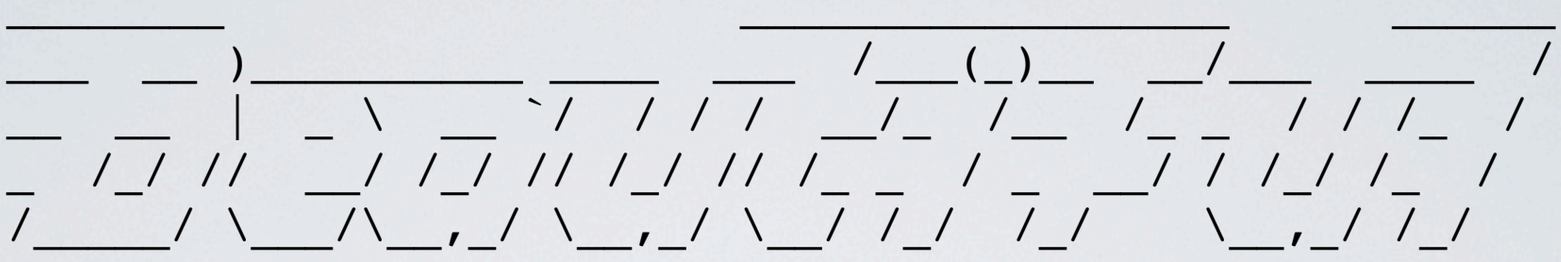


Jesse G. Donat <donatj@gmail.com>

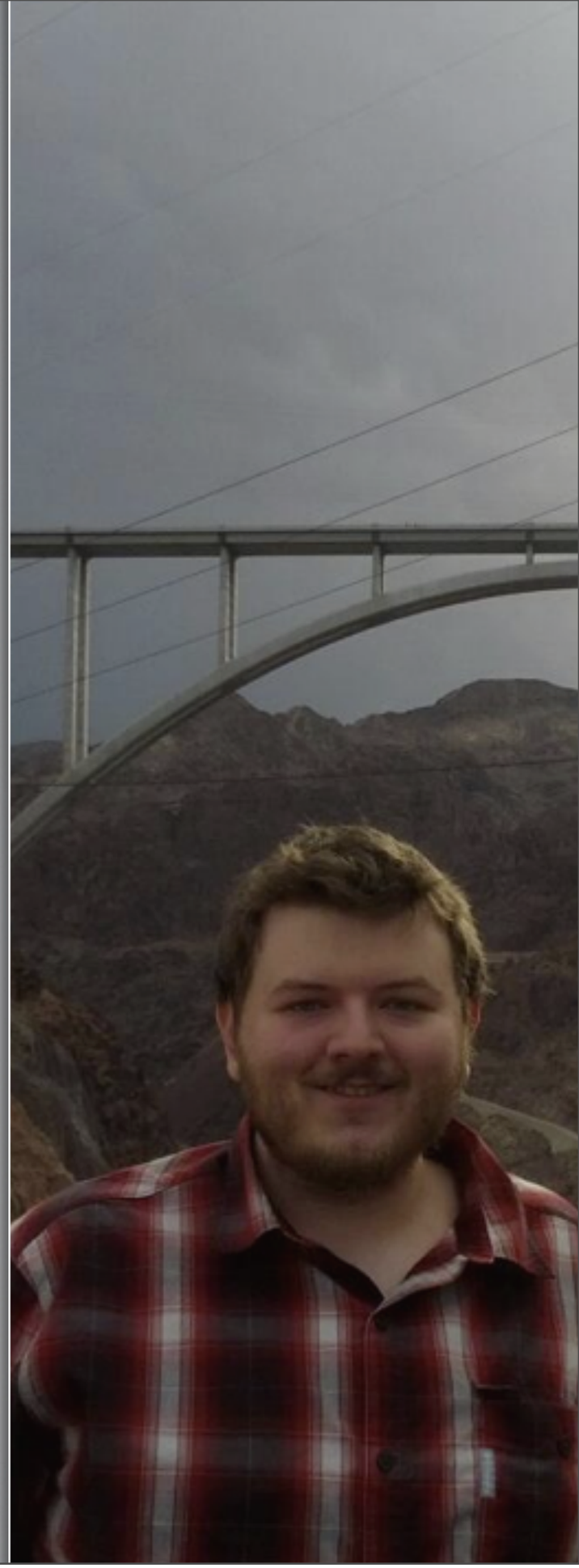
donatstudios.com

github.com/donatj

@donatj

ABOUT ME

- Programming Professional for 7 years.
- First computer was a monochrome Epson running DOS 3.22
- Started programming at age 11 (QBasic)
- Works for Capstone Digital
- Spent my 21st birthday in a Japanese Police Department



WHAT?

Using PHP as a Shell Script, while making it both useful and visually appealing.

That's What!

UNIX Like

Sorry, Windows Developers :(

High Level Skim

WHY COMMAND LINE?

Why not?

Data Processing

Long Running Processes

Tools / Utilities

Component Testing

WHY PHP?

Why not?

Familiarity

Flexibility

Ease of Use

Code Reuse

EXAMPLES

Composer

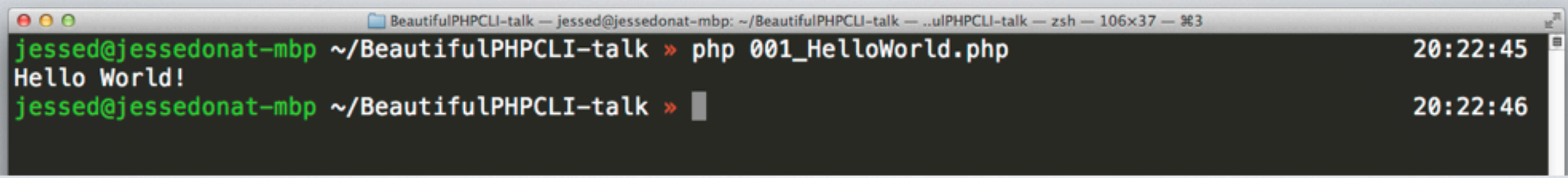
PHPUnit

phpDocumentor

Phing

HOW?

```php
<?php

echo "Hello World!";
echo PHP_EOL;
```



GOOD ENOUGH?

```php
<?php

echo "Hello World!";
echo PHP_EOL;
```

```
BeautifulPHPCLI-talk — jessed@jessedonat-mbp: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 106×37 — ⌘3
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » php 001_HelloWorld.php                20:22:45
Hello World!
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk »                                       20:22:46
```

WE CAN DO BETTER!

```php
#!/usr/bin/env php
<?php

fwrite( STDOUT, "Hello World!" );
fwrite( STDOUT, PHP_EOL );
```

```
BeautifulPHPCLI-talk — jessed@jessedonat-mbp: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 106×37 — ⌘3
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » chmod +x 002_HelloWorld.php           20:36:03
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » ./002_HelloWorld.php                  20:36:09
Hello World!
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk »                                       20:36:15
```

#! ???

- *Known as shebang*, hash-bang, pound-bang, etc
- "Interpreter Directive"
- Absolute path
  - *#!/usr/bin/php*
- *#!/usr/bin/env php*
  - More portable
  - Executes from your $PATH
- Script must be set executable
  - *chmod +x my_awesome_script.php*

OTHER NOTES

When using hash-bang, must use UNIX linefeeds

Make sure *max_execution_time* not overridden as included frameworks may reset this.

SYSTEM CALLS

- `*call*` - Backtick operator
  - Returns result as string
  - Easy, least flexible.
  - Can be difficult to spot in code.
- *shell_exec(string* ***$cmd*** *)*
  - Identical to the Backtick operator
- *exec(string* ***$command*** *[, array &****$output*** *[, int &****$return_var*** *]])*
  - Returns only the last line of the result
  - Entire output is retrieved by reference through the second parameter
  - Third parameter is the status code by reference
- *passthru(string* ***$command*** *[, int &****$return_var*** *])*
  - Sends result directly to output
  - Second parameter is the status code

PROCESS CONTROL

- proc_* php.net/manual/en/function.proc-open.php
- Good for long running external commands where you want to parse the output in real time
- Complicated to use

```php
define('BUF_SIZ',  20);     # max buffer size
define('FD_WRITE', 0);      # STDIN
define('FD_READ',  1);      # STDOUT
define('FD_ERR',   2);      # STANDARDERROR

function proc_exec($cmd) {
    $descriptorspec = array(
        0 => array("pipe", "r"),
        1 => array("pipe", "w"),
        2 => array("pipe", "w")
    );

    $ptr = proc_open($cmd, $descriptorspec, $pipes);
    if (!is_resource($ptr))
        return false;

    while (($buffer = fgets($pipes[FD_READ], BUF_SIZ)) != NULL
            || ($errbuf = fgets($pipes[FD_ERR], BUF_SIZ)) != NULL) {
        if (!isset($flag)) {
            $pstatus = proc_get_status($ptr);
            $first_exitcode = $pstatus["exitcode"];
            $flag = true;
        }
        if (strlen($buffer)) fwrite(STDOUT, $buffer);
        if (strlen($errbuf)) fwrite(STDERR, "ERR: " . $errbuf);
    }

    foreach ($pipes as $pipe) { fclose($pipe); }

    /* Get the expected *exit* code to return the value */
    $pstatus = proc_get_status($ptr);

    if (!strlen($pstatus["exitcode"]) || $pstatus["running"]) {
        /* we can trust the retval of proc_close() */
        if ($pstatus["running"])
            proc_terminate($ptr);
        $ret = proc_close($ptr);
    } else {
        if ((($first_exitcode + 256) % 256) == 255
                && (($pstatus["exitcode"] + 256) % 256) != 255)
            $ret = $pstatus["exitcode"];
        elseif (!strlen($first_exitcode))
            $ret = $pstatus["exitcode"];
        elseif ((($first_exitcode + 256) % 256) != 255)
            $ret = $first_exitcode;
        else
            $ret = 0; /* we "deduce" an EXIT_SUCCESS ;) */
        proc_close($ptr);
    }

    return ($ret + 256) % 256;
}
```

TERMINAL INFORMATION

tput makes a very good friend!

Size in columns	**tput cols**
Size in rows	tput lines
Supported Number of Colors	tput colors

```
$rows = intval(`tput lines`);
$cols = intval(`tput cols`);
echo "Your terminal is {$rows}x{$cols}!";
```

STREAMS


Processes: 150 total, 4 running, 2 stuck, 144 sleeping, 868 threads 18:40:05
Load Avg: 0.84, 0.84, 0.80 CPU usage: 3.53% user, 2.97% sys, 93.49% idle
SharedLibs: 596OK resident, 9636K data, 0B linkedit.
MemRegions: 31450 total, 2431M resident, 61M private, 745M shared.
PhysMem: 1152M wired, 4681M active, 2307M inactive, 8140M used, 44M free.
VM: 336G vsize, 1043M framework vsize, 1734755(0) pageins, 15915(0) pageouts.
Networks: packets: 3596445/2565M in, 3514593/2472M out.
Disks: 1435780/32G read, 585368/15G written.
PID COMMAND %CPU TIME #TH #WQ #POR #MREG RPRVT RSHRD RSIZE
21089 SyncServer 0.0 00:00.13 6 4 60 110 10%2K+ 12M+ 6896K
jessed@JesseDonat-MBP ~/Sites/myon ±|myon.node ✗| php server/nodes/005_book_summary_conversion.php
Book Summary Conversion
Progress 35/2900 [.....] Book 39: [########.....]
[0] 0:top* "JesseDonat-MBP.local" 18:40 03-May-12
VT220
Help

STANDARD STREAMS

Standard Output	Standard Error	Standard Input
STDOUT The main application output to the terminal. Stream most easily piped for logging or secondary processing.	*STDERR* A second output stream to the terminal. Useful for error messages as well as control codes and other you wouldn't want sent to a piped output.	*STDIN* The main input stream into a program. Generally directly from the end users keyboard unless otherwise piped.

STANDARD OUT NOTES

- ~~*echo "Hello World!";*~~
  - php://output is not necessarily standard out
  - Susceptible to output buffering
- *fwrite(STDOUT, "Hello World!");*
  - More explicit!

READING STANDARD IN

Its just like reading from a file, very slowly.

```php
fwrite( STDERR, "Are you sure (y/n): " );
if( fread( STDIN, 1 ) == 'y' ) {
    fwrite( STDOUT, "Super Cool!" );
}else{
    fwrite( STDOUT, "LAME!" );
}

fwrite( STDOUT, PHP_EOL );
```

```
BeautifulPHPCLI-talk — jessed@JesseDonat-MBP: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 71×8 — ⌘2
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk » php 007a_stdin_read.php
Are you sure (y/n): y
Super Cool!
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk » php 007a_stdin_read.php
Are you sure (y/n): n
LAME!
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk »                  1:23:14
```

```php
while( $line = fgets( STDIN ) ) {
  fwrite( STDOUT, $line );
}
```

```
BeautifulPHPCLI-talk — php 007b_stdin_read.php — php — php — 71×8 — ⌘2
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk » php 007b_stdin_read.php
I repeat
I repeat
everything typed!!!
everything typed!!!
Just like `cat`
Just like `cat`
```

```
BeautifulPHPCLI-talk — jessed@JesseDonat-MBP: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 71×9 — ⌘2
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk » echo "\nI repeat like ca
t" | php 007b_stdin_read.php

I repeat like cat
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk » echo "\nI repeat like ca
t" | cat

I repeat like cat
jessed@JesseDonat-MBP ~/BeautifulPHPCLI-talk »                  1:30:50
```

COMMAND LINE ARGUMENTS

BASIC ARGUMENTS

- Global *$argv*
- First argument is always the executed script
- Useful for simple single scripts accepting a small number of parameters

```
BeautifulPHPCLI-talk — jessed@jessedonat-mbp: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 93×9 — ⌘2
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » ./010_arguments.php "cool text" 100 -a=7
Array
(
    [0] => ./010_arguments.php
    [1] => cool text
    [2] => 100
    [3] => -a=7
)
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk »                              22:19:58
```

COMPLEX ARGUMENTS

```php
$shortopts  = "r:";  // Required value
$shortopts .= "o::"; // Optional value
$shortopts .= "abc"; // These options do not accept values

$longopts   = array(
    "reqval:",       // Has a value
    "optval::",      // Optional value
    "noval",         // No value
);
$options = getopt($shortopts, $longopts);
var_dump($options);
```

```
BeautifulPHPCLI-talk — jessed@jessedonat-mbp: ~/BeautifulPHPCLI-talk — ..ulPHPCLI-talk — zsh — 110×19 — ⌘2
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » ./011_adv_args.php --reqval 7 --noval --optval=3      22:53:10
array(3) {
  ["reqval"]=>
  string(1) "7"
  ["noval"]=>
  bool(false)
  ["optval"]=>
  string(1) "3"
}
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk » ./011_adv_args.php --reqval 7 --noval=1 --optval      22:53:14
array(3) {
  ["reqval"]=>
  string(1) "7"
  ["noval"]=>
  bool(false)
  ["optval"]=>
  bool(false)
}
jessed@jessedonat-mbp ~/BeautifulPHPCLI-talk »                                                       22:53:25
```

FURTHER INFORMATION

Other Simple Operations

Is the code running in the CLI?	`$is_cli = substr(php_sapi_name(), 0, 3) == 'cli';`
Is a stream being piped?	`$is_piped = !posix_isatty(STDOUT);`

CONTROL CODES

It's about to get good!

- More are prefixed with the ASCII "Escape" character: "\033"
- Let you control terminal behaviors and format output.
  - Cursor Control (Positioning Text)
  - Colors
  - Erasing (Screen, Lines, Etc)
  - Raise a "Bell"
  - More!
- bit.ly/controlcodes

www.termsys.demon.co.uk/vtansi.htm

CURSOR CONTROL

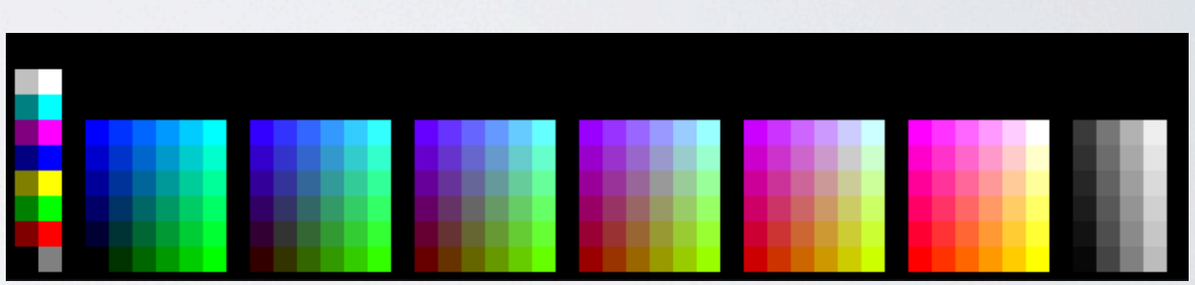
Up	`\033[{COUNT}A`
Down	`\033[{COUNT}B`
Forward (Right)	`\033[{COUNT}C`
Backwards (Left)	`\033[{COUNT}D`
Force Position	`\033[{ROW};{COLUMN}f`
Save	`\0337`
Restore	`\0338`

ERASING

Erase Screen	`\033[2J`
Current Line	`\033[2K`
Current Line UP	`\033[1J`
Current Line Down	`\033[J`
To Start of Line	`\033[1K`
To End of Line	`\033[K`

COLORS!

- Standard 8-Color Set
  - Black / Red / Green / Yellow / Blue / Magenta / Cyan / White
- Extended xterm-256 Color Set



  - Not Well Supported

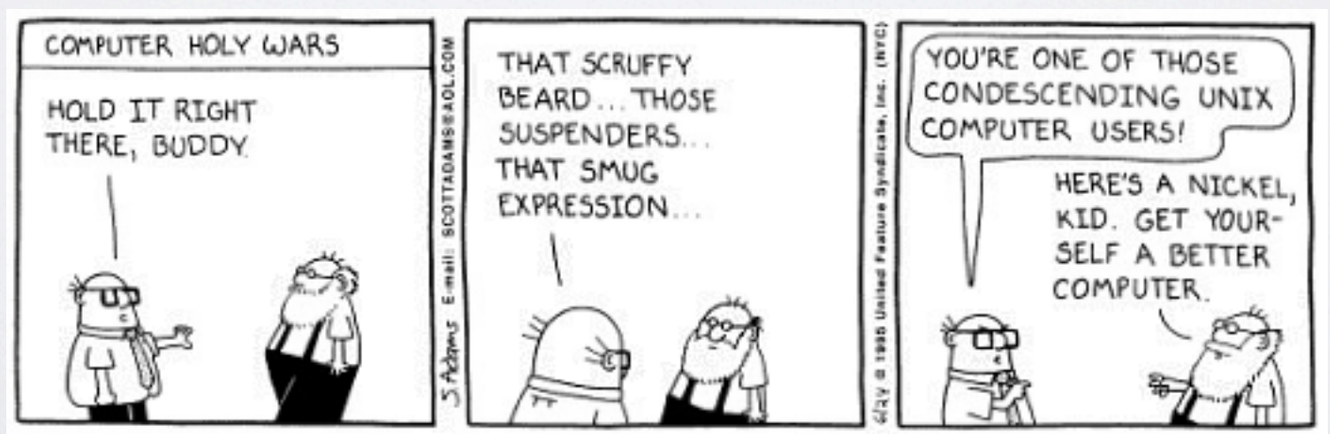
Detection!

```
#!/usr/bin/env php
<?php

$colors = max(8, intval(`tput colors`));
fwrite( STDOUT, "Terminal supports {$colors} colors!" );
fwrite( STDOUT, PHP_EOL );
```

Great for warnings (red!)

Great for separating ideas.

Hated by UNIX beards world over



STANDARD STYLING

Attribute	Code
Reset All	0
Bright (Bold)	1
Dim	2
Underline	4
Blink	5
Reverse (Inverted)	7
Hidden	8

\033[{*attributes separated by semicolon*}m

```
fwrite( STDOUT, "\033[31m"   . "Red Text"                . PHP_EOL );
fwrite( STDOUT, "\033[4m"    . "Red And Underlined Text" . PHP_EOL );
fwrite( STDOUT, "\033[0m"    . "Normal Text"             . PHP_EOL );
fwrite( STDOUT, "\033[31;4m" . "Red and Underlined Text" . PHP_EOL );
```

```
Red Text
Red And Underlined Text
Normal Text
Red and Underlined Text
```

Color	Foreground Code	Background Code
Black	30	40
Red	31	41
Green	32	42
Yellow	33	43
Blue	34	44
Magenta	35	45
Cyan	36	46
White	37	47

RESET!

BeautifulPHPCLI-talk — bash — bash — bash — 106×30 — ⌘3

```
bash-3.2$ php 004_style_examples.php
Red Text
Red And Underlined Text
Normal Text
Red and Underlined Text

bash-3.2$ I am typing in bash and it is a crying shame...
```

EOL AT THE END!

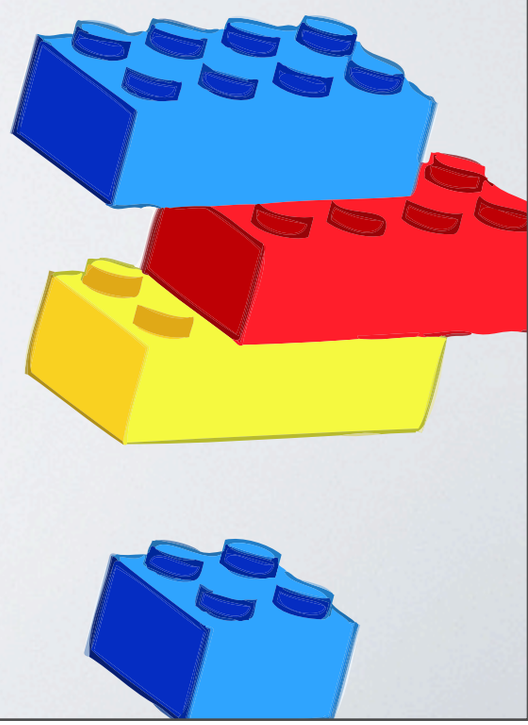
BeautifulPHPCLI-talk — bash — bash — bash — 106×30 — ⌘3

```
bash-3.2$ php 005_no_eol.php
Hello World!bash-3.2$
```

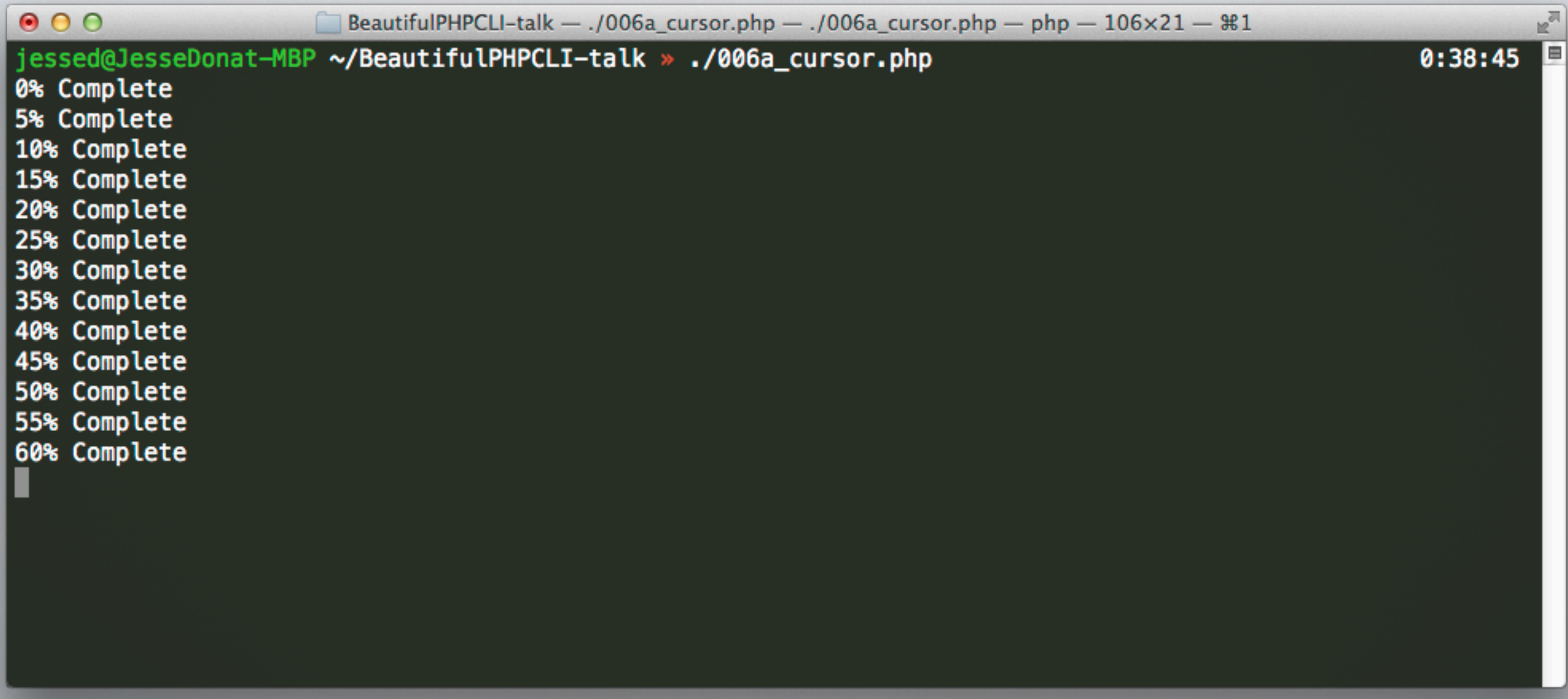
OTHER NOTES

- Best Practice: send all control codes to standard error
- "\007" is the ASCII Bell Character
- PHP errors by default output to standard out, not standard error
- Status Codes
  - Let the world outside your script know there was a problem
  - *die(1); //*non-zero implies error

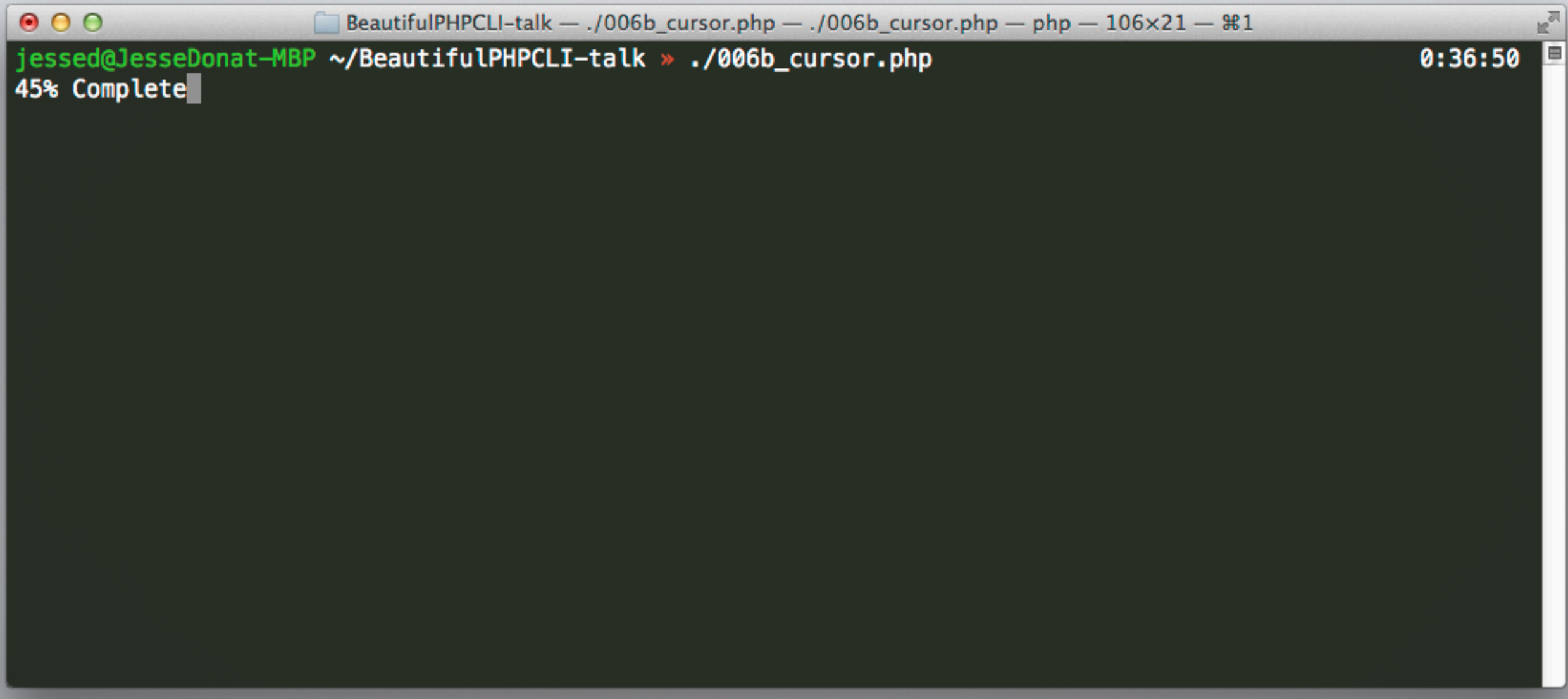
PUTTING IT TOGETHER!



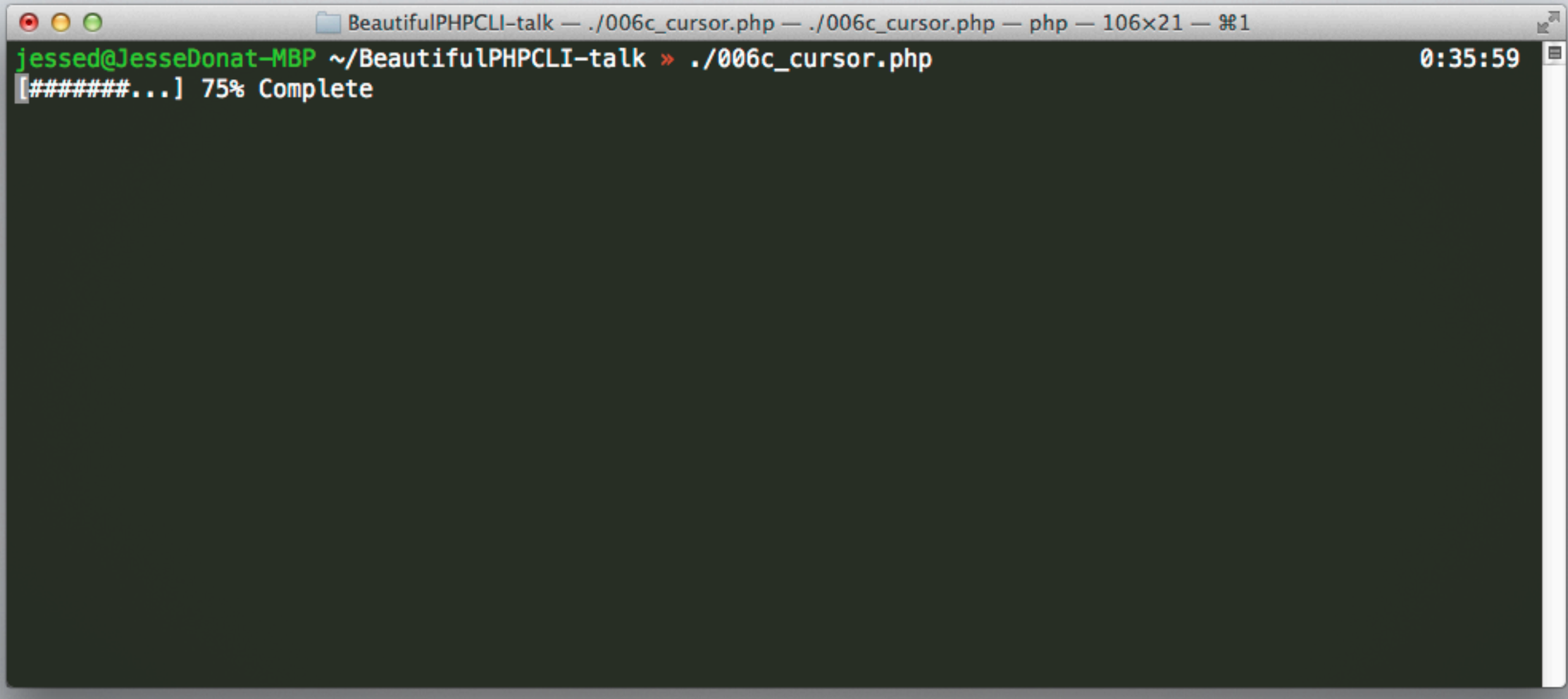
```php
for($i = 0; $i <= 100; $i += 5) {
    fwrite( STDOUT, "{$i}% Complete" . PHP_EOL );
    usleep( 100000 );
}
```



```php
fwrite( STDOUT, "\0337" ); // Save Position
for($i = 0; $i <= 100; $i += 5) {
    fwrite( STDOUT, "\0338" . "{$i}% Complete" ); // Restore Position and Write Percentage
    usleep( 100000 );
}
fwrite( STDOUT, PHP_EOL );
```

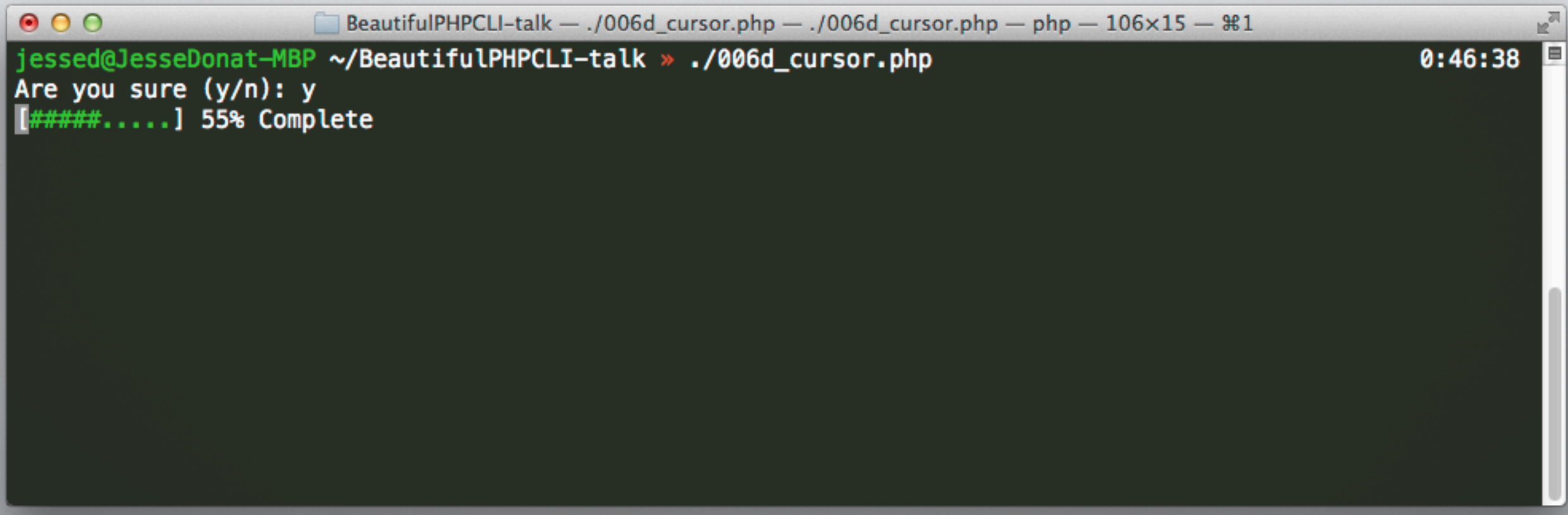


```php
fwrite( STDOUT, "\0337" ); // Save Position
for($i = 0; $i <= 100; $i += 5) {
    $step = intval($i / 10);
    fwrite( STDERR, "\0338"); // Restore Position
    fwrite( STDERR, '[' . str_repeat('#', $step) . str_repeat('.', 10 - $step) . ']'  ); // Write Progress Bar
    fwrite( STDOUT, " {$i}% Complete" . PHP_EOL );
    fwrite( STDERR, "\033[1A"); //Move up, undo the PHP_EOL
    usleep( 100000 );
}
```



ONE STEP FURTHER!

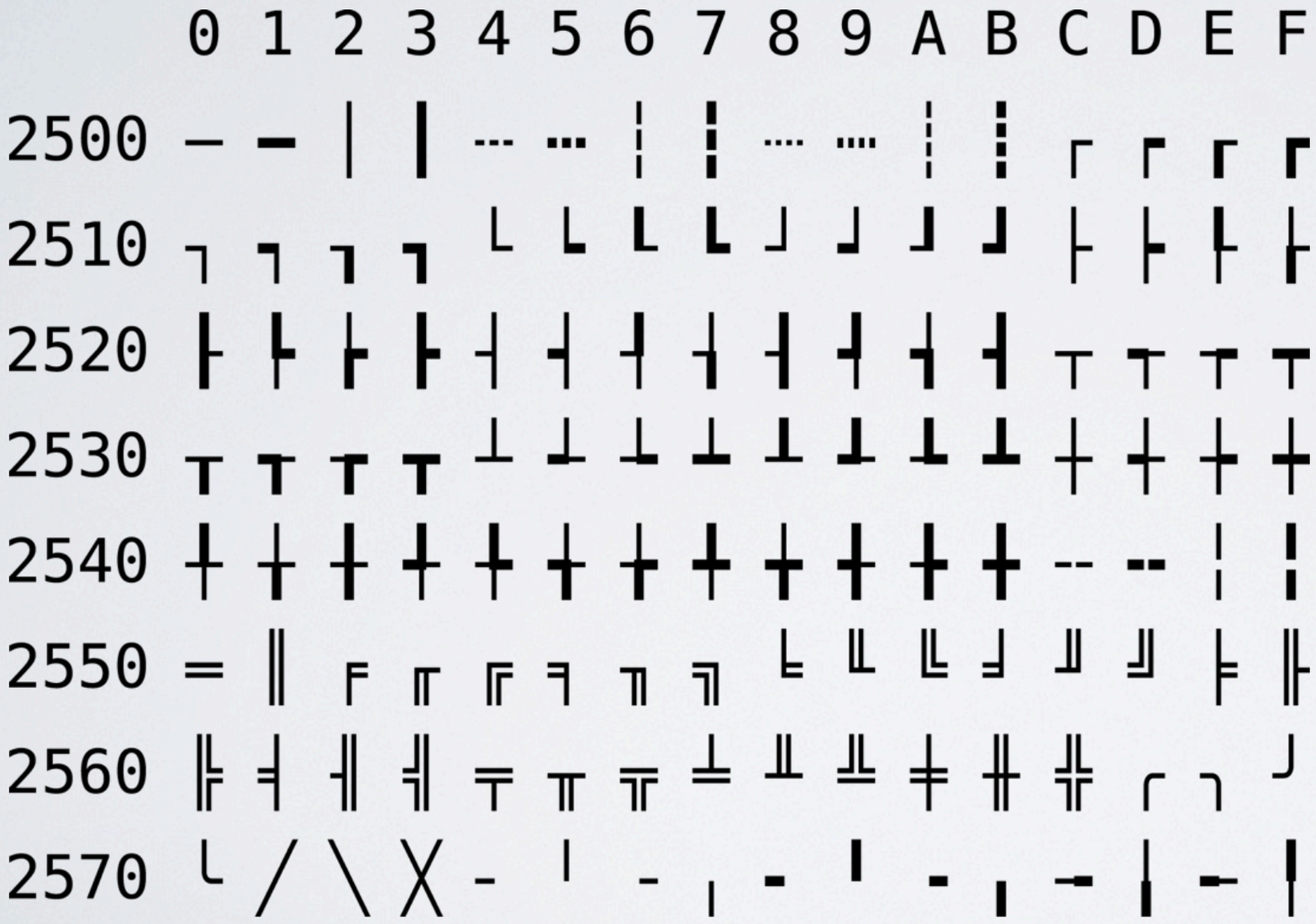
```php
fwrite( STDERR, "Are you sure (y/n): " );
if( fread( STDIN, 1 ) != 'y' ) { die(1); }

fwrite( STDOUT, "\0337" ); // Save Position
for($i = 0; $i <= 100; $i += 5) {
    $step = intval($i / 10);
    fwrite( STDERR, "\0338"); // Restore Position
    fwrite( STDERR, "[\033[32m" . str_repeat('#', $step) . str_repeat('.', 10 - $step) . "\033[0m]" ); // Write Progress Bar
    fwrite( STDOUT, " {$i}% Complete" . PHP_EOL );
    fwrite( STDERR, "\033[1A"); //Move up, undo the PHP_EOL
    usleep( 100000 );
}

fwrite( STDERR, "\007" ); //Bell on completion

fwrite( STDERR, PHP_EOL );
```

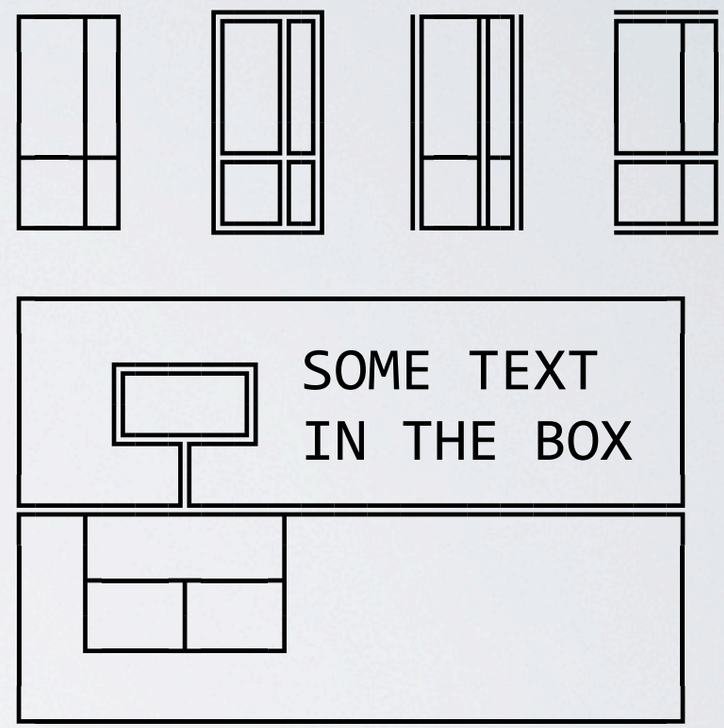


MORE PROGRESS BARS!

- CLI-Toolkit StatusGUI ProgressBar
  - **bit.ly/cli-tk** github.com/donatj/CLI-Toolkit
- PEAR Console_ProgressBar
  - **bit.ly/pear-pb** pear.php.net/package/Console_ProgressBar

UTF-8 BOX DRAWING CHARACTERS





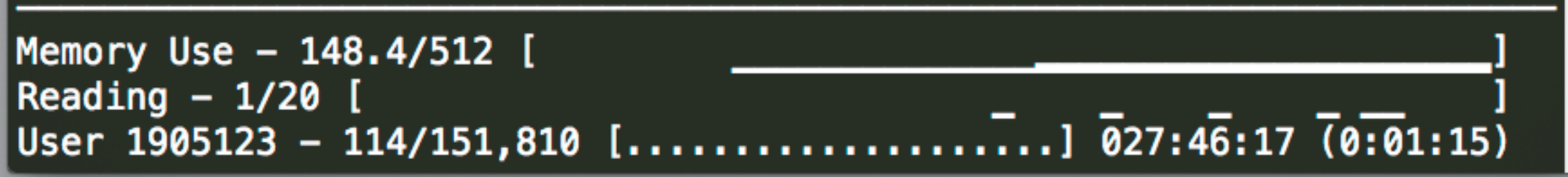
bit.ly/boxchars

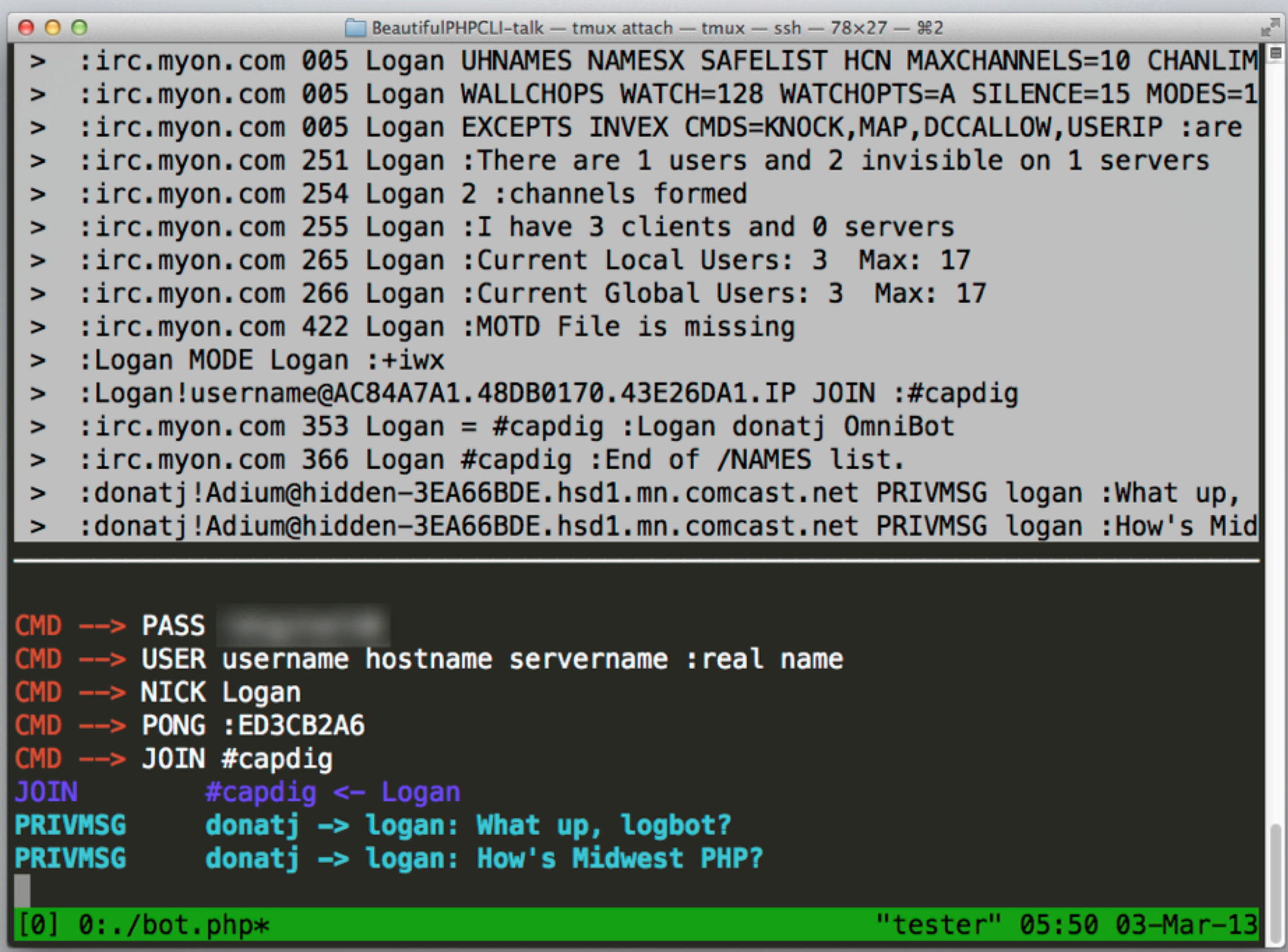
en.wikipedia.org/wiki/Box-drawing_character

EXAMPLES

myon — php server/sql/001_rollup_user_touchpath_daily_fix.php — php — php — 71×25 — ⌘3

```
Caching 78fc5773c63b46c2c0bb9e610d647a29863a0a4e 88
Caching 5c282b6c7de052777b1c06f8152ea67a7cbca143 89
Caching 9c04f53727e6614eb8230de1d75cc6300c07ad30 90
Caching 4155d22aff157193313cf682ed175913415a1d90 91
Caching ce8a2402da33da16065aa344e7956aefb2d65b31 92
Caching f25f156f02400d5b8ec208ca0fdd3e41770ef028 93
Caching 660da89c08052e94fb4c7c509862114fba32f528 94
Caching fef588774041e7c321e41d0fa2355ea14d447179 95
Caching de3fbc18bad5ba4e7a6df5f4fafd004b5dbee4f7 96
Caching 79ad32b0747d0fe9eaa641ed13ac667905cec5c3 97
Caching 8976130b2f589597f0811a8081e623780a19b5ab 98
Caching afe6977c164ea604e981739644218513e99b77de 99
Caching 71de009c00c1887961cf7a141a2969b1003c4c40 100
Caching 66ed2ab5eb765b2aee80c861040523b49be9b474 101
-----------------
Over Achiever - User 1905147 read in nodes 312967,312784
-----------------
Caching 5d1cdd46ceaa477bde42c497d56d92e6b7043604 102
Caching 554585b34b9986e9e3ecbef069c8ed92892e3c27 103
Caching feee51a119522e49dbdd60d9c56307be3517b163 104
Caching 60a76622da40b6a340bcb856e7b89e8e851dfd14 105
```





https://github.com/donatj/Pinch

THINGS TO LOOK AT

- PHAR for Redistribution
- PHP-CLI.com
- CLI Toolkit - **bit.ly/cli-tk**
  github.com/donatj/CLI-Toolkit
- Symfony 2 - **bit.ly/cli-sf**
  github.com/symfony/symfony/tree/master/src/Symfony/Component/Console

CLOSING

joind.in/8231

Code Samples: **bit.ly/bpcs-samples**

donatstudios.com

github.com/donatj

@donatj

donatj on freenode