

# CS 350

## Spring 2005

### Use of gprof

#### Activities:

1. On a UNIX or Linux machine, type “man gprof” or “info gprof” to learn how to use gprof.
2. Copy the contents of the directory ~cs350/Recitation5 into a directory in your UNIX account. After copying, you should have the following files:
  - a. TestProg: the executable for a program to be tested.
  - b. input01, input02, input03, input04: the input files for testing the program TestProg
  - c. expected01, expected02, expected03, expected04: the files created by the person testing the program TestProg. expected01 is what the tester thinks a correctly implemented TestProg will produce when it reads the file input01, etc.
  - d. TestTool.cpp: this is the source code for a test tool described briefly below. You should experiment with it to see what does and to understand how to use it. You might benefit from looking at the source code.
3. Compile the program TestTool.cpp with the pg flag. (That is, type “g++ -pg -o TestTool TestTool.cpp”) This tells the compiler to include extra code in the executable so that the data needed by gprof will be created whenever the program is run.
4. Run the program by typing “TestTool 0.0001 4 TestProg” on the command line. Depending on how your account is set up, you may need to type “./TestTool” instead of just “TestTool”.
5. Run gprof. (Type “gprof TestTool > OutputReport”)
6. Read the report produced by gprof and answer the following questions:
  - a. How often is the function PrintMiss called?
  - b. How often is the function EvalStrng called?
  - c. How many seconds does the CreatInt function take to run?
  - d. What function takes to largest amount of CPU time?  
The function and all of its children?  
The function ignoring its children?
  - e. What functions call CreatInt?
  - f. What functions are called by PrintMiss?

#### Purpose of TestTool

TestTool was written by a 350 student in a previous semester. The program aids in testing by comparing the actual output produced by a program with what a tester has prepared as the expected output. It assumes that the person testing the code will have prepared several test cases (consisting of input files and, for each, an expected output file). The tool runs the program and compares the output actually produced by the program with what the tester provided as “expected output.” The tool also requires that the person testing specify how precise floating point comparisons should be to be regarded as a match.

#### TestTool parameters:

1. The precision to be used for floating point comparisons.

2. The number of input files (test cases) that should be used. The program being tested will be run once with each input file.
3. The name of the executable of the program being tested.

**Assignment:**

1. Find two commercial code profiling tools. Describe each briefly (one or two sentences) and where they can be found.
2. Answer the following 2 questions:  
For what activities is gcov a better tool than gprof?  
For what activities is gprof a better tool than gcov?  
Include answers to these questions in your report.
3. Use the submit command to submit the report to cs350
4. Submit the run report generated by gprof to cs350, along with a file containing your answers to the questions of activity 6 above.

**Due Date:** Thursday, March 3, by midnight.