gprof(GNU profiler)是GNU binutils工具集中的一个工具，linux系统当中会自带这个工具。

操作系统性能研究技术的核心是内核跟踪机制，他是一个强大的系统内部行为观测机制。

内核探测技术是将跟踪代码添加到所要跟踪的源码中，这些跟踪代码作为探测点可以获得衡量系统性能的信息。

静态探测

动态探测

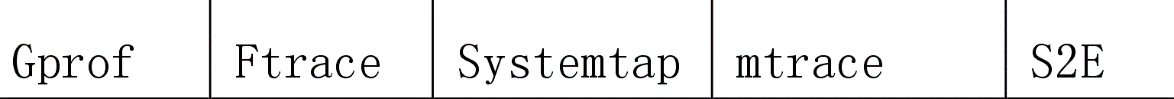
函数间的调用关系，cpu的消耗时间 进程切换和内存访问

GCC编译工具链是GCC编译器的一整套工具，主要作用就是 将源代码转化为可执行应用程序。

Gprof：通过Gcc早对原目标代码进行编译时和链接用户程序的时候在用户的应用程序的每个函数入口出都加入一个名为mcount的函数，也就是说应用程序里每一个函数都会调用mcount，而mcount函数通过函数调用堆栈的形式查找子函数和父函数的返回地址，生成函数调用表，其中，mcount函数插入每个函数中时，主要依赖于Gcc编译器在编译时执行汇编语言。

异常包括很多方面，有出错（fault），有陷入（trap），也有可编程异常（programmable exception）。出错（fault）和陷入（trap）最重要的一点区别是他们发生时所保存的EIP值的不同。出错（fault）保存的EIP指向触发异常的那条指令；而陷入（trap）保存的EIP指向触发异常的那条指令的下一条指令。

本文对gprof的改进：生成函数调用，获取函数时间



生成函数调用

之前：通过Gcc编译时静态插入探测函数获取

改进后：通过将Ftrace、Systemtap和S2E等谈的工具跟踪到的数据进行转换获取

获取函数时间

之前：对每个运行中的函数运行时间进行统计，采样统计（存在误差）

改进后：来源于其他跟踪工具跟踪的真实时间

本文的最终目的是将Ftrace、Systemtap和S2E等的跟踪结果作为Gprof的输入信息，使用Gprof的功能对数据进行分析和转换，最终将函数关系以图的形式展示出来

<https://blog.csdn.net/luckyapple1028/article/details/52972315>

Linux内核调试技术——kprobe使用与实现

Ftrace是linux内核提供的一种调试工具。使用 ftrace 可以对内核中发生的事情进行跟踪。现在不用单独编译，Linux5.11.0已经支持ftrace使用。

从2.6.9 Linux内核开始，Kprobes被加入内核源码，并处于不断完善之中，越来越多的功能被添加到 Kprobes 内核调试技术中来。Kprobes 目前已经能在 i386，x86\_64， ppc64， ia64，sparc64等CPU平台上正常工作。

systemtap没有合并到内核主版本中，如果要使用需要单独安装。

Ftrace原理

1.编译阶段。通过编译选项 -pg -mrecord-mcount 在每个支持 trace 的函数中插入 bl 0 <\_mcount> 指令。

2.链接阶段。会根据重定位段将 bl 0 <\_mcount> 指令地址重定位为 \_mcount 函数地址。

3.运行阶段。

在risc-v上的ftrace

在线实验平台，追踪加一个模块，追踪函数

kprobe的工作过程大致如下：

1）注册kprobe。注册的每个kprobe对应一个kprobe结构体，该结构中记录着插入点（位置），以及该插入点本来对应的指令original\_opcode；

2）替换原有指令。使能kprobe的时候，将插入点位置的指令替换为一条异常(BRK)指令，这样当CPU执行到插入点位置时会陷入到异常态；

3）执行pre\_handler。进入异常态后，首先执行pre\_handler，然后利用CPU提供的单步调试（single-step）功能，设置好相应的寄存器，将下一条指令设置为插入点处本来的指令，从异常态返回；

4）再次陷入异常态。上一步骤中设置了single-step相关的寄存器，所以originnal\_opcode刚一执行，便会二进宫：再次陷入异常态，此时将single-step

清除，并且执行post\_handler，然后从异常态安全返回。

步骤2），3），4）便是一次kprobe工作的过程，它的一个基本思路就是将本来执行一条指令扩展成执行kprobe->pre\_handler ---> 指令 ---> kprobe-->post\_hander这样三个过程。

https://www.cnblogs.com/honpey/p/4575902.html