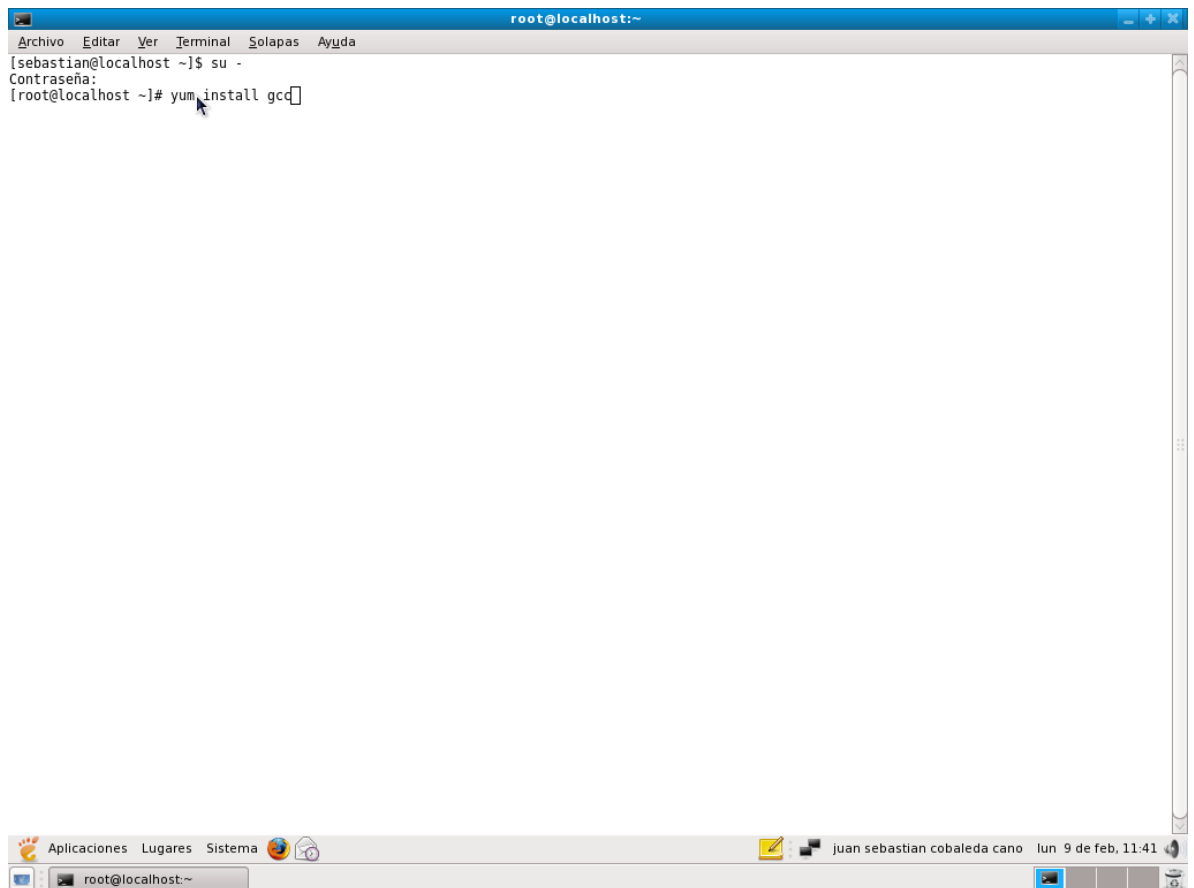


Programar en C y C++ usando Fedora

Tu Fedora te da la posibilidad de compilar tus programas, para eso tienes que tener instalado el paquete (GCC) .este paquete es de suma importancia para los programadores y desarrolladores de software, pues es una colección de compiladores de diferentes lenguajes de programación.

Para empezar con ello tienes que tener instalado el GCC, para ello procedemos de la siguiente manera:

- Instalando el paquete directamente así:



The screenshot shows a terminal window titled "root@localhost:~". The user has entered the command `su -` to become the root user. The prompt changes from `[sebastian@localhost ~]$` to `[root@localhost ~]#`. The user then enters the command `yum install gcc`. The terminal window has a menu bar with "Archivo", "Editar", "Ver", "Terminal", "Solapas", and "Ayuda". The desktop environment is visible at the bottom, showing a taskbar with icons for "Aplicaciones", "Lugares", "Sistema", and a user profile "juan sebastian cobaleda cano". The system clock shows "lun 9 de feb, 11:41".

```
root@localhost:~  
[sebastian@localhost ~]$ su -  
Contraseña:  
[root@localhost ~]# yum install gcc
```

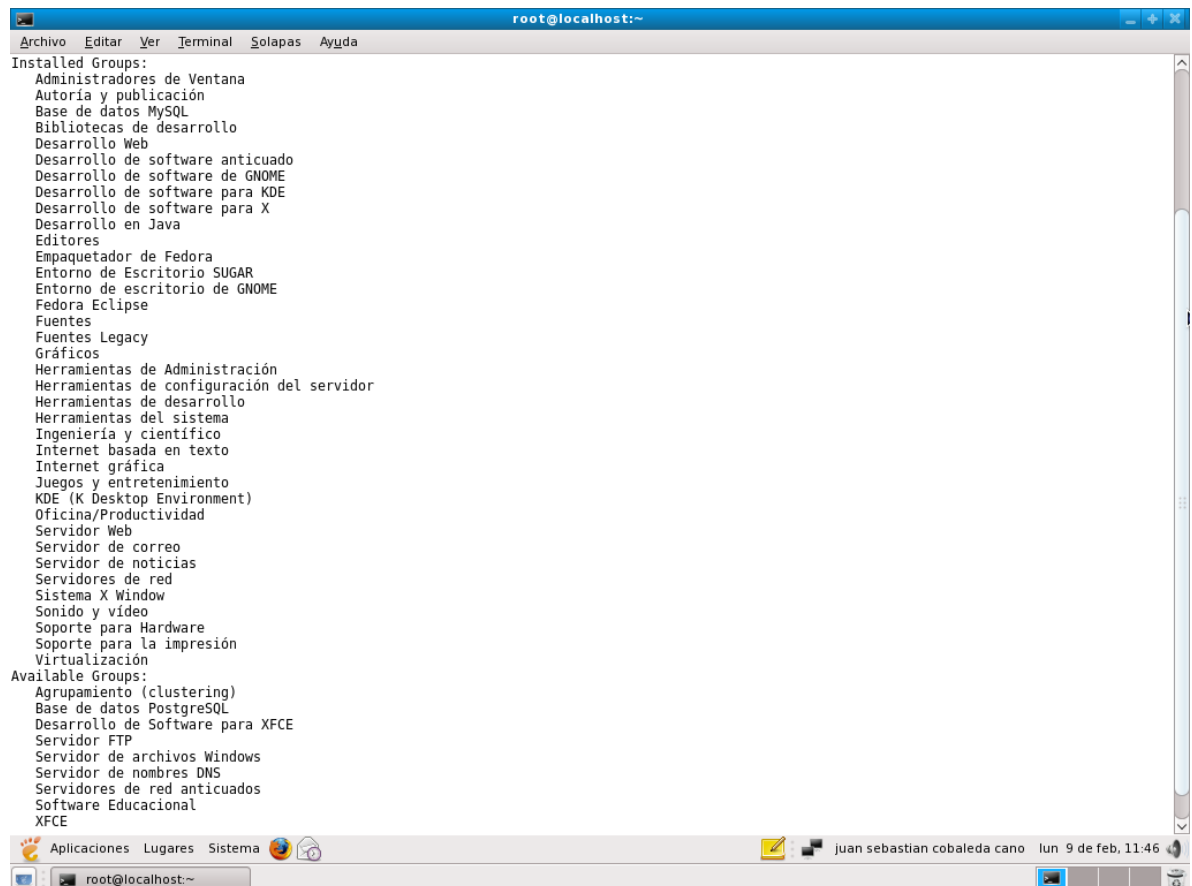
Abrimos el terminal, ingresamos como administrador y copiamos la ste cadena:

“ Yum install gcc “



- La otra manera es instalar desde los grupos.

Los grupos son aplicaciones y paquetes que vienen diseñados para un determinado fin, como por ejemplo diferentes entornos de desarrollo, servidor ftp, herramientas y aplicaciones lúdicas, etc.



Para visualizar los grupos que tenemos instalados copiamos en la terminal:

“ yum grouplist ”

Y para instalar los grupos que no tenemos instalados procedemos así:

“ yum groupinstall ‘ el nombre del grupo que falte ‘ ”



Para instalar un grupo que contiene el gcc procede asi:

“ yum groupinstall ‘ Herramientas de desarrollo ’ “

```
root@localhost:~  
Archivo Editar Ver Terminal Solapas Ayuda  
Package diffstat-1.43-7.fc9.i386 already installed and latest version  
Package matching cvs-1.11.22-14.fc9.i386 already installed. Checking for update.  
Package texinfo-4.11-8.fc9.i386 already installed and latest version  
Package matching automake15-1.5-23.noarch already installed. Checking for update.  
Package python-ldap-2.3.1-3.fc9.i386 already installed and latest version  
Package matching subversion-1.4.6-7.i386 already installed. Checking for update.  
Package byacc-1.9.20070509-2.fc9.i386 already installed and latest version  
Package frysk-0.4-0.fc9.i386 already installed and latest version  
Package gcc-gfortran-4.3.0-8.i386 already installed and latest version  
Package elfutils-0.137-3.fc9.i386 already installed and latest version  
Package rcs-5.7-32.i386 already installed and latest version  
Package automake16-1.6.3-14.noarch already installed and latest version  
Package automake17-1.7.9-11.noarch already installed and latest version  
Package matching automake14-1.4p6-15.fc7.noarch already installed. Checking for update.  
Package matching patchutils-0.2.31-5.fc9.i386 already installed. Checking for update.  
Package ccache-2.4-13.fc9.i386 already installed and latest version  
Package ctags-5.7-1.fc9.i386 already installed and latest version  
Package oprofile-gui-0.9.3-18.fc9.i386 already installed and latest version  
Package systemtap-0.8-1.fc9.i386 already installed and latest version  
Package swig-1.3.33-2.fc9.i386 already installed and latest version  
Package l:doxygen-1.5.7-1.fc9.i386 already installed and latest version  
Package indent-2.2.10-1.fc9.i386 already installed and latest version  
Package matching l:valgrind-3.3.0-3.i386 already installed. Checking for update.  
Package oprofile-0.9.3-18.fc9.i386 already installed and latest version  
Package cscope-15.6-2.fc9.i386 already installed and latest version  
Resolving Dependencies  
--> Running transaction check  
---> Package binutils.i386 0:2.18.50.0.6-7.fc9 set to be updated  
--> Finished Dependency Resolution  
  
Dependencies Resolved  
  
=====
```

Package	Arch	Version	Repository	Size
Updating: binutils	i386	2.18.50.0.6-7.fc9	updates-newkey	3.0 M

```
=====
```

Transaction Summary

Install	0 Package(s)
Update	1 Package(s)
Remove	0 Package(s)

Total download size: 3.0 M
Is this ok [y/N]: y
Downloading Packages:
binutils-2.18.50.0.6-7.fc9.i386.rpm 9% [====] 66 kB/s | 280 kB 00:41 ETA

Aplicaciones Lugares Sistema juan sebastian cobaleda cano lun 9 de feb, 11:49
root@localhost:~

Después de instalar el gcc, podemos empezar a programar en C y C++.

Primero vamos a comenzar con un ejemplo básico de programación en C++, que nos muestra en la terminal algún mensaje que de antemano hallamos establecido.

Vamos a empezar con el mensaje “ FEDORA COLOMBIA ” visualizado en el terminal.



Este es el código escrito en C++:

```
#include <iostream>

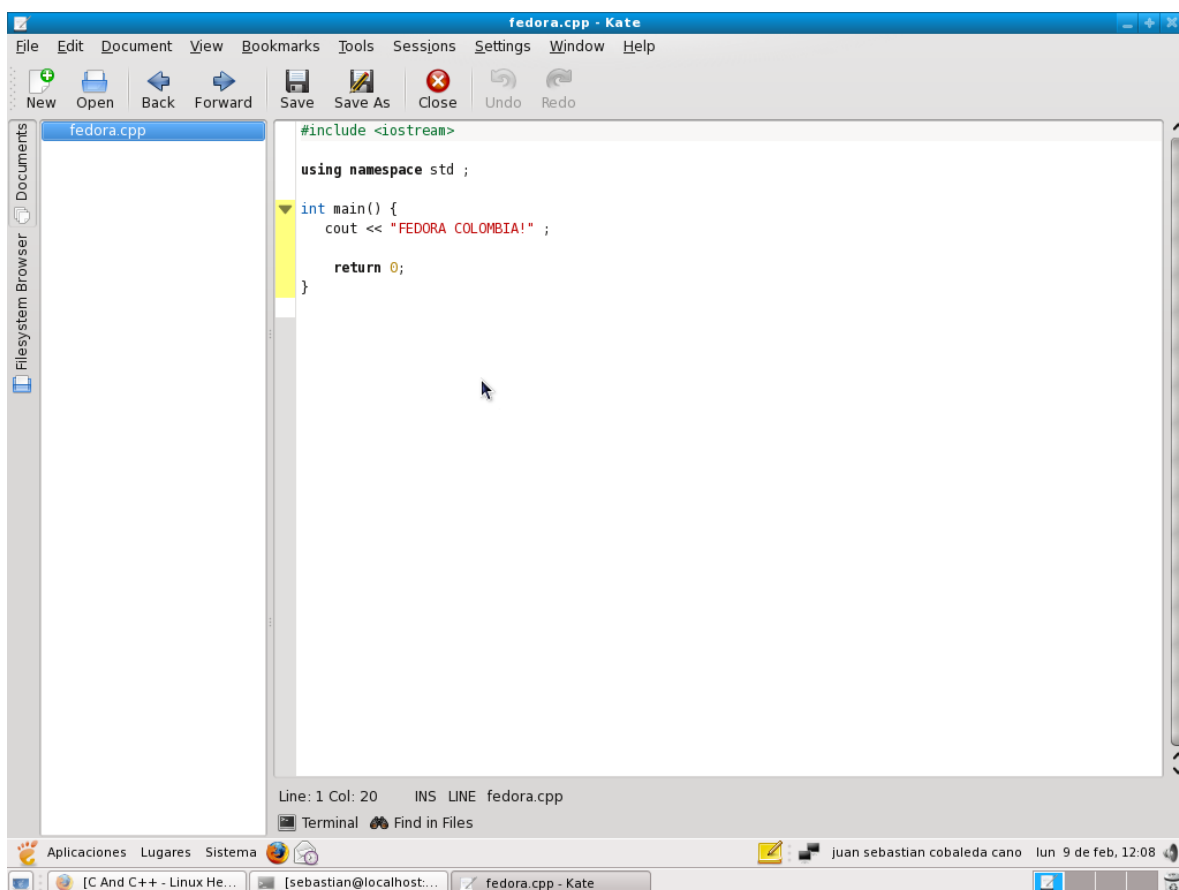
using namespace std ;

int main() {

cout << "FEDORA COLOMBIA!"<<endl ;

return 0;

}
```





Para copiar el código abrimos cualquier editor de texto que tengamos instalado, copiamos todo el código y guardamos el archivo con el nombre que queramos pero con la extensión `.c` o `.cpp` según sea el caso, en este tutorial se uso el nombre `fedora.cpp`

Después de haber guardado el archivo, abrimos la terminal, nos desplazamos hasta la ubicación del archivo y compilamos de la siguiente manera:

`"g++ fedora.cpp -o fedoracol "`

El nombre `"fedoracol"` es para renombrar el archivo de salida de la compilación

Después para ejecutar el archivo creado procedemos así en la terminal:

`" ./fedoracol "` o cualquier nombre que le hallamos dado al archivo.

A screenshot of a Linux terminal window titled 'sebastian@localhost: ~/Escritorio'. The terminal shows the following commands and output:

```
[sebastian@localhost ~]$ cd Escritorio
[sebastian@localhost Escritorio]$ g++ fedora.cpp -o fedoracol
[sebastian@localhost Escritorio]$ ./fedoracol
FEDORA COLOMBIA!
[sebastian@localhost Escritorio]$
```

The terminal window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Terminal', 'Solapas', and 'Ayuda'. The desktop environment is visible at the bottom, showing a taskbar with icons for 'Aplicaciones', 'Lugares', 'Sistema', and a panel with 'C And C++ - Linux Hell...', '[fedora.cpp - Kate]', and 'sebastian@localhost:...'. The system clock shows 'lun 9 de feb, 12:16'.



Para compilar programas en C utilizamos la sentencia:

```
"gcc nombre_del_archivo.c -o nombre_de_salida"
```

Para compilar programas en C++ utilizamos la sentencia:

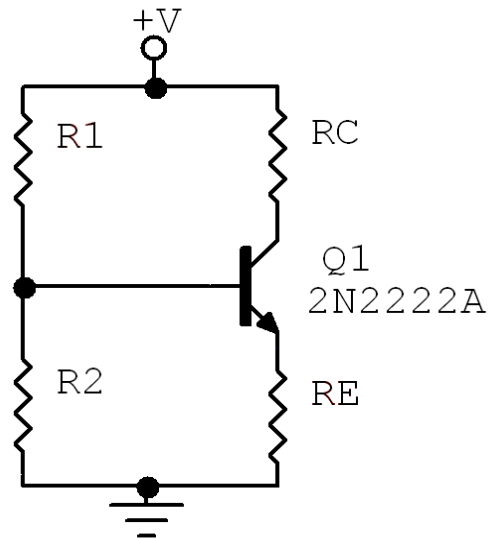
```
"g++ nombre_del_archivo.cpp -o nombre_de_salida"
```

En la cadena de sentencias para compilar. El " – o " sirve para indicar con que nombre estará el archivo de salida.

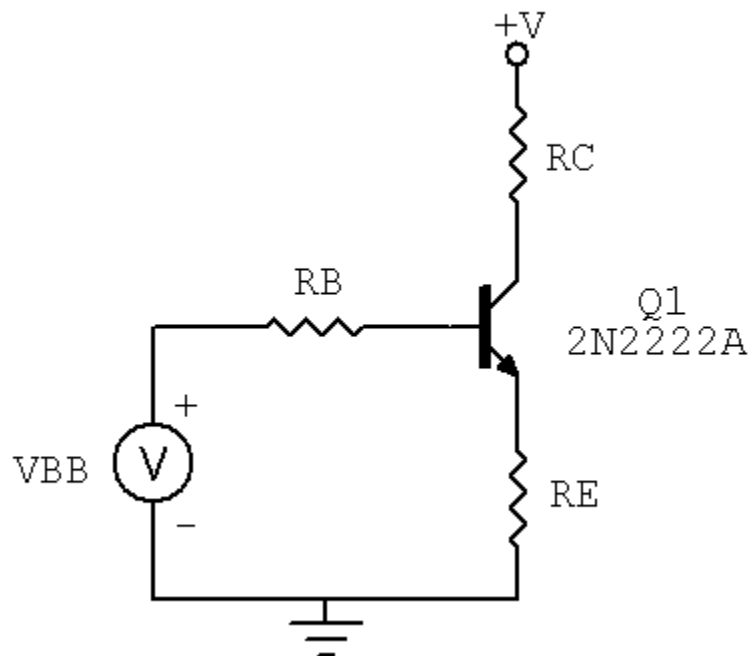
Este archivo de salida no tiene extensión (no confundir con las aplicaciones de salida.exe de Windows).

Ahora que todos sabemos programar y compilar con nuestro Fedora, dispongámonos a hacer una aplicación Electrónica.

Con este programa escrito en C++ vamos a hallar el voltaje en el colector y emisor (VCE) de un transistor NPN 2N2222A, también hallaremos la corriente en el colector (IC) del mismo transistor, todo variando los valores de las resistencias R1, R2, RC, RE y el voltaje con el que se alimentara el circuito, también dispusimos de otra variable llamada el beta del transistor (B), que es variable y depende de las características específicas de cada transistor, esta es una variable de fabrica por así decirlo, y varia según el tipo del transistor y según las condiciones de trabajo a las que lo tengamos enfrentado.



El circuito anterior se puede también representar de la siguiente manera:





Hasta ahora todo parecerá muy complicado y difícil de entender, pero mas adelante se comprenderá que todo viene dado mediante las ecuaciones matemáticas, y que así como este ejemplo estarás en la capacidad de desarrollar cualquier algoritmo matemático.

Mediante el modelado matemático del circuito anterior se llega a las siguientes ecuaciones:

$$R_B = (R_1 * R_2) / (R_1 + R_2)$$

$$V_{BB} = (V * R_2) / (R_1 + R_2)$$

$$I_B = (V_{BB} - 0.7) / (R_B + R_E * \beta)$$

$$I_C = \beta * I_B$$

$$V_{CE} = V - ((R_C + R_E) * I_C)$$

Ahora lo que tenemos que hacer es crear el algoritmo en C para incluir las anteriores ecuaciones y hallar los resultados dependiendo de los valores de las resistencias, el voltaje y el beta (B) del transistor.

Como podrán ver esto es un simple modelado matemático, que incluye las operaciones matemáticas mas básicas; ahora estas en la capacidad de crear programas y correrlos en el Fedora de una manera muy practica y lúdica.



Este es el código del ejemplo anterior:

```
#include <iostream>
#include <stdio.h>

using namespace std ;

main()
{

float V,R1,R2,RC,RE,B,VBB,IB,IC,VCE,RB;

cout<< "ingrese el valor de V "<<endl;
cin>>V;

cout<< "ingrese el valor de R1 "<<endl;
cin>>R1;

cout<< "ingrese el valor de R2 "<<endl;
cin>>R2;

cout<< "ingrese el valor de RC "<<endl;
cin>>RC;

cout<< "ingrese el valor de RE "<<endl;
cin>>RE;

cout<< "ingrese el valor del beta (B) "<<endl;
cin>>B;

RB=(R1*R2)/(R1+R2);
VBB=(V*R2)/(R1+R2);

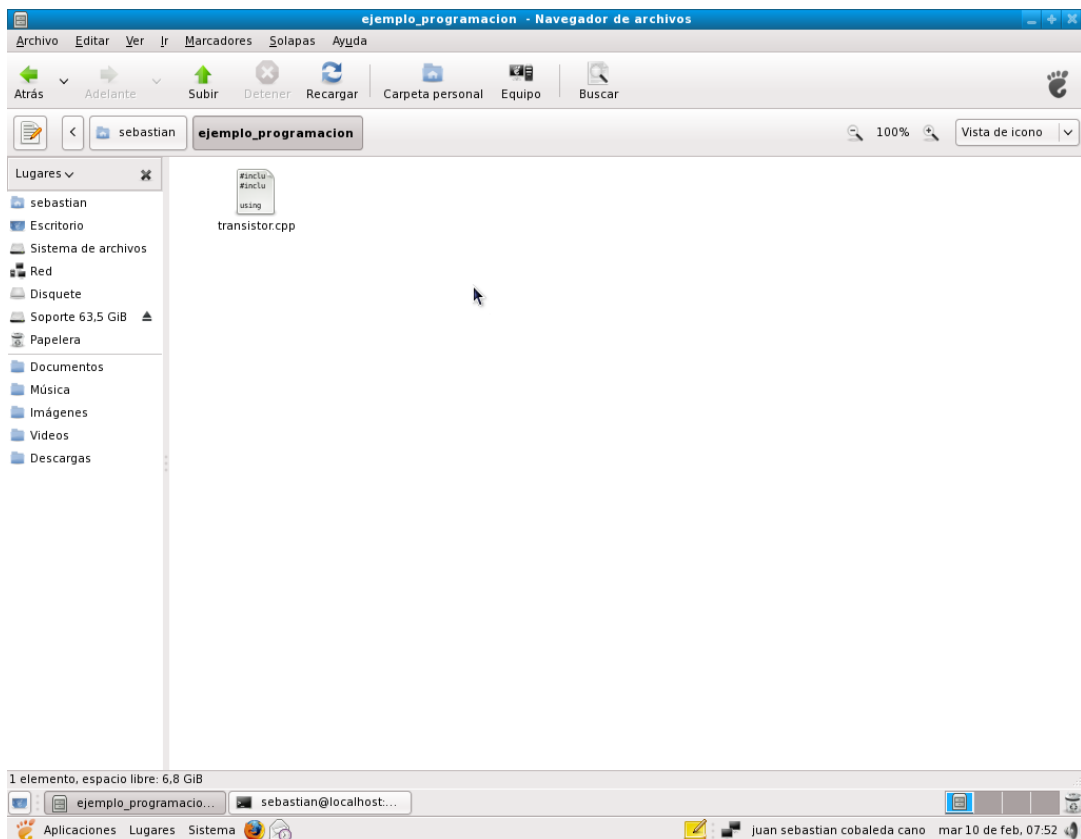
IB=(VBB-0.7)/(RB+RE*B);

IC=B*IB;
```



```
VCE=V- ((RC+RE) *IC) ;  
  
cout<<" el valor de VCE es "<<VCE<<endl;  
  
cout<<" el valor de IB es "<<IB<<endl;  
  
cout<<" el valor de IC es "<<IC<<endl;  
  
cout<<" el valor de RB es "<<RB<<endl;  
  
cout<<" el valor de VBB es "<<VBB<<endl;  
  
getchar ();  
  
return 0;  
}
```

Abrimos un editor de texto, copiamos todo el contenido y guardamos el archivo con la extension .cpp





Después por medio de la terminal navegamos hasta el directorio donde este guardado nuestro archivo y lo compilamos por medio de la terminal:

A screenshot of a terminal window titled "sebastian@localhost:~/ejemplo_programacion". The window has a menu bar with "Archivo", "Editar", "Ver", "Terminal", "Solapas", and "Ayuda". The terminal shows the command "g++ transistor.cpp -o ejemplo_transistor" being entered and executed. The prompt is "[sebastian@localhost ejemplo_programacion]\$". The window is part of a desktop environment with a taskbar at the bottom showing "Aplicaciones", "Lugares", "Sistema", and a system tray with the name "juan sebastian cobeleda cano" and the date "mar 10 de feb, 07:53".

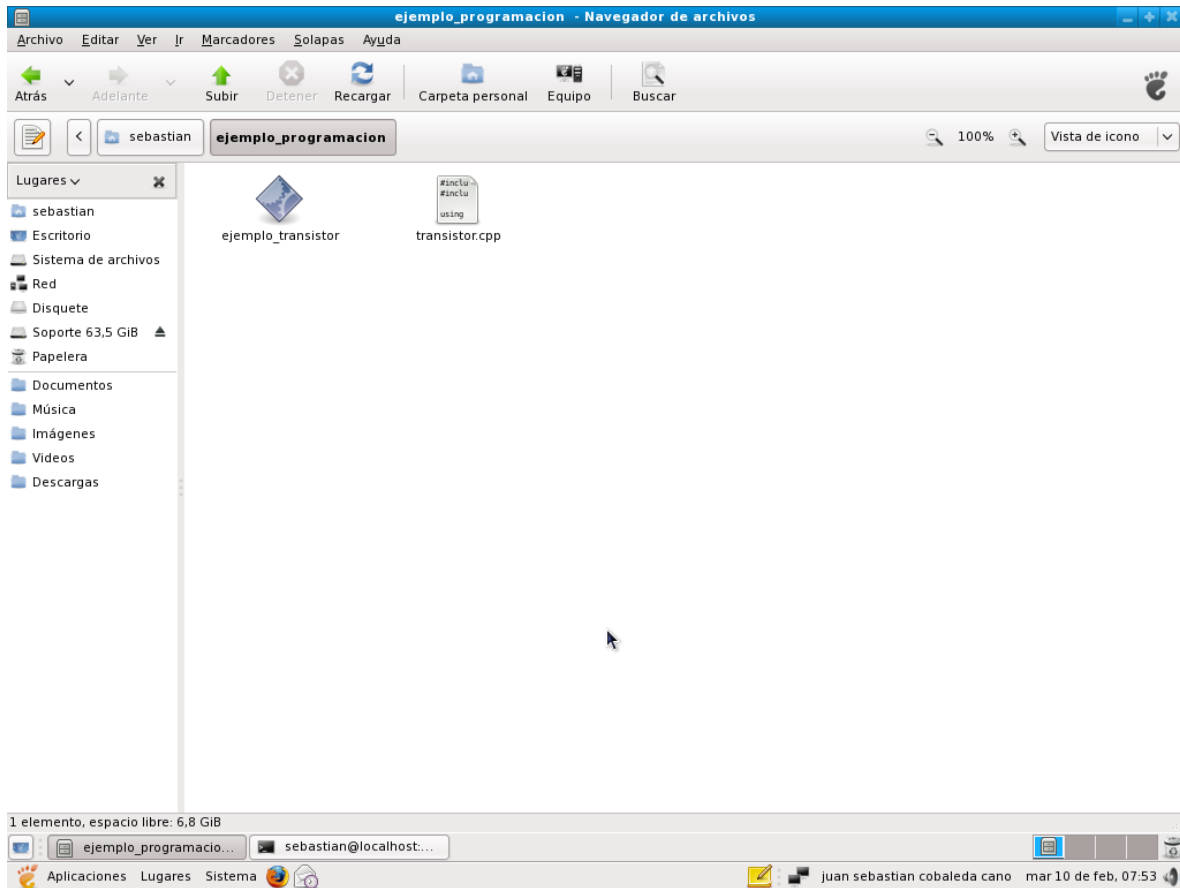
```
sebastian@localhost:~/ejemplo_programacion
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[sebastian@localhost ejemplo_programacion ]$ g++ transistor.cpp -o ejemplo_transistor
[sebastian@localhost ejemplo_programacion ]$
```

“ g++ transistor.cpp -o ejemplo_transistor “

Para compilar usamos “ g++ “ porque el código esta escrito en C++ como lo dije anteriormente.



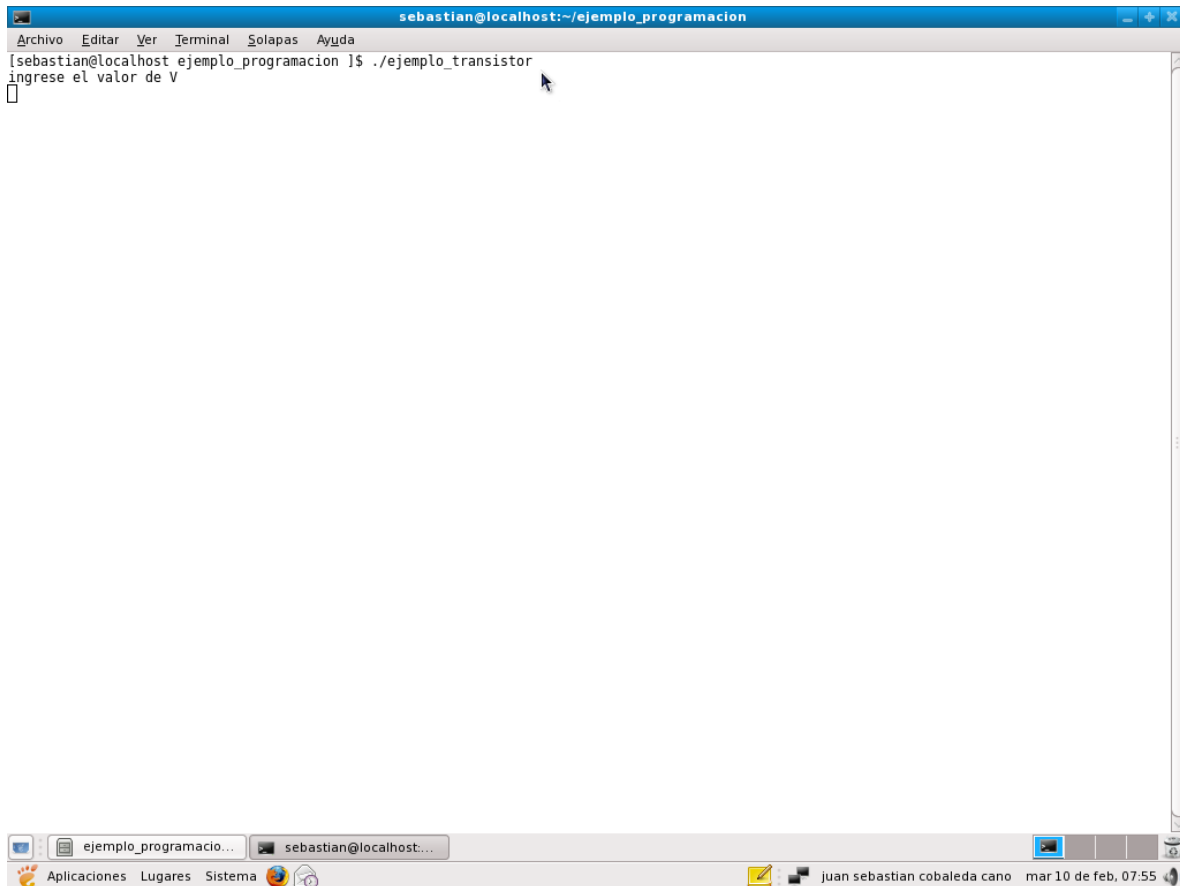
Así se nos presenta el archivo de salida con el nombre que le indicamos en el mismo directorio donde tenemos nuestro archivo “.cpp “.



Ahora que ya tenemos nuestro archivo de salida, solo tenemos que abrir el terminal para poder ejecutar nuestro nuevo programa.

Por medio de la terminal nos desplazamos hasta donde se encuentran los archivos, y copiamos lo siguiente.

“ ./ejemplo_transistor “



Ahora se nos empezara a desplegar el listado de instrucciones que anteriormente digitamos, le damos los valores que deseemos para este caso explicito.

Como lo dije anteriormente todo es resultado del modelado matemático, y así como este ejemplo podrás practicar con cualquier ecuación matemática.



Y por ultimo así se nos visualizan los resultados en el terminal:

```
sebastian@localhost:~/ejemplo_programacion
Archivo Editar Ver Terminal Solapas Ayuda
[sebastian@localhost ejemplo_programacion]$ g++ transistor.cpp -o ejemplo_transistor
[sebastian@localhost ejemplo_programacion]$ ./ejemplo_transistor
ingrese el valor de V
12
ingrese el valor de R1
1000
ingrese el valor de R2
1000
ingrese el valor de RC
1200
ingrese el valor de RE
1500
ingrese el valor del beta (B)
130
el valor de VCE es 2.4844
el valor de IB es 2.711e-05
el valor de IC es 0.0035243
el valor de RB es 500
el valor de VBB es 6
[sebastian@localhost ejemplo_programacion]$
```

Compilar proyectos en nuestro Fedora es muy fácil, ahora puedes ponerte a practicar y desarrollar tus propias aplicaciones.

Cualquier inquietud o sugerencia será bien recibida.



Juan Sebastián Cobaleda Cano
Embajador Fedora en Colombia
cobaleda@fedoraproject.org