PEC3 Control de Versiones y Documentación

Francisco Maya Sarasty

19-12-2015

Contents

1 Introducción

La tercera prueba de evaluación continua (PEC) de la asignatura de Desarrollo de Software del Master Universitario en Software Libre trata el tema de control de versiones y documentación. En esta PEC se crea un repositorio para el desarrollo controlado de la librería de filtros y se genera la documentación del código mediante la utilización de herramientas especializadas para cada trabajo.

En un principio se hace referencia al control de versiones mediante la creación de un repositorio con Git y se continua con la documentación del código mediante la utilización de Doxygen y LaTeX.

2 Control de Versiones

1. Nos ubicamos en el directorio para el cual aplicaremos el sistema de control de versiones

```
$ git init

> Initialized empty Git repository in
    /home/francisco/ProMaestria/DesarrolloSW/PEC3/.git/
```

   Ahora tenemos nuestro directorio "PEC3" donde se encuentra el repositorio en /.git/

2. Verificamos el estado actual del proyecto para ver que ha cambiado

```
$ git status

> En la rama master
> Commit inicial
> nada que hacer...
```

3. Se crea los archivos en nuestro repositorio o directorio de trabajo, así que creamos el directorio "libsrc/" y dentro de este, los archivos "filtros.c" y "filtros.h"

4. Verificamos el estado del proyecto

```
$ git status

> En la rama master
> Commit inicial
> Archivos sin seguimiento:
>   libsrc/
> no se ha agregado nada al commit pero existen archivos sin
    seguimiento...
```

   Muestra que nuestros archivos aun no estan siendo manejados por Git

5. Ahora se escribe "git add ." para hacer seguimiento a todos los archivos o se hace para cada archivo con un comando similar a "git add libsrc/filtros.c"

```
$ git add libsrc/filtros.c
$ git status

> En la rama master
> Commit inicial
> Cambios para hacer commit:
```

```
>   new file:  libsrc/filtros.c
> Archivos sin seguimiento:
>   libsrc/filtros.h
```

Nos muestra que se agrego en la lista para hacer "commit" el archivo "filtros.c", tal como se le especifico en el comando.

6. A continuacion agregamos todos los archivos para que queden en lista de seguimiento.

```
$ git add .
$ git status

> En la rama master
> Commit inicial
> Cambios para hacer commit:
>   new file:  libsrc/filtros.c
>   new file:  libsrc/filtros.h
```

Nos muestra los archivos pendientes por hacer "commit", en este caso los dos de la biblioteca filtros.

7. Para aplicar los cambios hacemos el "commit" correspondiente con un mensaje para precisar sobre qué trata este "commit".

```
$ git commit -m "Agregar biblioteca de filtros"

> [master (root-commit) 8926324] Agregar biblioteca de filtros
>  2 files changed, 107 insertions(+)
>  create mode 100644 libsrc/filtros.c
>  create mode 100644 libsrc/filtros.h
```

8. Se ejecuta nuevamente git status, lo que no muestra que no hay nada pendiente

```
$ git status

> En la rama master
> nothing to commit, working directory clean
```

9. Una vez tenemos nuestro repositorio podemos hacer el "push" hacia el servidor de GitHub, para lo cual utilizaré mi propia cuenta de este sitio (https://github.com/franciscomaya/pec3.git)

```
$ git remote add origin https://github.com/franciscomaya/pec3.git
```

10. Ahora se debe enviar los "commits" cuando se termine de modificar nuestra biblioteca

```
$ git push -u origin master

> Username for 'https://github.com': franciscomaya
> Password for 'https://franciscomaya@github.com':
> Counting objects: 5, done.
> Delta compression using up to 4 threads.
> Compressing objects: 100% (4/4), done.
> Writing objects: 100% (5/5), 984 bytes | 0 bytes/s, done.
> Total 5 (delta 0), reused 0 (delta 0)
> To https://github.com/franciscomaya/pec3.git
>  * [new branch]    master -> master
> Branch master set up to track remote branch master from origin.
```

Nos pide nuestro usuario y clave de Github y envia los archivos de nuestra biblioteca, como se puede comprobar en https://github.com/franciscomaya/pec3

11. Ahora supongamos que ha pasado ya un tiempo considerable desde nuestro último aporte y probablemente haya cambios en el repositorio de Github realizados por otros usuarios, revisamos los cambios en el repositorio de Github y bajamos cualquier nuevo cambio con el siguiente comando

```
$ git pull origin master

> remote: Counting objects: 4, done.
> remote: Compressing objects: 100% (3/3), done.
> remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
> Unpacking objects: 100% (4/4), done.
> De https://github.com/franciscomaya/pec3
>  * branch          master    -> FETCH_HEAD
>    8926324..54fcae2 master   -> origin/master
> Updating 8926324..54fcae2
> Fast-forward
>  libsrc/filtros.c | 1 +
>  1 file changed, 1 insertion(+)
```

Donde se evidencia que se modificó un archivo.

12. Si quisieramos colaborar en otro proyecto, es decir, queremos tener una copia completa de un repositorio de otro usuario, para eso se debe clonar su repositorio ubicándonos en el directorio donde se realizará la copia

```
$ git clone https://github.com/franciscomaya/pec3.git

> Clonar en pec3...
> remote: Counting objects: 9, done.
> remote: Compressing objects: 100% (7/7), done.
```

```
> remote: Total 9 (delta 1), reused 4 (delta 0), pack-reused 0
> Unpacking objects: 100% (9/9), done.
> Checking connectivity... hecho.
```

Una vez hecho esto, tendremos nuestro repositorio local listo para realizar los trabajos que tengamos planeados para ese proyecto

3 Documentación

Documentamos los archivos de nuestra biblioteca (filtros.h y filtros.c), siguiendo los pasos que se describen a continuación:

1. Para generar una descripción de cada archivo utilizamos la etiqueta "file" y las etiquetas habituales como lo son "brief", "author" y "date" donde se explica algo general de lo que se encuentra en nuestro archivo.

2. En nuestro archivo "filtros.c" para cada función se documenta de manera abreviada y detallada el funcionamiento de la misma, sus parámetros, lo que devuelve y se enlaza hacia los otros filtros de la biblioteca, así como a las funciones internas que cada función utiliza.

3. Una vez terminamos de documentar nuestro trabajo, es momento de pasar a Doxygen para generar el código HTML de la documentación. Para facilitar nuestro trabajo utilizamos doxygen-gui para agilizar su configuración.

4. Ya en Doxygen GUI, especificamos el directorio de trabajo donde Doxygen se ejecutará, para este caso: /home/francisco/ProMaestria/DesarrolloSW/PEC3

5. Configuramos Doxygen utilizando el asistente, entre otras, ingresamos la siguiente configuración en la pestaña "Wizard":

```
Project
 Project name: PEC3
 Project synopsis: Tercera prueba de evaluacion continua
 Source code directory:
     /home/francisco/ProMaestria/DesarrolloSW/PEC3
  * Marcando la casilla "Scan recursively" para que busque en
      todos los directorios de la ruta especificada
 Destination directory:
     /home/francisco/ProMaestria/DesarrolloSW/PEC3/doc

Mode
 Select the desired extraction mode:
  Seleccionando "All entities"
  * Marcando la casilla "Include cross-referenced source code in
      the output"
 Select programming language to optimize the results for
  Seleccionando "Optimize for C or PHP output"

Output
 Select the output format(s) to generate
  Marcando "HTML -> plain HTML"

Diagrams
 Diagrams to generate
  Seleccinando "Use built-in class diagram generator"
```

6. En la pestaña "Expert" cambiamos "OUTPUT LANGUAGE" A "Spanish" para que nuestra documentación de HTML aparezca en español.

7. En la pestaña "Run", hacemos click sobre el botón "Show configuration" para revisar el proceso.

8. Por último se hace click en el botón "Run doxygen" y podemos dirigirnos hacia la ruta que especificamos en "Destination directory" para abrir el archivo index.html y revisar la documentación generada.

4 Conclusiones