**NAME**

**cal** — displays a calendar

**SYNOPSIS**

**cal** [ **-mjy13**][[*month*] *year*]

**DESCRIPTION**

**Cal** displays a simple calendar. If arguments are not specified, the current month is displayed. The options are as follows:

**-1** Display single month output (use if cal was built with -3 as default to get older traditional output)

**-3** Display prev/current/next month output (use if cal was built with traditional -1 as default to get newer improved output)

**-m** Display Monday as the first day of the week. (The default is Sunday.)

**-j** Display Julian dates (days one-based, numbered from January 1).

**-y** Display a calendar for the current year.

A single parameter specifies the year (1 - 9999) to be displayed; note the year must be fully specified: "`cal 89`" will *not* display a calendar for 1989. Two parameters denote the month (1 - 12) and year. If no parameters are specified, the current month's calendar is displayed.

A year starts on Jan 1.

The Gregorian Reformation is assumed to have occurred in 1752 on the 3rd of September. By this time, most countries had recognized the reformation (although a few did not recognize it until the early 1900's.) Ten days following that date were eliminated by the reformation, so the calendar for that month is a bit unusual.

**HISTORY**

A **cal** command appeared in Version 6 AT&T UNIX.

## NAME

**col** — filter reverse line feeds from input

## SYNOPSIS

**col** [ **-bfx** ] [ **-l** *num* ]

## DESCRIPTION

**Col** filters out reverse (and half reverse) line feeds so that the output is in the correct order with only forward and half forward line feeds, and replaces white-space characters with tabs where possible. This can be useful in processing the output of nroff(1) and tbl(1).

**Col** reads from the standard input and writes to the standard output.

The options are as follows:

**-b** Do not output any backspaces, printing only the last character written to each column position.

**-f** Forward half line feeds are permitted (‘‘fine’’ mode). Normally characters printed on a half line boundary are printed on the following line.

**-x** Output multiple spaces instead of tabs.

**-l** *num* Buffer at least *num* lines in memory. By default, 128 lines are buffered.

The control sequences for carriage motion that **col** understands and their decimal values are listed in the following table:

| | |
|---|---|
| ESC−7 | reverse line feed (escape then 7) |
| ESC−8 | half reverse line feed (escape then 8) |
| ESC−9 | half forward line feed (escape then 9) |
| backspace | moves back one column (8); ignored in the first column |
| carriage return | (13) |
| newline | forward line feed (10); also does carriage return |
| shift in | shift to normal character set (15) |
| shift out | shift to alternate character set (14) |
| space | moves forward one column (32) |
| tab | moves forward to next tab stop (9) |
| vertical tab | reverse line feed (11) |

All unrecognized control characters and escape sequences are discarded.

**Col** keeps track of the character set as characters are read and makes sure the character set is correct when they are output.

If the input attempts to back up to the last flushed line, **col** will display a warning message.

## SEE ALSO

expand(1), nroff(1), tbl(1)

## HISTORY

A **col** command appeared in Version 6 AT&T UNIX.

**NAME**

**colcrt** — filter nroff output for CRT previewing

**SYNOPSIS**

**colcrt** [ **-** ] [ **-2** ] [ *file . . .* ]

**DESCRIPTION**

**Colcrt** provides virtual half-line and reverse line feed sequences for terminals without such capability, and on which overstriking is destructive. Half-line characters and underlining (changed to dashing '−') are placed on new lines in between the normal output lines.

Available options:

**-** Suppress all underlining. This option is especially useful for previewing *allboxed* tables from tbl(1).

**-2** Causes all half-lines to be printed, effectively double spacing the output. Normally, a minimal space output format is used which will suppress empty lines. The program never suppresses two consecutive empty lines, however. The **-2** option is useful for sending output to the line printer when the output contains superscripts and subscripts which would otherwise be invisible.

**EXAMPLES**

A typical use of **colcrt** would be

```
tbl exum2.n | nroff −ms | colcrt − | more
```

**SEE ALSO**

nroff(1), troff(1), col(1), more(1), ul(1)

**BUGS**

Should fold underlines onto blanks even with the '**-**' option so that a true underline character would show.

Can't back up more than 102 lines.

General overstriking is lost; as a special case '**|**' overstruck with '−' or underline becomes '+'.

Lines are trimmed to 132 characters.

Some provision should be made for processing superscripts and subscripts in documents which are already double-spaced.

**HISTORY**

The **colcrt** command appeared in 3.0BSD.

**NAME**

**colrm** — remove columns from a file

**SYNOPSIS**

**colrm** [*startcol* [*endcol*]]

**DESCRIPTION**

**Colrm** removes selected columns from a file. Input is taken from standard input. Output is sent to standard output.

If called with one parameter the columns of each line will be removed starting with the specified column. If called with two parameters the columns from the first column to the last column will be removed.

Column numbering starts with column 1.

**SEE ALSO**

awk(1), column(1), expand(1), paste(1)

**HISTORY**

The **colrm** command appeared in 3.0BSD.

**NAME**

**column** — columnate lists

**SYNOPSIS**

**column** [ **-tx** ] [ **-c** *columns* ] [ **-s** *sep* ] [ *file . . .* ]

**DESCRIPTION**

The **column** utility formats its input into multiple columns. Rows are filled before columns. Input is taken from *file* operands, or, by default, from the standard input. Empty lines are ignored.

The options are as follows:

**-c** Output is formatted for a display *columns* wide.

**-s** Specify a set of characters to be used to delimit columns for the **-t** option.

**-t** Determine the number of columns the input contains and create a table. Columns are delimited with whitespace, by default, or with the characters supplied using the **-s** option. Useful for pretty-printing displays.

**-x** Fill columns before filling rows.

**Column** exits 0 on success, >0 if an error occurred.

**ENVIRONMENT**

COLUMNS The environment variable COLUMNS is used to determine the size of the screen if no other information is available.

**EXAMPLES**

```
(printf "PERM LINKS OWNER GROUP SIZE MONTH DAY
HH:MM/YEAR NAME\n" \
; ls -l | sed 1d) | column -t
```

**SEE ALSO**

colrm(1), ls(1), paste(1), sort(1)

**HISTORY**

The **column** command appeared in 4.3BSD−Reno.

## NAME

cygstart − start a program or open a file or URL

## SYNOPSIS

**cygstart** [−**oxefp?**] [−**a** *action*] [−**d** *directory*] [−−**hide**] [−−**maximize**] [−−**minimize**] [−−**restore**] [−−**show**] [−−**showmaximized**] [−−**showminimized**] [−−**showminnoactive**] [−−**showna**] [−−**shownoactivate**] [−−**shownormal**] [−−**usage**] [−−**version**] [−−**license**] [−−**reference**] *file* [*arguments*]

## DESCRIPTION

**cygstart** is a command-line tool which allows you to let Windows start a program or open a file or URL in its associated application. It is similar to the Windows command-line **start** command.

## OPTIONS

*Action options*

**−a**, **−−action**=*STRING*
Use specified action instead of default

**−o**, **−−open**
Short for: **−−action** *open*

**−x**, **−−explore**
Short for: **−−action** *explore*

**−e**, **−−edit**
Short for: **−−action** *edit*

**−f**, **−−find**
Short for: **−−action** *find*

**−p**, **−−print**
Short for: **−−action** *print*

*Directory options*

**−d**, **−−directory**=*STRING*
Set working directory

*Show options*

**−−hide** Hides the window and activates another window

**−−maximize**
Maximizes the specified window

**−−minimize**
Minimizes the specified window and activates the next top-level window in the z-order

**−−restore**
Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window

**−−show**
Activates the window and displays it in its current size and position

**−−showmaximized**
Activates the window and displays it as a maximized window

**−−showminimized**
Activates the window and displays it as a minimized window

**−−showminnoactive**
Displays the window as a minimized window. The active window remains active

**−−showna**
Displays the window in its current state. The active window remains active

**−−shownoactivate**
Displays a window in its most recent size and position. The active window remains active

**−−shownormal**
Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time

*Help options*

**−?**, **−−help**
Show this help message

**−−usage**
Display brief usage message

**−−version**
Display version information

**−−license**
Display licensing information

**−−reference**
Open MSDN reference for ShellExecute

## EXAMPLES

Start Bash in a new window

**$ cygstart bash**

Open the Cygwin website in your default browser:

**$ cygstart http://www.cygwin.com**

Print a text file

**$ cygstart --print README.txt**

Open a Word document in a maximized window

**$ cygstart --maximize ~/projects/whatever/design.doc**

## COPYRIGHT



## AUTHOR

Michael Schaap <cygstart(at)mscha.org>

## SEE ALSO

**<http://msdn.microsoft.com/library/en-us/shellcc/platform/Shell/reference/functions/shellexecute.asp>**

## NAME

ddate − converts Gregorian dates to Discordian dates

## SYNOPSIS

**ddate [+format] [date]**

## DESCRIPTION

**ddate** prints the date in Discordian date format.

If called with no arguments, **ddate** will get the current system date, convert this to the Discordian date format and print this on the standard output. Alternatively, a Gregorian date may be specified on the command line, in the form of a numerical day, month and year.

If a format string is specified, the Discordian date will be printed in a format specified by the string. This mechanism works similarly to the format string mechanism of **date(1),** only almost completely differently. The fields are:

%A Full name of the day of the week (i.e., Sweetmorn)

%a Abbreviated name of the day of the week (i.e., SM)

%B Full name of the season (i.e., Chaos)

%b Abbreviated name of the season (i.e., Chs)

%d Ordinal number of day in season (i.e., 23)

%e Cardinal number of day in season (i.e., 23rd)

%H Name of current Holyday, if any

%N Magic code to prevent rest of format from being printed unless today is a Holyday.

%n Newline

%t Tab

%X Number of days remaining until X-Day. (Not valid if the SubGenius options are not compiled in.)

%{

%} Used to enclose the part of the string which is to be replaced with the words "St. Tib’s Day" if the current day is St. Tib’s Day.

%. Try it and see.

## EXAMPLES

```
% ddate
Sweetmorn, Bureaucracy 42, 3161 YOLD

% ddate +'Today is %{%A, the %e of %B%}, %Y. %N%nCelebrate %H'
Today is Sweetmorn, the 42nd of Bureaucracy, 3161.

% ddate +"It's %{%A, the %e of %B%}, %Y. %N%nCelebrate %H" 26 9 1995
It's Prickle-Prickle, the 50th of Bureaucracy, 3161.
Celebrate Bureflux

% ddate +'Today's %{%A, the %e of %B%}, %Y. %N%nCelebrate %H' 29 2 1996
Today's St. Tib's Day, 3162.
```

## BUGS

**ddate(1)** will produce undefined behaviour if asked to produce the date for St. Tib's day and its format string does not contain the St. Tib's Day delimiters %{ and %}.

## NOTE

After 'X-Day' passed without incident, the Church of the SubGenius declared that it had got the year upside down - X-Day is actually in 8661 AD rather than 1998 AD. Thus, the True X-Day is Cfn 40, 9827.

## AUTHOR

Original program by Druel the Chaotic aka Jeremy Johnson (mpython@gnu.ai.mit.edu)
Major rewrite by Lee H:. O:. Smith, KYTP, aka Andrew Bulhak (acb@dev.null.org)
Five tons of flax.

## DISTRIBUTION POLICY



## SEE ALSO

date(1),
http://www.subgenius.com/
Malaclypse the Younger, *Principia Discordia, Or How I Found Goddess And What I Did To Her When I Found Her*

## NAME

getopt − parse command options (enhanced)

## SYNOPSIS

**getopt** optstring parameters **getopt** [options] [--] optstring parameters **getopt** [options] **-o**|**--options** optstring [options] [--] parameters

## DESCRIPTION

**getopt** is used to break up (*parse*) options in command lines for easy parsing by shell procedures, and to check for legal options. It uses the GNU **getopt**(3) routines to do this. The parameters **getopt** is called with can be divided into two parts: options which modify the way getopt will parse (*options* and *-o|--options optstring* in the **SYNOPSIS),** and the parameters which are to be parsed (*parameters* in the **SYNOPSIS).** The second part will start at the first non-option parameter that is not an option argument, or after the first occurence of '**--**'. If no '**-o**' or '**--options**' option is found in the first part, the first parameter of the second part is used as the short options string. If the environment variable **GETOPT_COMPATIBLE** is set, or if its first parameter is not an option (does not start with a '**-**', this is the first format in the **SYNOPSIS), getopt** will generate output that is compatible with that of other versions of **getopt**(1). It will still do parameter shuffling and recognize optional arguments (see section **COMPATIBILITY** for more information). Traditional implementations of **getopt**(1) are unable to cope with whitespace and other (shell-specific) special characters in arguments and non-option parameters. To solve this problem, this implementation can generate quoted output which must once again be interpreted by the shell (usually by using the **eval** command). This has the effect of preserving those characters, but you must call **getopt** in a way that is no longer compatible with other versions (the second or third format in the **SYNOPSIS).** To determine whether this enhanced version of **getopt**(1) is installed, a special test option (**-T**) can be used.

## OPTIONS

-a, --alternative

Allow long options to start with a single '**-**'.

-h, --help

Output a small usage guide and exit succesfully. No other output is generated.

-l, --longoptions longopts

The long (multi-character) options to be recognized. More than one option name may be specified at once, by separating the names with commas. This option may be given more than once, the *longopts* are cummulative. Each long option name in *longopts* may be followed by one colon to indicate it has a required argument,and by two colons to indicate it has an optional argument.

-n, --name progname

The name that will be used by the **getopt**(3) routines when it reports errors. Note that errors of **getopt**(1) are still reported as coming from getopt.

-o, --options shortopts

The short (one-character) options to be recognized. If this options is not found, the first parameter of **getopt** that does not start with a '**-**' (and is not an option argument) is used as the short options string. Each short option character in *shortopts* may be followed by one colon to indicate it has a required argument, and by two colons to indicate it has an optional argument. The first character of shortopts may be '**+**' or '**-**' to influence the way options are parsed and output is generated (see section **SCANNING MODES** for details).

-q, --quiet

Disable error reporting by getopt(3).

-Q, --quiet-output

Do not generate normal output. Errors are still reported by **getopt**(3), unless you also use *-q*.

-s, --shell shell

Set quoting conventions to those of shell. If no -s argument is found, the BASH conventions are used. Valid arguments are currently '**sh**' '**bash**', '**csh**', and '**tcsh**'.

-u, --unquoted

Do not quote the output. Note that whitespace and special (shell-dependent) characters can cause havoc in this mode (like they do with other **getopt**(1) implementations).

-T --test

Test if your **getopt**(1) is this enhanced version or an old version. This generates no output, and sets the error status to 4. Other implementations of **getopt**(1), and this version if the environment variable **GETOPT_COMPATIBLE** is set, will return '--' and error status 0.

-V, --version

Output version information and exit succesfully. No other output is generated.

## PARSING

This section specifies the format of the second part of the parameters of **getopt** (the *parameters* in the **SYNOPSIS**). The next section (**OUTPUT**) describes the output that is generated. These parameters were typically the parameters a shell function was called with. Care must be taken that each parameter the shell function was called with corresponds to exactly one parameter in the parameter list of **getopt** (see the **EXAMPLES**). All parsing is done by the GNU **getopt**(3) routines. The parameters are parsed from left to right. Each parameter is classified as a short option, a long option, an argument to an option, or a non-option parameter. A simple short option is a '-' followed by a short option character. If the option has a required argument, it may be written directly after the option character or as the next parameter (ie. separated by whitespace on the command line). If the option has an optional argument, it must be written directly after the option character if present. It is possible to specify several short options after one '-', as long as all (except possibly the last) do not have required or optional arguments. A long option normally begins with '--' followed by the long option name. If the option has a required argument, it may be written directly after the long option name, separated by '=', or as the next argument (ie. separated by whitespace on the command line). If the option has an optional argument, it must be written directly after the long option name, separated by '=', if present (if you add the '=' but nothing behind it, it is interpreted as if no argument was present; this is a slight bug, see the **BUGS**). Long options may be abbreviated, as long as the abbreviation is not ambiguous. Each parameter not starting with a '-', and not a required argument of a previous option, is a non-option parameter. Each parameter after a '--' parameter is always interpreted as a non-option parameter. If the environment variable **POSIXLY_CORRECT** is set, or if the short option string started with a '+', all remaining parameters are interpreted as non-option parameters as soon as the first non-option parameter is found.

## OUTPUT

Output is generated for each element described in the previous section. Output is done in the same order as the elements are specified in the input, except for non-option parameters. Output can be done in *compatible* (*unquoted*) mode, or in such way that whitespace and other special characters within arguments and non-option parameters are preserved (see **QUOTING**). When the output is processed in the shell script, it will seem to be composed of distinct elements that can be processed one by one (by using the shift command in most shell languages). This is imperfect in unquoted mode, as elements can be split at unexpected places if they contain whitespace or special characters. If there are problems parsing the parameters, for example because a required argument is not found or an option is not recognized, an error will be reported on stderr, there will be no output for the offending element, and a non-zero error status is returned. For a short option, a single '-' and the option character are generated as one parameter. If the option has an argument, the next parameter will be the argument. If the option takes an optional argument, but none was found, the next parameter will be generated but be empty in quoting mode, but no second parameter will be generated in unquoted (compatible) mode. Note that many other **getopt**(1) implemetations do not support optional arguments. If several short options were specified after a single '-', each will be present in the output as a separate parameter. For a long option, '--' and the full option name are generated as one parameter. This is done regardless whether the option was abbreviated or specified with a single '-' in the input. Arguments are handled as with short options. Normally, no non-option parameters output is generated until all options and their arguments have been generated. Then '--' is generated as a single parameter, and after it the non-option parameters in the order they were found, each as a separate parameter. Only if the first character of the short options string was a '-', non-option parameter output is generated at the place they are found in the input (this is not supported if the first format of the **SYNOPSIS** is used; in that case all preceding occurences of '-' and '+' are ignored).

## QUOTING

In compatible mode, whitespace or 'special' characters in arguments or non-option parameters are not handled correctly. As the output is fed to the shell script, the script does not know how it is supposed to break the output into separate parameters. To circumvent this problem, this implementation offers quoting. The idea is that output is generated with quotes around each parameter. When this output is once again fed to the shell (usually by a shell **eval** command), it is split correctly into separate parameters. Quoting is not enabled if the environment variable **GETOPT_COMPATIBLE** is set, if the first form of the **SYNOPSIS** is used, or if the option '**-u**' is found. Different shells use different quoting conventions. You can use the '**-s**' option to select the shell you are using. The following shells are currently supported: '**sh**', '**bash**', '**csh**' and '**tcsh**'. Actually, only two 'flavors' are distinguished: sh-like quoting conventions and csh-like quoting conventions. Chances are that if you use another shell script language, one of these flavors can still be used.

## SCANNING MODES

The first character of the short options string may be a '**-**' or a '**+**' to indicate a special scanning mode. If the first calling form in the **SYNOPSIS** is used they are ignored; the environment variable **POSIXLY_CORRECT** is still examined, though. If the first character is '**+**', or if the environment variable **POSIXLY_CORRECT** is set, parsing stops as soon as the first non-option parameter (ie. a parameter that does not start with a '**-**') is found that is not an option argument. The remaining parameters are all interpreted as non-option parameters. If the first character is a '**-**', non-option parameters are outputed at the place where they are found; in normal operation, they are all collected at the end of output after a '**--**' parameter has been generated. Note that this '**--**' parameter is still generated, but it will always be the last parameter in this mode.

## COMPATIBILITY

This version of **getopt**(1) is written to be as compatible as possible to other versions. Usually you can just replace them with this version without any modifications, and with some advantages. If the first character of the first parameter of getopt is not a '**-**', getopt goes into compatibility mode. It will interpret its first parameter as the string of short options, and all other arguments will be parsed. It will still do parameter shuffling (ie. all non-option parameters are outputed at the end), unless the environment variable **POSIXLY_CORRECT** is set. The environment variable **GETOPT_COMPATIBLE** forces **getopt** into compatibility mode. Setting both this environment variable and **POSIXLY_CORRECT** offers 100% compatibility for 'difficult' programs. Usually, though, neither is needed. In compatibility mode, leading '**-**' and '**+**' characters in the short options string are ignored.

## RETURN CODES

**getopt** returns error code **0** for succesful parsing, **1** if **getopt**(3) returns errors, **2** if it does not understand its own parameters, **3** if an internal error occurs like out-of-memory, and **4** if it is called with **-T**.

## EXAMPLES

Example scripts for (ba)sh and (t)csh are provided with the **getopt**(1) distribution, and are optionally installed in **/usr/local/lib/getopt** or **/usr/lib/getopt**.

## ENVIRONMENT

POSIXLY_CORRECT

This environment variable is examined by the **getopt**(3) routines. If it is set, parsing stops as soon as a parameter is found that is not an option or an option argument. All remaining parameters are also interpreted as non-option parameters, regardless whether they start with a '**-**'.

GETOPT_COMPATIBLE

Forces **getopt** to use the first calling format as specified in the **SYNOPSIS**.

## BUGS

**getopt**(3) can parse long options with optional arguments that are given an empty optional argument (but can not do this for short options). This **getopt**(1) treats optional arguments that are empty as if they were not present. The syntax if you do not want any short option variables at all is not very intuitive (you have to set them explicitely to the empty string).

## AUTHOR

Frodo Looijaard <frodol@dds.nl>

## SEE ALSO

**getopt**(3), **bash**(1), **tcsh**(1).

## NAME

lpr − Spool files to a printer

## SYNOPSIS

**lpr** [**−D**] [**−d** *device* ] [**−h**] [**−l**] [**−P** *device* ]

## DESCRIPTION

**lpr** spools a file to the specified printer device. No formatting is done -- data is sent "raw". This is useful, for example, for sending Postscript data to a Postscript printer.

## OPTIONS

**-D** enables some debugging output.

**-d** *device*
specifies the *device* to which to send the output.

**-h** does nothing and is accepted for compatibility only.

**-l** disables CR/LF translation. Normally, files are converted to DOS/Windows-style line endings (CR+LF) during the spooling process. Some drivers appear to require this translation, while most don’t seem to care. Including this option on the lpr command line disables any such translations.

**-P** *device*
an alias for **-d**.

## DEVICES

A device name may be a UNC path (\\server_name\printer_name), a reserved DOS device name (e.g., prn, lpt1), or a local port name that is mapped to a printer share. Note that forward slashes may be used in a UNC path also (e.g., //server_name/printer_name).

## ENVIRONMENT

A default device name may be specified in the **PRINTER** environment variable. Specifying a device via a **-d** or **-P** will override the environment variable setting.

## NOTES

Make sure that the default paper size setting is correct for the program that is formatting the page. For example, for enscript, make sure the **DefaultMedia** setting is correct in **/etc/enscript.cfg**. If this setting is incorrect, it is possible that no output at all will be produced by the printer. This is **not** an lpr issue.

## AUTHORS

Written by Rick Rankin.

## NAME

mcookie − generate magic cookies for xauth

## SYNOPSIS

**mcookie [−v] [−f** *filename* **]**

## DESCRIPTION

**mcookie** generates a 128-bit random hexadecimal number for use with the X authority system. Typical usage:

xauth add :0 . ‘mcookie‘

The "random" number generated is actually the output of the MD5 message digest fed with various piece of random information: the current time, the process id, the parent process id, the contents of an input file (if **−f** is specified), and several bytes of information from the first of the following devices which is present: */dev/random*, */dev/urandom*, files in */proc*, */dev/audio*.

## BUGS

The entropy in the generated 128-bit is probably quite small (and, therefore, vulnerable to attack) unless a non-pseudorandom number generator is used (e.g., */dev/random* under Linux).

It is assumed that none of the devices opened will block.

## FILES

*/dev/random*
*/dev/urandom*
*/dev/audio*
*/proc/stat*
*/proc/loadavg*

## SEE ALSO

**X**(1), **xauth**(1), **md5sum**(1)

## NAME

mkshortcut − create a Windows shortcut

## SYNOPSIS

**mkshorcut** [−**a** *ARGS*] [−**d** *DESC*] [−**i** *ICONFILE* [−**j** *INT*] ] [−**n** *NAME* ] [−**s** *norm*|*min*|*max* ] [−**w** *PATH* ] [−A] [−D|−P] *TARGET*

## OPTIONS

**−a**, **−−arguments**=*ARGS*
Arguments to use (see example below).

**−d**, **−−desc**=*"DESC"*
Text for description/tooltip (defaults to POSIX path of TARGET). Note that *"DESC"* can contain spaces, but in that case must be enclosed in quotes.

**−h**, **−−help**
Output usage information (to stdout) and exit.

**−i**, **−−icon**=*ICONFILE*
Specify *ICONFILE* containing the icon to use for the shortcut; defaults to **TARGET**. *ICONFILE* must be a full filename, including an extension if applicable.

**−j**, **−−iconoffset**=*NUM*
Requires **−i**. Use *NUM* icon in *ICONFILE*, offset from 0. Defaults to 0.

**−n**, **−−name**=*"NAME"*
Name to use for the shortcut file. Note that *"NAME"* can contain spaces, but in that case must be enclosed in quotes. The Windows extention **.lnk** is automatically appended to *"NAME"* if not present.

**−s**, **−−show**=*norm*|*min*|*max*
For norm, min, and max the new window will be normal, minimized, and maximized, respectively. Note that if you use the properties dialog to inspect properties of shortcuts for which you request minimized windows the dialog may indicate that normal windows will be displayed. Fortunately, that indicator is often wrong.

**−w**, **−−workingdir**=*"PATH"*
PATH to use for the working directory (defaults to directory path of TARGET).

**−v**, **−−version**
Output version information (to stdout) and exit.

**−A**, **−−allusers**
Requires either **−D** or **−P**. Instead of using the current user's *"Desktop"* or *"Start Menu/Programs"* folders, use the *"All Users"* version. Has no effect on Win95 without multiple users.

**−D**, **−−desktop**
Instead of creating the shortcut relative to the current directory, create it relative to the Windows *"Desktop"* directory. The **−A** option can also be used to use the *"All Users/Desktop"* directory instead.

**−P**, **−−smprograms**
Instead of creating the shortcut relative to the current directory, create it relative to the Windows *"Start Menu/Programs"* directory. The **−A** option can also be used to use the *"All Users/Start Menu/Programs"* directory instead.

## NOTES

All filename arguments must be in unix (POSIX) format, not in Windows (C:\) format. If **mkshortcut** encounters a syntax error, it will return an exit value of 1 and output usage information to stderr.

If you don't need any of **mkshortcut**'s options and just want a simple link to a file or directory, you might want to just use a symbolic link with **ln**.

## EXAMPLES

Create a shortcut to the Cygwin website in the *"Start Menu/Programs"* directory:

$ mkshortcut -P http://www.cygwin.com

Create a shortuct to **rxvt** on the *"Desktop"* that looks like *"Internet Explorer"* but really starts up an interactive **bash** shell:

$ mkshortcut -a ’-rv -fn "FixedSys" -e /bin/bash --login -i’ \
-i /c/WINNT/system32/SHELL32.DLL -j 106 -n "Internet Explorer" \
-D /bin/rxvt

## COPYRIGHT



## AUTHOR

Joshua Daniel Franklin, joshuadfranklin@yahoo.com

## SEE ALSO

**ln**(1)

## NAME

namei - follow a pathname until a terminal point is found

## SYNOPSIS

**namei** *[-mx] pathname [ pathname ... ]*

## DESCRIPTION

*Namei* uses its arguments as pathnames to any type of Unix file (symlinks, files, directories, and so forth). *Namei* then follows each pathname until a terminal point is found (a file, directory, char device, etc). If it finds a symbolic link, we show the link, and start following it, indenting the output to show the context.

This program is useful for finding a "too many levels of symbolic links" problems.

For each line output, *namei* outputs a the following characters to identify the file types found:

f: = the pathname we are currently trying to resolve
d = directory
l = symbolic link (both the link and it's contents are output)
s = socket
b = block device
c = character device
- = regular file
? = an error of some kind

*Namei* prints an informative message when the maximum number of symbolic links this system can have has been exceeded.

## OPTIONS

**-x** Show mount point directories with a 'D', rather than a 'd'.

**-m** Show the mode bits of each file type in the style of ls(1), for example 'rwxr-xr-x'.

## AUTHOR

Roger Southwick (rogers@amadeus.wr.tek.com)

## BUGS

To be discovered.

## CAVEATS

*Namei* will follow an infinite loop of symbolic links forever. To escape, use SIGINT (usually ^C).

## SEE ALSO

ls(1), stat(1)

## NAME

readshortcut − read data from a windows shortcut (.lnk) file

## SYNOPSIS

**readshortcut** [OPTION]∗ SHORTCUT

## DESCRIPTION

**readshortcut** is a command-line tool for reading Windows shortcut files (also known as OLE links). The most practical use is to resolve the target that a shortcut points to. It can be easily run from a script or batch file for non-interactive use.

## OPTIONS

**−h**, **−−help**
Output detailed usage information and exit.

**−−usage**
Output basic usage information and exit.

**−v**, **−−version**
Display the program version and exit.

**−−license**
Display the license agreement and exit.

**−f**, **−−fields**
Show field names (Target, Working Directory, Arguments, Show Command, Icon Library, Icon Library Offset, Description).

**−u**, **−−unix**
Use Unix path format for display (default).

**−w**, **−−windows**
Use Windows path format for display.

**−t**, **−−target**
Display shortcut target (default).

**−g**, **−−working**
Display shortcut working directory.

**−r**, **−−args**
Display shortcut arguments

**−s**, **−−showcmd**
Display shortcut "show" command value (Normal, Minimized, or Maximized).

**−i**, **−−icon**
Display icon library location.

**−j**, **−−offset**
Display icon library offset.

**−d**, **−−desc**
Display shortcut description.

**−a**, **−−all**
Display all information.

## NOTES

The filename (shortcut) argument may be unix (POSIX) format or Windows (C:\) format.

Information extracted from the shortcut is always displayed in the following order: Target, Working Directory, Arguments, Show Command, Icon Library, Icon Library Offset, Description. Each element appears on a separate line.

## BUGS

There may be problems if the SHORTCUT argument is longer than 256 characters

## COPYRIGHT



## AUTHOR

Rob Siklos, (r o b 3 @ s i k l o s . c a), Toronto, Canada

## SEE ALSO

**mkshortcut**(1) **readlink**(1)

## NAME

rename − Rename files

## SYNOPSIS

**rename** *from to file...*

## DESCRIPTION

**rename** will rename the specified files by replacing the first occurrence of *from* in their name by *to*. For example, given the files *foo1*, ..., *foo9*, *foo10*, ..., *foo278*, the commands

```
rename foo foo0 foo?
rename foo foo0 foo??
```

will turn them into *foo001*, ..., *foo009*, *foo010*, ..., *foo278*. And

```
rename .htm .html *.htm
```

will fix the extension of your html files.

## SEE ALSO

**mv**(1)

**NAME**

**rev** — reverse lines of a file

**SYNOPSIS**

**rev** [*file*]

**DESCRIPTION**

The **rev** utility copies the specified files to the standard output, reversing the order of characters in every line. If no files are specified, the standard input is read.