Open Computing ‘‘Hands-On’’: ‘‘Wizard’s Grabbag’’ Column: December 1994: Listings

Listing 1: The `dateorder` script reports files named as month, year (*mmmyy*) in chronological order.

A. Listing of the `dateorder` Perl program:

```
#! /usr/bin/perl
# @(#) dateorder    Return mmmyy command-line args in yr, month order.
# Author: Robert Nicholson , April 1994
# Usage: dateorder mmmyr-string ...

# Given the following command-line argument list:
# INDEX apr90 jan89 dec90 apr89 jan90 dec89
# Dateorder returns:
# jan89 apr89 dec89 jan90 apr90 dec90

die "Usage: $0 mmmyr-string [mmmyr-string...]\n" unless @ARGV;
%monthorder =
    ("jan", 1, "feb", 2, "mar", 3, "apr",  4, "may", 5,  "jun",  6,
     "jul", 7, "aug", 8, "sep", 9, "oct", 10, "nov", 11, "dec", 12 );

# Only collect arguments with mmmyy (month-year) prefix:
$i = 0;                                     # initialize
foreach $file (@ARGV) {
    if (&month_and_year_prefix($file)) {    # if proper prefix
        $monthfiles[$i++]=$file;            # save argument
    }
}

@sorted_months = sort by_date @monthfiles;  # order by year, month
print "@sorted_months\n";                   # space-separated display

# Comparator to sort by year and then by month in that year:
sub by_date {
    # First three characters are the month string:
    $amon = substr($a, 0, 3);               # mmm saved in amon
    $bmon = substr($b, 0, 3);               # mmm saved in bmon
    # The next two are the year:
    $ayr = substr($a, 3, 2);                # yy saved in ayr
    $byr = substr($b, 3, 2);                # yy saved in byr

    # Lookup index with lower-case string:
    $amonthindex = $monthorder{"\L$amon"};  # 1-12 in amonthindex
    $bmonthindex = $monthorder{"\L$bmon"};  # 1-12 in bmonthindex

    # Catenate the year and adjusted month index:
    $aall = $ayr.($amonthindex+11);         # create four-digit integer
    $ball = $byr.($bmonthindex+11);         # for easy comparision
    $aall <=> $ball;                        # by "spaceship" operator
}

# Determine if argument is mmmyy (month and year prefix):
sub month_and_year_prefix {
    local($filename) = @_;

    return 0 if (length($filename) < 5);# files with names too short
```

```
    $month = substr($filename,0,3);      # save month prefix
    $year = substr($filename,3,2);       # followed by the year

    foreach $key (keys %monthorder) {    # for each valid month
        return 1 if (-f $filename        # only consider plain files
            && $monthorder{$month}       # with valid month abbreviation
            && ($year =~ /\d\d/))        # and two digits for the year
    }
    return 0;                            # not the correct prefix
}
```

B. Sample usage:

```
$ echo *
dec91.Z index91.Z jan91.Z jul91.Z mar91.Z may91.Z nov91.Z sep91.Z
$ dateorder *
jan91.Z mar91.Z may91.Z jul91.Z sep91.Z nov91.Z dec91.Z
$ {}
```

C. A short simulation using conventional Unix tools, which isn't portable (see text for discussion):

```
$ cd /MirrorFiles/ftp.uu.net/published/unix-world/grabbag
$ ls -l 1*/* | sort +0.47 -0.51 +0.52M
-rw-r--r--    1 bne         10451 Feb 20  1994 1990/nov90.Z
-rw-r--r--    1 bne          4100 Feb 20  1994 1994/jan94
-rw-r--r--    1 bne          9566 Feb 20  1994 1994/feb94
-rw-r--r--    1 bne          5171 Mar 13  1994 1994/mar94
-rw-r--r--    1 bne          8271 Apr  6 01:48 1994/apr94
-rw-r--r--    1 bne          7395 May 24 01:28 1994/may94
-rw-r--r--    1 bne         16664 May 24 01:28 1994/jun94
-rw-r--r--    1 bne          4043 Jul 27 01:23 1994/jul94
-rw-r--r--    1 bne          5429 Jul 30 13:24 1994/aug94
$ {}
```

D. Another simulation command line:

```
$ ls -l 1*/** | \
egrep -i '(jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec)[0-9][0-9]'| \
sort +0.54n +0.52M
```

Listing 2: Definitions of the directory-climbing functions.

A. Listing of the contributed `up()` function:

```
# @(#) up  Move up directory hierarchy specified number of levels
# Author: David Wood, March 1990
#
up()
{
    count=${1:-1}    # default one level, otherwise use argument
    while [ $count -gt 0 ]; do  # while need to climb
        cd ..                   # go up a level
        count=`expr $count - 1` # one less level to go
    done
}
```

B. Listing of the version ported to the Korn shell:

```
# @(#) up  Move up a directory tree specified number of levels
# Author: David Wood, March 1990
# Adapted for Korn shell by Becca Thomas, December 1992
#
function up
{
    Usage="Usage: `basename $0` [level-count]\\nWhere, level-count >0"
    case $# in
        0|1)    ;;                     # correct argument count
        *)  print -u2 $Usage; return ;; # no more than one argument
    esac
    integer count=${1:-1}   # default one level, otherwise use argument
    if (( $count <= 0 )); then      # if argument is not positive
        print -u2 "level-count must be positive integer\n$Usage";return
    fi
    while (( $count -gt 0 )); do    # while need to climb
        cd ..                       # go up a level
        let count=count-1           # one less level to go
    done
}
```


Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content   Search

Last Modified: Tuesday, 22-Aug-95 16:18:57 PDT