```
#include <iostream>
#include <list>
#include <sys/stat.h>
#include <errno.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <string.h>
using namespace std;

// man readdir(3), lstat, opendir, closedir.

// Format of a directory entry:
// http://www.delorie.com/gnu/docs/glibc/libc_270.html

// Testing the type of a file:
// http://www.delorie.com/gnu/docs/glibc/libc_286.html
// (Test the st_mode field returned by stat on a given file.)

// To get Linux to use ascii ordering "export LANG=us.ascii"

// handy macro for iterating through an stl container
#define each(I) \
  for( typeof((I).begin()) it=(I).begin(); it!=(I).end(); ++it )


int visit( string root ) {             // recursive visitor function

  // OPEN root
  DIR* dirp;                                              // open DIR
  if ( ! ( dirp = opendir( root.c_str() ) ) ) {
    cerr << "Cannot open directory " << root << ".\n";
    return -1;
  }

  // CREATE TWO LISTS OF FILE NAMES: file and hardSubdirectory
  list<string> file;          // names of each file in this dirctory
  list<string> hardSubdirectory;    // each hard-linked subdirectory
  while ( dirent* dp = readdir(dirp) ) {
    string s = dp->d_name;          // converts C string to C++ string
    if ( s == "." || s == ".." ) continue;             // skip these
    s = root + "/" + s;                      // prepend the current path
    file.push_back( s );
    if ( ( dp->d_type & DT_DIR ) && !(dp->d_type & DT_LNK) ) {
      hardSubdirectory.push_back( s );
    }
  }
  closedir(dirp);         // close DIR asap, to reset internal data

  // EMIT root's HEADER, INCLUDING ITS TOTAL SIZE
  cout << root << ":" << endl;
  cout << "total ";
  int size = 0;
  each( file ) {
    string filename = *it;
    struct stat st;   // "struct" needed because stat() is a defined
    if ( lstat( filename.c_str(), &st ) == 0 ) size += st.st_blocks;
  }
  cout << size/2 << endl;    // kilobytes-per-block correcton factor

  // lstat() AND REPORT ON EACH FILE WHOSE NAME IS IN root
  file.sort();
  each( file ) {
    string filename = *it;

    struct stat st;       // "struct stat" because stat() is defined
    if ( lstat( filename.c_str(), &st ) != 0 ) {
      cerr << "Cannot stat file " << filename
           << ": " << strerror(errno) << endl;
      return -1;
    }

    cout << ( (S_ISDIR(st.st_mode) != 0) ? 'd' : '-' )
```



```
         << ( (st.st_mode & S_IRUSR) ? 'r' : '-' )
         << ( (st.st_mode & S_IWUSR) ? 'w' : '-' )
         << ( (st.st_mode & S_IXUSR) ? 'x' : '-' )
         << ( (st.st_mode & S_IRGRP) ? 'r' : '-' )
         << ( (st.st_mode & S_IWGRP) ? 'w' : '-' )
         << ( (st.st_mode & S_IXGRP) ? 'x' : '-' )
         << ( (st.st_mode & S_IROTH) ? 'r' : '-' )
         << ( (st.st_mode & S_IWOTH) ? 'w' : '-' )
         << ( (st.st_mode & S_IXOTH) ? 'x' : '-' )
    ;

    char date[64];
    strftime( date, 15, "%b %d %H:%M ", localtime( &st.st_mtime ) );

    printf(
      "%2i %7s %7s %8ld %8s ",           // format string
      st.st_nlink,                       // number of links
      getpwuid(st.st_uid)->pw_name,      // password name
      getgrgid(st.st_gid)->gr_name,      // group name
      st.st_size,                        // size of file
      date                               // time of last modification
    );

    cout << *it << endl;

  }

  // RECURSE THROUGH root's HARD-LINKED SUBDIRECTORIES AND RETURN
  each( hardSubdirectory ) {
    cout << endl;
    visit( *it );
  }

  return 0;                                              // return success

}


int main( int argc, char* argv[] ) {
  return visit( argc > 1 ? argv[1] : "." );
}
```