dtt – disk “top ten” list

Edward K. McGuire

March 7, 2024
Version 2.2

Abstract

dtt types a “top ten” ranking of object collections found on a file-structured storage device such as a Unix or Windows volume. In other words, it floats the really big chunks to the top.

Contents

1 Synopsis

dtt [*options*] [*files*]

2 Description

Object measurements are aggregated into collections by directory name, or optionally by filename suffix. A grand total is reported, followed by the top ten collections in order, largest to smallest.

The measurements available for ranking are:

- physical size,
- virtual (apparent) size,
- directory entries, and
- inodes.

When using the directory name aggregation method, *dtt* defines a directory's measurement as its own measurement plus the sum of the measurements of its contents. To avoid double-counting, if any directory would be in the top ten list along with one or more of its descendant directories or objects, only the descendants are reported, not the directory.

When using the filename suffix aggregation method, *dtt* defines a suffix's measurement as the sum of the measurements of all files having that suffix. Objects having no suffix are aggregated using an empty (blank) suffix.

3 Options

Options, given in the argument list, alter the behavior of *dtt*. Giving an unknown option is a fatal error: *dtt* types a diagnostic message on *stderr* and stops.

5a ⟨*dtt set option arguments* 5a⟩≡ (28a)

```
use Getopt::Long ;
Getopt::Long::Configure( 'bundling' ) ;
GetOptions(
⟨dtt option specifications 5b⟩
) or exit 1 ;
```

3.1 Measurement Control Options

3.1.1 Physical size

When no measurement control options are given, *dtt* measures physical size. This is useful when the focus of the investigation is on disk blocks in use on the local volume.

3.1.2 Virtual size

The `--apparent-size` option measures virtual (apparent) size. This is useful when the focus of the investigation is large objects, regardless of disk space currently in use. Virtual size is often less than physical size, because file lengths do not generally extend exactly to their block boundaries. But it can also be much more than physical size, in the case of sparse or offline files. The `--apparent-size`, `--dirent`, and `--inodes` options conflict with one another.

5b ⟨*dtt option specifications* 5b⟩≡ (5a) 6a▷

```
'--apparent-size' => sub {
        $opt_apparent = 1 ;
        $opt_dirent = 0 ; $opt_inodes = 0 ;
} ,
```

Uses `opt_apparent` 5d, `opt_dirent` 6c, and `opt_inodes` 6c.

5c ⟨*dtt option help* 5c⟩≡ (10h) 6b▷

```
--apparent-size         measure virtual (apparent) size
```

5d ⟨*dtt option variables* 5d⟩≡ (28a) 6c▷

```
my $opt_apparent = 0 ;
```

Defines:
`opt_apparent`, used in chunks 5b, 6a, 13b, 20d, and 22a.

3.1.3 Directory entries or inodes

The `--dirent` (`-d`) option counts directory entries instead of file sizes. The `--inodes` option counts inodes instead of file sizes. A file with multiple hard links will be counted each time by `--dirent`, but will be counted once by `--inodes`. The `--apparent-size`, `--dirent`, and `--inodes` options conflict with one another.

When `--dirent` or `--inodes` are used, *dtt* operates as if `--block-size=1` were given as the first argument. In these modes, environment variables `DTT_BLOCK_SIZE`, `BLOCK_SIZE`, and `BLOCKSIZE` are ignored.

Other values of `--block-size|B` can be given, to scale a `--dirent` or `--inodes` report. For example, `--dirent --block-size=KB` would report the number of directory entries scaled to 1000 files per unit. `--dirent --block-size=K` would scale the report to 1024 files per unit, so that 65536 entries would be reported as 64. (If doing that is confusing, consider not doing that.)

6a ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁5b 7a▷

```
'dirent|d' => sub {
        $opt_dirent = 1 ;
        $opt_apparent = 0 ; $opt_inodes = 0 ;
        $opt_implicit = 1 ;
} ,
'inodes'   => sub {
        $opt_inodes = 1 ;
        $opt_apparent = 0 ; $opt_dirent = 0 ;
        $opt_implicit = 1 ;
} ,
```

Uses `opt_apparent` 5d, `opt_dirent` 6c, `opt_implicit` 7c, and `opt_inodes` 6c.

6b ⟨*dtt option help* 5c⟩+≡ (10h) ◁5c 7b▷

```
--dirent          | -d  measure directory entry counts, not sizes
--inodes                measure inode counts, not sizes
```

6c ⟨*dtt option variables* 5d⟩+≡ (28a) ◁5d 7c▷

```
my $opt_dirent = 0 ;
my $opt_inodes = 0 ;
```

Defines:
`opt_dirent`, used in chunks 5b, 6a, 13b, 20d, and 22a.
`opt_inodes`, used in chunks 5b, 6a, 13b, 20d, and 22a.

3.2 Scaling Control Options

3.2.1 Scaling to a given unit of measure

The `--block-size=s` (`-B s`) option scales measurements to a given unit of measure. The option value can be `n`, or it can be an integer having an optional suffix. The `--human-readable`

and `--si` options conflict with this option.

The option value `n` means "native blocksize". It scales measurements to the `st_blksize` property of the file system being reported on, such as `4096`.

Optional suffixes used with an integer correspond to those supported by *du* and *df*: `K` or `KiB` for customary (binary) kilobytes (1024 bytes), `KB` for metric kilobytes (1000 bytes), and so on for `M` (mega), `G` (giga), `T` (tera), `P` (peta), `E` (exa), `Z` (zetta), `Y` (yotta), `R` (ronna), and `Q` (quetta). Scaled values are rounded to whole numbers and typed with no suffix.

When `--block-size` is not present, *dtt* operates as if `--block-size` were given as the first argument. It takes the option value from the environment. *dtt* searches for the variables `DTT_BLOCK_SIZE`, `BLOCK_SIZE`, and `BLOCKSIZE` in the environment. Searching in this order, the first value found determines the runtime value. If none of these variables are found, the value `512` is used. (But see `--dirent` and `--inodes` for a special case change to this behavior.)

The option value `n` is only compatible with measurements of multiple filesystems when they agree on `st_blksize`. Multiple blocksizes cause *dtt* to operate as if `--block-size` is not present. If the option value `n` is set by an environment variable, the value `512` is used.

7a ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁6a 8a▷

```
'block-size|B=s' => sub {
$opt_blocksize = validate_blocksize( $_[1] ) ;
$opt_human = 0 ;
$opt_si = 0 ;
} ,
```

Uses `opt_blocksize` 7c, `opt_human` 9a, `opt_si` 9d, and `validate_blocksize` 7d.

7b ⟨*dtt option help* 5c⟩+≡ (10h) ◁6b 8b▷

```
--block-size=s    | -Bs scale counts to blocks of size s
```

7c ⟨*dtt option variables* 5d⟩+≡ (28a) ◁6c 9a▷

```
my $opt_blocksize ;
my $opt_implicit = validate_blocksize(
  exists $ENV{'DTT_BLOCK_SIZE'} ? $ENV{'DTT_BLOCK_SIZE'}
: exists $ENV{'BLOCK_SIZE'}     ? $ENV{'BLOCK_SIZE'}
: exists $ENV{'BLOCKSIZE'}      ? $ENV{'BLOCKSIZE'}
: 512
) ;
```

Defines:
`opt_blocksize`, used in chunks 7–9, 19a, 20d, and 22a.
`opt_implicit`, used in chunks 6a, 19a, 20d, and 22a.
Uses `validate_blocksize` 7d.

7d ⟨*dtt function declarations* 7d⟩≡ (28a) 22a▷

```
sub validate_blocksize {
        CASE: {
                if ( $_[0] eq 'n' ) { return 'n' }
                if ( $_[0] =~ m '^[0-9]*[1-9][0-9]*' ) {
```

```
                        my $multiplicand = $& ;
                        if ( $' eq '' ) { return 1 * $multiplicand ; }
                        if ( $' !~ m "[${\KMGTPEZYRQ}]" ) { last CASE }
                        if ( $' eq '' or $' eq 'iB' ) {
                                my $multiplier = BINARY_FACTORS->{$&} ;
                                return $multiplier * $multiplicand ;
                        }
                        if ( $' eq 'B' ) {
                                my $multiplier = METRIC_FACTORS->{$&} ;
                                return $multiplier * $multiplicand ;
                        }
                }
        }
        warn "$0: invalid blocksize '$_[0]', using 512\n" ;
        return 512 ;
}
```

Defines:
multiplicand, never used.
multiplier, never used.
validate_blocksize, used in chunk 7.
Uses BINARY_FACTORS 13a, KMGTPEZYRQ 13a, and METRIC_FACTORS 13a.

3.2.2 Scaling to human-readable binary units

The `--human-readable` (`-h`) option scales sizes individually to customary binary units of measure. Any count below 2^{10} (1024) is scaled to bytes, and typed as a counting number. Larger values are scaled to customary kilobytes or "kibibytes" (K) through quettabytes (Q), rounded, and typed as a number with a letter suffix. For example, a count of 2^{10} or more but less than 2^{20} is scaled to customary kilobytes, rounded, and typed with a K suffix. Rounded values below 10 are rounded to 1 decimal place. Rounded values of 10 or more are rounded to whole numbers. The `--block-size` and `--si` options conflict with this option.

8a ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁7a 9b▷

```
'human-readable|h' => sub {
$opt_human = 1 ;
undef $opt_blocksize ;
$opt_si = 0 ;
} ,
```

Uses opt_blocksize 7c, opt_human 9a, and opt_si 9d.

8b ⟨*dtt option help* 5c⟩+≡ (10h) ◁7b 9c▷

```
--human-readable  | -h  scale counts to customary binary units
```

9a ⟨*dtt option variables* 5d⟩+≡ (28a) ◁7c 9d▷
```
my $opt_human = 0 ;
```
Defines:
opt_human, used in chunks 7–9, 20d, and 22a.

3.2.3 Scaling to human-readable SI (metric) units

The `--si` option scales sizes individually to metric units of measure. It operates like `--human-readable`, using multiples of 1000 instead of 1024. The `--block-size` and `--human-readable` options conflict with this option.

9b ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁8a 9e▷
```
'si' => sub { $opt_si = 1 ; undef $opt_blocksize ; $opt_human = 0 } ,
```
Uses opt_blocksize 7c, opt_human 9a, and opt_si 9d.

9c ⟨*dtt option help* 5c⟩+≡ (10h) ◁8b 9f▷
```
--si                    scale counts to metric units
```

9d ⟨*dtt option variables* 5d⟩+≡ (28a) ◁9a 10a▷
```
my $opt_si = 0 ;
```
Defines:
opt_si, used in chunks 7–9, 20d, and 22a.

3.3 Aggregation Control Options

3.3.1 Aggregating by file and directory names

When no aggregation control options are given, *dtt* aggregates by file and directory names.

3.3.2 Aggregating by filename suffix

The `--suffix` (`-s`) option aggregates by filename suffix. This is useful when the focus of the investigation is particular file types. The suffixes recognized are:

- a final dot in the filename, optionally followed by other characters; or,
- a final tilde.

9e ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁9b 10b▷
```
'suffix|s' => \$opt_suffix,
```
Uses opt_suffix 10a.

9f ⟨*dtt option help* 5c⟩+≡ (10h) ◁9c 10c▷
```
--suffix          | -s  aggregate by suffix, not by name
```

10a ⟨*dtt option variables* 5d⟩+≡ (28a) ◁9d 10d▷
```
my $opt_suffix = '' ;
```
Defines:
opt_suffix, used in chunks 9e and 13b.

3.4 Miscellaneous Options

The `--one-file-system` (`-x`) option limits the *dtt* investigation to one filesystem. This makes *dtt* behave like `find -xdev` or `du -x`. When *dtt* detects a filesystem change, it types a message on *stderr*, skips the object, and continues.

10b ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁9e 10e▷
```
'one-file-system|x' => \$opt_onefilesystem ,
```
Uses opt_onefilesystem 10d.

10c ⟨*dtt option help* 5c⟩+≡ (10h) ◁9f 10f▷
```
--one-file-system | -x  limit work to one filesystem
```

10d ⟨*dtt option variables* 5d⟩+≡ (28a) ◁10a 10g▷
```
my $opt_onefilesystem = 0 ;
```
Defines:
opt_onefilesystem, used in chunks 10b and 18g.

The `--help` (`-h`) option types the command synopsis on *stdout*. No measurements are taken. The `--version` option conflicts with this option.

10e ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁10b 11a▷
```
'help' => sub { $opt_help = 1 ; $opt_version = 0 } ,
```
Uses opt_help 10g and opt_version 11g.

10f ⟨*dtt option help* 5c⟩+≡ (10h) ◁10c 11b▷
```
--help                  you're reading it -- "man dtt" for more
```

10g ⟨*dtt option variables* 5d⟩+≡ (28a) ◁10d 11c▷
```
my $opt_help = 0 ;
```
Defines:
opt_help, used in chunks 10 and 11e.

10h ⟨*dtt type help and exit* 10h⟩≡ (28a)
```
if ( $opt_help ) {
print <<\EOF ;
Usage: dtt [options] [objects]
Type a "top ten" ranking of objects found on a file-structured storage device
such as a Unix or Windows volume.

Options:
```

```
⟨dtt option help 5c⟩

Objects:
⟨dtt objects help 12b⟩
EOF
exit 0 ;
}
```

Uses opt_help 10g.

The --verbose (-v) option types more information on *stderr* as the count takes place. Currently that information consists of warnings when duplicate inodes are ignored, and warnings when a change in st_blksize is found.

11a ⟨dtt option specifications 5b⟩+≡ (5a) ◁10e 11e▷

```
'verbose|v' => \$opt_verbose,
```

Uses opt_verbose 11c.

11b ⟨dtt option help 5c⟩+≡ (10h) ◁10f 11f▷

```
--verbose          | -v  warn on duplicate inode or new blksize
```

11c ⟨dtt option variables 5d⟩+≡ (28a) ◁10g 11g▷

```
my $opt_verbose = '' ;
```

Defines:
opt_verbose, used in chunks 11a, 18g, and 19b.

The --version option types version information on *stdout*. No measurements are taken. The --help option conflicts with this option.

11d ⟨dtt version 11d⟩≡ (11h 36–38)

```
2.3a
```

11e ⟨dtt option specifications 5b⟩+≡ (5a) ◁11a 26a▷

```
'version' => sub { $opt_version = 1 ; $opt_help = 0 } ,
```

Uses opt_help 10g and opt_version 11g.

11f ⟨dtt option help 5c⟩+≡ (10h) ◁11b

```
--version                type version information and exit
```

11g ⟨dtt option variables 5d⟩+≡ (28a) ◁11c 26b▷

```
my $opt_version = 0 ;
```

Defines:
opt_version, used in chunks 10 and 11.

11h ⟨dtt type version and exit 11h⟩≡ (28a)

```
if ( $opt_version ) {
print <<\EOF ;
dtt ⟨dtt version 11d⟩
⟨boilerplate:copyright 25b⟩
```

```
License, warranty, and other information: <https://metaed.com/papers/dtt>.
EOF
exit 0 ;
}
```

Uses opt_version 11g.

3.5 Option Conflict Resolution

When the operator combines options, conflicts are resolved according to the following rules.

- Options are processed from left to right.
- When conflicting options are given, or the same option is given multiple times with different values, the rightmost option processed takes precedence.

3.6 Object List Arguments

After options, the operator can supply names of files and directories as arguments. These are the objects to be measured. If the operator defaults (supplies no names), the objects in the current working directory will be measured, including hidden objects, but excluding the . and .. entries.

12a ⟨*dtt default objects* 12a⟩≡ (12)

```
..?* .[!.]* *
```

12b ⟨*dtt objects help* 12b⟩≡ (10h)

```
filesystem path(s)
default objects: ⟨dtt default objects 12a⟩
```

12c ⟨*dtt set pathname arguments* 12c⟩≡ (28a)

```
@ARGV = glob( q( ⟨dtt default objects 12a⟩ ) ) unless @ARGV ;
#$opt_debug and warn "dtt DEBUG \$#ARGV=$#ARGV" ;
#$opt_debug and warn "dtt DEBUG \@ARGV=[@ARGV]" ;
```

Defines:
ARGV, used in chunk 13b.
Uses opt_debug 26b.

4 Phases Of Operation

The program operation has three phases:

1. measure,
2. reduce to a top-ten list, and
3. type the report.

13a ⟨*dtt constants* 13a⟩≡ (28a)

```
use constant KMGTPEZYRQ => 'KMGTPEZYRQ' ;
use constant BINARY_FACTORS => {
K => 2**10 , M => 2**20 , G => 2**30 , T => 2**40 , P => 2**50 ,
E => 2**60 , Z => 2**70 , Y => 2**80 , R => 2**90 , Q => 2**100 ,
} ;
use constant METRIC_FACTORS => {
K => 1000**1 , M => 1000**2 , G => 1000**3 , T => 1000**4 , P => 1000**5 ,
E => 1000**6 , Z => 1000**7 , Y => 1000**8 , R => 1000**9 , Q => 1000**10 ,
} ;
```

Defines:
BINARY_FACTORS, used in chunks 7d and 22c.
KMGTPEZYRQ, used in chunks 7d and 22c.
METRIC_FACTORS, used in chunks 7d and 22c.

4.1 Measure Phase

The Measure Phase measures each object in accord with the selected Measurement Control option. When an object is a directory, its descendants are also measured. Hidden descendants are measured, except the `.` and `..` entries. Measurements are aggregated in accord with the selected Aggregation Control option. Encountering an unreadable or nonexistent object is not a fatal error. *dtt* types a diagnostic message on *stderr* and continues with the next object to be measured.

13b ⟨*dtt lookup* 13b⟩≡ (28a)

```
sub walk_blocks_subtrees {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
                ⟨aggregate by subtree 18b⟩
                ⟨stat for id and blocks 18f⟩
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨count blocks 19d⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_blocks_subtrees( @entries ) ;
```

```
                ⟨aggregate child size 20a⟩
                ⟨total child size 20b⟩
            }
        }
        ⟨return total 20c⟩
}
sub walk_blocks_suffixes {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
            ⟨aggregate by suffix 18c⟩
            ⟨stat for id and blocks 18f⟩
            ⟨detect filesystem change 18g⟩
            ⟨detect duplicate inode 19b⟩
            ⟨record new inode 19c⟩
            ⟨count blocks 19d⟩
            ⟨detect directory 19g⟩
                $weight_children = walk_blocks_suffixes( @entries ) ;
                ⟨total child size 20b⟩
            }
        }
        ⟨return total 20c⟩
}
sub walk_bytes_subtrees {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
            ⟨aggregate by subtree 18b⟩
            ⟨stat for id and bytes 18e⟩
            ⟨detect filesystem change 18g⟩
            ⟨detect duplicate inode 19b⟩
            ⟨record new inode 19c⟩
            ⟨count bytes 19e⟩
            ⟨detect directory 19g⟩
                $weight_children = walk_bytes_subtrees( @entries ) ;
                ⟨aggregate child size 20a⟩
                ⟨total child size 20b⟩
            }
        }
        ⟨return total 20c⟩
}
sub walk_bytes_suffixes {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
            ⟨aggregate by suffix 18c⟩
            ⟨stat for id and bytes 18e⟩
```

```
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨count bytes 19e⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_bytes_suffixes( @entries ) ;
                        ⟨total child size 20b⟩
                }
        }
        ⟨return total 20c⟩
}
sub walk_count_dirent_subtrees {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
                ⟨aggregate by subtree 18b⟩
                ⟨stat for id only 18d⟩
                ⟨count slot 19f⟩
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_count_dirent_subtrees( @entries ) ;
                        ⟨aggregate child size 20a⟩
                        ⟨total child size 20b⟩
                }
        }
        ⟨return total 20c⟩
}
sub walk_count_dirent_suffixes {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
                ⟨aggregate by suffix 18c⟩
                ⟨stat for id only 18d⟩
                ⟨count slot 19f⟩
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_count_dirent_suffixes( @entries ) ;
                        ⟨total child size 20b⟩
                }
        }
        ⟨return total 20c⟩
}
```

```
sub walk_count_inodes_subtrees {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
                ⟨aggregate by subtree 18b⟩
                ⟨stat for id only 18d⟩
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨count slot 19f⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_count_inodes_subtrees( @entries ) ;
                        ⟨aggregate child size 20a⟩
                        ⟨total child size 20b⟩
                }
        }
        ⟨return total 20c⟩
}
sub walk_count_inodes_suffixes {
        ⟨declare counters 17b⟩
        ⟨iterate over readable arguments 18a⟩
                ⟨aggregate by suffix 18c⟩
                ⟨stat for id only 18d⟩
                ⟨detect filesystem change 18g⟩
                ⟨detect duplicate inode 19b⟩
                ⟨record new inode 19c⟩
                ⟨count slot 19f⟩
                ⟨detect directory 19g⟩
                        $weight_children = walk_count_inodes_suffixes( @entries ) ;
                        ⟨total child size 20b⟩
                }
        }
        ⟨return total 20c⟩
}
( $starting_st_dev, undef, undef, undef, undef, undef, undef, undef, undef,
        undef, undef, $starting_st_blksize ) = lstat( $ARGV[0] ) ;
#$opt_debug && warn "\$starting_st_blksize=$starting_st_blksize" ;
my $grand_total =
        $opt_dirent
                ? $opt_suffix
                        ? walk_count_dirent_suffixes( @ARGV )
                        : walk_count_dirent_subtrees( @ARGV )
        : $opt_inodes
                ? $opt_suffix
                        ? walk_count_inodes_suffixes( @ARGV )
```

```
                        : walk_count_inodes_subtrees( @ARGV )
        : $opt_apparent
                ? $opt_suffix
                        ? walk_bytes_suffixes( @ARGV )
                        : walk_bytes_subtrees( @ARGV )
        :       $opt_suffix
                        ? walk_blocks_suffixes( @ARGV )
                        : walk_blocks_subtrees( @ARGV )
        ;
```

Defines:
grand_total, used in chunk 20d.
walk_blocks_subtrees, never used.
walk_blocks_suffixes, never used.
walk_bytes_subtrees, never used.
walk_bytes_suffixes, never used.
walk_count_dirent_subtrees, never used.
walk_count_dirent_suffixes, never used.
walk_count_inodes_subtrees, never used.
walk_count_inodes_suffixes, never used.
Uses ARGV 12c, opt_apparent 5d, opt_debug 26b, opt_dirent 6c, opt_inodes 6c, opt_suffix 10a, starting_st_blksize 17a, starting_st_dev 17a, and weight_children 17b.

17a ⟨*dtt database* 17a⟩≡ (28a)

```
my %weight ;
my %known_st_ino ;
my $starting_st_dev ;
my $starting_st_blksize ;
my $st_blksize_changed = 0 ;
```

Defines:
known_st_ino, used in chunk 19.
st_blksize_changed, used in chunks 18g and 19a.
starting_st_blksize, used in chunks 13b, 18g, 20d, and 22a.
starting_st_dev, used in chunks 13b and 18g.
weight, used in chunks 19 and 20.

17b ⟨*declare counters* 17b⟩≡ (13b)

```
my $aggregation_name ;
my $weight_children ;
my $weight_all = 0 ;
my $st_dev ;
my $st_ino ;
my $st_size ;
my $st_blksize ;
my $st_blocks ;
```

Defines:
aggregation_name, used in chunks 18–20.
st_blksize, used in chunk 18.
st_blocks, used in chunks 18f and 19d.
st_dev, used in chunks 18 and 19.

st_ino, used in chunks 18 and 19.
st_size, used in chunks 18e and 19e.
weight_all, used in chunks 19 and 20.
weight_children, used in chunks 13b and 20.

18a ⟨*iterate over readable arguments* 18a⟩≡ (13b)

```
#$opt_debug and warn "dtt DEBUG \$#_=$#_" ;
#$opt_debug and warn "dtt DEBUG \@_=[@_]" ;
foreach ( @_ ) {
#$opt_debug and warn "dtt DEBUG \$_='$_'" ;
lstat or do { warn "$0: $!: cannot lstat: '$_'\n" ; next ; } ;
```

Uses opt_debug 26b.

18b ⟨*aggregate by subtree* 18b⟩≡ (13b)

```
$aggregation_name = $_ ;
```

Uses aggregation_name 17b.

18c ⟨*aggregate by suffix* 18c⟩≡ (13b)

```
/(\.[^.\/]*|~)$/ ;
$aggregation_name = $1 ;
```

Uses aggregation_name 17b.

18d ⟨*stat for id only* 18d⟩≡ (13b)

```
( $st_dev, $st_ino, undef, undef, undef, undef, undef, undef, undef, undef,
        undef, $st_blksize, undef ) = stat ( _ ) ;
```

Uses st_blksize 17b, st_dev 17b, and st_ino 17b.

18e ⟨*stat for id and bytes* 18e⟩≡ (13b)

```
( $st_dev, $st_ino, undef, undef, undef, undef, undef, $st_size, undef, undef,
        undef, $st_blksize, undef ) = stat( _ ) ;
```

Uses st_blksize 17b, st_dev 17b, st_ino 17b, and st_size 17b.

18f ⟨*stat for id and blocks* 18f⟩≡ (13b)

```
( $st_dev, $st_ino, undef, undef, undef, undef, undef, undef, undef, undef,
        undef, $st_blksize, $st_blocks ) = stat( _ ) ;
```

Uses st_blksize 17b, st_blocks 17b, st_dev 17b, and st_ino 17b.

18g ⟨*detect filesystem change* 18g⟩≡ (13b)

```
if ( $st_dev ne $starting_st_dev ) {
        if ( $opt_onefilesystem ) {
                warn "$0: Filesystem changed, skipping: $_\n" ;
                next ;
        }
        unless ( $st_blksize == $starting_st_blksize ) {
                $st_blksize_changed = 1 ;
                $opt_verbose and warn "$0: Blocksize changed: '$_'\n" ;
        }
}
```

Uses opt_onefilesystem 10d, opt_verbose 11c, st_blksize 17b, st_blksize_changed 17a, st_dev 17b, starting_st_blksize 17a, and starting_st_dev 17a.

19a ⟨*dtt check for blksize changed* 19a⟩≡ (28a)

```
if ( $st_blksize_changed ) {
        if ( $opt_blocksize eq 'n' ) {
                warn "$0: Ignoring 'n' scaling, using implicit scaling'\n" ;
                undef $opt_blocksize ;
        }
        if ( $opt_implicit eq 'n' ) {
                warn "$0: Ignoring implicit 'n' scaling, using '512'\n" ;
                $opt_implicit = 512 ;
        }
}
```

Uses opt_blocksize 7c, opt_implicit 7c, and st_blksize_changed 17a.

19b ⟨*detect duplicate inode* 19b⟩≡ (13b)

```
if ( $known_st_ino{$st_dev . ' ' . $st_ino} ) {
        $opt_verbose and warn "$0: Duplicate inode, skipping: $_\n" ;
        next ;
}
```

Uses known_st_ino 17a, opt_verbose 11c, st_dev 17b, and st_ino 17b.

19c ⟨*record new inode* 19c⟩≡ (13b)

```
$known_st_ino{$st_dev . ' ' . $st_ino} = 1 ;
```

Uses known_st_ino 17a, st_dev 17b, and st_ino 17b.

19d ⟨*count blocks* 19d⟩≡ (13b)

```
$weight{$aggregation_name} += $st_blocks ;
$weight_all += $st_blocks ;
```

Uses aggregation_name 17b, st_blocks 17b, weight 17a, and weight_all 17b.

19e ⟨*count bytes* 19e⟩≡ (13b)

```
$weight{$aggregation_name} += $st_size ;
$weight_all += $st_size ;
```

Uses aggregation_name 17b, st_size 17b, weight 17a, and weight_all 17b.

19f ⟨*count slot* 19f⟩≡ (13b)

```
$weight{$aggregation_name} ++ ;
$weight_all ++ ;
```

Uses aggregation_name 17b, weight 17a, and weight_all 17b.

19g ⟨*detect directory* 19g⟩≡ (13b)

```
if ( -d _ ) {
        my $dh ;
        unless ( opendir( $dh, $_ ) ) {
                warn "$0: $!: opendir $_\n" ;
                next ;
        }
        my $root_dir = $_ eq '/'
                ? '/'
```

```
                : $_ . '/' ;
        my @entries =
                map { $root_dir . $_ }
                grep { !/^\.\.?$/ }
                readdir( $dh ) ;
        closedir( $dh ) ;
```

Defines:
  dh, never used.
  entries, never used.
  root_dir, never used.

20a ⟨*aggregate child size* 20a⟩≡ (13b)

```
  $weight{$aggregation_name} += $weight_children ;
```

Uses aggregation_name 17b, weight 17a, and weight_children 17b.

20b ⟨*total child size* 20b⟩≡ (13b)

```
  $weight_all += $weight_children ;
```

Uses weight_all 17b and weight_children 17b.

20c ⟨*return total* 20c⟩≡ (13b)

```
  return $weight_all ;
```

Uses weight_all 17b.

4.2 Reduce Phase

The Reduce Phase iterates over the accumulated objects in order by decreasing measurement, saving the top ten. As noted above, when using the directory aggregation method, it discards any directory object which would be in the top ten with one of its child objects. The Reduce Phase also adds the "grand total" object that shows the total of all measurements.

20d ⟨*dtt reduce* 20d⟩≡ (28a)

```
  my $top_counter = 10 ;
  my %topten ;
  sub purgeparents {
          my $parent_name ;
          $parent_name = $_[0] ;
          $parent_name =~ s/(.*)\/.*/$1/ or return ;
          if ( exists $topten{$parent_name} ) {
                  delete $topten{$parent_name} ;
                  ++$top_counter ;
          }
          purgeparents( $parent_name ) ;
  }
  foreach ( sort { $weight{$b} <=> $weight{$a} || $a cmp $b } keys %weight ) {
          purgeparents( $_ ) ;
```

```
        $topten{$_} = $weight{$_} ;
        last unless --$top_counter ;
  }
  my $grand_text = '(grand total - ' ;
  if ( $opt_dirent or $opt_inodes ) {
             if ( $opt_dirent ) { $grand_text .= 'directory entry count' }
        elsif ( $opt_inodes ) { $grand_text .= 'inodes count' }
        $grand_text .=
          $opt_human                    ? ( ' - customary binary units' )
        : $opt_si                       ? ( ' - si metric units' )
        : $opt_blocksize eq 'n'         ? ( ' x' . $starting_st_blksize )
        : defined $opt_blocksize        ? ( ' x' . $opt_blocksize )
        :                                 ( '' )
        ;
  }
  else {
        $grand_text .= $opt_apparent ? 'virtual size' : 'physical size' ;
        $grand_text .=
          $opt_human                    ? ( ' - customary binary units' )
        : $opt_si                       ? ( ' - si metric units' )
        : $opt_blocksize eq 'n'         ? ( ' ' . $starting_st_blksize  . 'B-blocks' )
        : defined $opt_blocksize        ? ( ' ' . $opt_blocksize        . 'B-blocks' )
        : $opt_implicit eq 'n'          ? ( ' ' . $starting_st_blksize  . 'B-blocks' )
        :                                 ( ' ' . $opt_implicit         . 'B-blocks' )
        ;
  }
  $grand_text .= ')' ;
  $topten{$grand_text} = $grand_total ;
```

Defines:
grand_text, never used.
parent_name, never used.
purgeparents, never used.
top_counter, never used.
topten, used in chunk 21.

Uses grand_total 13b, opt_apparent 5d, opt_blocksize 7c, opt_dirent 6c, opt_human 9a, opt_implicit 7c, opt_inodes 6c, opt_si 9d, starting_st_blksize 17a, and weight 17a.

4.3 Report Phase

The aggregated objects and measurements are typed on *stdout*, ordered by decreasing size and scaled in accord with the selected Scaling Control option.

21 ⟨*dtt report* 21⟩≡ (28a)

```
  format STDOUT =
```

```
@>>>>>>>>>>>>>>>> @*
scale_any( $topten{$_} ) , $_
.
foreach ( sort { $topten{$b} <=> $topten{$a} || $a cmp $b } keys %topten ) {
        #$opt_debug and warn "\$_=$_ \$topten{\$_}=$topten{$_}" ;
        write ;
}
```

Uses opt_debug 26b, scale_any 22a, and topten 20d.

22a ⟨*dtt function declarations* 7d⟩+≡ (28a) ◁7d 22b▷

```
sub scale_any {
        my $unscaled = $_[0] ;
        $unscaled *= 512 unless $opt_dirent or $opt_inodes or $opt_apparent ;
        return
          $opt_human                    ? scale_human( $unscaled )
        : $opt_si                       ? scale_si(    $unscaled )
        : $opt_blocksize eq 'n'         ? scale_fixed( $unscaled , $starting_st_blksize )
        : defined $opt_blocksize        ? scale_fixed( $unscaled , $opt_blocksize )
        : $opt_implicit eq 'n'          ? scale_fixed( $unscaled , $starting_st_blksize )
        :                                 scale_fixed( $unscaled , $opt_implicit )
        ;
}
```

Defines:
scale_any, used in chunk 21.
unscaled, never used.
Uses opt_apparent 5d, opt_blocksize 7c, opt_dirent 6c, opt_human 9a, opt_implicit 7c, opt_inodes 6c, opt_si 9d, scale_fixed 22b, scale_human 22c, scale_si 22c, and starting_st_blksize 17a.

22b ⟨*dtt function declarations* 7d⟩+≡ (28a) ◁22a 22c▷

```
sub scale_fixed { int( $_[0] / $_[1] + 0.5 ) }
```

Defines:
scale_fixed, used in chunk 22a.

22c ⟨*dtt function declarations* 7d⟩+≡ (28a) ◁22b

```
sub scale_human {
        my $to_count = $_[0] ;
        foreach ( split // , reverse ${\KMGTPEZYRQ} ) {
                $opt_debug and warn "\$_=$_" ;
                if ( $to_count >= BINARY_FACTORS->{$_} ) {
                        $to_count /= BINARY_FACTORS->{$_} ;
                        return ( $to_count < 10 ? int( 10 * $to_count + 0.5 ) / 10
                                                : int( $to_count + 0.5 ) ) . $_ ;
                }
        }
        return $to_count ;
}
sub scale_si {
```

```
        my $to_count = $_[0] ;
        foreach ( split // , reverse ${\KMGTPEZYRQ} ) {
                $opt_debug and warn "\$_=$_" ;
                if ( $to_count >= METRIC_FACTORS->{$_} ) {
                        $to_count /= METRIC_FACTORS->{$_} ;
                        return ( $to_count < 10 ? int( 10 * $to_count + 0.5 ) / 10
                                                : int( $to_count + 0.5 ) ) . $_ ;
                }
        }
        return $to_count ;
}
```

Defines:
scale_human, used in chunk 22a.
scale_si, used in chunk 22a.
to_count, never used.
Uses BINARY_FACTORS 13a, KMGTPEZYRQ 13a, METRIC_FACTORS 13a, and opt_debug 26b.

5 Author

Edward K. McGuire

6 Copyright And License

7 Notes

7.1 Details Of Operation

dtt is careful of multiple entries in the directory structure which catalog the same file (duplicate inodes). Storage allocated to an inode is not counted more than once.

The output will only be accurate if the operator's credentials permit the program to collect the information. The standard error output should be monitored for any errors indicating that the program did not have permission to scan a directory.

The native *dtt* unit of measure is the 512-byte block. This is for compatibility with *Unix dt/du*. If the system has *Gnu dt/du*, the program installer can configure *dtt* to use 1024-byte blocks using an environment variable.

7.2 Language Processor

The language of the program is strict Perl 5. It is syntax-compatible with Perl 5.6.0, but requires 5.6.1 or newer to work properly. Every release of the program is tested against the two newest Perl 5 stable release series, at their latest patch levels, and the Perl 5 releases that are included with the distros the program supports, plus the oldest Perl that still works with it. The program is not tested against the experimental release track.

This release of *dtt* has been tested with:

- Perl 5.38.2 (current stable release)
- Perl 5.36.1 (previous stable release)
- Perl 5.34.1 (stable release packaged with target platform Slackware 15.0)
- Perl 5.6.1 (oldest release that works)

25a ⟨*boilerplate:dtt* 25a⟩≡ (25d 30h 32c 34b 36f 39 40a)

```
This file is part of the dtt package.
```

25b ⟨*boilerplate:copyright* 25b⟩≡ (11h 25d 30h 32c 34b 36f 39 40a)

```
Copyright (c) 2011, 2022, 2023, 2024 Edward K. McGuire, Fort Worth, Texas.
```

25c ⟨*boilerplate:noweb* 25c⟩≡ (25d 30h 32c 34b 36f 39 40a)

```
# It was compiled from dtt.nw using Norman Ramsey's Noweb.
```

25d ⟨*dtt pragmas* 25d⟩≡ (28a)

```
#! /usr/bin/perl
# ⟨boilerplate:dtt 25a⟩
# ⟨boilerplate:copyright 25b⟩
# ⟨boilerplate:noweb 25c⟩
use v5.6.1 ;
use strict ;
use warnings ; no warnings 'uninitialized' ;
```

7.3 Acknowledgements

dtt was inspired by *du* in standard Unix. In fact, *du* is used to test *dtt*. What *du* doesn't do, and *dtt* does well, is aggregation by whole directory subtree or suffix with no redundant reporting.

I am indebted to N. Ramsey for *Noweb – A Simple, Extensible Tool for Literate Programming*, used to prepare *dtt* and its documentation for public release. Migrating *dtt* to *Noweb* has made *dtt* better software.

T. Oetiker, *et al.*, *The Not So Short Introduction to LaTeX*, was an essential reference.

P. Volkerding, *et al.*, *Slackware 15.0*, provided an ideal test platform for development.

D. Book, *Perl::MinimumVersion*, and K. Liu, *App::perlbrew*, made possible the support of *dtt* with older Perl releases.

7.4 Bugs

26a ⟨*dtt option specifications* 5b⟩+≡ (5a) ◁11e

```
'debug|d' => \$opt_debug,
```

Uses `opt_debug` 26b.

26b ⟨*dtt option variables* 5d⟩+≡ (28a) ◁11g

```
my $opt_debug = '' ;
```

Defines:
`opt_debug`, used in chunks 12c, 13b, 18a, 21, 22c, and 26a.

Gnu du seems to round up, but *dtt* rounds to nearest number. This is arguably a bug in *Gnu du*.

The `--suffix` results should be tested against a real-world filesystem ... but it's not clear how. Also, The three measurement types are well-tested, but the many new output scaling options, not so much.

dtt is careful never to read file blocks. On a sparse volume, such as a OneDrive or Google Drive File Stream, reading file blocks causes those blocks to be loaded over the network to the local cache. The cost of such an operation is heavy network traffic and loss of free blocks on the local disk. However, *dtt* must read directories. This unavoidably causes directory files to be cached, and incurs the network traffic cost to cache them.

A conflict between multiple blocksizes and `--blocksize=n` should maybe fall back to a local system default scale factor, instead of always 512.

7.5 Roadmap

dtt should be packaged for more platforms than *Slackware 15.0*. When packaged for *Slackware 15.0*, the default unit of measure should be customized to 1024 to match the behavior of *Gnu df/du* on the target platform; the same for other platforms having *Gnu df/du*.

How to handle an overlayfs is an interesting question.

Consider implementing `-n|--number` or something like it to change top "ten" to top "some other number". Consider adding `-i` for `--inodes`. Suggested by pan64 at LinuxQuestions.org.

8 Compilation, Testing, and Packaging

The source package `dtt.nw` is a single file that contains the text of the program, and the documentation files, plus all the commands needed to compile them.

The compilation process expects *Noweb*, LaTeX, and *latex2man* to be present in the system library. The testing process further expects *Perl::MinimumVersion* and *App::perlbrew*, and the various Perl releases to be tested against.

8.1 Executable

The file `dtt` is the executable program. *Noweb* compiles it from code blocks in the master source file:

28a ⟨*dtt* 28a⟩≡
- ⟨*dtt pragmas* 25d⟩
- ⟨*dtt constants* 13a⟩
- ⟨*dtt database* 17a⟩
- ⟨*dtt option variables* 5d⟩
- ⟨*dtt function declarations* 7d⟩
- ⟨*dtt set option arguments* 5a⟩
- ⟨*dtt type help and exit* 10h⟩
- ⟨*dtt type version and exit* 11h⟩
- ⟨*dtt set pathname arguments* 12c⟩
- ⟨*dtt lookup* 13b⟩
- ⟨*dtt check for blksize changed* 19a⟩
- ⟨*dtt reduce* 20d⟩
- ⟨*dtt report* 21⟩

This code is written to file `dtt`.

28b ⟨*makefile all targets* 28b⟩≡ (32c) 29c▷
```
all :: dtt
```

28c ⟨*makefile object targets* 28c⟩≡ (32c) 29a▷
```
dtt : dtt.nw.sentinel ; chmod 755 $@
```

28d ⟨*makefile test targets* 28d⟩≡ (32c)
```
test :: ; ./test-dtt
```

28e ⟨*makefile clean targets* 28e⟩≡ (32c) 29b▷
```
clean :: ; rm -f dtt
```

8.2 Manuals

The full technical manual is produced from a documentation master in LaTeX format, `dtt.tex`, which is derived from the *Noweb* source.

29a ⟨*makefile object targets* 28c⟩+≡ (32c) ◁28c 29d▷
```
dtt.tex : dtt.nw ; noweave -latex $< >$@
```

29b ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁28e 29e▷
```
clean :: ; rm -f dtt.tex
```

From `dtt.tex`, the full technical manual is produced in PDF:

29c ⟨*makefile all targets* 28b⟩+≡ (32c) ◁28b 29f▷
```
all :: dtt.pdf
```

29d ⟨*makefile object targets* 28c⟩+≡ (32c) ◁29a 29g▷
```
dtt.pdf : dtt.tex ; latexmk -pdf $<
```

29e ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁29b 29h▷
```
clean :: ; latexmk -C dtt
```

and in HTML:

29f ⟨*makefile all targets* 28b⟩+≡ (32c) ◁29c 29k▷
```
all :: dtt.html
```

29g ⟨*makefile object targets* 28c⟩+≡ (32c) ◁29d 29i▷
```
dtt.html : dtt.nw
        noweave -html -filter 'l2h' -index -autodefs c $< >$@.tmp
        htmltoc <$@.tmp >$@
        rm $@.tmp
```

29h ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁29e 29j▷
```
clean :: ; rm -f dtt.html dtt.html.tmp
```

The operating manual is produced from a smaller documentation master in LaTeX format, `dtt-nocode.tex`, which is also derived from the *Noweb* source but excludes the code blocks.

29i ⟨*makefile object targets* 28c⟩+≡ (32c) ◁29g 30a▷
```
dtt-nocode.tex : dtt.nw ; noweave -latex -filter 'elide "*"' $< >$@
```

29j ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁29h 30b▷
```
clean :: ; rm -f dtt-nocode.tex
```

From `dtt-nocode.tex` is generated an operating manual master in *troff* format, using *latex2man*. This becomes the man page `dtt.1`. As part of the process, this section on compilation, testing, and packaging is also excluded, being bracketed by `%@% IF NOTMANPAGE %@%` and `%@% END-IF %@%` markers.

29k ⟨*makefile all targets* 28b⟩+≡ (32c) ◁29f 30c▷
```
all :: dtt.1
```

30a ⟨*makefile object targets* 28c⟩+≡ (32c) ◁29i 30d▷
```
dtt.1 : dtt-nocode.tex dtt.trans
        latex2man -CMANPAGE -t./dtt.trans -M dtt-nocode.tex dtt.1
```

30b ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁29j 30e▷
```
clean :: ; rm -f dtt.1
```

Then, from the *troff* master, a PDF is generated.

30c ⟨*makefile all targets* 28b⟩+≡ (32c) ◁29k 33a▷
```
all :: dtt.1.pdf
```

30d ⟨*makefile object targets* 28c⟩+≡ (32c) ◁30a 30f▷
```
dtt.1.pdf : dtt.1 ; man -Tpdf ./$< >$@
```

30e ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁30b 30g▷
```
clean :: ; rm -f dtt.1.pdf
```

8.2.1 *latex2man* translation rules

As *latex2man* converts LaTeX input to *troff*, *HTML*, or *texinfo* output, it makes sense of the TeX macros it encounters. Out of the box, it recognizes many stock TeX and LaTeX macros, plus a set of macros in its own LaTeX package. Other macros can be used, provided that mappings to *troff*, *HTML*, and *texinfo* macros are given. The file `dtt.trans` gives these translations. In most cases, the translations simply no-op the macros.

30f ⟨*makefile object targets* 28c⟩+≡ (32c) ◁30d 33b▷
```
dtt.trans : dtt.nw.sentinel ;
```

30g ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁30e 33c▷
```
clean :: ; rm -f dtt.trans
```

30h ⟨*dtt.trans* 30h⟩≡ 30i▷
```
# ⟨boilerplate:dtt 25a⟩
# ⟨boilerplate:copyright 25b⟩
# ⟨boilerplate:noweb 25c⟩
```
This code is written to file `dtt.trans`.

These mappings no-op macros used from the `noweb` package:

30i ⟨*dtt.trans* 30h⟩+≡ ◁30h 31a▷
```
$manMacro1a{'nwbegindocs'} = '\n.\" '    ; $manMacro1b{'nwbegindocs'} = '' ;
$htmlMacro1a{'nwbegindocs'} = ''         ; $htmlMacro1b{'nwbegindocs'} = '' ;
$texiMacro1a{'nwbegindocs'} = ''         ; $texiMacro1b{'nwbegindocs'} = '' ;
$manMacro1a{'nwenddocs'} = '\n.\" '      ; $manMacro1b{'nwenddocs'} = '' ;
$htmlMacro1a{'nwenddocs'} = ''           ; $htmlMacro1b{'nwenddocs'} = '' ;
```

```
$texiMacro1a{'nwenddocs'} = ''          ; $texiMacro1b{'nwenddocs'} = '' ;
$manMacro{'nwdocspar'} = '' ;
$htmlMacro{'nwdocspar'} = '' ;
$texiMacro{'nwdocspar'} = '' ;
$manMacro{'nowebindex'} = '' ;
$htmlMacro{'nowebindex'} = '' ;
$texiMacro{'nowebindex'} = '' ;
```

No-op the table of contents.

31a ⟨*dtt.trans* 30h⟩+≡ ◁30i 31b▷

```
$manMacro{'tableofcontents'} = '' ;
$htmlMacro{'tableofcontents'} = '' ;
$texiMacro{'tableofcontents'} = '' ;
```

?

31b ⟨*dtt.trans* 30h⟩+≡ ◁31a 31c▷

```
#$manMacro{'PROG'} = '' ;
#$htmlMacro{'PROG'} = '' ;
#$texiMacro{'PROG'} = '' ;
#$manMacro{'TEXTTT'} = '' ;
#$htmlMacro{'TEXTTT'} = '' ;
#$texiMacro{'TEXTTT'} = '' ;
#$manMacro1a{'TEXTTT'} = ''            ; $manMacro1b{'TEXTTT'} = '' ;
#$htmlMacro1a{'TEXTTT'} = ''           ; $htmlMacro1b{'TEXTTT'} = '' ;
#$texiMacro1a{'TEXTTT'} = ''           ; $texiMacro1b{'TEXTTT'} = '' ;
```

No-op newpage.

31c ⟨*dtt.trans* 30h⟩+≡ ◁31b 31d▷

```
$manMacro{'newpage'} = '' ;
$htmlMacro{'newpage'} = '' ;
$texiMacro{'newpage'} = '' ;
```

No-op italics.

31d ⟨*dtt.trans* 30h⟩+≡ ◁31c 31e▷

```
$manMacro1a{'textit'} = ''             ; $manMacro1b{'textit'} = '' ;
$htmlMacro1a{'textit'} = ''            ; $htmlMacro1b{'textit'} = '' ;
$texiMacro1a{'textit'} = ''            ; $texiMacro1b{'textit'} = '' ;
```

Superscript.

31e ⟨*dtt.trans* 30h⟩+≡ ◁31d 32a▷

```
$manMacro1a{'textsuperscript'} = '^('   ; $manMacro1b{'textsuperscript'} = ')' ;
$htmlMacro1a{'textsuperscript'} = '^('  ; $htmlMacro1b{'textsuperscript'} = ')' ;
$texiMacro1a{'textsuperscript'} = '^('  ; $texiMacro1b{'textsuperscript'} = ')' ;
```

No-op href.

32a ⟨*dtt.trans* 30h⟩+≡ ◁31e 32b▷
```
$manMacro2a{'href'}  = '\n.\" ' ; $manMacro2b{'href'}  = '\n' ; $manMacro2c{'href'} = '' ;
$htmlMacro2a{'href'} = '' ; $htmlMacro2b{'href'} = '' ; $htmlMacro2c{'href'} = '' ;
$texiMacro2a{'href'} = '' ; $texiMacro2b{'href'} = '' ; $texiMacro2c{'href'} = '' ;
```

Carry over the LaTeX macro as the text "LaTeX".

32b ⟨*dtt.trans* 30h⟩+≡ ◁32a
```
$manMacro1a{'LaTeX'} = ''              ; $manMacro1b{'LaTeX'} = 'LaTeX' ;
$htmlMacro1a{'LaTeX'} = ''             ; $htmlMacro1b{'LaTeX'} = 'LaTeX' ;
$texiMacro1a{'LaTeX'} = ''             ; $texiMacro1b{'LaTeX'} = 'LaTeX' ;
```

8.3 Makefile

The build process is managed by *make*. The file `makefile` contains the commands to (i) produce the executable file, the operator manual, and the developer manual; and, (ii) packages everything up into target-system installers.

The `makefile` expects an argument, normally one of:

make (by itself) types help for the options.

make all compiles the executable file, manual, technical paper, and package files from source.

make test runs correctness testing against the files compiled by "make all".

make slackpkg packages the files into a Slackware binary package.

make clean deletes intermediate files.

make commit commits changes to the *git* repository.

make website publishes the latest version.

Here, `makefilie` is outlined. Elsewhere, individual objects add their build, test, and clean targets.

32c ⟨*makefile* 32c⟩≡
```
# ⟨boilerplate:dtt 25a⟩
# ⟨boilerplate:copyright 25b⟩
# ⟨boilerplate:noweb 25c⟩
⟨makefile macros 38d⟩
usage ::
        @echo usage:
```

```
        @echo make all - build from source
        @echo make test - test the build
        @echo make slackpkg - package the build for Slackware
        @echo make clean - erase the build
        @echo make commit - push changes to git
        @echo make website - push website source files to web server
all ::
⟨makefile all targets 28b⟩
test :: all
⟨makefile test targets 28d⟩
slackpkg :: all
⟨makefile slackpkg targets 37e⟩
⟨makefile object targets 28c⟩
clean :: ; rm -f *~ .*~
⟨makefile clean targets 28e⟩
website ::
⟨makefile website targets 37f⟩
commit :: ; git commit -av -uno
```

This code is written to file `makefile`.

The makefile also produces itself from source.

33a ⟨makefile all targets 28b⟩+≡ (32c) ◁30c 33d▷

```
all :: makefile
```

33b ⟨makefile object targets 28c⟩+≡ (32c) ◁30f 33e▷

```
makefile : dtt.nw.sentinel ;
```

33c ⟨makefile clean targets 28e⟩+≡ (32c) ◁30g 34a▷

```
clean :: ; rm -f makefile
```

8.4 Testing

The test script `test-dtt` checks the output of *dtt*. It runs the `dtt` executable, and compares its output to that of other utilities, to the extent that is possible.

Testing is best done on a quiet system. Changes to files during the test run can yield false positives. To keep things quiet as possible, the test is run against the `/usr` subtree, in which change is relatively rare. To avoid rounding differences generating false positives, file lengths are compared by byte, not by block.

33d ⟨makefile all targets 28b⟩+≡ (32c) ◁33a 36c▷

```
all :: test-dtt
```

33e ⟨makefile object targets 28c⟩+≡ (32c) ◁33b 36a▷

```
test-dtt : dtt.nw.sentinel ; chmod 755 $@
```

34a ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁33c 36b▷

```
clean :: ; rm -f test-dtt
```

34b ⟨*test-dtt* 34b⟩≡ 34c▷

```
#! /bin/sh -e
# ⟨boilerplate:dtt 25a⟩
# ⟨boilerplate:copyright 25b⟩
# ⟨boilerplate:noweb 25c⟩
```

This code is written to file `test-dtt`.

The syntax test verifies the code is still syntax-compatible with Perl 5.6.0 as reported by `perlver`.

34c ⟨*test-dtt* 34b⟩+≡ ◁34b 34d▷

```
echo syntax test:
EXPLVER=v5.6.0
PERLVER=‘  perlver dtt  |  awk ’/dtt/ {print $6}’  ‘
echo ’ EXPLVER=’$EXPLVER
echo ’ PERLVER=’$PERLVER
test "$PERLVER" = "$PERLVER"
```

The results test verifies the code returns the same count as similar utilities. The code is testes against every Perl installation loaded into *perlbrew*.

- To list Perls:`perlbrew list`
- To add a new Perl: `perlbrew install perl-(version-string)`.
- To drop an old Perl: `perlbrew uninstall perl-(version-string)`.

34d ⟨*test-dtt* 34b⟩+≡ ◁34c

```
echo results test:
FINDCT=‘  find /usr -xdev  |  wc -l   |  awk ’{print $1;}’  ‘
DUPHYS=‘  du -B512 -s -x /usr  | awk ’{print $1;}’ ‘
DUVIRT=‘  du --apparent-size -B1 -s -x /usr | awk ’{print $1;}’ ‘
DUINOD=‘  du --inodes -s -x /usr | awk ’{print $1;}’ ‘
for TESTPERL in ‘ perlbrew list ‘
do
        echo testing with $TESTPERL
        echo ’    dirent count test’
        DTT=‘
                perlbrew --verbose \
                        exec --with $TESTPERL \
                        perl ./dtt -x --dirent /usr  |
                head -1  |
                awk ’{print $1;}’
        ‘
```

```
        echo '        FINDCT='$FINDCT
        echo '           DTT='$DTT
        test "$DTT" -eq "$FINDCT"
        echo '    inodes count test'
        DTT=`
                perlbrew --verbose \
                        exec --with $TESTPERL \
                        perl ./dtt -x --inodes /usr  |
                head -1  |
                awk '{print $1;}'
        `
        echo '        DUINOD='$DUINOD
        echo '           DTT='$DTT
        test "$DTT" -eq "$DUINOD"
        echo '    blocks test'
        DTT=`
                perlbrew --verbose \
                        exec --with $TESTPERL \
                        perl ./dtt -x /usr  |
                head -1  |
                awk '{print $1;}'
        `
        echo '        DUPHYS='$DUPHYS
        echo '           DTT='$DTT
        test "$DTT" -eq "$DUPHYS"
        echo '    bytes test'
        DTT=`
                perlbrew --verbose \
                        exec --with $TESTPERL \
                        perl ./dtt -x --apparent-size -B1 /usr  |
                head -1  |
                awk '{print $1;}'
        `
        echo '        DUVIRT='$DUVIRT
        echo '           DTT='$DTT
        test "$DTT" -eq "$DUVIRT"
done
```

8.5 Sentinel file

A sentinel file represents multiple outputs from a single source file. `dtt.nw` is the common source for the executable and its manuals. `dtt.nw.sentinel` represents these multiple outputs.

36a ⟨*makefile object targets* 28c⟩+≡ (32c) ◁33e 36d▷
```
dtt.nw.sentinel : dtt.nw ; noweb $<
```

36b ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁34a 36e▷
```
clean :: ; rm -f dtt.nw.sentinel
```

8.6 Packaging for Slackware

The *dtt* `makefile` can build a package for *Slackware 15.0*.

Produce the Slackware 15.0 package description.

36c ⟨*makefile all targets* 28b⟩+≡ (32c) ◁33d 36g▷
```
all :: slack-desc
```

36d ⟨*makefile object targets* 28c⟩+≡ (32c) ◁36a 36h▷
```
slack-desc : dtt.nw.sentinel ;
```

36e ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁36b 37a▷
```
clean :: ; rm -f slack-desc
```

36f ⟨*slack-desc* 36f⟩≡
```
|2345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678|
|⟨boilerplate:dtt 25a⟩
|⟨boilerplate:copyright 25b⟩
|⟨boilerplate:noweb 25c⟩
dtt: dtt ⟨dtt version 11d⟩ (disk "top ten" list)
dtt:
dtt: dtt types a "top ten" ranking of directory trees and files found on a file-
dtt: structured storage device such as a Unix or Windows volume. In other words,
dtt: it floats the really big chunks to the top.
dtt:
dtt:
dtt:
dtt:
dtt:
dtt: dtt, metaed, com
```
This code is written to file `slack-desc`.

Produce the *README* from `slack-desc` by recognizing lines that start with "dtt:".

36g ⟨*makefile all targets* 28b⟩+≡ (32c) ◁36c 38e▷
```
all :: README
```

36h ⟨*makefile object targets* 28c⟩+≡ (32c) ◁36d 38f▷
```
README : slack-desc
        <slack-desc >README sed ' s/^dtt: *\(..*\)/\1/  ;  tPRINT  ;  d  ;  :PRINT '
```

37a ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁36e 37g▷
```
clean :: ; rm -f README
```

This *make* recipe is basically the slackbuild script for the package. It should eventually become exactly that, so it can be submitted to *SB.org*.

The `slackpkg` target is elsewhere made dependent on `all` so it does not even get started until everything has been compiled.

37b ⟨*slackware BIN* 37b⟩≡ (37e)
```
/usr/bin
```

37c ⟨*slackware DOC* 37c⟩≡ (37e)
```
/usr/doc/dtt-⟨dtt version 11d⟩
```

37d ⟨*slackware MAN* 37d⟩≡ (37e)
```
/usr/man/man1
```

37e ⟨*makefile slackpkg targets* 37e⟩≡ (32c)
```
slackpkg ::
        sudo rm -rf staging
        sudo mkdir -p staging/install
        sudo cp slack-desc staging/install
        sudo mkdir -p staging⟨slackware BIN 37b⟩
        sudo install dtt staging⟨slackware BIN 37b⟩
        sudo mkdir -p staging⟨slackware DOC 37c⟩
        sudo cp dtt.pdf dtt.html dtt.nw README CHANGELOG.html staging⟨slackware DOC 37c⟩
        sudo mkdir -p staging⟨slackware MAN 37d⟩
        sudo cp dtt.1 staging⟨slackware MAN 37d⟩
        sudo gzip staging⟨slackware MAN 37d⟩/dtt.1
        cd staging && sudo makepkg -l y -c n ../dtt-⟨dtt version 11d⟩-noarch-1me.tgz
```

37f ⟨*makefile website targets* 37f⟩≡ (32c) 39a▷
```
website :: slackpkg
        cp dtt-⟨dtt version 11d⟩-noarch-1me.tgz $(PAPERS)
```

37g ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁37a 38c▷
```
clean :: ; sudo rm -rf staging
```

8.7 Packaging for Cygwin

https://cygwin.com/packaging-package-files.html

37h ⟨*dtt.cygport* 37h⟩≡
```
NAME="git"
VERSION=⟨dtt version 11d⟩
RELEASE=1
```

```
CATEGORY="Utils"
SUMMARY="Types a top ten ranking of directories and files"
DESCRIPTION="dtt is a utility that aggregates disk usage by directory branch,
filename, or suffix and types a top ten ranking. It can report physical size,
virtual size, or file counts. It offers several ways to scale totals for easy
reading or for post-processing by a script."
HOMEPAGE="https://metaed.com/papers/dtt"
LICENSE="MIT-style"
```

This code is written to file `dtt.cygport`.

38a ⟨*makefile DISABLED* 38a⟩≡

```
CYGMAN = /usr/share/man/man1
CYGDOC = /usr/share/doc/dtt-⟨dtt version 11d⟩
cygpackage :: all
        sudo rm -rf cygstaging
        sudo mkdir -p cygstaging$(BIN)
        sudo install dtt cygstaging$(BIN)
        sudo mkdir -p cygstaging$(CYGDOC)
        sudo cp dtt.pdf dtt.html dtt.nw CHANGELOG.html cygstaging$(CYGDOC)
        sudo mkdir -p cygstaging$(CYGMAN)
        sudo cp dtt.1 cygstaging$(CYGMAN)
        sudo gzip cygstaging$(CYGMAN)/dtt.1
        cd cygstaging && sudo tar cfv ../dtt-⟨dtt version 11d⟩-1.tar.xz --xz *
```

38b ⟨*makefile website targets DISABLED* 38b⟩≡

```
website :: cygpackage
        cp dtt-⟨dtt version 11d⟩-1.tar.xz $(PAPERS)
```

38c ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁37g 38g▷

```
clean :: ; sudo rm -rf cygstaging
```

8.8 Packaging for website

These *make* targets update the website directly.

38d ⟨*makefile macros* 38d⟩≡ (32c)

```
PAPERS = /var/www/metaed.com/root/papers/dtt
```

38e ⟨*makefile all targets* 28b⟩+≡ (32c) ◁36g 46a▷

```
all :: header.html footer.html htaccess
```

38f ⟨*makefile object targets* 28c⟩+≡ (32c) ◁36h 46b▷

```
header.html footer.html htaccess : dtt.nw.sentinel ;
```

38g ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁38c 46d▷

```
clean :: ; rm -f header.html footer.html htaccess
```

39a ⟨*makefile website targets* 37f⟩+≡ (32c) ◁37f 46c▷

```
website :: $(PAPERS)/header.html $(PAPERS)/footer.html $(PAPERS)/.htaccess
website :: $(PAPERS)/dtt.pdf $(PAPERS)/dtt.nw $(PAPERS)/dtt.1.pdf
#website :: $(PAPERS)/dtt.html
$(PAPERS) : ; mkdir -p $@
$(PAPERS)/header.html    : $(PAPERS) header.html    ; cp header.html    $@
$(PAPERS)/footer.html    : $(PAPERS) footer.html    ; cp footer.html    $@
$(PAPERS)/.htaccess      : $(PAPERS) htaccess       ; cp htaccess       $@
$(PAPERS)/dtt.pdf        : $(PAPERS) dtt.pdf        ; cp dtt.pdf        $@
$(PAPERS)/dtt.nw         : $(PAPERS) dtt.nw         ; cp dtt.nw         $@
$(PAPERS)/dtt.1.pdf      : $(PAPERS) dtt.1.pdf      ; cp dtt.1.pdf      $@
#$(PAPERS)/dtt.html      : $(PAPERS) dtt.html       ; cp dtt.html       $@
```

39b ⟨*header.html* 39b⟩≡

```
<!-- ⟨boilerplate:dtt 25a⟩ -->
<!-- ⟨boilerplate:copyright 25b⟩ -->
<!-- ⟨boilerplate:noweb 25c⟩ -->
<h1> metaed.com/papers/dtt </h1>
<p>
This is the home of <code>dtt</code>, a utility that types a \top ten" ranking
of directory trees and files found on a file-structured storage device such as a
Unix or Windows volume. In other words, it floats the really big chunks to the
top.
```

This code is written to file `header.html`.

39c ⟨*footer.html* 39c⟩≡

```
<!-- ⟨boilerplate:dtt 25a⟩ -->
<!-- ⟨boilerplate:copyright 25b⟩ -->
<!-- ⟨boilerplate:noweb 25c⟩ -->
<p>
⟨boilerplate:copyright 25b⟩
All rights reserved.
Redistribution and use of this software, with or without modification, is
permitted, provided that the following conditions are met:
<p>
1. Redistribution of this software must retain the copyright notice above, this
list of conditions, and the disclaimer below.
<p>
THIS SOFTWARE IS PROVIDED BY THE AUTHOR \AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA OR PROFITS; OR
```

```
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCY OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.
```

This code is written to file `footer.html`.

40a ⟨*htaccess* 40a⟩≡ 40b▷

```
# ⟨boilerplate:dtt 25a⟩
# ⟨boilerplate:copyright 25b⟩
# ⟨boilerplate:noweb 25c⟩
<IfModule autoindex_module>
Options         +Indexes
HeaderName      header
ReadmeName      footer
IndexOptions    FancyIndexing IgnoreClient VersionSort Charset=UTF-8 \
                NameWidth=* DescriptionWidth=*
```

This code is written to file `htaccess`.

Here we tell the autoindex module not to index hidden files. The syntax of IndexOptions filename patterns makes it impossible to ignore two-character hidden files without also ignoring `..` (parent directory), so we ignore three-character hidden files and longer.

40b ⟨*htaccess* 40a⟩+≡ ◁40a 40c▷

```
IndexIgnore     .??*
IndexIgnore     *~
```

Also do not index the header and footer pieces.

40c ⟨*htaccess* 40a⟩+≡ ◁40b 40d▷

```
IndexIgnore     header.html footer.html
```

Here we give a short description of each file. Again the syntax is painful. Notably, the autoindex module does *partial* matching of listed filename(s) to URLs, causing unintended matches. Effectively, each name has an implied wildcard at the beginning and end. This ambiguity of a name has to be resolved by the order of the directives. The lines are carefully grouped by filetype and sorted for correct matching.

`.nw` files:

40d ⟨*htaccess* 40a⟩+≡ ◁40c 40e▷

```
AddDescription  "Current source (Noweb)"                dtt.nw
```

`.pdf` files:

40e ⟨*htaccess* 40a⟩+≡ ◁40d 41a▷

```
AddDescription  "Current technical paper (PDF)"         dtt.pdf
AddDescription  "Current man page (PDF)"                dtt.1.pdf
```

`.tgz` files:

41a ⟨*htaccess* 40a⟩+≡ ◁40e 41b▷

```
AddDescription  "dtt 1.0.1a Slackware 15.0 installer"   dtt-1.0.1a-noarch-1me.tgz
AddDescription  "dtt 1.1a Slackware 15.0 installer"     dtt-1.1a-noarch-1me.tgz
AddDescription  "dtt 2a Slackware 15.0 installer"       dtt-2a-noarch-1me.tgz
AddDescription  "dtt 2b Slackware 15.0 installer"       dtt-2b-noarch-1me.tgz
AddDescription  "dtt 2.1a Slackware 15.0 installer"     dtt-2.1a-noarch-1me.tgz
AddDescription  "dtt 2.1b Slackware 15.0 installer"     dtt-2.1b-noarch-1me.tgz
AddDescription  "dtt 2.1g Slackware 15.0 installer"     dtt-2.1g-noarch-1me.tgz
AddDescription  "dtt 2.2g Slackware 15.0 installer"     dtt-2.2g-noarch-1me.tgz
AddDescription  "dtt 2.3a Slackware 15.0 installer"     dtt-2.3a-noarch-1me.tgz
```

`.xz` files:

41b ⟨*htaccess* 40a⟩+≡ ◁41a 41c▷

```
AddDescription  "dtt 2.1a Cygwin installer"             dtt-2.1a-1.tar.xz
```

`.html` files:

41c ⟨*htaccess* 40a⟩+≡ ◁41b 41d▷

```
AddDescription  "Current changelog (HTML)"              CHANGELOG.html
```

And that is the end of the autoindex container.

41d ⟨*htaccess* 40a⟩+≡ ◁41c

```
</IfModule>
```

9 Changelog

Key to the Greek alphabet used in the release tags below:

- "a" stands for α (alpha) - experimental, for feedback on new features
- "b" stands for β (beta) - stable candidate, for field testing
- "g" stands for γ (gamma) - stable, for general use

2.2 (2024-02-07)

Summary: Version 2.2 is not a feature release. It is a simplification of the build process and general source and documentation cleanup.

Added

- Create `README` from the text in `slack-desc`

Fixed

- Eliminate unnecessary page breaks from developer manual
- Add line breaks in code where long lines did not fit between margins
- Rename some identifiers to avoid matching words in strings being cross referenced by *Noweb*

Changed

- Retire build-dtt.nw, moving the build process back into `dtt.nw`
- Convert changelog source from HTML to LaTeX, and generate the HTML from the LaTeX (like the manpage)
- Add *latex2man* conditionals to let the text be subsetted into the *man* page or the changelog file
- Tighten up whitespace in bulleted and numbered lists
- Wordsmith and copyedit here and there in the documentation
- Simplify production of the usage message
- Use starred versions of section macros in the changelog, so prenumbered text does not get numbered again by LaTeX

2.1g (2024-01-29)

Summary: 2.1 gamma (general release). No program changes from 2.1 beta. Cosmetic fixes to the website.

Fixed

- Corrected the file descriptions on the project landing page
- Updated the package description everywhere to be consistent

2.1b (2024-01-25)

Summary: 2.1 beta. Bug fixes, documentation cleanup, and the new –inodes option.

Changed

- Type –help and –version messages on stdout, not stderr
- Minor documentation copyediting
- Include build-dtt.nw and build-dtt.pdf in the Slack package

2.1a (2024-01-24)

Summary: 2.1 alpha. Bug fixes, documentation cleanup, and the new –inodes option.

Added

- Support for Cygwin packaging
- Clarify warning and fatal conditions and use of stdout/stderr
- Add –inodes option (like Gnu df/du –inodes)
- Support Perl 5.38.2 and clearly document the exact Perl versions tested against

Fixed

- Index some missing identifiers

Changed

- Separate the utility itself from its build process.
- Copyediting and minor rewriting of documentation to read better
- Treat an error in option parsing as fatal. Suggested by LinuxQuestions.org member 0XBF

2b (2023-05-02)

Summary: eliminate the one-filesystem limitation; backport all the way to Perl 5.6.1; make command-line options compatible with *Unix df/du* and *Gnu df/du* where practical; let installer

or end-user customize the reporting blocksize using an environment variable, and scale output by native filesystem blocksize in addition to classic metric and SI units. Plus much copyediting to make the manual clearer.

No code changes were introduced in promotion to beta.

Added

- Backport as far as possible: Perl 5.6.1
- Add –version option
- Add –block-size—B option
- Add –human-readable and –si
- Add –one-file-system—x option

Changed

- Copyedit error messages for clarity
- Make –help conflict with –version
- Change –bytes—b to –apparent-size, like Gnu df/du

  Note that –bytes had a side effect: it changed the output scale to bytes. This side effect is gone. –apparent-size can be combined with –block-size=1 to get this behavior.

- Change –help—h to just –help, to make -h available for human-readable option, like Gnu df/du
- Change –count—c to –dirent—d to resolve conflict with du -c
- Remove the one-filesystem limitation

Removed

- Take out erroneous single-letter-option negation (for backport)

2a (2023-05-01) – YANKED

1.1a (2023-04-25)

Third alpha release incorporates tests of *dtt*'s byte and slot counts. The program runs faster. The documentation is better.

Added

- Test `--bytes` using `du --apparent-size`
- Test `--count` using `find`
- Add roadmap

- Add changelog
- Add acknowledgements

Fixed

- Type clearer message when `--help` requested
- Copyedit "make test" output
- Copyedit documentation
- Clean up compiled file header lines

Changed

- Return success code when `--help` requested
- Reduce wasted CPU by breaking monolithic lookup phase into option-specific code sections with simpler logic
- Move option help to option processing sections

1.0.1a (2023-04-23)

Rerelease of 1.0a with the correct Slackware 15.0 package file name.

Added

- Add man page using `latex2man`
- Add license conditions
- Test block counting using `du`
- Add a Slackware 15.0 package file
- Add homepage using `autoindex_module`

Changed

- Copyedit documentation
- Break up option processing into sections for clarity
- Rename variables for clarity

1.0a (2023-04-21) – YANKED

First alpha release. This has been a useful personal tool for more than ten years, and I am sharing it with the Unix community. This release was yanked to correct the name of the Slackware 15.0 package file. For the list of changes introduced to make it sharable, see the 1.0.1a changelog.

latex2man does not recognize starred versions of LaTeX macros. The *sed* expression below lets `[[sub]sub]section*` macros be treated as `[[sub]sub]section` macros.

46a ⟨*makefile all targets* 28b⟩+≡ (32c) ◁38e
```
all :: CHANGELOG.html
```

46b ⟨*makefile object targets* 28c⟩+≡ (32c) ◁38f
```
#CHANGELOG.html : dtt.nw.sentinel ;
CHANGELOG.html : dtt-nocode.tex dtt.trans
        tmp=‘ echo ’mkstemp(/tmp/dttXXXXXX)’ | m4 ‘ ; \
        <dtt-nocode.tex >>$$tmp sed ’s/section[*]/section/’ ; \
        latex2man -H -CCHANGELOG -t./dtt.trans $$tmp $@
```

46c ⟨*makefile website targets* 37f⟩+≡ (32c) ◁39a
```
website :: $(PAPERS)/CHANGELOG.html
$(PAPERS)/CHANGELOG.html : $(PAPERS) CHANGELOG.html ; cp CHANGELOG.html $@
```

46d ⟨*makefile clean targets* 28e⟩+≡ (32c) ◁38g
```
clean :: ; rm -f CHANGELOG.html
```

10 Index of identifiers