Must Have Utilities

This is a collection of utilities on Unix which make life so much easier.

reptyr

Ever started a process on a terminal via ssh, only to realize this will take roughly the half life of the universe? You don't want to stop the process (it has already run a long time), but you really don't want to kill it when it could be done Real Soon Now.

`reptyr` to the rescue. `reptyr` is a utility which can steal a running process from one terminal and put it on another. For example, the correct way to do the above would have been to use `screen`, so it can run without a terminal. With `reptyr`, you can open a second ssh terminal, start `screen` (or `tmux`), “steal” the process from your running terminal, then let `screen` handle it (with no ssh connection).

I found this little jewel when I had a process running a scan of a 10T file system, and really wanted it to not run via my ssh session for the 60 hours it eventually took. I found the excellent article https://www.ostechnix.com/''reptyr''-move-running-process-new-terminal/ telling me everything I needed to get started, with links to additional information.

To test, I created a perl script below:

neverend.pl

```
#! /usr/bin/env perl

use warnings;
use strict;

select (STDERR);
$| = 1;

while ( 1 ) {
   print '.';
   sleep 1;
}
```

I then opened a terminal and started it running. Little periods over and over.

Now, I open a second terminal and figure out the PID of the script.

```
:~$ ps ax | grep neverend
30297 pts/1    S+     0:00 perl ./neverend.pl
30351 pts/2    S+     0:00 grep neverend
```

Remember, I'm now in the second terminal. Issue the command:

```
''reptyr'' //pid//
```

to steal the process from the other terminal:

```
:~$ ''reptyr'' 30297
[+] Allocated scratch page: 7f575938a000
[+] Looking up fds for tty in child.
[+] Resolved child tty: 8801
[+] Found an alias for the tty: 0
[+] Found an alias for the tty: 1
[+] Found an alias for the tty: 2
[+] Opened the new tty in the child: 3
[+] Target is not a session leader, attempting to setsid.
[+] Forked a child: 30356
[+] Change pgid for pid 30297
[+] Did setsid()
[+] Set the controlling tty
....
```

You'll see the process terminated on the original terminal

```
:~/temp$ ./neverend.pl
.....................................................................
.....................................................................
.....
[1]+  Stopped                 ./neverend.pl
```

At this point, you can exit the original terminal.

One great scenario would be to call `screen` before you steal the process via `reptyr`. Or, you could simply move your lazy rear from your computer to the terminal, log in and steal it that way, so the process is running on a terminal local to the server (obviously there are shortcomings).

pv

Long running processes can be a problem for me. Do I have time to go grab a coffee, or even supper, or will this thing finish Real Soon Now. The following code always drove me up a tree:

```
dd if=/some/large/disk/image | ssh backupserver 'dd
of=/name/of/image/on/server'
```

Yes, it is possible to get a status of dd, but I never remember how. pv to the rescue. pv is a filter, whose only job is to track throughput and give you a nice little progress bar.

Note: pv is a filter; it doesn't know anything about the size. If you want a progress bar with an ETA, you need to enter that as a command line parameter.

I have a great acronym for the parameters I use for pv all the time, the name petrs, followed by a space and the size of the data being processed. So, for example, if /some/large/disk/image was 100G, I'd write

```
dd if=/some/large/disk/image | pv -petrs 100G | ssh backupserver 'dd of=/name/of/image/on/server'
```

Look at the man page. Lots and lots of things you can do with this, and pv has a minuscule footprint. Never impacted performance that I can ever see.

screen

Remember the reptyr example above? That problem could have all been solved by using `screen` (or `tmux`, though I don't know anything about it). `screen` is a screen manager that allows you to abandon a terminal with a running process, then come back to it later. So, basically, multiple terminals (called sessions) in one console.

First, before anything, issue the screen function. You can include the name of the process you want to run also, if you like.

You can now view the output just like normal, but if you have better things to do, use the command

```
control-a D
```

to detach the screen. You'll see a notification similar to the following.

```
[detached from 30675.pts-2.wash]
```

and you are returned to the command prompt, without screen running. At a later date, you can log in and enter

```
screen -Dr
```

which will either give you a list of all sessions running and tell you how to connect (if more than one) or, if only one session, will immediately connect you.

`screen` is much, much more powerful. You can run multiple processes (sessions) simultaneously, do a full screen refresh, things like that. But, I won't go over that as you can rtfm or simply search the 'net for all the other options.

The real power is that you can begin a process over an ssh session, then log out of the ssh session, and come back hours (or days) later and see the results.

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/software/musthaveutilties

Last update: **2017/11/16 04:48**