SQL*Plus's Forgotten History

Harald van Breederode
Senior Principal DBA Instructor
Oracle University NL

You use `SQL*Plus` on a daily basis to perform various DBA activities but wonder why the `SQL*Plus` command history is restricted to the last command in the command buffer. You wish for less cumbersome SQL statement command line editing, and for a command history that persists across `SQL*Plus` sessions. In this edition of "OU Expert's Corner" we will demonstrate a cure for `SQL*Plus` amnesia, and provide full command line editing support using a program called `rlwrap`.

What is `rlwrap`

`RLWRAP` is a readline wrapper program written by Hans Lub from The Netherlands that provides command line tools with history and editing support. It is designed to run on `Unix` based systems including `Oracle Enterprise Linux` and works as a wrapper around the `GNU readline library`, which is also used in the familiar `Linux Bash shell`. One may also use `rlwrap` on Windows systems using the `Cygwin` toolset, but that is outside the scope of this article.

Downloading `rlwrap`

The quickest way to download `rlwrap` is to navigate to "google.com", enter "`rlwrap nl`" and hit the "I am feeling lucky" button. This should open the `rlwrap` homepage "http://utopia.knoware.nl/~hlub/uck/rlwrap/" where one may download the `rlwrap` distribution file (called `rlwrap-0.30.tar.gz` at the time of writing this article).

Installing `rlwrap`

After the `rlwrap` compressed tar file is downloaded one must compile and install this extremely useful utility.

This involves four steps listed below.

1. The first step is to uncompress and untar the file using the `tar` command:

```
Oracle> tar xzf rlwrap-0.30.tar.gz
```

Note: Red font denotes data entered at the terminal in all code examples.

2. The next step is to change the current working directory to the rlwrap source code tree directory and run the 'configure' command in order to adjust the source code tree to the specifics of the machine's operating system:

```
Oracle> cd rlwrap-0.30
Oracle> ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
.
.
.
<output omitted>
.
.
.
Now do:
    make (or gmake)  to build rlwrap
    make check       for instructions how to test it
    make install     to install it
```

3. Next, the `rlwrap` source code tree is ready to be compiled using the `make` command:

```
Oracle> make
make  all-recursive
make[1]: Entering directory `/fs02/home/oracle/rlwrap-0.30'
Making all in doc
...
<output omitted>
...
make[2]: Leaving directory `/fs02/home/oracle/rlwrap-0.30'
make[1]: Leaving directory `/fs02/home/oracle/rlwrap-0.30'
```

4. The final step is to install the program that was just compiled using "`make install`". The root user must execute this step, because it installs `rlwrap` into the so-called local bin directory.

```
Oracle> sudo make install
Making install in doc
...
<output omitted>
...
make[2]: Leaving directory `/fs02/home/oracle/rlwrap-0.30'
make[1]: Leaving directory `/fs02/home/oracle/rlwrap-0.30'
```

Note: If root access is prohibited copy the rlwrap to a personal bin directory instead.

Invoking SQL*PLUS with `rlwrap`

Now that the download, compilation and installation are complete, `rlwrap` is ready for use. Instead of invoking `SQL*Plus` in the usual manner, it is invoked as an argument of `rlwrap` as follows:

```
Oracle> rlwrap sqlplus / as sysdba
```

Note: `rlwrap` may be used in this manner for any program that lacks command line editing and history functionality. This includes both other Oracle command line utilities and any other non-Oracle utilities.

Automating SQL*PLUS with `rlwrap`

To automate the invocation of `sqlplus` under control of `rlwrap`, one may define an `alias` as follows:

```
Oracle> alias sqlplus
-bash: alias: sqlplus: not found

Oracle> alias sqlplus='rlwrap sqlplus'

Oracle> alias sqlplus
alias sqlplus='rlwrap sqlplus'
```

The first `alias` command checks for the existence of an `alias` for `sqlplus`. The second command defines such an `alias`, and the third command returns the `alias` definition.

Note: To automate the definition of this `alias` at login, add the second command above to $HOME/.bashrc file.

The appearance and behaviour of `SQL*Plus` are unaffected by this utility except for new editing and history capabilities provided by executing it through `rlwrap`:

1. The command line may be edited by navigating left or right with the arrow keys and overtyping.
2. One may retrieve previous commands with the up and down arrow keys and then edit them, submit them or both. See the example below.
3. All commands are saved in the command history for both the current session and from the previous sessions.

Here is an example:

```
SQL> select count(*) from dba_users;

  COUNT(*)
----------
        12

SQL> create user harald identified by prutser;

User created.

-- Press up arrow and rlwrap recalls the previous statement.

SQL> create user harald identified by prutser;

-- Press up arrow again to retrieve the one before that

SQL> select count(*) from dba_users;
```

Some important additional features of `rlwrap` are:

1. **Incremental Search**
2. **Custom Vocabulary**
3. **Application Specific History**

They are described in the following sections.

1. Incremental Search

Retrieval of previous commands using the up arrow key may be time consuming and error prone so `rlwrap` provides an incremental search feature to locate previous commands more easily. Key `CTRL+R` followed by a string uniquely identifying the required statement or command. The search is performed incrementally "on the fly", and the most recently issued command matching the search string is displayed. Each additional character entered refines the search, hence the term incremental search. If the first match displayed is not the desired command or statement, key `CTRL+R` again and `rlwrap` will display the next most recent match for the same string. This may be repeated until the end of the reverse history is reached or until the required command is found.

Here is an example:

```
SQL> select count(*) from dba_users;

  COUNT(*)
----------
        12

SQL> create user harald identified by prutser;

User created.

-- Press CTRL+R to perform an incremental search

SQL>(reverse-i-search)`':

-- While typing 'user' rlwrap shows us the best match so far

SQL> (reverse-i-search)`user': create user harald identified
by prutser;

-- Press CTRL+R again and rlwrap displays the next match

SQL>(reverse-i-search)`user': select count(*) from
dba_users;
```

2. Custom Vocabulary

To enable faster typing, `rlwrap` supports a custom vocabulary feature by loading all words in the "`$HOME/.<application>_completions`" file which if used, should contain vocabulary specific to the particular application. To use this feature the following steps are required:

- **2.1 Capture the required keywords to the completion file**
- **2.2 Manually add keywords if required**
- **2.3 Copy the completion file to the correct location**
- **2.4 Begin using Custom Vocabulary for Completion**

2.1. Capture the SQL Keywords to the Completion File

The creation of a completion file with a list of all reserved SQL words will save typing time for DBAs who work in command line.

A sample SQL script to do the capture looks as follows:

```
set pages 0
set trimspool on
set feedback off
set echo off
set heading off
set termout off

spool reserved_words

select lower(keyword)
from v$reserved_words
where keyword like '____%'
order by 1;

spool off
```

Note: The `WHERE` clause in the above `SELECT` statement filters out short reserved words which can be typed quickly and which do not benefit from the completion process.

The generated file will contain keywords as follows:

```
abort
access
...
create
create_stored_outlines
...
sequence
sequential
```

2.2. Manually add keywords if required

One may add words to the completion file for `SQL*Plus` commands, init.ora parameters, V$ view names, data dictionary view names, supplied `PL/SQL` Package names, or any other keywords specific to Oracle which are useful to DBAs.

2.3. Copy the completion file to the correct location

Copy the spooled reserved_words file to `$HOME/.sqlplus_completions` to enable custom vocabulary for `SQL`.

2.4 Using the completion file

Once loaded, use of the `TAB` key after entering the first few characters of a word causes `rlwrap` to perform word completion based on the vocabulary in the file. This may be very useful in Oracle as `SQL*Plus` and `SQL` have many word specific to the database environment. If two or more words satisfy the search after using the `TAB` key, `rlwrap` will expand the word as much as possible and then sound a beep. Keying `TAB` again, will display a list of possible completions for the characters already entered, and by typing one or more additional characters `rlwrap` should be able to complete the intended word.

Here is an example of completion:

```
SQL> cre -- Type cre
SQL> cre <TAB> -- TAB expands cre to create
SQL> create

SQL> create seq -- Type seq
SQL> create seq <TAB> -- TAB expands seq to seqen

SQL> create sequen <TAB> -- TAB beeps
SQL> create sequen <TAB> -- TAB lists possible expansions
sequence   sequential

SQL> create sequenc – Type a ‘c’
SQL> create sequenc <TAB> -- TAB expands sequenc to sequence
SQL> create sequence
```

3. Application Specific History

Under the covers, command line input from terminals is handled by `rlwrap`, allowing command history to be accessed and editing to be done, after which `rlwrap` passes the results to `SQL*Plus`. Command history is stored by default in an application specific history file in the home directory called "`.<application>_history`", where `<application>` is the name of the application controlled by `rlwrap`. Thus "`$HOME/.sqlplus_history`" is the name of the `rlwrap` command history file for `SQL*Plus`.

Conclusion

Using `rlwrap` improves productivity by helping DBAs avoid the re-typing of commands in command line tools such as `SQL*Plus`, `RMAN`, `LSNRCTL` and other DBA utilities. The auto-completion feature, whereby `rlwrap` completes reserved words or file names, helps to improve typing speed. As `rlwrap` is built on top of the `GNU readline library` it can be completely personalized. For more information see the "`readline`" manual page. For the additional help on `rlwrap`, read the `rlwrap` manual page with "`man rlwrap`".