Thomas Fahle

How To - Freien Speicherplatz auf einem Dateisystem ermitteln

Der freie Speicherplatz auf Festplatten bzw. Dateisystemen kann sowohl mit Filesys::Df als auch mit Filesys::DfPortable von Ian_Guthrie einfach ermittelt werden.

Filesys::DfPortable funktioniert unter Linux und Windows, während Filesys::Df nur mit unixoiden Betriebssystemen funktioniert, dafür aber ein bereits geöffnetes Datei Handle als Argument verarbeiten kann.

Filesys::Df

Die automatisch importierte Funktion `df` liefert eine Referenz auf einen Hash mit Informationen über die freien und belegten Blöcke des Dateisystems zurück.

Da nicht alle Dateisysteme Inodes unterstützen, ist es erforderlich, explizit zu fragen, ob Informationen zu Indodes vorliegen - siehe Listing 1.

Das Programm erzeugt z.B. folgende Ausgabe:

```
Total 1k blocks:             423301588
Total 1k blocks free:        230537836
Total 1k blocks avail to me: 209035272
Total 1k blocks used:        192763752
Percent full:                48
Total inodes:                26886144
Total inodes free:           26093840
Inode percent full:          3
```

Die Blockgröße lässt sich über den zweiten Parameter der Funktion `df()` angeben. Wer lieber in Bytes rechnet, verwendet einfach die Blockgröße 1 (siehe Listing 2).

Das Programm erzeugt z.B. folgende Ausgabe:

```
Total bytes:              433460826112
Total bytes free:         236072210432
Total bytes avail to me:  214053584896
Total bytes used:         197388615680
Percent full:             48%
Total inodes:             26886144
Total inodes free:        26093880
Inode percent full:       3
```

Filesys::Df kann auch Informationen über ein Dateisystem an Hand eines bereits geöffneten Dateihandles ermitteln. Das ist recht praktisch, wenn man wissen möchte, wie viel Platz für das gerade laufende Perl Programm ($0) noch zur Verfügung steht - siehe Listing 3.

Ausgabe wie oben.

Filesys::DfPortable

Filesys::DfPortable funktioniert, wie bereits gesagt, unter Linux und Windows und ist Filesys::Df ziemlich ähnlich.

Die automatisch importierte Funktion `dfportable()` liefert eine Referenz auf einen Hash mit Informationen über die freien und belegten Blöcke des Dateisystems zurück.

Die Blockgröße lässt sich über den zweiten Parameter der Funktion `dfportable()` angeben. Default Blockgröße ist dieses Mal 1, also Ausgabe in Bytes - siehe Listing 4.

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::Df;

# Default output is 1K blocks
my $df = df('/tmp');

if ( defined($df) ) {
    print 'Total 1k blocks:
           $df->{blocks}\n';
    print 'Total 1k blocks free:
           $df->{bfree}\n';
    print 'Total 1k blocks avail to me:
           $df->{bavail}\n';
    print 'Total 1k blocks used:
           $df->{used}\n';
    print 'Percent full:
           $df->{per}\n';

    # Only filesystems that support inodes
    if ( exists( $df->{files} ) ) {
        print 'Total inodes:
               $df->{files}\n';
        print 'Total inodes free:
               $df->{ffree}\n';
        print 'Inode percent full:
               $df->{fper}\n';
    }
}
else {
    warn 'Woops - something went wrong.\n';
}
```

Listing 1

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::Df;

my $df = df('/tmp', 1);    # output is bytes

if ( defined($df) ) {
    print 'Total bytes:
           $df->{blocks}\n';
    print 'Total bytes free:
           $df->{bfree}\n';
    print 'Total bytes avail to me:
           $df->{bavail}\n';
    print 'Total bytes used:
           $df->{used}\n';
    print 'Percent full:
           $df->{per}%\n';

    # Only for filesystems that
    # support inodes
    if ( exists( $df->{files} ) ) {
        print 'Total inodes:
               $df->{files}\n';
        print 'Total inodes free:
               $df->{ffree}\n';
        print 'Inode percent full:
               $df->{fper}\n';
    }
}
else {
    warn 'Woops - something went wrong.\n';
}
```

Listing 2

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::Df;

open( ME, '<', $0 ) or
      die 'Can't open myself $!';

# Get information for filesystem
# at file handle
my $df = df( \*ME, 1 );

if ( defined($df) ) {

    print 'Total bytes:
           $df->{blocks}\n';
    print 'Total bytes free:
           $df->{bfree}\n';
    print 'Total bytes avail to me:
           $df->{bavail}\n';
    print 'Total bytes used:
           $df->{used}\n';
    print 'Percent full:
           $df->{per}%\n';

    # Only for filesystems that
    # support inodes
    if ( exists( $df->{files} ) ) {
        print 'Total inodes:
               $df->{files}\n';
        print 'Total inodes free:
               $df->{ffree}\n';
        print 'Inode percent full:
               $df->{fper}\n';
    }
}
else {
    warn 'Woops - something went wrong.\n';
}
```

Listing 3

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::DfPortable;

# Display output in 1K blocks
my $dfp = dfportable('C:', 1024);

if ( defined($dfp) ) {
   print'Total 1k blocks:
         $dfp->{blocks}\n';
   print'Total 1k blocks free:
         $dfp->{bfree}\n';
   print'Total 1k blocks avail to me:
         $dfp->{bavail}\n';
   print'Total 1k blocks used:
         $dfp->{bused}\n';
   print'Percent full:
         $dfp->{per}\n';

   # Only filesystems that support inodes
   if ( exists( $dfp->{files} ) ) {
        print 'Total inodes:
               $dfp->{files}\n';
        print 'Total inodes free:
               $dfp->{ffree}\n';
        print 'Inode percent full:
               $dfp->{fper}\n';
   }

} else {
    warn 'Woops - something went wrong.\n';
}
```

Listing 4

Das Programm erzeugt z.B. folgende Ausgabe:

```
Total 1k blocks:              26109948
Total 1k blocks free:         4299136
Total 1k blocks avail to me: 4299136
Total 1k blocks used:         21810812
Percent full:                 84
```

Statt des Laufwerksbuchstabens können auch UNC_Pfade wie '\\\\Server\\Freigabe' verwendet werden.

Filesys::DfPortable unterstützt, wie Filesys::Df, Inodes - das ist meiner Ansicht nach unter Portabilitätsaspekten etwas sinnfrei, da Windows Inodes erst gar unterstützt.

Wer unter Windows einfach nur wissen möchte, wieviel Speicherplatz noch verfügbar ist, und daher nicht alle `bells and whistles` von Win32::DriveInfo benötigt, gewinnt mit Filesys::DfPortable ein wenig Portabilität.