

# Control de Versiones

PPROG

# Introducción

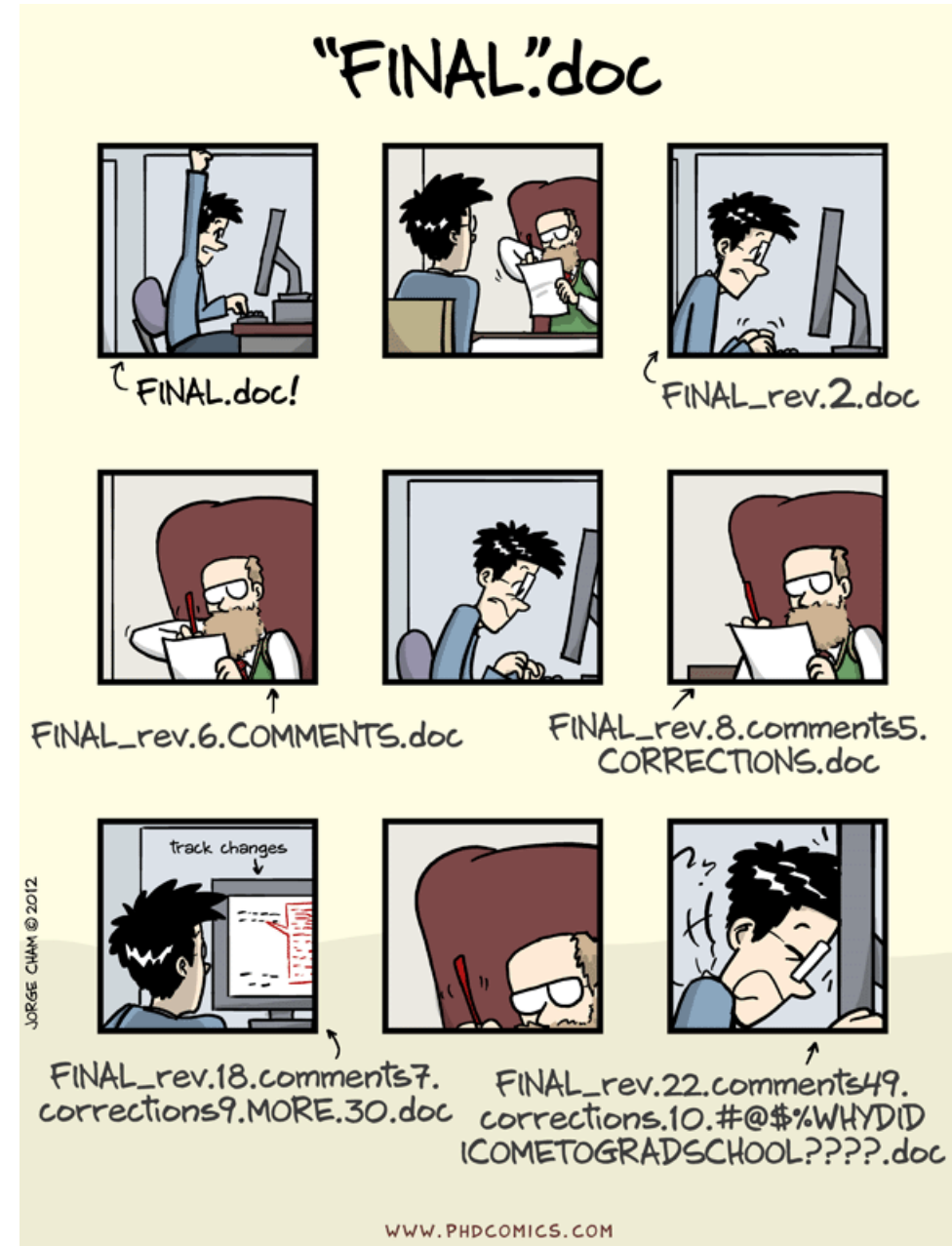
- Incluso cuando se trabaja solo es importante mantener un control de versiones eficiente
  - ▣ También aplicable a equipos “pequeños”
- No sólo hay que tener versiones del código
  - ▣ Documentos: memorias, manuales, actas, referencias
  - ▣ Scripts: makefile
  - ▣ Recursos multimedia: imágenes, vídeos
  - ▣ Ficheros de datos
  - ▣ Código ajeno: librerías, código externo

# Control de versiones no eficiente



# Control de versiones no eficiente

Aún así, mejor que no  
tener ningún sistema de  
control!



# Flujos de trabajo

- A nivel de carpeta
  - ▣ Copiar la versión actual en una carpeta temporal
    - Posibles nombres: “old”, “temp”, “20150209”
  - ▣ Editar en la carpeta
  - ▣ En caso de borrado accidental o necesidad de recuperar datos: acceder a la carpeta temporal
  - ▣ Desventajas: espacio en disco, frecuencia óptima, ¿qué ha cambiado?

# Flujos de trabajo

- A nivel de fichero
  - ▣ Duplicar el fichero que se va a editar
    - Posibles nombres: “file-01”, “file-01\_sec1”, “file-20150209”
  - ▣ Editar el fichero
  - ▣ En caso de borrado accidental o necesidad de recuperar datos: acceder a la versión anterior
  - ▣ Desventajas: frecuencia óptima, ¿qué ha cambiado?

# Flujos de trabajo

- A nivel de fichero (más profesional)
  - ▣ Editar el fichero que interesa
  - ▣ Una vez terminado, se envía un correo (a una cuenta personal) con el fichero
    - Describiendo los cambios
    - Bonus: fecha y nombre de fichero pueden ir en asunto
  - ▣ En caso de borrado accidental o necesidad de recuperar datos: acceder al correo y buscar
  - ▣ Desventajas: espacio en cuenta de correo, frecuencia óptima, internet

# Herramientas

- **diff** identifica diferencias entre ficheros línea a línea
- **merge** une dos ficheros en uno, teniendo en cuenta la parte común y no común de los mismos
  - “>” indica borrados
  - “<” indica inserciones

```
vivek@wks01:/tmp$ diff rhel.setup.sh rhel.setup-0.sh
21c21
< /bin/echo 'GATEWAY=192.168.1.254' >> /etc/sysconfig/network
---
> /bin/echo 'GATEWAY=192.168.1.253' >> /etc/sysconfig/network
23c23
< /bin/echo '192.168.1.254 server1.cyberciti.biz serer1' >> /etc/hosts
---
> /bin/echo '192.168.1.253 server2.cyberciti.biz serer2' >> /etc/hosts
vivek@wks01:/tmp$ colordiff rhel.setup.sh rhel.setup-0.sh
21c21
< /bin/echo 'GATEWAY=192.168.1.254' >> /etc/sysconfig/network
---
> /bin/echo 'GATEWAY=192.168.1.253' >> /etc/sysconfig/network
23c23
< /bin/echo '192.168.1.254 server1.cyberciti.biz serer1' >> /etc/hosts
---
> /bin/echo '192.168.1.253 server2.cyberciti.biz serer2' >> /etc/hosts
```

```
Hunk 1: Lines 7-8
7 7      import es.uam.eps.ir.beans.topic.TopicModelBean;
8      - import es.uam.eps.ir.configuration.ConfigurationTopicDetection;
9 8      import es.uam.eps.ir.lang.LanguageDetector;

Hunk 2: Lines 42-43
43 42      private Map<String, Set<SimilarUserTopicsBean>> currentUserNeighbourhoods;
44      -
45 43      private Map<String, TopicLanguageProbabilityBean[]> currentCommunity;

Hunk 3: Lines 131-135
133 131      TopicDetection model = currentModels[i];
134      - model.save(new File(getTopicFilename(saveFolder, langs, join, i)));
132      + if (model != null) {
133      +     model.save(new File(getTopicFilename(saveFolder, langs, join, i)));
134      + }
135 135      }

Hunk 4: Lines 188-192
188 188      String ret = rs.getString("c");
189      + if (ret == null) {
190      +     continue;
191      + }
189 192      for (String tweet : ret.split(separator)) {

Hunk 5: Lines 334-341
331 334      while (rs.next()) {
332      - String ret = rs.getString("c");
333 335      String id = rs.getString("id");
334      - TopicLanguageProbabilityBean[] topics = detectTopics(ret, 1, langs, separator);
335      - newCommunity.put(id, topics);
336      + String ret = rs.getString("c");
337      + if (ret != null) {
338      +     TopicLanguageProbabilityBean[] topics = detectTopics(ret, 1, langs, separator);
339      +     newCommunity.put(id, topics);
340      + }
336 341      }
```



# Alternativas

- Herramientas colaborativas (vistas en tema de trabajo en grupo)



- También son útiles con una sola persona

- Permiten automatizar los flujos de trabajo

- ❑ No hay que preocuparse de mantener las versiones

- Otras opciones

- ## ■ Sistemas de Control de Versiones

	Has añadido <a href="#">build</a> y 2 carpetas más.	Hace 46 min
	Has modificado el archivo <a href="#">main.c</a> .	Hace 47 min
	Has modificado el archivo <a href="#">main.c</a> .	Hace 47 min
	Has eliminado el archivo <a href="#">main.cpp</a> .	Hace 47 min
	Has añadido el archivo <a href="#">main.c</a> .	Hace 48 min
	Has añadido <a href="#">Makefile</a> y 12 archivos más.	Hace 48 min
	Has añadido <a href="#">basico</a> y 2 carpetas más.	Hace 48 min

# Sistemas de Control de Versiones

- Gestionar (versionar) **cambios** en el código
  - ▣ Quién hizo qué cambios y dónde
- Objetivo cuando se trabaja en grupo:
  - ▣ Construir el proyecto de manera colaborativa
  - ▣ Sin destruir el trabajo de otro cuando hay conflictos!

# Sistemas de Control de Versiones

- **Repositorio** donde se almacenan los ficheros
  - ▣ Centralizados: CVS, SVN
  - ▣ Distribuidos: Git, Mercurial, Bazaar
  
- Permiten manejar distintas versiones del código
  - ▣ Ej. deshacer cambios, volver a una versión anterior
  
- Algunas forjas conocidas:
  - ▣ Github, bitbucket, GoogleCode, Assembla