Using gcov with the Linux kernel
================================

1. Introduction
===============

gcov profiling kernel support enables the use of GCC's coverage testing
tool gcov [1] with the Linux kernel. Coverage data of a running kernel
is exported in gcov-compatible format via the "gcov" debugfs directory.
To get coverage data for a specific file, change to the kernel build
directory and use gcov with the -o option as follows (requires root):

```
# cd /tmp/linux-out
# gcov -o /sys/kernel/debug/gcov/tmp/linux-out/kernel spinlock.c
```

This will create source code files annotated with execution counts
in the current directory. In addition, graphical gcov front-ends such
as lcov [2] can be used to automate the process of collecting data
for the entire kernel and provide coverage overviews in HTML format.

Possible uses:

* debugging (has this line been reached at all?)
* test improvement (how do I change my test to cover these lines?)
* minimizing kernel configurations (do I need this option if the
  associated code is never run?)

--

[1] http://gcc.gnu.org/onlinedocs/gcc/Gcov.html
[2] http://ltp.sourceforge.net/coverage/lcov.php

2. Preparation
==============

Configure the kernel with:

```
        CONFIG_DEBUG_FS=y
        CONFIG_GCOV_KERNEL=y
```

and to get coverage data for the entire kernel:

```
        CONFIG_GCOV_PROFILE_ALL=y
```

Note that kernels compiled with profiling flags will be significantly larger and run slower. Also CONFIG_GCOV_PROFILE_ALL may not be supported on all architectures.

Profiling data will only become accessible once debugfs has been mounted:

```
mount -t debugfs none /sys/kernel/debug
```

3. Customization

To enable profiling for specific files or directories, add a line similar to the following to the respective kernel Makefile:

For a single file (e.g. main.o):

```
GCOV_PROFILE_main.o := y
```

For all files in one directory:

```
GCOV_PROFILE := y
```

To exclude files from being profiled even when CONFIG_GCOV_PROFILE_ALL is specified, use:

```
GCOV_PROFILE_main.o := n
```

and:

```
GCOV_PROFILE := n
```

Only files which are linked to the main kernel image or are compiled as kernel modules are supported by this mechanism.

4. Files

The gcov kernel support creates the following files in debugfs:

/sys/kernel/debug/gcov
: Parent directory for all gcov-related files.

/sys/kernel/debug/gcov/reset
: Global reset file: resets all coverage data to zero when written to.

/sys/kernel/debug/gcov/path/to/compile/dir/file.gcda
: The actual gcov data file as understood by the gcov tool. Resets file coverage data to zero when written to.

/sys/kernel/debug/gcov/path/to/compile/dir/file.gcno
: Symbolic link to a static data file required by the gcov tool. This file is generated by gcc when compiling with option -ftest-coverage.

5. Modules

==========

Kernel modules may contain cleanup code which is only run during
module unload time. The gcov mechanism provides a means to collect
coverage data for such code by keeping a copy of the data associated
with the unloaded module. This data remains available through debugfs.
Once the module is loaded again, the associated coverage counters are
initialized with the data from its previous instantiation.

This behavior can be deactivated by specifying the gcov_persist kernel
parameter:

        gcov_persist=0

At run-time, a user can also choose to discard data for an unloaded
module by writing to its data file or the global reset file.

6. Separated build and test machines

The gcov kernel profiling infrastructure is designed to work out-of-the
box for setups where kernels are built and run on the same machine. In
cases where the kernel runs on a separate machine, special preparations
must be made, depending on where the gcov tool is used:

a) gcov is run on the TEST machine

The gcov tool version on the test machine must be compatible with the
gcc version used for kernel build. Also the following files need to be
copied from build to test machine:

from the source tree:
  - all C source files + headers

from the build tree:
  - all C source files + headers
  - all .gcda and .gcno files
  - all links to directories

It is important to note that these files need to be placed into the
exact same file system location on the test machine as on the build
machine. If any of the path components is symbolic link, the actual
directory needs to be used instead (due to make's CURDIR handling).

b) gcov is run on the BUILD machine

The following files need to be copied after each test case from test
to build machine:

from the gcov directory in sysfs:
  - all .gcda files
  - all links to .gcno files

These files can be copied to any location on the build machine. gcov
must then be called with the -o option pointing to that directory.

Example directory setup on the build machine:

```
  /tmp/linux:    kernel source tree
  /tmp/out:      kernel build directory as specified by make O=
  /tmp/coverage: location of the files copied from the test machine

  [user@build] cd /tmp/out
  [user@build] gcov -o /tmp/coverage/tmp/out/init main.c
```

7. Troubleshooting

Problem: Compilation aborts during linker step.
Cause: Profiling flags are specified for source files which are not
          linked to the main kernel or which are linked by a custom
          linker procedure.
Solution: Exclude affected source files from profiling by specifying
          GCOV_PROFILE := n or GCOV_PROFILE_basename.o := n in the
          corresponding Makefile.

Problem: Files copied from sysfs appear empty or incomplete.
Cause: Due to the way seq_file works, some tools such as cp or tar
          may not correctly copy files from sysfs.
Solution: Use 'cat' to read .gcda files and 'cp -d' to copy links.
          Alternatively use the mechanism shown in Appendix B.

Appendix A: gather_on_build.sh

Sample script to gather coverage meta files on the build machine
(see 6a):

```
#!/bin/bash

KSRC=$1
KOBJ=$2
DEST=$3

if [ -z "$KSRC" ] || [ -z "$KOBJ" ] || [ -z "$DEST" ]; then
  echo "Usage: $0 <ksrc directory> <kobj directory> <output.tar.gz>" >&2
  exit 1
fi

KSRC=$(cd $KSRC; printf "all:\n\t@echo \${CURDIR}\n" | make -f -)
KOBJ=$(cd $KOBJ; printf "all:\n\t@echo \${CURDIR}\n" | make -f -)

find $KSRC $KOBJ \( -name '*.gcno' -o -name '*.[ch]' -o -type l \) -a \
                -perm /u+r,g+r | tar cfz $DEST -P -T -

if [ $? -eq 0 ] ; then
  echo "$DEST successfully created, copy to test system and unpack with:"
  echo "  tar xfz $DEST -P"
else
  echo "Could not create file $DEST"
```

```
fi
```

Appendix B: gather_on_test.sh
==========

Sample script to gather coverage data files on the test machine
(see 6b):

```
#!/bin/bash -e

DEST=$1
GCDA=/sys/kernel/debug/gcov

if [ -z "$DEST" ] ; then
  echo "Usage: $0 <output.tar.gz>" >&2
  exit 1
fi

TEMPDIR=$(mktemp -d)
echo Collecting data..
find $GCDA -type d -exec mkdir -p $TEMPDIR/\{\} \;
find $GCDA -name '*.gcda' -exec sh -c 'cat < $0 > '$TEMPDIR'/$0' {} \;
find $GCDA -name '*.gcno' -exec sh -c 'cp -d $0 '$TEMPDIR'/$0' {} \;
tar czf $DEST -C $TEMPDIR sys
rm -rf $TEMPDIR

echo "$DEST successfully created, copy to build system and unpack with:"
echo "  tar xfz $DEST"
```