Ruby

A Programmer's Best Friend

Ruby One-Liners

based on http://www.fepus.net/ruby1line.txt

By Nicolas Cavigneaux

http://www.bounga.org

File spacing

Double space a file

```
$ cat <file> | ruby -pe 'puts'
```

Triple space a file

```
$ cat <file> | ruby -pe '2.times {puts}'
```

Undo double-spacing

```
$ cat <file> | ruby -lne 'BEGIN{$/="\n\n"}; puts $_'
$ cat <file> | ruby -ne 'BEGIN{$/="\n\n"}; puts $_.chomp'
$ cat <file> | ruby -e 'puts STDIN.readlines.to_s.gsub(/\n\n/, "\n")'
```

Add a blank line every 5 lines

```
$ cat <file> | ruby -pe 'puts if $. % 6 == 0'
```

Numbering

Number each line (*left justified*)

```
$ cat <file> | ruby -ne 'printf("%-6s%s", $., $_)'
```

Number each line (*right justified*)

```
$ cat <file> | ruby -ne 'printf("%6s%s", $., $_)'
```

Number each line (*only print non-blank lines*)

```
$ cat <file> | ruby -e 'while gets; end; puts $.'
```

Count lines

```
$ cat <file> | ruby -ne 'END {puts $.}'
$ cat <file> | ruby -e 'while gets; end; puts $.'
```

Text conversions

converting newlines

Convert DOS newlines (*CR/LF*) to Unix format (*LF*)

```
$ cat <file> | ruby -ne 'BEGIN{$\="\n"}; print $_.chomp'
```

Convert Unix newlines (*LF*) to DOS format (*CR/LF*)

```
$ cat <file> | ruby -ne 'BEGIN{$\="\r\n"}; print $_.chomp'
```

Text conversions
deleting whitespaces

Delete leading whitespaces (*spaces*/*tabs*/*etc*) from beginning of each line

```
$ cat <file> | ruby -pe 'gsub(/^\s+/, "")'
```

Delete trailing whitespaces (*spaces*/*tabs*/*etc*) from end of each line

```
$ cat <file> | ruby -pe 'gsub(/\s+$/, $/)'
```

Delete both leading and trailing whitespaces from each line

```
$ cat <file> | ruby -pe 'gsub(/^\s+/, "").gsub(/\s+$/, $/)'
```

Text conversions
indentation

Insert 4 blank spaces at the beginning of each line

```
$ cat <file> | ruby -pe 'gsub!($_, "    #{$_}")'
```

Align all text flush right on a 79-column width

```
$ cat <file> | ruby -ne 'printf("%79s", $_)'
```

Center all text in middle of 79-column width

```
$ cat <file> | ruby -ne 'puts $_.chomp.center(79)'
$ cat <file> | ruby -lne 'puts $_.center(79)'
```

Text conversions
substitutions

Substitute (*find and replace*) "foo" with "bar" on each line

```
$ cat <file> | ruby -pe 'gsub(/foo/, "bar")'
```

Substitute "foo" with "bar" **only** for lines which contain "baz"

```
$ cat <file> | ruby -pe 'gsub(/foo/, "bar") if $_ =~ /baz/'
```

Substitute "foo" with "bar" **except** for lines which contain "baz"

```
$ cat <file> | ruby -pe 'gsub(/foo/, "bar") unless $_ =~ /baz/'
```

Substitute "foo" **or** "bar" **or** "baz" with "baq"

```
$ cat <file> | ruby -pe 'gsub(/(foo|bar|baz)/, "baq")'
```

Text conversions

reverse

Reverse order of lines

```
$ cat <file> | ruby -ne 'BEGIN{@arr=Array.new}; @arr.push $_; END
{puts @arr.reverse}'
```

Reverse each character on the line

```
$ cat <file> | ruby -ne 'puts $_.chomp.reverse'
$ cat <file> | ruby -lne 'puts $_.reverse'
```

Text conversions

join

Join pairs of lines side-by-side

```
$ cat <file> | ruby -pe '$_ = $_.chomp + " " + gets if $. % 2'
```

If a line ends with a backslash, append the next line to it

```
$ cat <file> | ruby -pe 'while $_.match(/\\$/); $_ = $_.chomp.chop + gets; end'
$ cat <file> | ruby -e 'puts STDIN.readlines.to_s.gsub(/\\\n/, "")'
```

If a line begins with an equal sign, append it to the previous line

```
$ cat <file> | ruby -e 'puts STDIN.readlines.to_s.gsub(/\n=/, "")'
```

Substitute "foo" **or** "bar" **or** "baz" with "baq"

```
$ cat <file> | ruby -pe 'gsub(/(foo|bar|baz)/, "baq")'
```

Selective printing
head / tail

Print first line of a file

```
$ cat <file> | ruby -pe 'puts $_; exit'
```

Print first 10 lines of a file

```
$ cat <file> | ruby -pe 'exit if $. > 10'
```

Print the last line of a file

```
$ cat <file> | ruby -ne 'line = $_; END {puts line}'
```

Print last 10 lines of a file

```
$ cat <file> | ruby -e 'puts STDIN.readlines.reverse!.slice
(0,10).reverse!'
```

Selective printing

regexp matching

Print only lines that match a regular expression

```
$ cat <file> | ruby -pe 'next unless $_ =~ /regexp/'
```

Print only lines that **do not** match a regular expression

```
$ cat <file> | ruby -pe 'next if $_ =~ /regexp/'
```

Print the line immediately before a regexp

```
$ cat <file> | ruby -ne 'puts @prev if $_ =~ /regex/; @prev = $_;'
```

Print the line immediately after a regexp

```
$ cat <file> | ruby -ne 'puts $_ if @prev =~ /regex/; @prev = $_;'
```

Selective printing
grep emulation

Grep for foo **and** bar **and** baz (*in any order*)

```
$ cat <file> | ruby -pe 'next unless $_ =~ /foo/ && $_ =~ /bar/ && $_
=~ /baz/'
```

Grep for foo **and** bar **and** baz (*in order*)

```
$ cat <file> | ruby -pe 'next unless $_ =~ /foo.*bar.*baz/'
```

Grep for foo **or** bar **or** baz

```
$ cat <file> | ruby -pe 'next unless $_ =~ /(foo|bar|baz)/'
```

Selective printing
paragraph printing

Print paragraph if it contains regexp

```
$ cat <file> | ruby -ne 'BEGIN{$/="\n\n"}; print $_ if $_ =~ /
regexp/'
```

Print paragraph if it contains foo **and** bar **and** baz (*in any order*)

```
$ cat <file> | ruby -ne 'BEGIN{$/="\n\n"}; print $_ if $_ =~ /foo/
&& $_ =~ /bar/ && $_ =~ /baz/'
```

Print paragraph if it contains foo **and** bar **and** baz (*in order*)

```
$ cat <file> | ruby -ne 'BEGIN{$/="\n\n"}; print $_ if $_ =~ /
(foo.*bar.*baz)/'
```

Print paragraph if it contains foo **or** bar **or** baz

```
$ cat <file> | ruby -ne 'BEGIN{$/="\n\n"}; print $_ if $_ =~ /(foo|
bar|baz)/'
```

Selective printing
line length based

Print only lines of 65 characters or greater

```
$ cat <file> | ruby -pe 'next unless $_.chomp.length >= 65'
$ cat <file> | ruby -lpe 'next unless $_.length >= 65'
```

Print only lines of 65 characters or less

```
$ cat <file> | ruby -pe 'next unless $_.chomp.length < 65'
$ cat <file> | ruby -lpe 'next unless $_.length < 65'
```

Selective printing
line numbers based

Print section of file based on line numbers (*eg. lines 2-7 inclusive*)

```
$ cat <file> | ruby -pe 'next unless $. >= 2 && $. <= 7'
```

Print line number 52

```
$ cat <file> | ruby -pe 'next unless $. == 52'
```

Print every 3rd line starting at line 4

```
$ cat <file> | ruby -pe 'next unless $. >= 4 && $. % 3 == 0'
```

Selective printing
regexp based

Print section of file from regex to end of file

```
$ cat <file> | ruby -pe '@found=true if $_ =~ /regex/; next unless
@found'
```

Print section of file between two regular expressions, /foo/ and /bar/

```
$ cat <file> | ruby -ne '@found=true if $_ =~ /foo/; next unless
@found; puts $_; exit if $_ =~ /bar/'
```

Print all the file **except** between two regular expressions, /foo/ and /bar/

```
$ cat <file> | ruby -ne '@found = true if $_ =~ /foo/; puts $_
unless @found; @found = false if $_ =~ /bar/'
```

Selective deleting

remove duplicate

Print file and remove duplicate, consecutive lines from a file

```
$ cat <file> | ruby -ne 'puts $_ unless $_ == @prev; @prev = $_'
```

Print file and remove duplicate, non-consecutive lines from a file

```
$ cat <file> | ruby -e 'puts STDIN.readlines.sort.uniq!.to_s'
```

Delete all consecutive blank lines from a file except the first

```
$ cat <file>  | ruby -e 'BEGIN{$/=nil}; puts
STDIN.readlines.to_s.gsub(/\n(\n)+/, "\n\n")'
```

Delete all leading blank lines at top of file

```
$ cat <file> | ruby -pe '@lineFound = true if $_ !~ /^\s*$/; next
if !@lineFound'
```

Selective deleting
given lines

Print file except for first 10 lines

```
$ cat <file> | ruby -pe 'next if $. <= 10'
```

Print file except for last 10 lines

```
$ cat <file> | ruby -e 'lines=STDIN.readlines; puts lines
[0,lines.size-10]'
```

Print file except for every 8th line

```
$ cat <file> | ruby -pe 'next if $. % 8 == 0'
```

Print file except for blank lines

```
$ cat <file> | ruby -pe 'next if $_ =~ /^\s*$/'
```