

Using Screen to Manage Multiple Remote and Interrupted SSH Sessions

Contributed by Administrator
 Tuesday, 10 February 2009
 Last Updated Sunday, 20 June 2010

Attaching to Remote SSH Sessions

In a day of laptops and remote systems, it's often impractical to keep the same ssh session going to a specific server indefinitely. There are times when I want to reconnect from home to an ssh session that I started at work. Unfortunately, ssh doesn't support that sort of thing. But the screen utility lets you do something similar to this.

Screen is great for letting you start a terminal session, walk away from it, and then come back later. Maybe you need to start a long running process such as a complicated data conversion or a multi-hour build. You can use screen to start the ball rolling, go home, and resume the already-in-progress and uninterrupted activity that you started at work.

What screen is and does

According to the first line of the screen man page, "Screen is a full-screen window manager that multiplexes a physical terminal between several processes (typically interactive shells)." That's a mouthful. It's also a little misleading.

When I think of a window manager, I think of an application that manages GUI windows on a desktop. When the screen documentation refers to a "window," it means a virtual terminal that is running some application. By "window manager," it means that screen manages one or more virtual terminals. By "full-screen," it means that screen can expand to the limits of the real terminal that contains the screen session. And by "multiplexes," it means that screen can control a number of virtual terminals and switch the view to any one of them and not interrupt the running of any of the others.

While that is the gist of what screen does, that's not the whole picture. When you execute the screen utility, you are really spawning a management process and connected to it. This management process is also known as a screen "session." You can disconnect from the process at any time and leave everything running just as if it were running in a terminal. You can also create a number of different virtual terminals, each running its own application, and switch among them. When you create these virtual terminals, you can almost think of them as part of a tabbed terminal emulator such as Terminal.app on Mac, gnome-terminal on Gnome, or konsole on KDE, except without the visible tabs.

Creating screen sessions

You create a screen session by running the "screen" command line utility. If you run "screen" with no command line arguments, screen will create a new session for you and drop you into a shell. The first thing you see will be a banner that looks like this: Screen version 4.00.03jw4 (FAU) 2-May-06 Copyright (c) 1993-2002 Juergen Weigert, Michael Schroeder Copyright (c) 1987 Oliver Laumann This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program (see the file COPYING); if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen@uni-erlangen.de 'Press Space or Return to end.'

After you hit space or return (or enter), you will be dropped into a shell. At this point, you can list all the screen sessions that you are running. If you run "screen" with a "-ls" argument, screen will display a list of your screen

```
sessions:jmjones@dinktrepid:~$ screen -ls
```

There is a screen on:

```
7596.pts-1.dinktrepid (01/30/2009 07:45:37 AM) (Attached)
1 Socket in /var/run/screen/S-jmjones.
```

The state of this screen session is "Attached." When you start a new screen session with no arguments, you will automatically be "attached" to it. The number "7596" indicates the process ID of the screen management process. I'll discuss later how this information is useful. (Hint, I said something about detaching and re-attaching to screen sessions earlier.)

Running Several Sessions

What if I need to start several screen sessions? Let's say I want one to manage a long data load, one to run a build, and

one to run a command line bittorrent to download the latest DVD of Suse. If I have run "screen" with no arguments in different terminal windows (or tabs), I will now have three attached screen sessions:

```
jmjones@dinktrepid:~$ screen -ls
There are screens on:
  7957.pts-5.dinktrepid (01/30/2009 07:55:03 AM) (Attached)
  7859.pts-3.dinktrepid (01/30/2009 07:54:54 AM) (Attached)
  7596.pts-1.dinktrepid (01/30/2009 07:45:37 AM) (Attached)
3 Sockets in /var/run/screen/S-jmjones.
```

There's a slight problem with this; how can I tell which process ID corresponds to which function? Maybe I can remember which order I started them in and find the right one. But there is an easier way. You can pass screen a certain command line argument that will let you see the text you passed in the "screen -ls" list. Basically, it lets you "tag" a screen session with a piece of text. Most of the time, I use this command to identify my purpose for creating the screen session. Here are three commands to create screen sessions with the same reasons as before, only explicitly declaring the purpose for creating the session.

```
jmjones@dinktrepid:~$ screen -S data_load
jmjones@dinktrepid:~$ screen -S build
jmjones@dinktrepid:~$ screen -S torrent
```

screen uses the text after the "-S" option as the "name" for the session. Now, when I look at a list of screen sessions, I can see the names.

```
jmjones@dinktrepid:~$ screen -ls
There are screens on:
  15966.torrent (01/30/2009 08:06:04 AM) (Attached)
  15937.build (01/30/2009 08:05:57 AM) (Attached)
  15899.data_load (01/30/2009 08:05:47 AM) (Attached)
3 Sockets in /var/run/screen/S-jmjones. Detaching from and Attaching to screen Sessions
```

What do you do when you want to take your laptop home? Normally, closing your Terminal.app or gnome-terminal or konsole results in your processes dying a sad, lonely death. But with screen, you don't have to worry about that. You can detach from the screen session by hitting CONTROL-a followed by CONTROL-d. Most screen commands are accessible by holding down CONTROL-a, letting the keys up, then hitting the next key sequence. In this case, holding down the CONTROL key, then pressing the "a" key and letting it up, then pressing the "d" key (which keeping the CONTROL key depressed) and letting it up will detach the current screen session. After CONTROL-a CONTROL-d, you will drop back into the shell you were in when you started screen.

So, what do you when you get home and want to connect back up to the screen sessions that you detached from at the office? This is where the "screen -ls" command comes in. You have a couple of options to connect back to the "build" session I showed you earlier. You can re-attach by process ID:screen -r 15937

Or you can re-attach by name:screen -r build

Creating new windows

When I created individual screen sessions for the build, the data load, and the bittorrent client, it wasn't entirely necessary. All that I really had to do was to create a window (virtual terminal) for each of the applications. From within a running screen session, if you type CONTROL-a CONTROL-c, you will get a new shell in a new virtual terminal (or window) in the screen session. You can create as many virtual terminals as you want. You navigate to the next window by typing CONTROL-a CONTROL-n and you can go to the previous window by typing CONTROL-a CONTROL-p. Or, you can flip back to the last window you were on by typing CONTROL-a CONTROL-a.

Creating named screen sessions definitely has its uses. Being able to identify what you were working on and even collaborating with a co-worker is invaluable. But being able to create multiple "tabs" in a screen session is a huge bonus.

Conclusion

screen will do much more than what I've shown in this article. To get a fuller list of what you can do, try CONTROL-a CONTROL-? from a running screen session or give "man screen" a read. The commands I've shown you are the commands you will probably use most of the time and with them, you will be extremely productive with screen. With these commands, you can create a bundle of shells that you can disconnect from, let them run, then reconnect to when you need to resume your work.