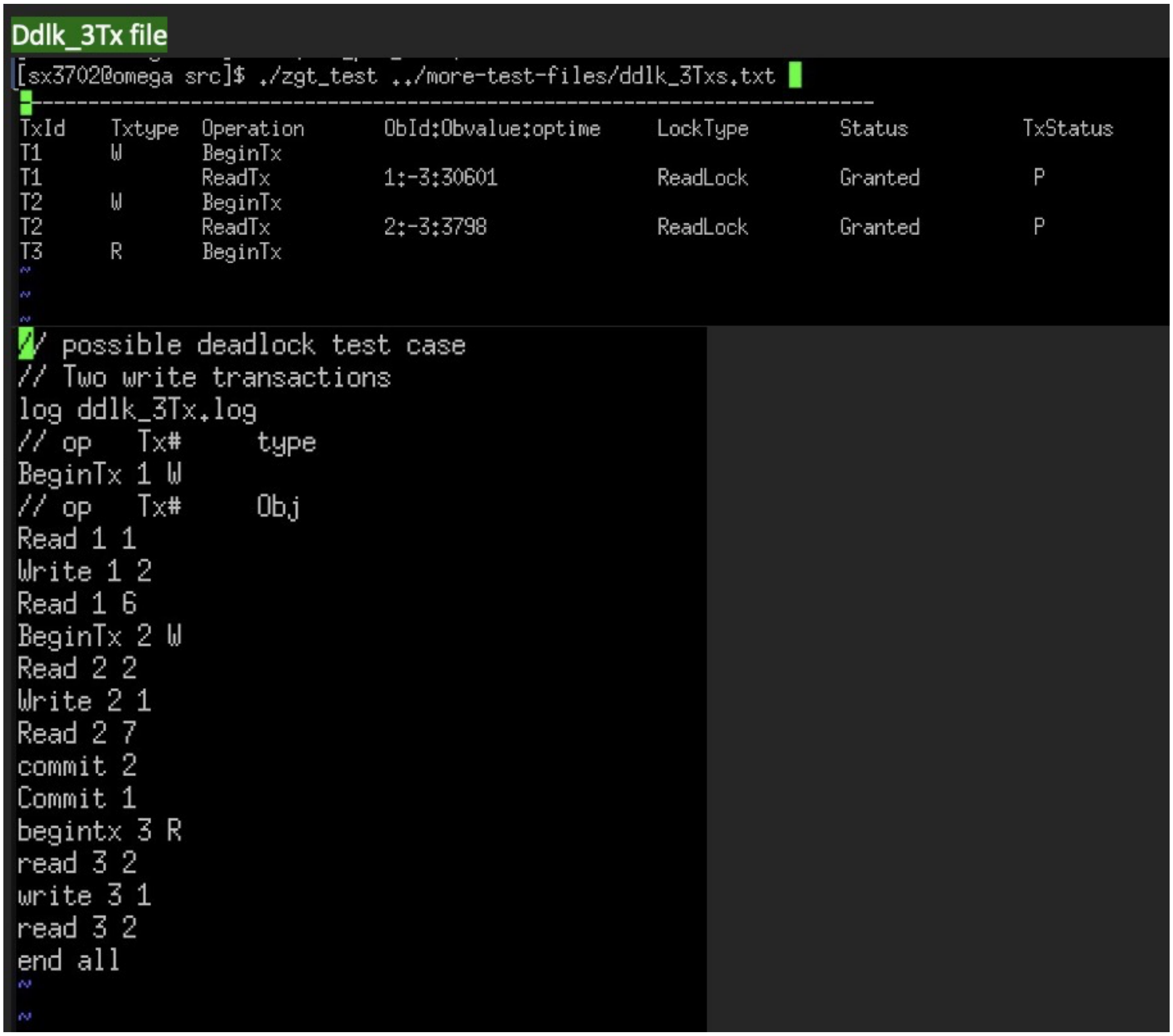


disj_multi_accesses.txt file

```
sx3702@omega src]$ ./zgt_test ../more-test-files/disj_multi_accesses.txt
```

```
---------------------------------------------------------------------------
TxId    Txtype  Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1      W       BeginTx
T1              ReadTx          1:-3:30601              ReadLock        Granted          P
T2      W       BeginTx
T2              ReadTx          5:-3:3798               ReadLock        Granted          P
T2              WriteTx         5:2:3798                WriteLock       Granted          P
T2              WriteTx         6:5:3798                WriteLock       Granted          P
T2              ReadTx          6:2:3798                ReadLock        Granted          P
T2              CommitTx
T1              ReadTx          2:-3:30601              ReadLock        Granted          P
T1              WriteTx         3:5:30601               WriteLock       Granted          P
T1              WriteTx         4:5:30601               WriteLock       Granted          P
T1              ReadTx          1:-6:30601              ReadLock        Granted          P
T1              WriteTx         2:2:30601               WriteLock       Granted          P
T1              WriteTx         4:10:30601              WriteLock       Granted          P
T1              WriteTx         4:15:30601              WriteLock       Granted          P
T1              CommitTx
~
```

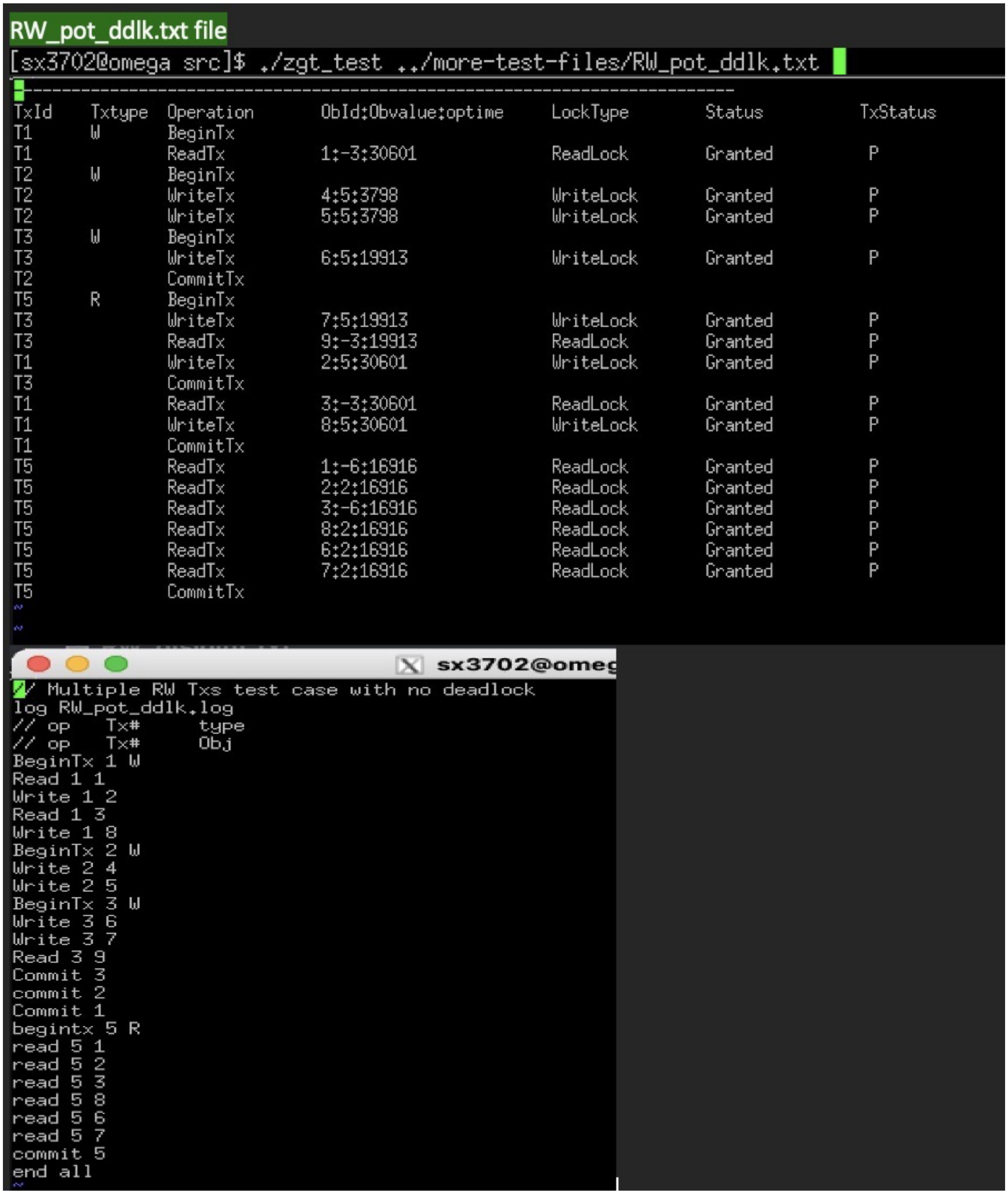
```
// serial history
// 2 transactions
// same disjoint objects accessed
// multiple times
Log disj_multi_accesses.log
BeginTx 1 W
Read    1 1
Read    1 2
Write   1 3
Write   1 4
read    1 1
write   1 2
write   1 4
write   1 4
commit 1
begintx 2 W
read    2 5
write   2 5
write   2 6
read    2 6
commit  2
end all
~
```



Ddlk_3Tx file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/ddlk_3Txs.txt
 -----------------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime     LockType        Status          TxStatus
T1       W        BeginTx
T1                ReadTx           1:-3:30601              ReadLock        Granted          P
T2       W        BeginTx
T2                ReadTx           2:-3:3798               ReadLock        Granted          P
T3       R        BeginTx
~
~
~
```

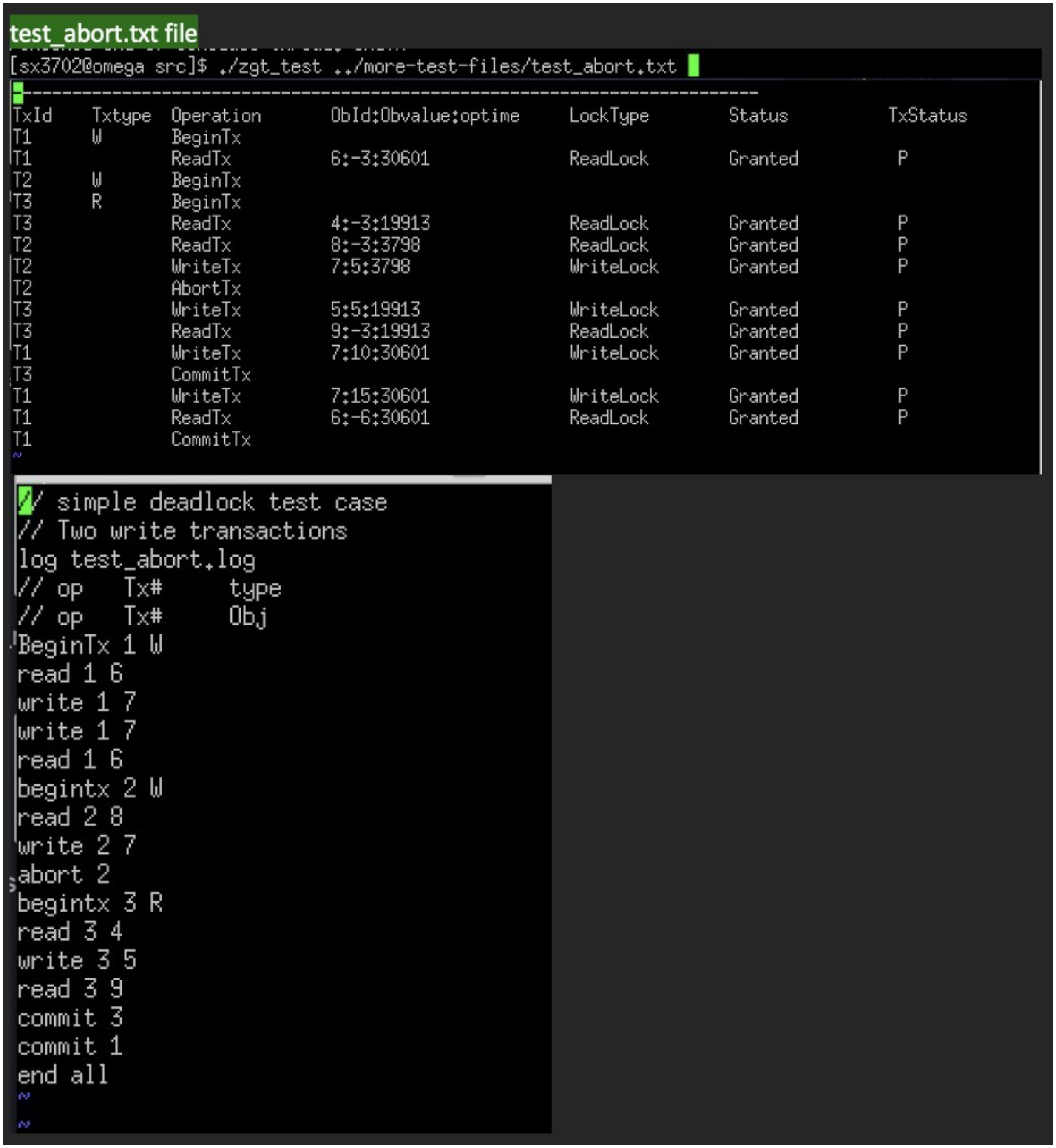
```
// possible deadlock test case
// Two write transactions
log ddlk_3Tx.log
// op    Tx#     type
BeginTx 1 W
// op    Tx#     Obj
Read 1 1
Write 1 2
Read 1 6
BeginTx 2 W
Read 2 2
Write 2 1
Read 2 7
commit 2
Commit 1
begintx 3 R
read 3 2
write 3 1
read 3 2
end all
~
~
```



RW_pot_ddlk.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/RW_pot_ddlk.txt
-----------------------------------------------------------------------------
TxId     Txtype   Operation       ObId:Obvalue:optime     LockType        Status          TxStatus
T1       W        BeginTx
T1                ReadTx          1:-3:30601              ReadLock        Granted          P
T2       W        BeginTx
T2                WriteTx         4:5:3798                WriteLock       Granted          P
T2                WriteTx         5:5:3798                WriteLock       Granted          P
T3       W        BeginTx
T3                WriteTx         6:5:19913               WriteLock       Granted          P
T2                CommitTx
T5       R        BeginTx
T3                WriteTx         7:5:19913               WriteLock       Granted          P
T3                ReadTx          9:-3:19913              ReadLock        Granted          P
T1                WriteTx         2:5:30601               WriteLock       Granted          P
T3                CommitTx
T1                ReadTx          3:-3:30601              ReadLock        Granted          P
T1                WriteTx         8:5:30601               WriteLock       Granted          P
T1                CommitTx
T5                ReadTx          1:-6:16916              ReadLock        Granted          P
T5                ReadTx          2:2:16916               ReadLock        Granted          P
T5                ReadTx          3:-6:16916              ReadLock        Granted          P
T5                ReadTx          8:2:16916               ReadLock        Granted          P
T5                ReadTx          6:2:16916               ReadLock        Granted          P
T5                ReadTx          7:2:16916               ReadLock        Granted          P
T5                CommitTx
~
~
```

```
// Multiple RW Txs test case with no deadlock
log RW_pot_ddlk.log
// op    Tx#      type
// op    Tx#      Obj
BeginTx 1 W
Read 1 1
Write 1 2
Read 1 3
Write 1 8
BeginTx 2 W
Write 2 4
Write 2 5
BeginTx 3 W
Write 3 6
Write 3 7
Read 3 9
Commit 3
commit 2
Commit 1
begintx 5 R
read 5 1
read 5 2
read 5 3
read 5 8
read 5 6
read 5 7
commit 5
end all
~
```

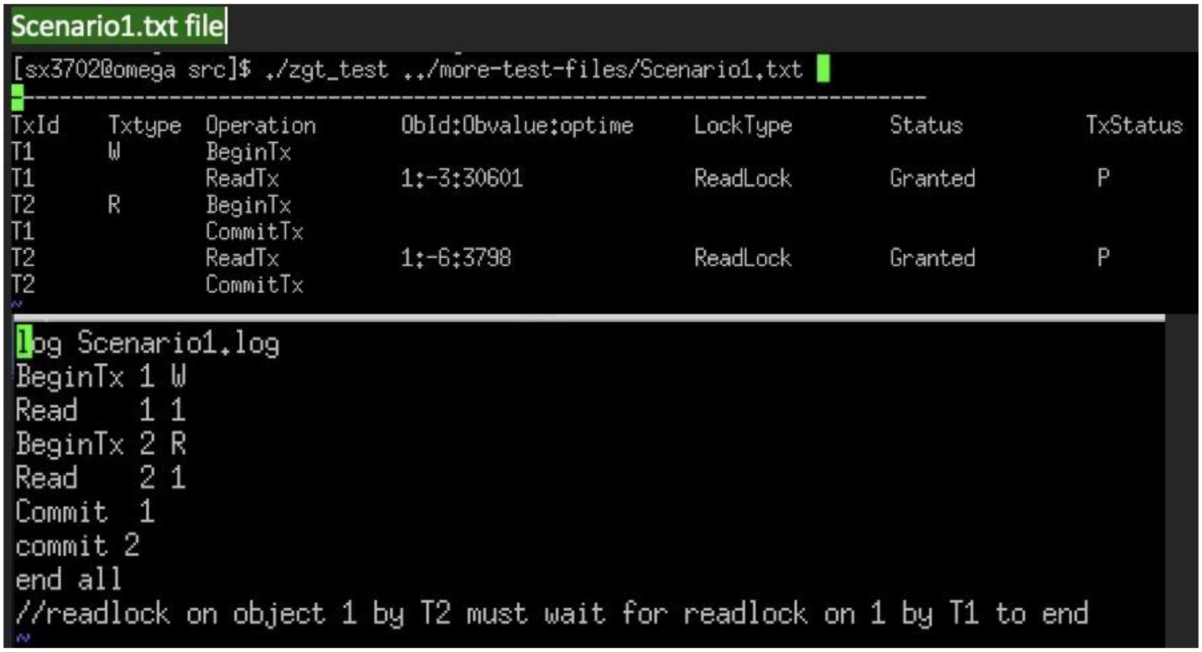


test_abort.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/test_abort.txt
```

```
---------------------------------------------------------------------------------
TxId     Txtype  Operation      ObId:Obvalue:optime    LockType       Status         TxStatus
T1       W       BeginTx
T1               ReadTx         6:-3:30601             ReadLock       Granted         P
T2       W       BeginTx
T3       R       BeginTx
T3               ReadTx         4:-3:19913             ReadLock       Granted         P
T2               ReadTx         8:-3:3798              ReadLock       Granted         P
T2               WriteTx        7:5:3798               WriteLock      Granted         P
T2               AbortTx
T3               WriteTx        5:5:19913              WriteLock      Granted         P
T3               ReadTx         9:-3:19913             ReadLock       Granted         P
T1               WriteTx        7:10:30601             WriteLock      Granted         P
T3               CommitTx
T1               WriteTx        7:15:30601             WriteLock      Granted         P
T1               ReadTx         6:-6:30601             ReadLock       Granted         P
T1               CommitTx
~
```

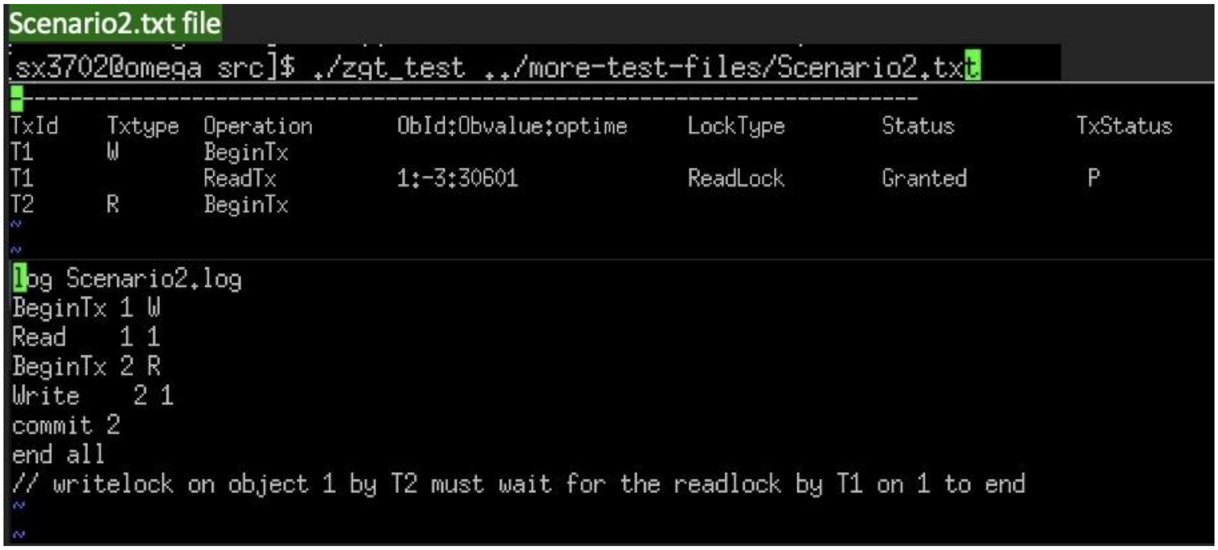
```
// simple deadlock test case
// Two write transactions
log test_abort.log
// op   Tx#     type
// op   Tx#     Obj
BeginTx 1 W
read 1 6
write 1 7
write 1 7
read 1 6
begintx 2 W
read 2 8
write 2 7
abort 2
begintx 3 R
read 3 4
write 3 5
read 3 9
commit 3
commit 1
end all
~
~
```



Scenario1.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/Scenario1.txt
-----------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime      LockType        Status          TxStatus
T1       W        BeginTx
T1                ReadTx           1:-3:30601               ReadLock        Granted          P
T2       R        BeginTx
T1                CommitTx
T2                ReadTx           1:-6:3798                ReadLock        Granted          P
T2                CommitTx
~
```

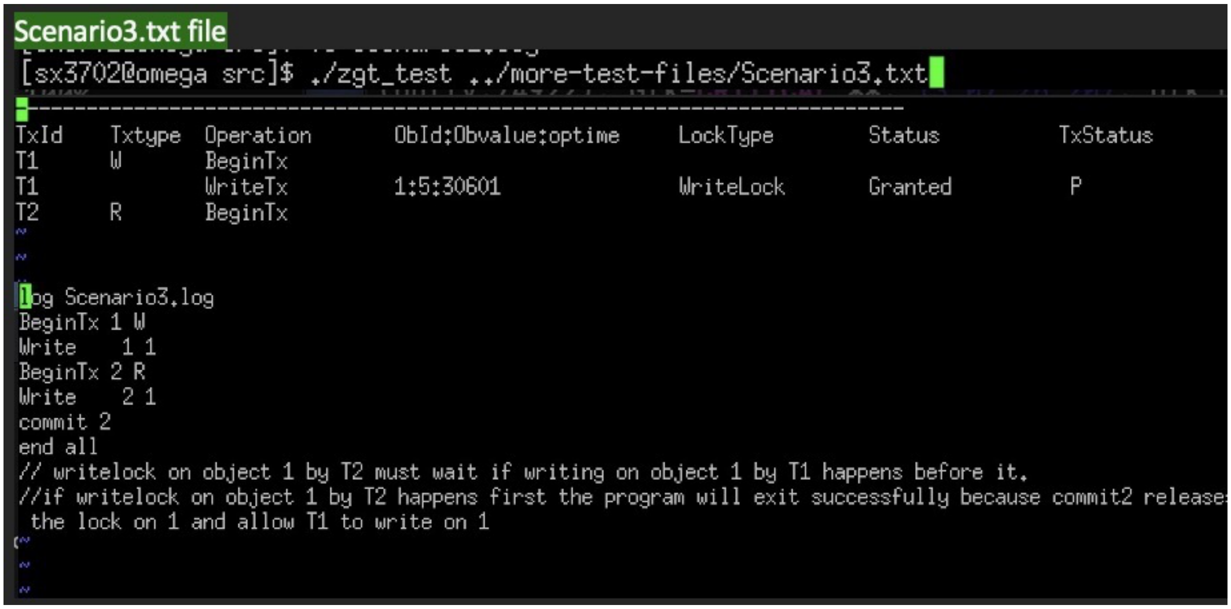
```
log Scenario1.log
BeginTx 1 W
Read    1 1
BeginTx 2 R
Read    2 1
Commit  1
commit 2
end all
//readlock on object 1 by T2 must wait for readlock on 1 by T1 to end
~
```



Scenario2.txt file

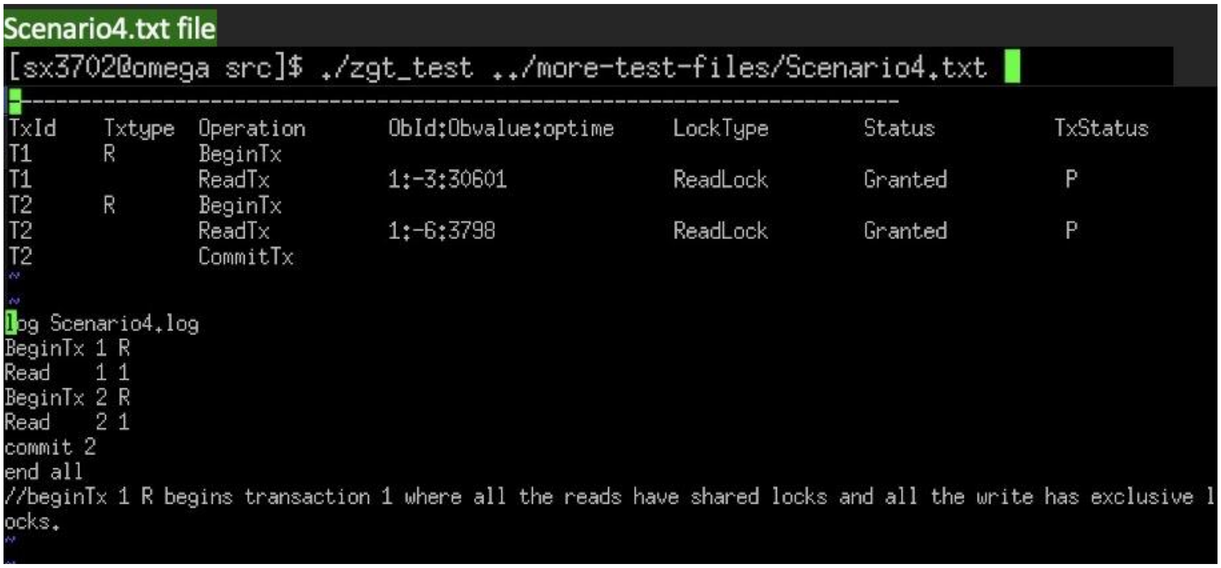
```
[sx3702@omega src]$ ./zgt_test ../more-test-files/Scenario2.txt
-----------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime      LockType        Status          TxStatus
T1       W        BeginTx
T1                ReadTx           1:-3:30601               ReadLock        Granted          P
T2       R        BeginTx
~
~
```

```
log Scenario2.log
BeginTx 1 W
Read    1 1
BeginTx 2 R
Write    2 1
commit 2
end all
// writelock on object 1 by T2 must wait for the readlock by T1 on 1 to end
~
~
```



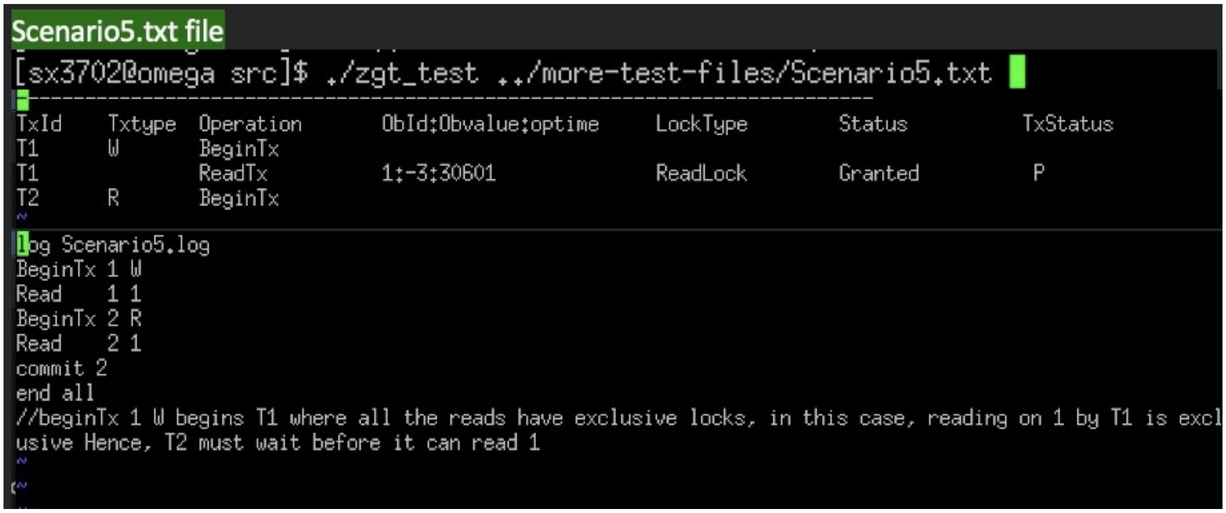
Scenario3.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/Scenario3.txt
-------------------------------------------------------------------------------
TxId     Txtype   Operation        ObId:Obvalue:optime      LockType        Status          TxStatus
T1       W        BeginTx
T1                WriteTx          1:5:30601                WriteLock       Granted          P
T2       R        BeginTx
~
~

log Scenario3.log
BeginTx 1 W
Write    1 1
BeginTx 2 R
Write    2 1
commit 2
end all
// writelock on object 1 by T2 must wait if writing on object 1 by T1 happens before it.
//if writelock on object 1 by T2 happens first the program will exit successfully because commit2 release
 the lock on 1 and allow T1 to write on 1
~
~
~
```



Scenario4.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/Scenario4.txt
-----------------------------------------------------------------------------
TxId     Txtype  Operation       ObId:Obvalue:optime        LockType        Status          TxStatus
T1       R       BeginTx
T1               ReadTx          1:-3:30601                 ReadLock        Granted          P
T2       R       BeginTx
T2               ReadTx          1:-6:3798                  ReadLock        Granted          P
T2               CommitTx
~
~
log Scenario4.log
BeginTx 1 R
Read    1 1
BeginTx 2 R
Read    2 1
commit 2
end all
//beginTx 1 R begins transaction 1 where all the reads have shared locks and all the write has exclusive l
ocks.
~
```



Scenario5.txt file

```
[sx3702@omega src]$ ./zgt_test ../more-test-files/Scenario5.txt
-----------------------------------------------------------------------------
TxId     Txtype  Operation       ObId:Obvalue:optime       LockType        Status          TxStatus
T1       W       BeginTx
T1               ReadTx          1:-3:30601                ReadLock        Granted          P
T2       R       BeginTx
~
log Scenario5.log
BeginTx 1 W
Read    1 1
BeginTx 2 R
Read    2 1
commit 2
end all
//beginTx 1 W begins T1 where all the reads have exclusive locks, in this case, reading on 1 by T1 is excl
usive Hence, T2 must wait before it can read 1
~
~
```