File Version Numbers [B]

In the course of normal system administration, occasions exist when it is important to make a new version of a file, but it is also important to keep an old file. A sequence of operations often used to replace a real production file with a new one and keep an old version of a file is as follows:

```
cp file.REAL file.NEW
edit file.NEW
cp file REAL file.OLD
mv file.NEW file.REAL
```

The preceding sequence can lead to problems; if a small error was made in the generation of the new file and a subsequent version is made, reusing the sequence will cause the loss of the previous real file. Because this is a well-known problem, you should use one of the following two sequences. To correct the error, use the following command lines:

```
cp file.REAL file.NEW
edit file.NEW
cp file.NEW file.REAL
```

To start again but to keep a copy of the broken new file, use the following command lines:

```
cp file.OLD file.NEW
cp file.REAL file.OLD2
edit file.REAL
```

A better strategy is to dispense with the `.OLD` file naming convention and use the following sequences. The first time you want to alter a file, use the following sequence:

```
cp file.REAL file.000
cp file.000 file.001
edit file.001
```

Each time you are ready to live with the latest version of a file, copy the highest number file to *file* `.REAL`.

This method of file version numbering has three main advantages over the `.OLD` file naming scheme:

- You can quickly see a version history.

- You can make as many versions of the file as you like without losing the real file.
- You have a back-up copy of the real file in case it gets damaged in production.

Each time you make a new version, you can add a comment in the file's history file, as follows:

```
echo "file.00x version comment" >> file.HISTORY
```