1)diff 的传统格式输出.

##

cat before.txt

输出:

This is a line to be deleted

This is a line that will be changed

This is a line that will be unchanged

cat after.txt

输出:

This is a line that has been changed

This is a line that will be unchanged

This is a line that has been added

##

diff before.txt after.txt

输 出:

1,2c1

< This is a line to be deleted

< This is a line that will be changed

> This is a line that has been changed

3a3

> This is a line that has been added

##

注释:

传统格式的输出

1,2c1 是指替换第 1 个文件的第 1,2 行到第 2 个文件的第 2 行,这里的 1,2 是指第 1 个文件的第 1,2 行,c 是替换的意思,最后的 1 是第 2 个文件的第 1 行

<号是指第 1 个文件更改或删除的行

---号是分割两个文件

>号是第 2 个文件中增加或删除的行

3a3 是指将第 2 个文件的第 3 行插入到第一个文件的第 3 行

也就是说第 1 个文件的:

< This is a line to be deleted

< This is a line that will be changed

被替换成第 2 个文件的:

> This is a line that has been changed

由于第 1 个文件的第 3 行和第 2 个文件的第 2 行一致,所以不做修改.

由于第 2 个文件的第 3 行是第 1 个文件所不具有的,所以在第 1 个文件的最后一行增加:

> This is a line that has been added

2)patch 命令的应用

用 diff 的传统格式输出:

##################################

diff before.txt after.txt >mypatch.txt

##################################

用 patch 修补 before.txt 文件,使 before.txt 和 after.txt 一致.

##################################

cat mypatch.txt |patch before.txt

输出:

patching file before.txt

##################################

比较两个文件，现在是一致的了.

##################################

cmp before.txt after.txt

##################################

用 patch 命令恢复 before.txt.

##################################

patch -R before.txt <mypatch.txt

输出:

patching file before.txt

##################################

注:-R 标记告诉 patch 在反向上应用区别或者撤销 patch.

再比较两个文件,现在不一致了.

##################################

cmp before.txt after.txt

输出:

before.txt after.txt differ: byte 17, line 1

##################################

3)diff 的统一格式输出.

##################################

diff -u before.txt after.txt |tee mypatch.diff

输出:

--- before.txt 2009-06-20 05:21:49.000000000 +0800

+++ after.txt   2009-06-20 04:03:16.000000000 +0800

@@ -1,3 +1,3 @@

-This is a line to be deleted

-This is a line that will be changed

+This is a line that has been changed

This is a line that will be unchanged

+This is a line that has been added

##################################

注释:

diff -u 选项是统一格式输出.

--- before.txt 2009-06-20 05:21:49.000000000 +0800

--- before.txt 是指旧文件

+++ after.txt 2009-06-20 04:03:16.000000000 +0800

+++ after.txt 是指新文件.

@@ -1,3 +1,3 @@

@@ -1,3 是指第 1 个文件一共有 3 行,+1,3 @@是指第 2 个文件一共有 3 行.

-This is a line to be deleted

-This is a line that will be changed

是被删除的行

+This is a line that has been changed

是增加的行

This is a line that will be unchanged

没有-号和+号是指该行不变,因为 after.txt 和 before.txt 都有这行.

+This is a line that has been added

是增加的行

diff 的统一格式比较与输出是按顺序进行的.

4)diff 命令在目录中的应用.

新建 old 和 new 目录,old 目录包含了初始内容,new 目录包含文件的最新版本.

##

mkdir old new

echo "This is one. It's unchanged." | tee old/one new/one

echo "This is two. It will change." > old/two

echo "This is two. It changed.">new/two

echo "This is three. It's new" > new/three

##

创建修补文件

##

diff -Nur old/ new/ >mypatch.diff

##

注:-r 选项按照文件目录递归创建修补文件.

-u 还是统一模式

-N 是指当 diff 遇到一个只存在于两个树中的一个树中的文件时,默认情况下跳过文件并且打印一个警告到 stderr.

这个行为可以通过-N 选项来更改,这也导致了 diff 认为丢失的文件实际上是存在的,但它是空的.采用这种方式,一个修补文件可以包括已经创建的文件.然后应用修补程序创建新的文件.

##

more mypatch.diff

输出:

diff -Nur old/three new/three

--- old/three 1970-01-01 08:00:00.000000000 +0800

+++ new/three   2009-06-20 06:55:34.000000000 +0800
@@ -0,0 +1 @@
+This is three. It's new
diff -Nur old/two new/two
--- old/two   2009-06-20 06:55:08.000000000 +0800
+++ new/two   2009-06-20 06:55:21.000000000 +0800
@@ -1 +1 @@
-This is two. It will change.
+This is two. It changed.
##
注释:
diff -Nur old/three new/three 是指下面比较 old/three new/three 两个文件.
因为没有 old/three 文件,所以在 old/three 中增加+This is three. It's new
diff -Nur old/two new/two 是指下面比较 old/two new/two 两个文件
因为 old/two 与 new/two 的第 3 行不一致,所以删除 This is two. It will change.增加 This is two. It changed.
打补丁到 old 目录,新建 old/three 以及更改 old/two
##
patch --dir old< mypatch.diff
ls -l   old/
输出:
one   three two
##
恢复 old 目录的内容,包括删除 old/three,以及恢复 old/two 文件
##
patch --dir old -R <mypatch.diff
输出:
ls -l old/
one two
##
5)检查和合并更改
用 vim 突出显示单个字符的更改来表示区别.
##
vim -d after.txt before.txt
##
用 gui 工具 gvimdiff 来显示两个文件.
##
gvimdiff after.txt before.txt
##
新建文件 orig.c
##
vi orig.c

```
void foo(void)
{
printf("This will be changed by me. \n");
printf("This will be unchanged,\n");
printf("This will be changed by you.\n");
}
```

##

复制文件 orig.c 到 me.c,更改第 4 行为 printf("This was changed by me. \n");

##

```
vi me.c
void foo(void)
{
printf("This was changed by me. \n");
printf("This will be unchanged,\n");
printf("This will be changed by you.\n");
}
```

##

复制文件 orig.c 到 you.c,更改第 7 行为 printf("This was changed by you.\n");

##

```
vi you.c
void foo(void)
{
printf("This will be changed by me. \n");
printf("This will be unchanged,\n");
printf("This was changed by you.\n");
}
```

##

版本工具如 cvs,subversion 使用 GNU 合并工具称为 diff3.

##

```
diff3 me.c orig.c you.c
```

输出:

```
====1
1:3c
printf("This was changed by me. \n");
2:3c
3:3c
printf("This will be changed by me. \n");
====3
1:7c
2:7c
printf("This will be changed by you.\n");
```

3:7c
printf("This was changed by you.\n");
注:
在没有参数的情况下,diff3 产生的输出说明了那行更改.
====1 和====3 指明造成同原始文件不同的是哪一个修改文件.
编号方式基于参数序列.
也就是第 1 个文件和第 3 个文件与原文件不同.
1:3c
printf("This was changed by me. \n");
3:3c
printf("This will be changed by me. \n");
1:3c 表示第 1 个文件的第 3 行与 3:3c 表示的第 3 个文件的第 3 行不同.
为什么不显示与原文件的比较呢。因为第 3 个文件的第 3 行与源文件(第 2 个文件)相同.所以与哪个文件比较无所谓了.
2:7c
printf("This will be changed by you.\n");
3:7c
printf("This was changed by you.\n");
2:7c 表示第 2 个文件的第 7 行与 3:7c 表示的第 3 个文件的第 7 行不同.
diff3 会试图为我们进行合并.合并是在源文件的基础上,依据两个新文件进行修改
源文件是第二个文件,第一个文件和第三个文件可以互换，但他们必须有共同的祖先，就是第二个文件.
##
diff3 -m me.c orig.c you.c |cat -n
输出:
```
1    void foo(void)
2    {
3        printf("This was changed by me. \n");
4
5        printf("This will be unchanged,\n");
6
7        printf("This was changed by you.\n");
8    }
```
##
为了测试更复杂的环境,新建一个文件 orig.c.1
内容如下:
##
vi orig.c.1
```
void foo(void)
{
printf("This will be changed by both of us.\n");
}
```

```
##########################################
用 diff3 -m 再次比较输出,如下:
##########################################
diff3 -m me.c orig.c.1 you.c
void foo(void)
{
<<<<<<< me.c
printf("This was changed by me. \n");
printf("This will be unchanged,\n");
printf("This will be changed by you.\n");
||||||| orig.c.1
printf("This will be changed by both of us.\n");
=======
printf("This will be changed by me. \n");
printf("This will be unchanged,\n");
printf("This was changed by you.\n");
>>>>>>> you.c
}
##########################################
注释:以上的格式,同 cvs update，需要人工合并文件的格式是一致的.
```