GNU Screen

Matt G. Habib & Ryan Curtin

LUG@GT

Introduction

What is GNU Screen and why should I care?

- Terminal multiplexer
- Preservation of sessions/jobs
- Access to the same job from multiple places
- Multiple-user shared sessions (group collaboration)
- Quick task switching

Quick Demonstration

To start a session, just type `screen`. Your `.screenrc` file will be read.

Screen Modes

A screen session can be in one of five different modes. These modes can be run by typing the command `screen -ls`. The following table lists the various modes a screen session can exist in.

Mode	Description
attached	session is running and has a controlling terminal
detached	session can be reattached using -r option
multi	session is in multiuser mode
unreachable*	either live on a different host or 'dead'
dead	should be throughly checked and removed using `-wipe` option

* An unreachable session is considered dead when its name matches either the name of the localhost or the specified parameter, if any.

Comandline Parameters

The following table lists a few commonly used options when running screen:

Option	Description
-ls	Prints a list of all screen sessions (previous slide)
-A	Adapt the size of all windows to the size of the current terminal
-R	Reattach a session and if necessary detach it first or even create it first
-x	Attach to a not detached screen session

Default Key Bindings (1/2)

Without these general key bindings, you will surely be lost!

Command	Description
C-a '	Prompt for a window name to switch to
C-a "	Present a list of all windows for selection
C-a 0 ... C-a 9	Switch to window number 0-9
C-a tab	Switch the focus to the next region
C-a C-a	Toggle to the previously displayed window
C-a A	Title the current window
C-a c	Create a new window with a shall
C-a d	Detach screen from the current terminal
C-a F	Resize the window to the current region size
C-a H	Begins/ends logging of the current window

Default Key Bindings (2/2)

Command	Description
C-a k	Destroy the current window
C-a m	Repeats the last message displayed in the window
C-a M	Toggles monitoring of the current window
C-a C-g	Toggles the visual bell
C-a n (p)	Switch to the next (or previous) window
C-a S	Horizontaolly splits the current region
C-a x	Lock the terminal
C-a X	Remove the current region (split screen)
C-a esc	Enter copy/scrollback mode
C-a *	Show a listing of all currently attached displays

The default key bindings can be customized in the screenrc file.

Custom Keybindings

You can define your own keybindings and actions to go with them.

```
bind [-c class] key [command [args]]
bindkey [input sequence] [command [args]]
command -c [class]
```

For instance, map `C-f 1` to switch to window 11:

```
bindkey "^F" command -c switch10
bind -c switch10 1 select 11
```

More Custom Keybindings

Make 'C-Y' start a window watching syslog:
```
bind ^Y screen -t log watch -n 5 rail -40
/var/log/syslog
```

F8 to turn the status bar on:
```
bindkey -k k8 hardstatus alwayslastline
```

F9 to turn the status bar off:
```
bindkey -k k9 hardstatus alwaysignore
```

Other Useful screenrc Commands

`defscrollback [num]` - define length of scrollback buffer

`startup_message off` - you don't want it on

`vbell on` - if you hate yourself

`vbell_msg [message]` - display message at the bottom for a bell alert

`nethack on` - slightly entertaining messages

`msgwait [seconds]` - set message timeout

Starting default screens

In your screenrc, use the `screen` command:

```
screen [-n #] [-t title] [-L] [-M] [command
[args]]
```

`-L` - enable logging
`-M` - enable monitoring

Example:
```
screen -t "yes sir" yes
screen -n 5 -t "python shell" python
```

Password Protection

Prevents malicious users from baggy-pantsing your beautiful screen session.

Generate hash:
`C-a :password`
Password is placed in copy buffer (`C-a ]`)

Now place hashed password in screenrc:
`password [HASH]`

Copy/Paste

`C-a Esc` - Enter copy mode

When in copy mode, `Y` grabs the current line and exits copy mode. `y` starts marking from the beginning of the line. `W` marks exactly one word and exits copy mode.

`C-a [` - Copy selected text into copy buffer
`C-a ]` - Paste text in copy buffer

In copy mode, `/` searches forward, `?` searches backwards (vi-style).

Logging

Dump current screen to file ("screen capture"):
`C-a h` dumps to hardcopy.n in the default directory.
`:hardcopy filename` dumps to filename

Log screen to file:
`C-a H` logs to screenlog.n (unless logfile is defined) in default directory.
`:logfile filename` defines the file to log to.

Hit `C-a H` again to stop logging.

Hardstatus Line

```
hardstatus [on|off]
hardstatus [always]lastline|message|ignore
[string]
hardstatus string [string]
```

- The first form toggles the hardware status line.

- The second form identifies how to handle the hardstatus line.
  - ◆ 'lastline' will reserve the last line of the display for the hardstatus.
  - ◆ 'message' uses screen's message mechanism
  - ◆ 'ignore' tells screen never to display the hardstatus.
  - ◆ If 'always' is prepended to the type (e.g. 'alwayslastline'), screen will always show the type selected.

- The third form specifies the contents of the hardstatus line.

Hardstatus Line String (1/2)

```
hardstatus on
hardstatus alwayslastline
backtick 42 1 1 cat /tmp/.temp
hardstatus string "%{kw}%-w%{r}%n,%t%{-}%+w
%42`%=%D %m/%d/%y ::  [%c:%s]"
```

Explanation:

% is the escape character

Char	Description
%{kw}	set all the following stuff with a black foreground and white background
%-w	all numbers up the active one with number and title
%{r}	set the color of active window red
%n, %t	[number of window], [title of the current window]

Hardstatus Line String (2/2)

```
hardstatus string "%{kw}%-w%{r}%n,%t%{-}%+w
%42`%=%D %m/%d/%y ::  [%c:%s]"
```

Char	Description
%{-}	Don't change any colors
%+w	all windows after the active one with number and title
%42'	Look for the backtick process numbered 42 and display result
%=	pad everything after this to the right
%D	Name of the day
%m/%d/%y	date (numbered format)
%c:%s	time

Backtick

The backtick command is a handy tool to run a specified command and return the output for substitution of the '%'' string escape. The form is as follows:

```
backtick ID LIFESPAN AUTOREFRESH COMMAND
[ARGS]
```

Command	Description
ID	The identifier in which to return the string
LIFESPAN	The number of seconds the output is considered valid
AUTOREFRESH	Triggers an automatic refresh for caption and hardstatus strings (seconds)

```
backtick 42 1 1 cat /tmp/.temp
```

Horizontal Screen Split

`C-a S` - split current region horizontally

`C-a <Tab>` - focus on next region

`:resize [(+/-)lines]` - resize region; +N, -N, N, =, max, min

`C-a F` - fit window to region size (works with non-split too)

`C-a Q` - kill all regions except current region

`C-a X` - remove current region (if more than one)

You can split a screen as many times as you like.
More trickery is required for vertical screen split.

Vertical Screen Split

Patch source with vertical-split patch: `http://vsp4sdl.yuggoth.org/wrp_vertical_split_0.3_4.0.2.diff.bz2`

Then compile from source.

`C-a V` - split vertically

Region control is the same, but there is not a splitting line (could be written in).

Multiuser Screen Sessions

Collaborate with friends!

First allow multiuser: `:multiuser on`

Allow user: `:acladd username [password]`

Other user resumes session: `screen -x -S firstuser/sessionname`

Access Control

`:aclchg [usernames] [permission bits] [screen list]`

Permission bits are like UNIX permissions: `rwx`.

`r` - see the window
`w` - allow input to be typed into the window
`x` - execute commands in the window (screen commands)

Example:
`:aclchg chad -r-w-x 1`
This disallows chad from doing anything in screen 1.

Misc. ACL Commands

`:aclgrp username [groupname]` - control users as groups

`:acldel username` - remove user from access; detaches them automatically

`:wall message` - write messages to all windows (appears in status line)

`:multiuser off` - disallow multiuser and detach all other users

`C-a *` - display currently attached users (also works in single-user mode)

`:writelock on|off|auto` - auto gives writelock to the first user to switch to the window; on preserves writelock on switches; off gives everyone write permission

Increasing Maximum Windows

Source needs to be patched.

In `screen/config.h` the variable `MAXWIN` is defined:
`#define MAXWIN 40`

Try something a little more usable:

`#define MAXWIN 100000000000000`

Use A Persistent Shell

Keep a persistent shell open, so you don't have to type `ssh box` every time you switch back to the window.

```bash
#!/bin/bash
while [ 1 = 1 ]; do
  $@
  echo "Being persistent..."
  sleep 5
done
```

Questions and Comments?

Matt G. Habib & Ryan Curtin