# Code coverage practice session

## Typographical conventions

We use the following conventions in this guide:

emacs The name of a specific command or file

*file*  You should replace *file* with a specific name

### **Exit abc** Output that you see on the screen

## Getting Started with gcov

1. Login into the Linux server with your login Ids
2. Create a new directory called code\_cov in your home directory <home>

mkdir code\_cov

1. Go inside the directory you have created in (2) /<home>/code\_cov

cd code\_cov

1. Copy the following files from the path as mentioned by the trainer:
   1. sample.c
   2. link.c
   3. link.h
2. Take a look at the example programs sample.c and link.c

## Compilation

1. Compile the files sample.c and sample1.c and put the output in the executable file called output

gcc –o output –ftest-coverage –fprofile-arcs sample.c link.c

The .gcno file is generated when the source file is compiled with the GCC -ftest-coverage option and .gcda file is generated when a program containing object files built with the GCC -fprofile-arcs option is executed

## Execution

1. Execute the file output

./output

1. Now run gcov for each source file one by one

gcov sample.c

***File `sample.c'***

***Lines executed:50.00% of 10***

***sample.c:creating `sample.c.gcov'***

View the output file sample.c.gcov with the vi editor

vi sample.c.gcov

Analyse this file and notice that each executable statement is either preceeded by a number or by ######. The number specifies the number of times that statement got executed while ###### represents that this source code statement did not get executed at all.

gcov link.c

***File `link.c'***

***Lines executed:100.00% of 2***

***sample1.c:creating `link.c.gcov'***

1. Run output again, this time with command line arguments:

./output a a b b

***Check!***

***oops***

***Check!***

#### This function is just called to link this file

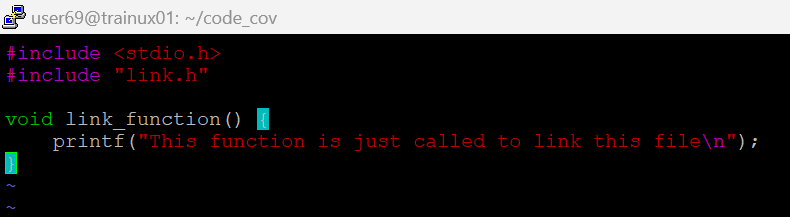
1. Now run gcov for sample.c again

What do you observe?

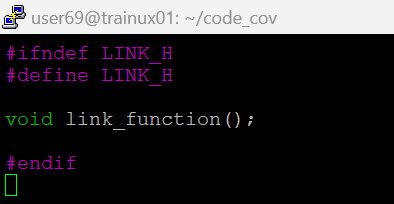
**NOTE : If the code coverage is not 100% , it can be achieved by using gdb (For that, you need to compile with -g as well as -ftest-coverage -fprofile-arcs options)**

**Food for thought : What happens to the currently achieved coverage, when you modify a .c file? Is the earlier coverage data still valid?**

**Link.c**



**Link.h**



**Sample.c**

