El sistema de control de versiones git

Elaborado por: Ukranio Coronilla

A lo largo del curso y sobre todo para aplicaciones distribuidas tendremos que trabajar todos los miembros del equipo sobre el mismo proyecto. En este caso tendremos que lidiar con los siguientes problemas:

- Dos o más programadores modifican el mismo archivo y es necesario manejar dicho conflicto.
- Se requiere tener al menos una versión (la mejor para que la revise el profesor) siempre lista, mientras se tiene otra en desarrollo.
- Si se hace un cambio no deseado, poder regresar a una versión anterior estable.
- Se inicia un fork o proyecto diferente que reutiliza un código ya existente.

La solución a estos problemas no es sencilla por lo que es necesario utilizar un sistema de control de versiones (CVS). Los sistemas CVS se han estado utilizando por ejemplo, para desarrollo de software libre por parte de muchos programadores que trabajan de manera simultánea en todo el mundo.

En esta práctica veremos el CVS llamado git (creado originalmente por Linus Torvalds para desarrollar el kernel de LINUX), el cual es un sistema distribuido que almacena un repositorio (conjunto de archivos informáticos) en la computadora de cada programador, y donde los repositorios son locales hasta el momento de mandar a sincronizar con los demás repositorios.

Git es un software libre y su código se encuentra disponible para Linux, Windows, Mac y Solaris.

Instale git en la terminal de Linux UBUNTU con la instrucción:

```
sudo apt-get install git
```

Para identificar a un programador de manera univoca se utiliza su nombre y su correo electrónico. Estos los configuramos con los siguientes comandos de git:

```
git config --global user.name "Tu nombre"
git config --global user.email "correo@electronico"
```

existen gran número de opciones a configurar y puedes verlas con el comando

```
git config –help
```

También puedes ver las opciones que has configurado con:

```
git config --list
```

Para manejar color en los tipos de texto se utiliza la opción:

```
git config --global color.ui true
```

El repositorio de git se almacena en el directorio que deseemos utilizar. Solo hay que colocarnos dentro de dicho directorio y ejecutamos:

```
git init
```

Cree un directorio y dentro ejecute `git init`.

En git existe una lista virtual de archivos dentro del directorio de trabajo conocida como **INDEX**. Si quiero comenzar a trabajar con un archivo que contendrá código, primero debo crear el archivo (puede estar vacío) y después añadirlo al INDEX con la siguiente instrucción:

```
git add nombre_del_archivo
```

Añada dentro de su directorio un archivo de texto sencillo.

También se pueden añadir de manera simultánea varios archivos si estos se encuentran en un directorio y se ejecuta

```
git add nombre_del_directorio
```

Un comando muy útil para ver el estado actual del archivo de trabajo es:

```
git status
```

Observe al ejecutarlo que muestra el archivo que se ha añadido al repositorio. Ahora modifique dicho archivo y posteriormente cheque el estado actual del repositorio con `git status` para ver qué cambios existen.

Observe que `git` le informa de las posibles acciones que puede llevar a cabo, y también que si modifica un archivo es necesario ejecutar `git add` para que los cambios se almacenen en el **INDEX**. Observe que sucede si utiliza:

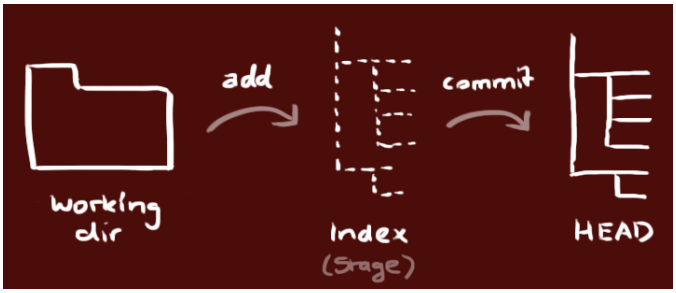
```
git checkout nombre_del_archivo
```

Modifique de nuevo el archivo para después añadirlo al **INDEX**.

En git cuando se desea guardar una versión definitiva en el repositorio se utiliza la orden de confirmación **commit**. Esto provoca que se mande al HEAD (termino git que indica que se trata de la versión actual del repositorio) el contenido del **INDEX**.

```
git commit
```

Al ejecutarlo se abrirá un editor por defecto del sistema para que se añadan los comentarios sobre los cambios efectuados. Si no se añaden comentarios recibirá un error y no se llevará acabo el **commit**.



Puedes cambiar el editor por otro, por ejemplo si se desea cambiarlo por el editor pico:

```
git config --global core.editor pico
```

Es importante recordar que para que se guarden los cambios en el commit es necesario antes ejecutar el `git add` para añadir al INDEX, y también se recomienda usar `git status` con frecuencia para probar que todo vaya bien.

Ahora con lo que hemos visto vamos a hacer un ejercicio para mostrar la característica principal de git o de cualquier sistema CVS, la capacidad de mantener distintos repositorios sincronizados a través de exportar o importar cambios.

Vamos a crear una carpeta donde almacenaremos nuestro proyecto con el nombre `directorio_de_trabajo`:

```
mkdir directorio_de_trabajo
```

Observamos las opciones configuradas y si es necesario las configuramos:

```
git config –list
```

Indicamos que `directorio_de_trabajo` es el directorio donde vamos a trabajar con git, por lo que entramos al directorio y ejecutamos:

```
git init
```

Vamos a copiar en la carpeta los archivos Coordenada.cpp, Coordenada.h, Makefile, prac_compo.cpp, Rectangulo.cpp y Rectangulo.h que obtuvimos al resolver el ejercicio 1 de la práctica 4_Composicion_en_C++_ver4.pdf. Todos estos archivos los añadimos al INDEX. Uno por uno o más fácil usando el comodín ***** :

```
git add *
```

Podemos ver el estado actual del repositorio local:

```
git status
```

Posteriormente hacemos el commit para enviar el INDEX al HEAD, y para evitar que se abra el editor usamos la opción -m como sigue:

```
git commit -m "Solución del ejercicio 1 en la práctica 4"
```

Esto va a crear nuestro primer repositorio en la computadora local y la información relacionada se va a almacenar en el archivo oculto `.git` (compruebelo).

```
cd .git
ls
cat *
```

Ahora queremos clonar nuestro repositorio a un repositorio remoto al que puedan acceder otros programadores para realizar cambios y que el CVS se encargue de mantener sincronizados todos los repositorios. Para ello vamos a utilizar un servidor en la web que mantenga nuestro repositorio, el cual sea confiable, disponible y seguro. En este curso utilizaremos ***Bitbucket*** pero el más famoso y utilizado es ***Github***.

Github es un sitio web comercial que alberga los proyectos de una comunidad mundial de programadores, el cual permite entre otras cosas que interactúen o se califiquen (poner estrellas a los proyectos) con el objeto de darse a conocer. Sin embargo las cuentas gratuitas tienen repositorios públicos, y por consiguiente cualquiera puede ver y copiar el código que estás desarrollando, lo cual no conviene en este curso por obvias razones.

Bitbucket es un sitio similar a Github pero tiene la particularidad de ofrecer espacio privado e ilimitado para repositorios en equipos de hasta 5 personas, exactamente lo que queremos en nuestro curso. Lo primero que haremos es registrarnos en:

https://bitbucket.org

Después de responder el e-mail para verificar el correo de usuario y contestar la pequeña encuesta damos click en "*Crete repository*" y creamos un repositorio para esta práctica con el nombre de `holabitbucket`. Dejamos marcada la opción de repositorio privado, no incluimos un archivo README y el control de versiones como git.

Al hacerlo nos devolverá la URL que podremos utilizar, algo similar a:

https://usuario@bitbucket.org/usuario/holabitbucket.git

Dado que bitbucket es más confiable, disponible y seguro que nuestra computadora, asumiremos que el origen del código y el repositorio principal se encuentra en bitbucket.

Ahora añadimos a nuestro directorio de trabajo la ubicación del repositorio en bitbucket que acabamos de crear y que se supone tendrá el origen de todo el código. Nos colocamos dentro de `directorio_de_trabajo` y ejecutamos `git remote add origin` con la URL recibida:

```
git remote add origin https://usuario@bitbucket.org/usuario/holabitbucket.git
```

Y entonces al ejecutar

```
git remote –v
```

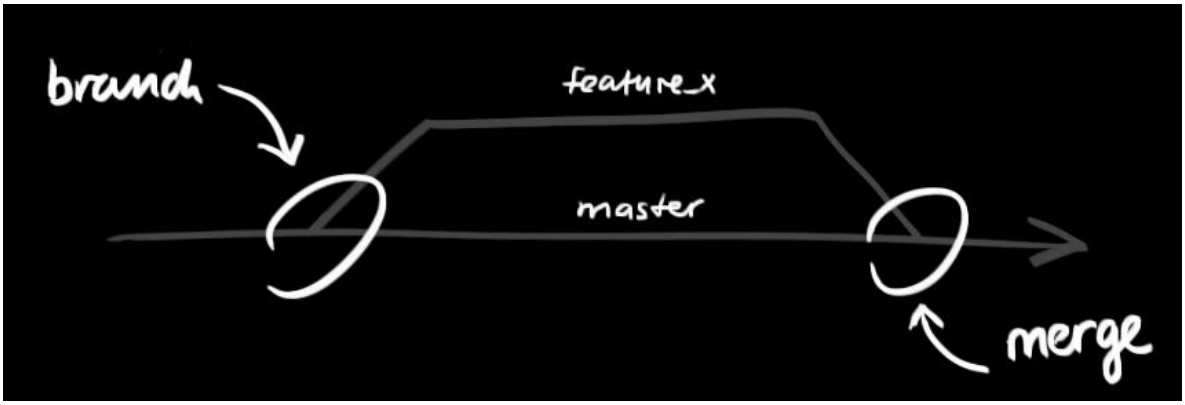
podremos ver algo como:

```
origin https://usuario@bitbucket.org/usuario/holabitbucket.git (fetch)
origin https://usuario@bitbucket.org/usuario/holabitbucket.git (push)
```

indicando que existe un repositorio llamado **origin** que se usará tanto para enviar (push) como para recibir (fetch) los cambios que se realicen. Aunque solo hemos añadido un repositorio llamado **origin**, es posible añadir más repositorios (con distinto nombre) y que todos ellos se sincronicen. También es posible eliminar alguno con:

```
git remote rm NOMBRE
```

Observe que git no comprueba que ese repositorio exista en Internet o si se tiene acceso, por ello no hay garantía de que pueda usarlo en ese momento.



Para enviar los cambios del repositorio local al repositorio de gitbucket solo ejecutamos:

```
git push origin master
```

Donde master es la rama (branch) de desarrollo principal por default que utiliza bitbucket, pero puede haber más ramas de desarrollo. Cada rama se usa para desarrollar alguna característica adicional y al final se vuelven a mezclar las ramas con el master para obtener una nueva versión completa. Observe que le pedirá su contraseña. Al terminar puede verificar que el repositorio ya está disponible en el servidor de bitbucket en una dirección similar a:

https://bitbucket.org/usuario/holabitbucket/src/master/

Trate de ingresar a esta liga desde otra computadora y observe que no será posible debido a que su repositorio es privado, cosa que no sucede con las cuentas gratuitas de github.

Ahora vamos a permitir que otros compañeros trabajen con nuestro proyecto. Para eso damos click en el proyecto y a continuación en *Settings* (columna de la izquierda). Posteriormente click en User and group access y en la ventana de entrada de *Users* introducimos al usuario para que bitbucket lo busque y lo agregue. Seleccionamos *Read*(permisos de lectura) *Write*(permisos de escritura) o *Admin*(todos los permisos), y posteriormente click al botón *Add* (en este caso usaremos la opción más común *Write*).

El usuario al que le dimos permiso, ahora podrá visualizar el nuevo repositorio al darle click en *Your work* de la página principal. También podrá clonar el repositorio ya existente y como lo indica bitbucket, con una instrucción similar a la siguiente se clona un repositorio que está disponible en el sitio web de bitbucket.org :

git clone https://usuario_invitado@bitbucket.org/usuario_admin/repositorio.git

Al clonar se almacenan no solo los archivos, sino también el historial y mucha otra información, por ejemplo dentro de la carpeta que se clonó se puede ejecutar el comando para ver las opciones activadas del repositorio (`git config --list`). Observe que *remote.origin.url* contiene la URL donde se almacena el repositorio que se clonó. Git reconoce la ruta donde se encuentra y con ello el repositorio en el cual está trabajando.

Ejercicios:

Uno de los miembros del equipo (ADMIN) debe crear un nuevo proyecto con el código de los archivos Coordenada.cpp, Coordenada.h, Makefile, prac_compo.cpp, Rectangulo.cpp y Rectangulo.h que obtuvimos al resolver el ejercicio 1 de la práctica 4_Composicion_en_C++_ver4.pdf . Los demás miembros del equipo deberán clonar el proyecto en sus respectivas computadoras.

Cada usuario debe crear una nueva clase que incluya la clase coordenada (por ejemplo la clase Cuadrado). Con lo visto hasta ahora documente los pasos a seguir para que todos los miembros del grupo tengan las clases creadas por todos (usar fetch).

Ahora cada usuario modifique el archivo Makefile y el main para que haga uso de la clase que elaboró. Documente que sucede en estos casos donde todos los usuarios están modificando el mismo archivo. Investigue y practique con el uso de ramas en git y comandos como los siguientes:

git branch nueva-rama

git checkout nueva-rama

git checkout master

git branch

git diff

git log

git blame

Recursos consultados:

http://rogerdudler.github.io/git-guide/

Aprende Git: ... y, de camino, GitHub Edición Kindle

por Juan Julián Merelo Guervós (Autor), Ángel Pablo Hinojosa Gutiérrez (Autor)