Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias

Análisis y Diseño 2

Ing. Ricardo Morales

Aux. Kenny Eguizábal



**Práctica2: Manual de GPROF**

Virginia Chavarría Guzmán

200920081

Guatemala,19 Abril de 2016

GPROF

# HISTORIA

GPROF originalmente fue escrito por un grupo liderado por Sunsan L. Graham en la universidad de California, Berkeley para Berkeley unix, otra implementación fue escrita como parte del proyecto GNU por GNU Binutils en 1988 por Jay Fenlason

# ¿QUE ES?

GPROF es una herramienta de código abierto que puede mejorar el rendimiento del código fuente de los programas. Puede ayudar a identificar en que parte del código está el problema para que se pueda corregir. GNU GPROF este puede ayudar a identificar los cuellos de botella (“bottlenecks”) que se han generado al escribir un programa para así optimizar el rendimiento.

# PARA QUE SIRVE

La herramienta GPROF nos proporciona perfiles de la ejecución de un programa, es decir, nos da información acerca de cuánto tiempo se emplea en cada función y de cuántas veces se llama. Sirve para detectar dónde el programa está invirtiendo la mayor parte de su tiempo de ejecución.

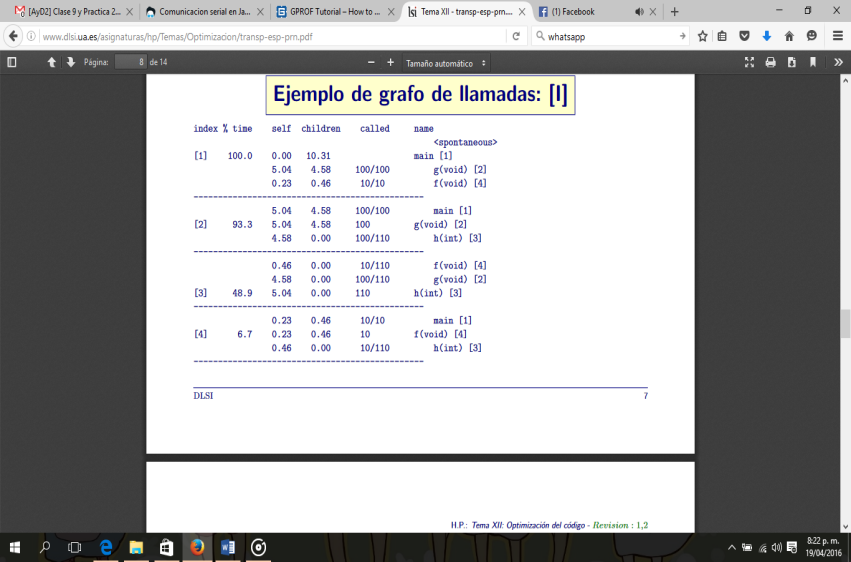
Esta herramienta nos muestra dos tipos de perfiles que son los siguientes:

* **Perfil plano:** Para cada función proporciona el tiempo y el número de veces que se llama cada función.

****

* **Call Graph**: Lo mismo que la anterior, pero con información acerca

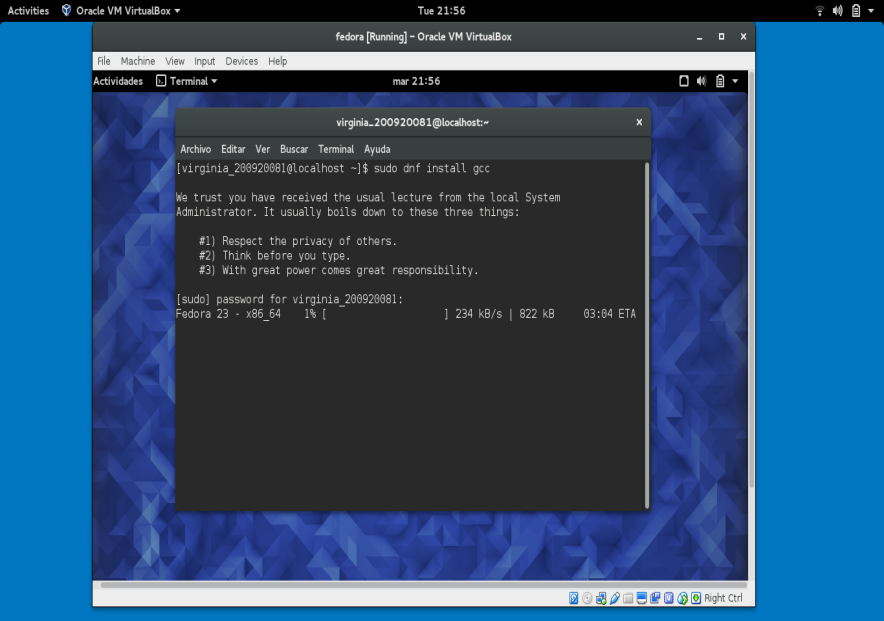
de que funciones la llama y que funciones llama.



# USO

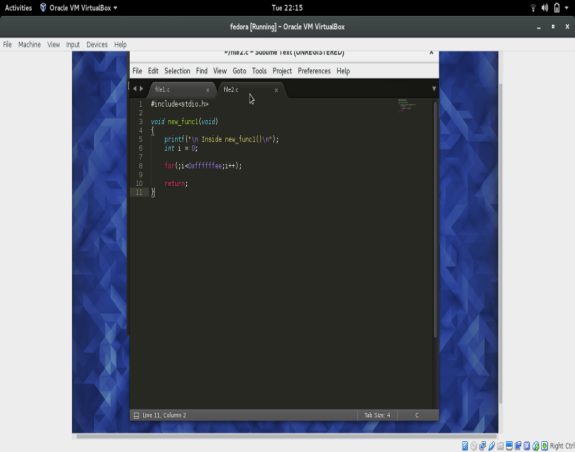
Usaremos para este ejemplo el O.S Fedora 23.

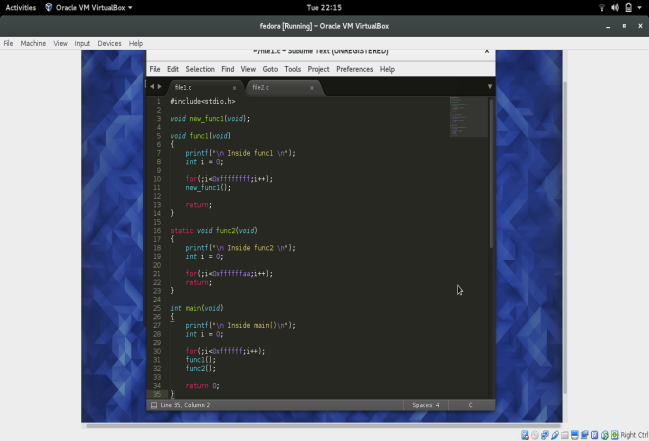
Primero que nada se debe tener instalado GCC de no tenerlo se instala de la siguiente manera



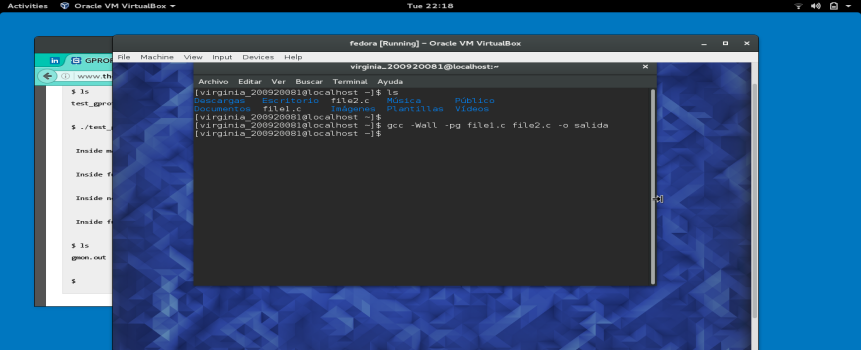
Teniendo instalado GCC se procedera a usar GPROF, para esto se tiene 2 archivos

File1.c y file2.c en las funciones los “for” se usan para consumir un poco de tiempo de ejecución.

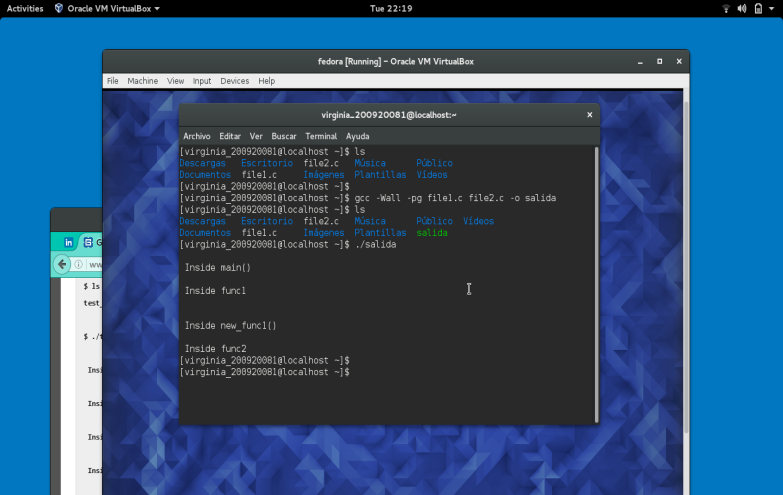


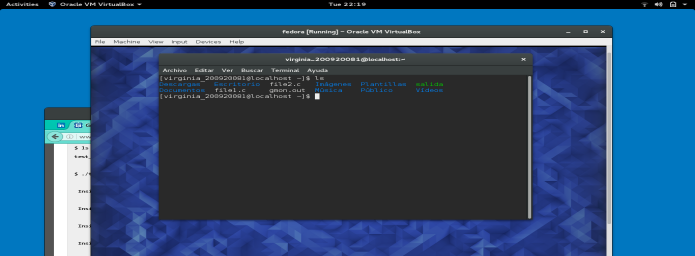


Notese que para generar codigo extra para analizar el programa con GPROF se añade “-pg” al momento de compilar:

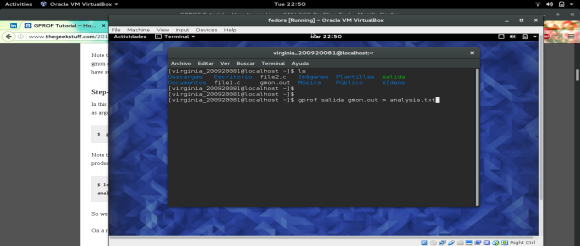
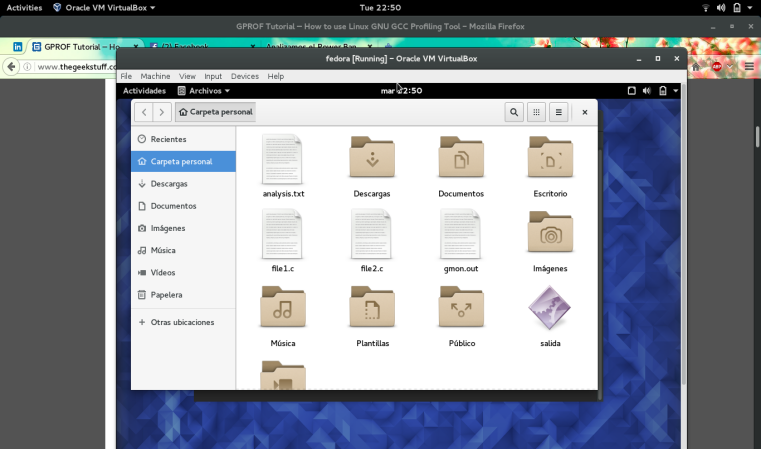


Luego ejecutamos la salida así **“./salida”**, al ejecutar estos notamos que este nos ha generado un archivo llamado gmon.out en el directorio donde se encuentra nuestro proyecto.

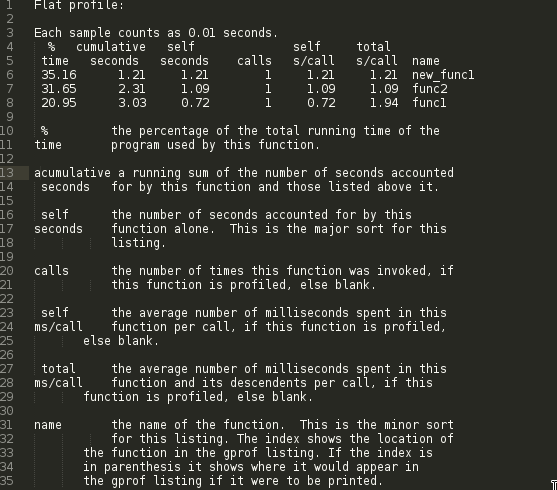
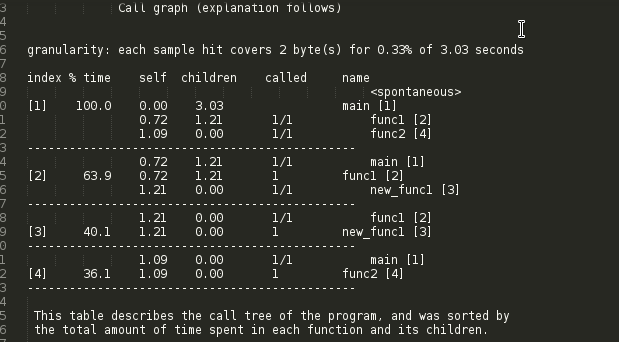




Par poder correr GPROF se usa gmon.out para generar un archivo de análisis del programa



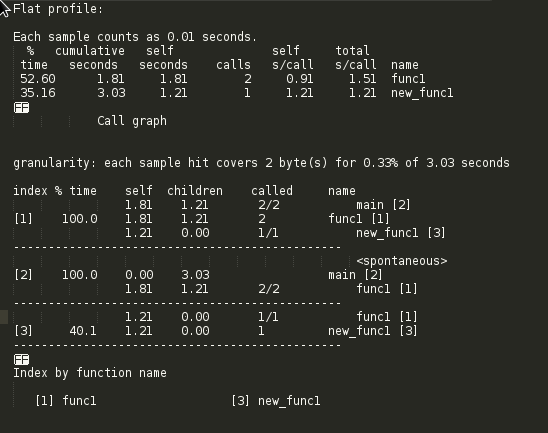
Como se mencionó anteriormete GPROF nos generá dos tipos de perfiles el flat profile y call graph en el archivo se encuentra explicado a detalle.



Si queremos modificar nuestras salidas gprof nos ofrece varias flags que podremos usar como:

**-a:** Suprime la salida de un método static/private



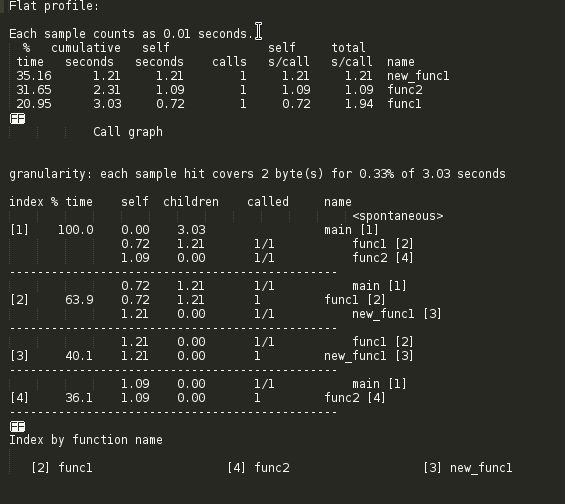


Que a nuestra salida nos muestra lo siguiente:

Y que como podemos ver se ha suprimido el método fun2 el cual era estatico.

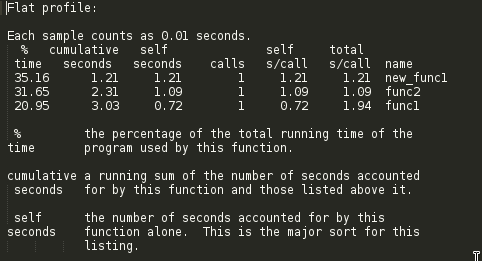
**-b:** Suprime toda la documentación inecesaria usando





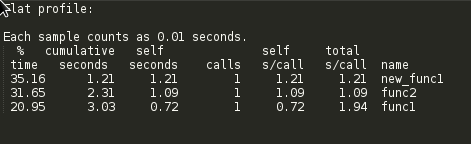
**-p:** Solo muestra el flat profile





O se puede mezclar para mostrar solo el flat profile sin la documentacion extra usando

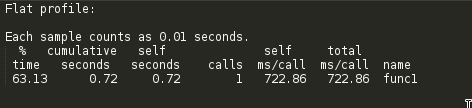
**>gprof -p -b salida gmon.out > analysis2.txt**



Si se desea mostrar la información de X función se muestra así:

**>gprof -pfunc1 -b salida gmon.out > analysis2.txt**

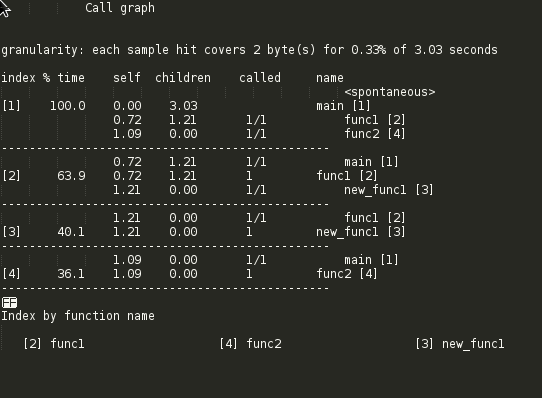




Para suprimir del archivo el flat profile se utiliza **-P**

**>gprof -P -b salida gmon.out > analysis2.txt**

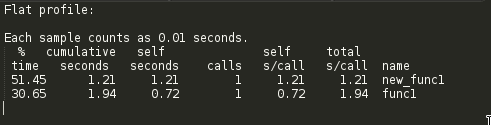




Para excluir una cierta función del flat profile seria :

**>gprof -Pfunc2 -b salida gmon.out > analysis2.txt**

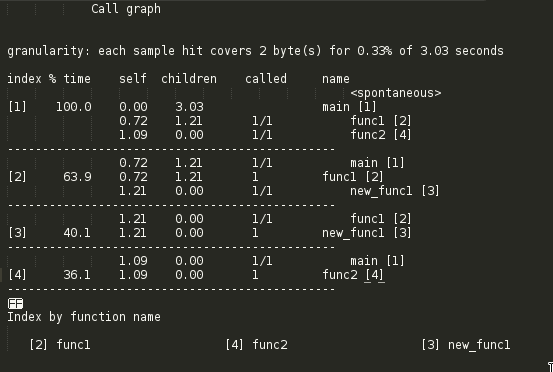




Para imprimir unicamente el call graph se utiliza **-q**

**>gprof -q -b salida gmon.out > analysis2.txt**

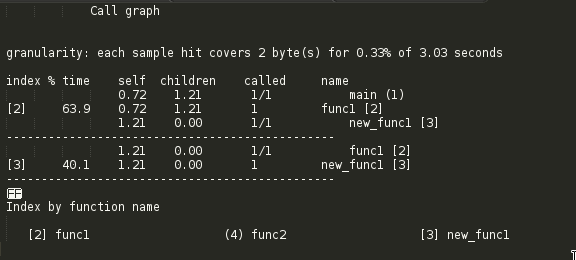




Para imprimir cierta concidencia con el nombre de una función del call graph

**>gprof -qfunc1 -b salida gmon.out > analysis2.txt**

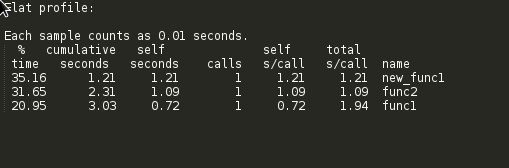




Para suprimir el call graph de la salida se utiliza **-Q**

**>gprof -Q -b salida gmon.out > analysis2.txt**

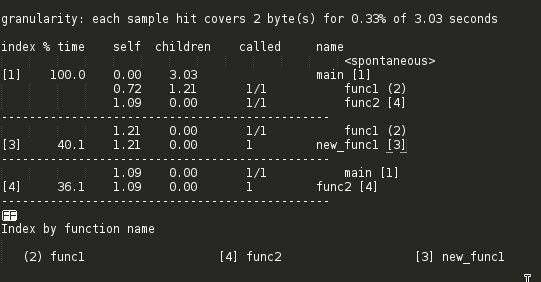




Para suprimir cierta función del Call graph se usa **-Qnombre**

**>gprof -Qfunc1 -b salida gmon.out > analysis2.txt**





# EXPLICACIÓN

A continuación se explicará que es lo que muestra cada columna de los despliegues de GPROF tanto de flat profile como de Call graph.

**Flat profile:**

**%time**: el porcentaje del tiempo total de ejecución del programa utilizado por esta función.

**Cumulative seconds:** suma acumulada del número en segundos usados por cada función.

**Self seconds:**  el número en segundos de tiempo usados por cada función individualmente.

**Calls:**  el número de veces que esta función fue invocada , si esta función fue perfilada entonces la muestra en blanco.

**Self s/call:**  el promedio de número en milisegundos gastados por cada función, si la función fue perfilada entonces lo muestra en blanco.

**Total s/call:** el promedio de número de milisegundos gastados por la función y sus hijos, al igual que anterior si esta es perfilada entonces lo muestra en blanco.

**Name:**  nombre la función.

**Call graph:**

Cada entrada en esta tabla consiste en varias líneas.

**Index:**  un número único dado a cada elemento de la tabla, estos son ordenados numéricamente, este es impreso.

**Self:** este es el total de tiempo gastado en esta función.

**%time:**  tiempo en milisegundos gastados en la ejecución de cada función y sus hijos.

**Children:** este es número total de tiempo propagado en esta función por sus hijos.

**Called:**  es el número de veces que la función fue llamada. Si esta función es llamada recursivamente solo incluirá las llamados no recursivas.

**Name:**  El nombre de la función actual, el index es impreso después del nombre, si esta función es un miembro de un ciclo, el número del ciclo es impreson entre el nombre de la función y el index de esta.

Si hay ciclos en esta tabla, entonces habrá una entrada para el ciclo como un todo, esta muestra quien la llamó (padre) y los miembros del ciclo (hijos)