Snippets

Pieter P

CMake

Compilation timing

```
1   set_property(GLOBAL PROPERTY RULE_LAUNCH_COMPILE "${CMAKE_COMMAND} -E time")
```

Stripping debug information

```
1   # Strip and install debug information
2   function(myproject_install_debug_syms target component dest_lib dest_bin)
3       if (MSVC)
4           install(FILES "$<TARGET_PDB_FILE:${target}>"
5               DESTINATION ${dest_bin}
6               CONFIGURATIONS Debug RelWithDebInfo
7               COMPONENT ${component}
8               OPTIONAL EXCLUDE_FROM_ALL)
9       elseif (CMAKE_STRIP AND CMAKE_OBJCOPY)
10          set(DEBUG_FILE "$<TARGET_FILE_NAME:${target}>.debug")
11          add_custom_command(TARGET ${target} POST_BUILD
12              COMMAND "${CMAKE_STRIP}" "--only-keep-debug" "$<TARGET_FILE:${target}>" "-o" "${DEBUG_FILE}"
13              COMMAND "${CMAKE_STRIP}" "--strip-debug" "$<TARGET_FILE:${target}>"
14              COMMAND "${CMAKE_OBJCOPY}" "--add-gnu-debuglink=${DEBUG_FILE}" "$<TARGET_FILE:${target}>"
15              COMMAND "${CMAKE_COMMAND}" "-E" "echo" "Stripped into ${DEBUG_FILE}"
16              WORKING_DIRECTORY $<TARGET_FILE_DIR:${target}>)
17          install(FILES "$<TARGET_FILE_DIR:${target}>/${DEBUG_FILE}"
18              DESTINATION ${dest_lib}
19              CONFIGURATIONS Debug RelWithDebInfo
20              COMPONENT ${component}
21              EXCLUDE_FROM_ALL)
22      endif()
23  endfunction()
24
25  # Usage
26  include(GNUInstallDirs)
27  foreach(target IN LISTS MYPROJECT_INSTALL_TARGETS)
28      get_target_property(target_TYPE ${target} TYPE)
29      if (${target_TYPE} STREQUAL "SHARED_LIBRARY")
30          myproject_install_debug_syms(${target} debug
31                                       ${CMAKE_INSTALL_LIBDIR}
32                                       ${CMAKE_INSTALL_BINDIR})
33      endif()
34  endforeach()
```

Hiding symbols with default visibility in shared library

```
1   function(configure_visibility target)
2       set_target_properties(${target} PROPERTIES CXX_VISIBILITY_PRESET "hidden"
3                                                   VISIBILITY_INLINES_HIDDEN true)
4       if (CMAKE_SYSTEM_NAME MATCHES "Linux")
5           target_link_options(${target} PRIVATE "LINKER:--exclude-libs,ALL")
6       endif()
7   endfunction()
```

The target properties ensure that the source files comprising the given target are compiled with the visibility set to hidden, which means that unless a symbol is explicitly marked "export", it won't be exported in the shared library.

The linker option ensures that symbols with default visibility in any static libraries the target depends on are not exported in the shared library either.
From `man ld(1)`:

```
--exclude-libs lib,lib,...
    Specifies a list of archive libraries from which symbols should not
    be automatically exported.  The library names may be delimited by
    commas or colons.  Specifying "--exclude-libs ALL" excludes symbols
    in all archive libraries from automatic export.
```

For explicitly exporting symbols that are part of the public API, see GenerateExportHeader.

Linux compilation and development

Inspecting binaries

```
1  # Dependencies and dynamic section of a shared library
2  readelf -d libfile.so
3  # List of symbols in shared library (1)
4  nm -CD --defined-only --size-sort libfile.so
5  # List of symbols in shared library (2)
6  readelf --wide --symbols --demangle libfile.so
7  # Filter symbols and prevent line wrapping
8  readelf --wide --symbols --demangle libfile.so | grep name | bat --wrap=never
```

Checking a debug link

```
1  # Dependencies and dynamic section of a shared library
2  objcopy -O binary --dump-section .gnu_debuglink=>(cut -d '' -f 1 -) libfile.so
```

Check the GLIBC version requirements of an ELF file

```
1  # Print private headers containing version references
2  objdump -p libfile.so
```

C++

Reversing a linked list

```cpp
1  #include <utility> // std::exchange
2
3  struct Node {
4      Node *next = nullptr;
5  };
6
7  Node *reverse_linked_list(Node *fwd) {
8      Node *rev = nullptr;
9      while (fwd)
10         rev = std::exchange(fwd, std::exchange(fwd->next, rev));
11     return rev;
12 }
```

BlueZ

Send and receive MIDI over BLE

```
1  bluetoothctl
2  scan le
3  scan off
4  pair F4:12:FA:E3:47:51
5  connect F4:12:FA:E3:47:51
6  menu gatt
7  list-attributes
8  select-attribute /org/bluez/hci0/dev_F4_12_FA_E3_47_51/service000a/char000b
9  notify on
10 read
11 write "0x80 0x80 0x90 0x12 0x13" 0 command
12 back
13 disconnect
```

Plot a dense matrix using matplotlib

```python
"""Plot a dense matrix using matplotlib, with special colors for zeros and
inf/nan. Useful for visualizing differences between matrices with possibly
missing values.w"""

import matplotlib.pyplot as plt
import matplotlib.colors as mcolors


def imshow_matrix(
    data,
    cmap: mcolors.Colormap | str | None = None,
    *,
    inf_color="red",
    zero_color="white",
):
    """Prepare data, colormap and normalization for imshow of a dense matrix.
    Zeros are shown as white, inf/nan as red (customizable)."""
    data = np.copy(data)
    min_val = np.min(data[np.isfinite(data) & (data > 0)])
    max_val = np.max(data[np.isfinite(data)])
    data[data == 0] = 0.99 * min_val
    cmap = plt.get_cmap(cmap).copy()
    cmap.set_bad(color=inf_color)  # inf/nan values
    cmap.set_under(color=zero_color)  # zero values
    norm = mcolors.LogNorm(vmin=min_val, vmax=max_val)
    return data, cmap, norm


import numpy as np

A = np.random.standard_normal((64, 96))
msk = np.random.uniform(0, 1, A.shape) > 0.5
np.fill_diagonal(A, np.nan)
A[msk] = 0
plt.imshow(*imshow_matrix(abs(A), cmap="viridis"))
plt.colorbar()
plt.show()
```

Plot a sparse matrix using matplotlib

```python
"""Plot a sparse matrix using matplotlib, with special colors for structural
zeros, numerical zeros, and inf/nan. Useful for visualizing differences between
sparse matrices with possibly missing values."""

import numpy as np
import scipy.sparse as spa
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors


def nonzero_indices(matrix: spa.csc_array):
    """Get the indices of the structural nonzeros in the given sparse matrix"""
    minor = matrix.indices
    indptr = matrix.indptr
    major = np.repeat(np.arange(len(indptr) - 1, dtype=minor.dtype), np.diff(indptr))
    return minor, major


def plot_sparse_matrix(
    matrix: spa.csc_array,
    cmap: mcolors.Colormap | str | None = None,
    *,
    inf_color="red",
    struc_zero_color="lightgray",
    num_zero_color="white",
):
    """Prepare data, colormap and normalization for imshow of a sparse matrix.
    Structural zeros are shown as gray, numerical zeros as white, inf/nan as
    red (customizable)."""
    data = np.array(matrix.data, dtype=np.float64)
    fin_nonzero = np.logical_and(np.isfinite(data), data != 0)
    finite_data = abs(data[fin_nonzero])
    minval, maxval = np.min(finite_data, initial=1.0), np.max(finite_data, initial=1.0)
    margin = 2 * np.sqrt(maxval / minval)
    under, over = minval / margin, maxval * margin
    assert matrix.shape is not None
    dense_matrix = np.full(matrix.shape, under, order="F")
    rows, cols = nonzero_indices(matrix)
    data[data == 0] = over
    dense_matrix.ravel("K")[rows + cols * matrix.shape[0]] = data
    cmap = plt.get_cmap(cmap).copy()
    cmap.set_extremes(under=struc_zero_color, over=num_zero_color, bad=inf_color)
    norm = mcolors.LogNorm(vmin=float(minval), vmax=float(maxval))
    return dense_matrix, cmap, norm


A = np.random.standard_normal((64, 64))
np.fill_diagonal(A, np.nan)
sparse_A = spa.triu(A, format="csc")
msk = np.random.uniform(0, 1, sparse_A.data.shape) > 0.7
sparse_A.data[msk] = 0  # Introduce some numerical zeros
# Note: many scipy operations implicitly drop numerical zeros
plt.imshow(*plot_sparse_matrix(abs(sparse_A), cmap="viridis"))
plt.colorbar()
plt.show()
```