Screen Tutorial

Introduction

This tutorial is hardly complete and only covers what is needed to use screen to keep our rails development server running. Screen allows us to create a terminal session that can remain open and in the same state even after we disconnect from the remote server. This is required because the rails server shuts down whenever the session that started it closes. Screen circumvents this problem by creating a persistent session that can be connected to or disconnected from without altering the session's state and any programs that were running in it. This tutorial will cover screen creation, disconnecting from screen sessions, reconnecting to screen sessions, and finally destroying screen sessions.

Screen Creation

This is the easiest part. Just call:

```
screen
```

This creates a screen session and gives it an automatically generated name that you will have to find by calling:

```
screen -ls
```

It will return a name that looks like this:

```
$ screen -ls
There are screens on:
    7139.pts-0.lawbuntu     (12/09/2013 10:16:17 AM)    (Detached)
```

If you would like a more memorable name call the following with <socketName> being the name that you would like to give your screen instance:

```
screen -S <socketName>
```

Keep in mind that the -S is a capital S and stands for Socket name. The socket name will also help you keep track of what you have running in that particular screen instance. Once you have created your screen, you will be immediately placed into that screen and any subsequent commands will live in that screen's session rather than your local or SSH session.

On our rails development server, our root user has a screen named "`6684.serve80`*" that is running, on port 80, the git master branch of our website's code.*

Disconnecting from a Screen

Again this step is easy to do but still takes some remembering powers. While you are connected to the screen session, press on the keyboard *Ctrl+A, Ctrl+D*. That's it. Hold the *Ctrl* button and

then while holding it press *A* THEN *D*. The handy mnemonic is "*press ctrl* **A** *then ctrl* **D**". Once disconnected, the screen is still available by either the name it was given or by the automatically generated name given by screen. After being disconnected you will be back in the session you were in when you created the screen.

Reconnecting to a Screen

Before being able to reconnect to a running screen, you need to make sure you know the name and that no one else is currently connected to it. Let's look at `screen -ls` again:

```
# screen -ls
There are screens on:
     7340.myScreen   (12/09/2013 10:42:19 AM)         (Detached)
```

There are three names you can use to reference this screen: "7340", "myScreen", or "7340.myScreen." The command for reconnecting to a currently running screen, with <screenName> being the name you are using to reference the screen is:

```
screen -r <screenName>
```

This will reconnect you to the screen in the state it was in when you left it.

Destroying a Screen Session

Finally to destroy a screen session, just *exit* while you are in the session by, while being in the session, call:

```
exit
```

This will close the session and destroy it as well as the screen that contained it.

Quick Note

As a reminder, a screen session is tied to the user that created it. Since we cannot log in as each other, the only user that we all have access to is root. For that reason we run our own personal instances of the rails server in our own screens and the main, port 80, server in a root screen. This can be done by running:

```
sudo su
```

and then by connecting to the (hopefully already running) server's screen named "serve80."

```
screen -r serve80
```

That screen will be already running the server. To stop the server, press *Ctrl+C* and then run it again with:

```
rails s -p 80
```

Using –p 80 will keep the main server running on port 80 so that it can be accessed at http://cheapasfree.com/ without any port numbers.