

Nano hacks: most useful tiny programs you've coded or come across

Asked 16 years, 2 months ago Modified 10 years, 3 months ago

Viewed 2k times



10



It's the first [great virtue](#) of programmers. All of us have, at one time or another automated a task with a bit of throw-away code. Sometimes it takes a couple seconds tapping out a one-liner, sometimes we spend an exorbitant amount of time automating away a two-second task and then never use it again.

What tiny hack have you found useful enough to **reuse**? To make go so far as to make an alias for?

Note: before answering, please check to make sure it's not already on [favourite command-line tricks using BASH](#) or perl/ruby one-liner questions.

scripting

automation

Share

edited May 23, 2017 at 12:33

Improve this question

Follow

community wiki

I'm not upvoting any answers until I see some code!
– [Chris Charabaruk](#) Oct 7, 2008 at 1:24

15 Answers

Sorted by:

Highest score (default)



i found this on dotfiles.org just today. it's very simple, but clever. i felt stupid for not having thought of it myself.

13



```
###
### Handy Extract Program
###
extract () {
    if [ -f $1 ] ; then
        case $1 in
            *.tar.bz2) tar xvjf $1 ;;
            *.tar.gz) tar xvzf $1 ;;
            *.bz2) bunzip2 $1 ;;
            *.rar) unrar x $1 ;;
            *.gz) gunzip $1 ;;
            *.tar) tar xvf $1 ;;
            *.tbz2) tar xvjf $1 ;;
            *.tgz) tar xvzf $1 ;;
            *.zip) unzip $1 ;;
            *.Z) uncompress $1 ;;
            *.7z) 7z x $1 ;;
            *) echo "'$1' cannot be
extracted via >extract<" ;;
        esac
    else
        echo "'$1' is not a valid file"
    fi
}
```

Share Improve this answer

answered Oct 7, 2008 at 2:19

Follow



joh6nn

308 ● 2 ● 5

-
- 1 Have this one in my bashrc, including '*.ace) unace x \$1 ;';
– [gnud](#) Oct 14, 2008 at 15:59
-



5



Here's a filter that puts commas in the middle of any large numbers in standard input.

```
$ cat ~/bin/comma
#!/usr/bin/perl -p

s/(\d{4,})/commify($1)/ge;

sub commify {
    local $_ = shift;
    1 while s/^( [ -+ ]? \d+ ) (\d{3}) /$1,$2/;
    return $_;
}
```

I usually wind up using it for long output lists of big numbers, and I tire of counting decimal places. Now instead of seeing

```
-rw-r--r--  1 alester alester 2244487404 Oct  6
15:38 listdetail.sql
```

I can run that as `ls -l | comma` and see

```
-rw-r--r--  1 alester alester 2,244,487,404 Oct
6 15:38 listdetail.sql
```

Share Improve this answer

answered Oct 7, 2008 at 3:17

Follow



[Andy Lester](#)

93.5k ● 15 ● 104 ● 159

Sure, ls -lh is an option, but this is a bigger issue than just directory options. – [Andy Lester](#) Oct 7, 2008 at 3:48



4



This script saved my career!

Quite a few years ago, i was working remotely on a client database. I updated a shipment to change its status. But I forgot the where clause.

I'll never forget the feeling in the pit of my stomach when I saw (6834 rows affected). I basically spent the entire night going through event logs and figuring out the proper status on all those shipments. **Crap!**

So I wrote a script (originally in awk) that would start a transaction for any updates, and check the rows affected before committing. This prevented any surprises.

So now I never do updates from command line without going through a script like this. Here it is (now in Python):

```
import sys
import subprocess as sp
pgm = "isql"
if len(sys.argv) == 1:
    print "Usage: \nsql sql-string [rows-affected]"
    sys.exit()
sql_str = sys.argv[1].upper()
```

```

max_rows_affected = 3
if len(sys.argv) > 2:
    max_rows_affected = int(sys.argv[2])

if sql_str.startswith("UPDATE"):
    sql_str = "BEGIN TRANSACTION\\n" + sql_str
    p1 = sp.Popen([pgm, sql_str], stdout=sp.PIPE,
                  shell=True)
    (stdout, stderr) = p1.communicate()
    print stdout
    # example -> (33 rows affected)
    affected = stdout.splitlines()[-1]
    affected = affected.split()[0].lstrip('(')
    num_affected = int(affected)
    if num_affected > max_rows_affected:
        print "WARNING! ", num_affected, "rows were
affected, rolling back..."
        sql_str = "ROLLBACK TRANSACTION"
        ret_code = sp.call([pgm, sql_str],
shell=True)
    else:
        sql_str = "COMMIT TRANSACTION"
        ret_code = sp.call([pgm, sql_str],
shell=True)
else:
    ret_code = sp.call([pgm, sql_str], shell=True)


```

Share Improve this answer

answered [Dec 22, 2008 at 16:12](#)

Follow

community wiki
[Dutch Masters](#)

2 That's why some genius invented **backups**. ;) – [leemes](#) Oct 16, 2012 at 21:31 



2



I use this script under assorted linuxes to check whether a directory copy between machines (or to CD/DVD) worked or whether copying (e.g. ext3 utf8 filenames -> fuseblk) has mangled special characters in the filenames.

```
#!/bin/bash
## dsum Do checksums recursively over a
directory.
## Typical usage: dsum <directory> > outfile

export LC_ALL=C # Optional - use sort order
across different locales

if [ $# != 1 ]; then echo "Usage: ${0/*\//}
<directory>" 1>&2; exit; fi
cd $1 1>&2 || exit
#findargs=-follow      # Uncomment to follow
symbolic links
find . $findargs -type f | sort | xargs -d'\n'
cksum
```

Share Improve this answer

answered Oct 7, 2008 at 1:46

Follow



Frentos

169 ● 1 ● 3



1



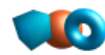
Sorry, don't have the exact code handy, but I coded a regular expression for searching source code in VS.Net that allowed me to search anything not in comments. It came in very useful in a particular project I was working on, where people insisted that commenting out code was good practice, in case you wanted to go back and see what the code used to do.



Share Improve this answer

answered Oct 7, 2008 at 0:51

Follow



Kibbee

66.1k ● 28 ● 144 ● 184



1



I have two ruby scripts that I modify regularly to download all of various webcomics. Extremely handy! *Note: They require wget, so probably linux. Note2: read these before you try them, they need a little bit of modification for each site.*



Date based downloader:



```
#!/usr/bin/ruby -w

Day = 60 * 60 * 24

Fromat = "hjlsdahjsd/comics/st%Y%m%d.gif"

t = Time.local(2005, 2, 5)

MWF = [1,3,5]

until t == Time.local(2007, 7, 9)
  if MWF.include? t.wday
    `wget #{t.strftime(Fromat)}`
    sleep 3
  end

  t += Day
end
```

Or you can use the number based one:

```
#!/usr/bin/ruby -w
```

```
Format = "http://fdsafdsa/comics/%08d.gif"
1.upto(986) do |i|
  `wget #{sprintf(Format, i)}`
  sleep 1
end
```

Share Improve this answer

answered Oct 7, 2008 at 1:45

Follow



[Frew Schmidt](#)

9,544 ● 17 ● 66 ● 88



1



Instead of having to repeatedly open files in SQL Query Analyser and run them, I found the syntax needed to make a batch file, and could then run 100 at once. Oh the sweet sweet joy! I've used this ever since.

```
isqlw -S servername -d dbname -E -i
F:\blah\whatever.sql -o F:\results.txt
```

Share Improve this answer

answered Oct 14, 2008 at 15:46

Follow



[Magnus Smith](#)

5,963 ● 8 ● 45 ● 65



0



This goes back to my COBOL days but I had two generic COBOL programs, one batch and one online (mainframe folks will know what these are). They were shells of a program that could take any set of parameters and/or files and be run, batch or executed in an IMS test region. I had them set up so that depending on the parameters I could access files, databases(DB2 or IMS DB) and or just manipulate working storage or whatever.

It was great because I could test that date function without guessing or test why there was truncation or why there was a database ABEND. The programs grew in size as time went on to include all sorts of tests and become a staple of the development group. Everyone knew where the code resided and included them in their unit testing as well. Those programs got so large (most of the code were commented out tests) and it was all contributed by people through the years. They saved so much time and settled so many disagreements!

Share Improve this answer

answered Oct 7, 2008 at 0:56

Follow



Taptronic

5,150 ● 9 ● 46 ● 59



0

I coded a Perl script to map dependencies, without going into an endless loop, For a legacy C program I inherited that also had a diamond dependency problem.



I wrote small program that e-mailed me when I received e-mails from friends, on an rarely used e-mail account.



I wrote another small program that sent me text messages if my home IP changes.

To name a few.

Share Improve this answer

answered Oct 7, 2008 at 1:03

Follow



J.J.

4,892 ● 1 ● 26 ● 29



0

Years ago I built a suite of applications on a custom web application platform in PERL. One cool feature was to convert SQL query strings into human readable sentences that described what the results were.



The code was relatively short but the end effect was nice.



[Share](#) [Improve this answer](#)

answered Oct 7, 2008 at 1:11

[Follow](#)



Patrick Manderson



0

I've got a little app that you run and it dumps a GUID into the clipboard. You can run it /noui or not. With UI, its a single button that drops a new GUID every time you click it. Without it drops a new one and then exits.



I mostly use it from within VS. I have it as an external app and mapped to a shortcut. I'm writing an app that relies heavily on xaml and guides, so I always find I need to paste a new guid into xaml...



[Share](#) [Improve this answer](#)

answered Oct 7, 2008 at 1:26

[Follow](#)



user1228



0

Any time I write a clever list comprehension or use of map/reduce in python. There was one like this:

```
if reduce(lambda x, c: locks[x] and c, locknames, True):
```



```
print "Sub-threads terminated!"
```



The reason I remember that is that I came up with it myself, then saw the exact same code on somebody else's website. Now-adays it'd probably be done like:

```
if all(map(lambda z: locks[z], locknames)):  
    print "ya trik"
```

[Share](#) [Improve this answer](#)

answered Oct 7, 2008 at 1:52

[Follow](#)



Claudiu

229k ● 173 ● 503 ● 697



0



I've got 20 or 30 of these things lying around because once I coded up the framework for my standard console app in windows I can pretty much drop in any logic I want, so I got a lot of these little things that solve specific problems.

I guess the ones I'm using a lot right now is a console app that takes stdin and colorizes the output based on xml profiles that match regular expressions to colors. I use it for watching my log files from builds. The other one is a command line launcher so I don't pollute my PATH env var and it would exceed the limit on some systems anyway, namely win2k.

[Share](#) [Improve this answer](#)

[Follow](#)

answered Oct 7, 2008 at 3:49



[Scott Dillman](#)

831 ● 1 ● 9 ● 16



0



I'm constantly connecting to various linux servers from my own desktop throughout my workday, so I created a few aliases that will launch an `xterm` on those machines and set the title, background color, and other tweaks:

```
alias x="xterm"          # local
alias xd="ssh -Xf me@development_host xterm -bg
aliceblue -ls -sb -bc -geometry 100x30 -title
Development"
alias xp="ssh -Xf me@production_host xterm -bg
thistle1 ..."
```

[Share](#) [Improve this answer](#)

community wiki
[Adam Liss](#)



0



I have a bunch of servers I frequently connect to, as well, but they're all on my local network. This Ruby script prints out the command to create aliases for any machine with ssh open:

```
#!/usr/bin/env ruby

require 'rubygems'
require 'dnssd'

handle = DNSSD.browse('_ssh._tcp') do |reply|
  print "alias #{reply.name}='ssh #{reply.name}.#\n#{reply.domain}';"
end

sleep 1
handle.stop
```

Use it like this in your `.bash_profile`:

```
eval `ruby ~/.alias_shares`
```

Share Improve this answer

edited Nov 7, 2008 at 2:13

Follow

community wiki

