Module02-05
Linux 开发环境 : CVS

Linux 开发环境 - CVS

- vim
- 使用 gcc/g++
- make 和 makefile
- 使用 gdb
- ➔ CVS
- Eclipse CDT

CVS - Concurrent Version System

在本次课程中，我们将通过一系列的操作来熟悉 CVS 这一古老而强大的版本控制工具，主要操作如下：

- 访问 CVS 仓库 (repository)
- 在 CVS 仓库中添加一个新项目、创建本地工作目录
- 同步本地与仓库中的代码
- 添加、删除文件与目录
- 解决冲突
- tag 和 sticky tag
- 为项目创建新版本
- 创建分支
- 合并分支

CVS - 一些约定

- 本次课程的一些约定：
  - 本次课程涉及的内容都是 cvs client 端操作，关于安装、管理（如初始化仓库等）等请参见课件《实验指导 - CVS 安装配置》
  - CVS 仓库： /projects（由于使用 cvsd 管理方式，目录 projects 并非实际位于根目录下）
  - 连接方式： pserver (password-authenticated server)
  - CVS Server 使用默认的 2401 端口
- 操作注意点：
  - 所有 cvs 命令，如果未指定文件名，默认对当前目录操作

CVS - 访问 CVS 仓库

- 访问 CVS 仓库的方式：
  - 通过 -d 选项 指定仓库位置：

```
# 一般的访问方式:
# cvs -d :method:user[:passwd]@server[:[port]]/repository cmd
#  如:
$ cvs -d :pserver:kwarph@localhost/projects login    # 先登录！
$ cvs update     # 在前面命令正确访问 CVS 仓库后，之后的命令不需再指定 -d 选项
```

  - 设置 CVSROOT 环境变量：

```
#CVSROOT 环境变量可以单次设置，也可以写入 ~/.bash_profile 或 ~/.bashrc :
$ export CVSROOT=:pserver:kwarph@localhost/projects

# 在设置好环境变量后，可以简单按如下方式执行 cvs 命令
$ cvs login        # 先登录！
$ cvs update
$ cvs diff -r1.1 -r1.5 main.cpp
```

CVS - 访问 CVS 仓库

- 访问 CVS 仓库的方式（续）
  - 关于访问 CVS 仓库的一些注意点：
    - 如果不需要频繁访问不同的 CVS 仓库（只访问一个 CVS 仓库），尽量定义 CVSROOT 环境变量，且将其写入 ~/bash_profile 或 ~/.bashrc
    - 在执行 cvs login 命令后执行其它命令

CVS - 向 CVS 仓库添加新项目

- 向 CVS 仓库添加新项目
  - import 命令

```
cvs import [-m comment] projname vendortag releasetag
```

  - 说明:
    - import 仅仅只在新项目第一次提交时使用
    - 如不带 -m 选项， cvs 将调用编辑器提示编辑本次 import 信息
    - projname：项目在仓库中的名称（可以与本地目录名不同）
    - vendortag：提供者
    - releasetag：版本标签（必须以字母开头，无特别意义）
  - 示例:

```
$ pwd
/home/kwarph/Training/Module02/context/make
$ cvs import -m 'import new project' make_proj tiger start
```

CVS - 创建本地工作目录

- 创建本地工作目录
  - 取出 CVS 仓库中的项目， checkout 命令，可简写为 co：

```
$ pwd
/home/kwarph/Training/Module02/context/cvs
$ cvs checkout make_proj    # 如果为第一次 checkout，则创建工作目录
```

  - 说明：
    - 通过 import 命令在服务器中创建一个新项目后，本地源目录与 CVS 仓库中的项目并无关联。
    - 第一次 checkout 后，会在本地创建一个工作目录，这个目录将于 CVS 仓库中的项目关联
    - 在工作目录以及各级子目录下，会有一个名为 CVS 目录，这个目录用于与 CVS 仓库中的项目形成关联

CVS - 工作目录的结构

- 工作目录一览

```
./make_proj/
|--- build/
|     |--- CVS/
|     |--- Makefile
|     |--- sub1/
|     |     |--- CVS/
|     +--- sub2/
|           |--- CVS/
|--- inc/
|     |--- CVS/
|     |--- *.h
|--- src/
|     |--- CVS/
|     |--- main.cpp
|     |--- sub1/
|     |     |--- CVS/
|     |     |--- *.cpp
|     +--- sub2/
|     |     |--- CVS/
|     |     |--- *.cpp
```

CVS - 删除本地工作目录

- 删除本地工作目录
  - release 命令
  - 示例：

```
$ pwd
/home/kwarph/Training/Module02/context/cvs/make_proj
$ cd ..
$ cvs -q release -d make_proj
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory
`make_proj': y
$ ls
$
```

CVS - 提交更改

- 提交更改
  - 修改工作目录中的文件

```
$ cat main.cpp
     1  #include <iostream>
     2
     3  int main() {
     4      std::cout << "hello, world!\n";
     5      return 0;
     6  }
# 做一些更改后:
$ cat main.cpp
     1  #include <iostream>
     2
     3  int main() {
     4      std::cout << "hello, world!\n";
     5      std::cout << "hello again!\n";
     6      return 0;
     7  }
```

CVS - 提交更改

- 提交更改（续）
  - commit 命令：提交更改后的文件或目录，可简写为 ci
    - `cvs commit [-m comment] file` 提交文件 file
    - `cvs commit [-m comment]` 提交当前目录
  - 示例：

```
# 注意：如果不使用 -m 选项指定信息，cvs 会开启编辑器提示输入信息
$ cvs commit -m 'add a line' main.cpp
/projects/make_proj/src/main.cpp,v  <--  main.cpp
new revision: 1.2; previous revision: 1.1
$ cvs commit -m 'test only'
cvs commit: Examining .
cvs commit: Examining sub1
cvs commit: Examining sub2
```

CVS - 处理冲突

- 处理冲突：
  - 冲突的产生：
    - 并行开发过程中多人同时修改同一个文件，在提交过程中有可能出现版本冲突的现象
    - 合并分支到主干时有可能发生冲突
  - 冲突的解决：

    (Demo)

CVS - 查看文件的状态

- 查看文件的状态
  - 命令 status：可简写为 st

```
$ cvs st main.cpp
===================================================
File: main.cpp          Status: Up-to-date

   Working revision:    2.2
   Repository revision: 2.2
/projects/make_proj/src/main.cpp,v
   Commit Identifier:   g6lTgNGFzUk9zOfu
   Sticky Tag:      rel_1_0 (revision: 2.2)
   Sticky Date:     (none)
   Sticky Options:  (none)
```

CVS - 查看版本之间的差别

- 查看版本之间的差别
  - diff 命令：比较文件各版本之间的差异
    - `cvs diff -r rev1 -r rev2 file`　　（常用方式）
  - 示例：

```
$ cvs diff -r1.1 -r1.2 main.cpp
Index: main.cpp
===================================================================
RCS file: /projects/make_proj/src/main.cpp,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
4a5          #4a5 意思是在原先第四行下添加第五行， a- 添加， d- 删除
>   std::cout << "hello again!\n";
```

CVS - 同步本地与仓库中的代码

- 同步本地与仓库中的代码
  - update 命令：简写为 up
    - `cvs update file`　　（同步指定文件）
    - `cvs update`　　（同步当前目录）
  - 示例：

```
$ vi main.cpp
$ cvs up
cvs update: Updating .
M main.cpp                    # 状态 M，表示本地文件做了修改，尚未提交
cvs update: Updating sub1
cvs update: Updating sub2
$ cvs ci main.cpp             # 提交更改
```

特别注意：良好的操作习惯是，首先 update，同步本地与 CVS 仓库的文件，然后才做修改，否则在并行开发过程中会出现文件版本冲突的现象。

CVS - 添加文件和目录

- 添加文件和目录
  - add 命令：
    - cvs add file(dir)
    - cvs add -kb file　　（添加二进制文件，使用 -kb 选项）
  - 示例：

```
$ echo some text > tmpfile
$ cvs add tmpfile
cvs add: scheduling file `tmpfile' for addition
cvs add: use `cvs commit' to add this file permanently
$ cvs ci -m 'a new file' tmpfile   # 提交后才真正添加到仓库
/projects/make_proj/src/tmpfile,v  <--  tmpfile
initial revision: 1.1
```

CVS - 删除文件

- 删除文件
  - remove 命令：简写为 rm
    - `cvs rmove file`
  - 示例：

```
$ rm tmpfile          # 先删除本地工作目录中的文件
$ cvs rm tmpfile      # 再执行 cvs remove 命令
cvs remove: scheduling `tmpfile' for removal
cvs remove: use `cvs commit' to remove this file
permanently
$ cvs ci -m 'remove tmpfile'   # 提交后才真正从 CVS 仓库中删除
cvs commit: Examining .
......
cvs commit: Examining sub2
/projects/make_proj/src/tmpfile,v  <--  tmpfile
new revision: delete; previous revision: 1.1
```

CVS - 删除目录

- 删除目录
  - 说明：删除目录相对比较麻烦，操作如下：

```
#1 先删除目录下的文件
$ cd src/sub1/
$ rm *.cpp
$ cvs rm dummy1.cpp dummy2.cpp #2 再执行 cvs remove 命令
......
$ cvs ci -m 'remove two files'   #3 提交后才真正从 CVS 仓库中删除
......
$ cd ..     #4 离开 sub1 目录
$ ls
CVS  main.cpp  sub1  sub2
$ cvs up -P     #5 执行 update -P
cvs update: Updating .
cvs update: Updating sub1
cvs update: Updating sub2
$ ls
CVS  main.cpp  sub2
```

CVS - 从本地工作目录创建 tag

- 从本地工作目录创建 tag
  - 为什么需要 tag：为项目打 tag 可以将项目当前的状态，如各个文件的版本等信息记录下来。日后我们可以根据 tag 将项目取出，并且各个文件的版本等信息同打 tag 时一致

```
# 切换到工作目录下，为本项目创建一个 tag：
$ cvs tag rel-1-0

# 更安全的做法是：在打 tag 时启用 -c 选项，要求本地工作目录与仓库的代码同步，否则命令执行失败
$ cvs tag -c rel-1-0

# 切换到其它路径，取出 rel-1-0
cvs co -r rel-1-0 make_proj
```

CVS - 处理粘性标签（sticky tag）

- 处理粘性标签（sticky tag）
  - 产生粘性标签的目的：可以保留某个或某些文件的老版本而且在更新的时候不会受影响
  - 怎样会产生粘性标签：

```
# 某些时候我们想将某个文件恢复到老版本:
$ cvs up -p -r1.2 main.cpp > main.cpp   # 正确做法

# 但是如果不小心操作成:
$ cvs up -r1.2 main.cpp      # 产生了一个 sticky tag !

# 要去除 sticky tag，使用 update -A 选项
$ cvs up -A
```

CVS - 开始新的版本

- 开始新的版本
  - 注意：为整个项目开始新的版本时，要求给定的版本号比项目中文件最高版本更新，如项目中版本最高的文件是 main.cpp，其当前版本为 2.5，则新版本的版本号不可小于 2.5

```
# 为某个文件或整个项目开始新的版本:
$ cvs ci -r 2.8 main.cpp
$ cvs ci -r 2.8          # 当前目录或整个项目
```

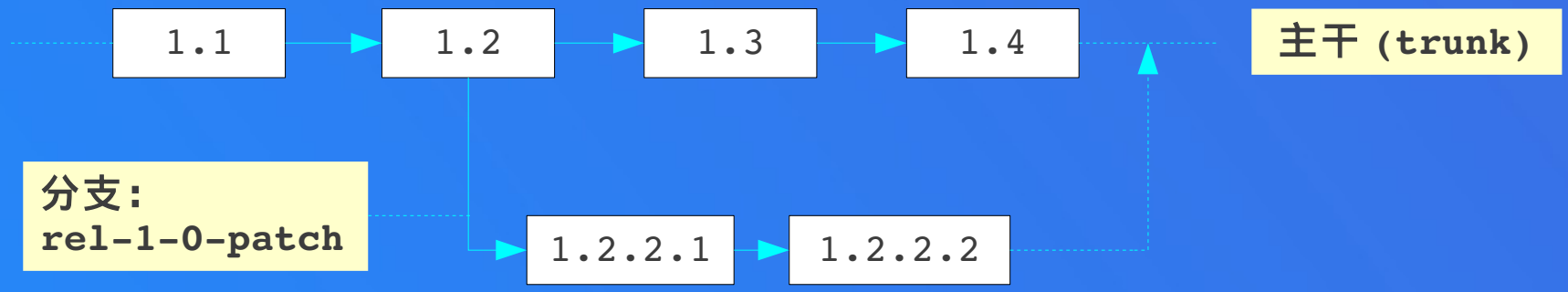
CVS - 建立分支

- 建立分支
  - 说明：某些情况下，在开始新版本后，还需回过头来维护已发行的老版本，这个时候就需要建立分支

```
# 使用 tag -b 从本地工作目录创建分支
$ cvs tag -b rel-1-0-patches

# 也可以使用 rtag -b 选项从仓库中已有的 tag（版本）创建分支
$ cvs rtag -b -r rel-1-0 rel-1-0-patches make_proj

# 访问分支方式 1：从仓库中取出，创建新的工作目录
$ cvs checkout -r rel-1-0-patches -d make_proj_p make_proj

# 访问分支方式 2：在现有的工作目录内更新（切换到分支版本）
$ cvs update -r rel-1-0-patches
```

CVS - 合并分支

- 合并分支
  - 示例：将下图所示的分支中的 1.2.2.2 版本合并到主干（以下图例代表单个文件的版本变迁）




```
$ pwd
/home/kwarph/Training/Module02/context/cvs/make_proj_p
$ cvs -q up -d -A    #1 -A 选项去掉粘帖 tag
RCS file: /projects/make_proj/src/main.cpp,v
retrieving revision 3.0
retrieving revision 3.1
Merging differences between 3.0 and 3.1 into main.cpp
main.cpp already contains the differences between 3.0 and 3.1
$ cvs -q update -d -j rel-1-0-patches    #2 -j 选项为合并
$ cvs -q ci -m 'merge from branch rel-1-0-patches'   #3
```

CVS - 小结

- CVS 最常用的命令：
  - import 、 update 、 checkout 、 commit 、 status 、 diff
- 使用 CVS 管理项目版本，需注意：
  - 由于 CVS 对目录的删除、重命名、文件或目录移位等操作没有十分有效的解决方案，所以在项目创建之处须计划好目录结构以及文件所属目录
  - 良好的提交习惯：
    - 在更改本地工作目录下的文件之前先与 CVS 仓库同步，降低冲突的频率
    - 一个重要原则：提交的项目代码最低要求是能够通过编译
- 一本优秀的书籍： Open Source Development with CVS

  http://cvsbook.red-bean.com/index.html