The Answer Girl
TEAMWORK

PATRICIA JUNG

In this issue, Answer Girl gives an introduction to version control with CVS (Concurrent Versions System). After reading this, when you see the phrase check-in you will no longer think about air travel.

The fact that the world of everyday computing, even under Linux, is often good for surprises, is a bit of a truism: Time and again things don't work, or not as they are supposed to. Answer Girl shows you how to deal elegantly with such little problems.

When you work on a source text for a long time, sooner or later the day will dawn when you wish you could get back that section you deleted last Monday. But deleted is deleted, and who doesn't make up their mind to do better next time?

Version control

Anyone who works alone and has a certain amount of staying power, may start off copying the latest version (for example *classscript-2.tex*) at the start of a session of work in a new file with a serial number (*classscript-3.tex*). Which version the one from last Monday was, you can find out from the date details, which states *ls -l classscript** as last amendment date.

So then why not just include the date stamp in the filename? With *date* a UNIX system supplies the latest date and time details to your doorstep, after all, and with the backwards-pointing inverted commas, you can induce the shell, first to execute the command contained therein and then to use the result in the complete command:

```
[trish@lunar answergirl]$ cp classscript.tex.
classscript`date`.tex
cp: copying multiple files, but last argum
ent (2001.tex) is not a directory
Try `cp –help' for more information.
```

cp is complaining that we want to copy more than one file and the last argument *2001.tex* is not a directory (because several source files cannot be copied into a single normal file). *2001.tex*? That looks like part of the *date* statement:

```
[trish@lunar answergirl]$ date
Mon Oct 15 02:24:09 CET 2001
```

Now the scales fall from our eyes. The spaces count, for the shell, obviously as separators between arguments, so in *cp* as filenames. With double inverted commas, though, the *bash* can be persuaded that the spaces are part of the argument string:

```
[trish@lunar answergirl]$ cp classscript.
tex "classscript`date`.tex"
[trish@lunar answergirl]$ ls -l classsc
ript*
-rw-r–r–  1 trish  users  8967 Oc
t 15 02:25 classscript.tex
-rw-r–r–  1 trish  users  8967 Oct 15 02:34
classscriptMon Oct 15 02:34:04

CET 2001.tex
```

The drawback here: *classscriptMon Oct 15 02:34:04 CET 2001.tex* not only looks ugly, but because of the spaces in the filename the file will force us more than once to place its name in some command line or other in inverted commas. So we would prefer a filename à la *classscript_dd_mm_yy.tex* or, in order that the files always appear nicely in the sequence followed in the calendar in the *ls* output-*classscript_yy_mm_dd.tex*.

As a glance at the *date* manpage shows, this works too. We just have to send *date* on its way with the desired format placeholder following a plus:

```
[trish@lunar answergirl]$ mv "classscript
Mon Oct 15 02:34:04 CET
2001.tex" classscript_`date +%y_%m_%d`.tex

-rw-r–r–  1 trish  users  8967 Oct 15 02:
25 classscript.tex
-rw-r–r–  1 trish  users  8967 Oct 15 02:
34 classscript_01_03_12.tex
```

... and for the less conscientious

So you already feel ill at ease with all this compulsory thinking? You're not the only one. And when it comes to working with co-author(s) on a manuscript, it's not only the discipline that gets difficult but also corrections. Who can guarantee me, after all, that my co-writer will not quietly remove the typing errors put in yesterday from her version, while I am completely transposing the sentences in precisely this chapter in my own version?

Not just to simplify the work routine, but also to avoid extra work, there is only one answer: there has to be professional version management. If you are working with office packages or suchlike, which store your work in proprietary binary formats (such as StarWriter), you will presumably fall back on the built-in version control function. **ASCII texts** on the other hand can be managed effortlessly with version managers, as used in large programming projects. The source code in a programming language is after all nothing but text.

A search produces astonishingly little choice. While companies would still rather license the commercial *Perforce* (*http://www.perforce.com/*), Open Source projects can ask for a free licence; the fully-functioning evaluation version allows only for two-person projects, it is not only occasional version controllers who tend most to fall back on the tried and trusted *C*oncurrent *V*ersions *S*ystem *cvs*.

This also comes with most distributions. Those unable to find a suitable package would be best heading for *http://rpmfind.net/linux/rpm2html/search.php?query=cvs* or for the source code *http://download.cyclic.com/pub/*.

Added on

So first to the *i*nstallation- in *rpm*-based systems such as with a

```
[root@lunar software]# rpm -i cvs-1.10.7
7-1.i386.rpm
```

There it is then, the great unknown. A timid

```
[trish@lunar answergirl]$ cvs –-help
```

does not exactly warm you up with its lovely muddle. But a closer look at the chaos does then help:

```
Usage: cvs [cvs-options] command [com7
mand-options-and-arguments]
[...]
```

In order to use *cvs*, then, we must at least state *cvs* and then specify a CVS command afterwards. In addition, the behaviour of *cvs* can be altered by means of *cvs-options*, which must be specified before the CVS command. To increase the complexity even more, each CVS command can also be followed by its own options and arguments.

After a nice deep breath, one of the next lines also decrypts itself:

```
[...]
  (specify – –help-commands for a li7
st of commands
[...]
```

In fact, *cvs —help-commands* outputs a whole range of commands and under one or other we can even get something of an idea:

```
[...]
     init        Create a CVS reposito7
ry if it doesn't exist
[...]
```

init, that sounds like *init*ialise, and to anyone who has ever come across the term *CVS repository* before in some open source project or other, this looks like just what we want: create a CVS depot, in which we can put, or – *check in* – our files.

```
[trish@lunar answergirl]$ cvs init
cvs init: No CVSROOT specified!  Plea7
se use the `-d' option
cvs [init aborted]: or set the CVSRO7
OT environment variable.
```

If only it were that simple ... Luckily, the *cvs* manpage explains the ominous option *-d* ("*d*irectory") to us:

```
CVS OPTIONS
[...]
  -d CVS_root_directory
     Use CVS_root_directory as the root
     directory pathname of the master
     source repository. Overrides the
     setting of the CVSROOT environment
     variable. This value should be
     specified as an absolute pathname.
```

So we are dealing with an option for the *cvs* command (unlike an option, which relates to a CVS command), which will have as argument the directory for our *D*epot. What matters here is: We must specify it with **full path**, for example *~/cvs/linuxcourse/coursedocuments*.

```
[trish@lunar answergirl]$ cvs -d ~/c7
vs/linuxcourse/coursedocuments init
cvs [init aborted]: cannot make directory7
/home/trish/cvs/linuxcourse/coursedocuments:7
No such file or directory
```

All right, now we'll just have to create the directory *~/cvs/linuxcourse* together with a *p*arent directory *~/cvs/* and try it again:

ASCII texts: Texts, whose characters are saved in the "American Standard Code for Information Interchange". This code in the 7-bit version encompasses only the characters which are found on an American keyboard and a few control codes such as *CR* (Carriage Return) for Enter (originally from a typewriter) or LF ("Line Feed") for a line break. In 8-bit ASCII, most special characters from languages with Latin alphabets can be coded. But anyone wanting to write with Cyrillic or Hebrew characters will have to use other codings such as UTF8. Most of the common text editors in this country use ASCII.

Full path: The route to a file starting from the root directory */*. So the full path leads to the program file */usr/sbin/groupadd* via the directories */—>usr—>sbin* and is therefore written */usr/sbin*. *Relative paths* on the other hand always start from the current working directory. If the full path to a program is not listed in the environment variable *PATH*, it is not enough to call up the name of the command. If the shell should acknowledge the command calls for *groupadd*, *usermod* and *useradd* with a *command not found* in the *co-combatants* box, the command with full path (*/usr/sbin/groupadd* etc.) will hopefully provide a remedy. If not, the question arises as to whether these commands are even installed.

SecureShell: Safe substitute for Internet services such as Telnet and RSH ("Remote Shell"), with the aid of which one can work on remote computers as if sitting right in front of them. Data transmission is encrypted when you do so. The SecureShell package usually comes with a secure substitute for RCP (Remote Copy) named scp.

Tunnelling: Using a service via a connection which another service makes. So for example CVS packages can be transmitted repackaged into SSH packages.

```
[trish@lunar answergirl]$ mkdir -p ~/cvs➥
/linuxcourse
[trish@lunar answergirl]$ cvs -d ~/cvs➥
/linuxcourse/coursedocuments init
```

No response this time, but in the best UNIX tradition that should actually mean that everything has gone smoothly. And so it has, *ls ~/cvs/linuxcourse/coursedocuments* shows that this directory has been created and also contains another subdirectory full of very oddly named files, called CVSROOT.

Now the started course documents just have to go in there and there was something else – that's right, I don't want to do the work involving writing scripts and slides all on my own, so my co-author must also have access to the depot. So as not to bore those who want to use their repository alone, the work steps necessary for this are shown separately in the *Co-authors* box.

Initial database

The CVS command overview lists, with

```
[...]
      import      Import sources into C➥
VS, using vendor branches
[...]
```

a command, with which it appears to be possible to *import* our initial database, which is currently in the working directory *~/course*, two directories *script* and *slides* each with a ***tex* file** and an illustration –o our depot:

```
[trish@lunar answergirl]$ cd ~/course
[trish@lunar course]$ cvs import
cvs import: No CVSROOT specified!  Plea➥
se use the `-d' option
cvs [import aborted]: or set the CVSRO➥
OT environment variable.
```

Obviously, if the depot is not specified with *-d*, *cvs* cannot even know where the data from the current directory should be imported to. But since we have no inclination to keep typing in the endless *-d ~/cvs/linuxcourse/coursedocuments*, we take to heart the last line in the error message and set the environment variable *CVSROOT*:

Co-authors

Anyone who creates a depot in their own home directory doesn't want the co-author/s to be able to poke around in all the files in ~ (*root* could also pack it away somewhere else, for example into */home/cvs*).

Classification

So the best thing to do is seize *root* and make a new group *course* with the Group number *101* not yet assigned in */etc/group*:

```
[trish@lunar answergirl]$ su
Password: root-Password
[root@lunar answergirl]# groupadd -g 101 course
```

The maintainer of the CVS depot made should of course be included by *root* in the new group. This is done by manually editing */etc/group* and obviously also with graphic user management tools. But before we start it as *root*, we'd be faster with a

```
[root@lunar answergirl]# usermod -G course trish
```

The capital *-G* here means: "Add another group to the other Groups of which the user is a member". A

```
[root@lunar answergirl]# groups trish
trish : users course
```

reveals that *trish* now, apart from belonging to the group *users*, is also a member of *course*.

New user

```
[root@lunar answergirl]# useradd fred
```

then makes an *Account* for the user *fred*. If he is only to use this for CVS purposes, this was a little premature, since then he should belong exclusively to the group *course*.

```
[root@lunar answergirl]# usermod -g course fred
```

The small *-g* swaps the *primary* Group for *fred* comes to the rescue, instead of adding an extra group.

Then *fred* gets another new user password ...

```
[root@lunar answergirl]# passwd fred
New user password: password_for_fred
Retype new user password: password_for_fred
passwd: all authentication tokens updated successfully
```

..., the home directory pre-defined by *useradd* is made ...

```
[root@lunar answergirl]# mkdir ~fred
```

... and handed over (with the *ch*ange-*own*er command *fred* becomes the owner of his home, and *course* as primary group inherits the group rights thereto):

```
[root@lunar answergirl]# chown fred:course ~fred
```

This means that *root* can now log out with *exit*. *trish* as CVS maintainer still has however, one more task to do: The CVS group *course* must be given read, write and in the case of directories, execution (and/or directory change) rights to the depot directory. When a check of the group rights shows that the rights are correct, but the name of the group is wrong, the command *chgrp* will help to correct this.

```
[trish@lunar course]$ export CVSROOT=~/cvs/linuxcourse/coursedocuments
```

So on to something new:

```
[trish@lunar course]$ cvs import
Usage: cvs import [-d] [-k subst] [-I ign] [-m msg] [-b branch]
  [-W spec] repository vendor-tag release-tags...
      -d     Use the file's modification time
             as the time of import.
      -k sub  Set default RCS keyword
             substitution mode.
      -I ign  More files to ignore (! to reset).
      -b bra  Vendor branch id.
      -m msg  Log message.
      -W spec Wrappers specification line.
(Specify the --help global option for a list of other help options)
```

So it's not that simple or perhaps with so little interaction *cvs* insists that we tell it explicitly what to call the depot ("repository"). The two additionally required arguments *vendor-tag* and *release-tags* are fortunately not necessary for simple version management, so that here we can enter any old thing.

The vendor tag, which is a sort of identification for the publisher of the data to be imported, then applies when one wishes to control amendments to sources from third parties by CVS, which are not themselves to flow back to the publisher (because the latter does not feel the amendments are important, correct or general enough). If a new original version comes out, the release tag makes it possible to distinguish between the versions.

Whatever the case may be, a

```
[trish@lunar course]$ cvs import linuxcourse trish v2001
```

alerts us, by calling up the *vi* editor (or the write program stored in the environment variables *VISUAL* or *EDITOR*), to the fact that it would like us to give it another short description of the data:

```
CVS: ----------------------
```

tex-file: Text file consists of content and TeX- or LaTeX commands, which typify the structure of the content. Using the text batch commands *tex* or *latex* this becomes the actual portrayal of the ready-to-print file.

Ownership

Since files can only be given to groups to which one belongs, *trish* has to log in again, only then does

```
[trish@lunar answergirl]$ chgrp -R course ~/cvs/linuxcourse/coursedocuments/
```

stop issuing error messages, because with the login process the membership data is refreshed. The *-R* in *chgrp* (just like *chown* and *chmod*) ensures that the owner's details for *~/cvs/linuxcourse/coursedocuments/* and all files/directories underneath are altered *r*ecursively in a round-up.

Where there are several CVS user(s) with different primary groups it's worth bearing in mind another problem: Since *trish* belongs to the depot directory, she can also check in data with her primary group *users*. But this would no longer be accessible for *course*-only members such as *fred*.

This problem can be solved by providing the depot directory with the *s*-right ("*s*et group ID on execution") for the group:

```
[trish@lunar answergirl]$ chmod g+s ~/cvs/linuxcourse/coursedocuments/
[trish@lunar answergirl]$ ls -al ~/cvs/linuxcourse/coursedocuments/
total 3
drwxrwsr-x   3 trish   course    1024 Oct 7 00:31 .
drwxr-xr-x   3 trish   users     1024 Oct 7 00:31 ..
drwxrwxr-x   2 trish   course    1024 Oct 7 00:31 CVSROOT
```

This ensures that all data written into the depot directory is part of the group *course*, even if *trish* checks in with a different primary group.

Distant relation

Now *fred* can log in to *lunar*, check out and edit the data in his home directory there, but it is fairly unlikely that he will want to be online the whole time he is working with the documents. The whole point of a revision control system is precisely that of co-ordinating the work of people who are working on several different computers.

Since *fred* now has a shell account on the CVS server *lunar*, he can now easily **tunnel** his CVS queries via **SecureShell**. To do so, on his Internet computer, he sets the variable *CVS_RSH* (*CVS R*emote *Sh*ell) on the command *ssh* (if necessary, also specifying **path details**):

```
[fred@fredsbox ~]$ export CVS_RSH=ssh
```

Naturally, he must also set his *CVSROOT* variable on the depot directory */home/trish/cvs/linuxcourse/coursedocuments*. Since this is on the remote computer *lunar.answergirl.co.uk*, this becomes somewhat more complicated: Using the keyword *ext* he specifies that the depot can be found on an *ext*ernal machine, then follows the address of the depot computer with the username first, and finally comes the destination directory.

To make it quite clear where each component ends, they are separated from each other by colons. To allow *cvs* to proceed in the certainty that the whole monstrosity is not just a somewhat odd directory name, there must also be an initial colon:

```
[fred@fredsbox ~]$ export CVSROOT=:ext:fred@lists.answergirl.co.uk:/home/trish/cvs/linuxcourse/coursedocuments
```

After that, *fred* can go online and check out the depot *linuxcourse* made in the main text with *cvs co linuxcourse*.

No more variables to set, ever again

Setting the variable *CVSROOT* and in the case of remote access to the CVS server *CVS_RSH* is not necessarily something one wants to do again by hand in every new shell. Those wishing to access just one CVS depot have it easy: They enter the *export* lines in the *~/.bashrc* read out when starting each *bash* or if applicable also in the initialisation file for **Login shells**, *~/.bash_profile* (at least when they are actually working with the Bash).

It gets more complicated when you are dealing with several depots. Then it is advisable to write the *export* lines into an otherwise empty text file and to read these in before the first access to a depot of a specified shell with the command

```
source file_with_CVS-variable
```

to *sources*.

Login shell: The command line interpreter seen after logging onto a virtual console, in most cases under Linux the *B*ourne *A*gain *Sh*ell *bash*. The fact that a Bash becomes a login shell is defined with the option *-login*, so that in X-terminal programs under X11 one can also get landed with login shells. If an *echo $variablename* should show that a variable set in *~/.bashrc* does not appear in the current shell, then one is usually dealing with a login shell, because in the Bash it doesn't care about the *~/.bashrc*, but about *~/.bash_profile*. If you have no success there either with the variables set there (as long as this is not due to the fact that errors slipped in during the setting), this has presumably been blocked by the Bash parameter *-noprofile*. The only remedy for this is to delve even deeper into the system or the previously mentioned *sources* of the variables shortly before use.

```
CVS: Enter Log.  Lines beginning with `CV
S:' are removed automatically
CVS:
CVS: ----------------------
```

An *o* brings us in the case of *vi* to a new line and into write mode, so that we can enter the text. By pressing the *Escape* key we get into the command mode of *vi* and save and end our entry with the sequence *:wq*. *cvs* now acknowledges the import with

```
cvs import: Importing /home/trish/cvs/linu
xcourse/coursedocuments/linuxcourse/script
N linuxcourse/script/unixcourse.tex
N linuxcourse/script/tree.eps
cvs import: Importing /home/trish/cvs/linu
xcourse/coursedocuments/linuxcourse/slides
N linuxcourse/slides/unixslide.tex
N linuxcourse/slides/tree.eps
```

No conflicts created by this import

This means the depot is now filled with data and the data directory *~/course* together with subdirectories have each been enriched by a directory named *CVS*.

In and out

It is now really easy to work with the data which has been checked in. At the start of a work session a

```
[trish@lunar course]$ cvs update
cvs update: Updating .
cvs update: Updating slides
cvs update: Updating script
```

brings the data in the data directory up to date; now if you want to update only a certain subdirectory, simply change to it before the command.

If you have reached the end of a work unit, check in the amendments to files in the respective subdirectories with (check *i*n).

```
[trish@lunar course]$ cvs ci
cvs commit: Examining .
cvs commit: Examining slides
cvs commit: Examining script
```

Here again you will need *vi* knowledge to describe the amendment. And of course, individual files can be "committed":

```
[trish@lunar course]$ cvs ci slides/uni
xslide.tex

CVS: ----------------------
CVS: Enter Log.  Lines beginning with `C
VS:' are removed automatically
CVS:
CVS: Committing in slides
CVS:
CVS: Modified Files:
CVS:   unixslide.tex
CVS: ----------------------
```

Qualms? If you quickly want to back out now, simply stop the editor without making any amendments. *:q!* in the command mode of *vi* will then make *cvs* start whinging:

```
Log message unchanged or not specified
a)bort, c)ontinue, e)dit, !)reuse this me
ssage unchanged for remaining dirs
Action: (continue)
```

An *a* confirms that we are serious about stopping unlike a simple *Enter*, which checks in nevertheless, and *e*, which brings us back into the editor.

Fresh Data

New ideas in new subdirectories – sometimes in retrospect it turns out to be quite helpful to give the initial passion for work some structure ...

```
[trish@lunar course]$ mkdir concept
[trish@lunar course]$ cd concept
```

If the concept, *concept.tex* is in this subdirectory, it should also be checked in – only how, if it's not yet in the depot?

```
[trish@lunar concept]$ cvs add concept.tex
```

```
cvs add: cannot open CVS/Entries for re➥
ading: No such file or directory
cvs [add aborted]: no repository
```

... so that was not quite the right idea: Of course, *cvs* is uncertain what to do with the file, since the subdirectory *concept* does not yet contain a *CVS* directory. So it's back one command or one directory ...

```
[trish@lunar concept]$ cd ..
```

... and one after the other:

```
[trish@lunar course]$ cvs add concept
? concept/concept.tex
Directory /home/trish/information/clas➥
sscript/concept added to the repository
[trish@lunar course]$ cd concept
[trish@lunar concept]$ cvs add concept.tex
cvs server: scheduling file `concept.te➥
x' for addition
cvs server: use 'cvs commit' to add th➥
is file permanently
```

That appears to have worked, except that the file is obviously not inside yet. So we follow the instruction to *commit* the new access too. Whether to do this you use the long *cvs* command *commit* or the short *ci*, the result is the same:

```
[trish@lunar concept]$ cvs commit
cvs commit: Examining .
CVS: ----------------------
CVS: Enter Log.  Lines beginning with `C➥
VS:' are removed automatically
CVS:
CVS: Committing in .
CVS:
CVS: Added Files:
CVS:    concept.tex
CVS: ----------------------
oNew concept
ESC:wq
RCS file:
/home/trish/information/classscript/conc➥
ept/concept.tex,v
done
Checking in concept.tex;
/home/trish/information/classscript/conc➥
ept/concept.tex,v  <--  concept.tex
initial revision: 1.1
done
```

Reconstruction

concept.tex changed back and forth, once the structure was in place, the file kept changing until it became the present course script. After some time and a few work phases the names at the front and back no longer matched each other anyway, while the previously checked-in *unixcourse.tex* has been edited in the *script* directory by nobody. In short: it is now time to make a cut and update the system, to overwrite *unixcourse.tex* with the content of *concept.tex* and to take *concept.tex* out of the repository. This is done with:-

Pre-programmed conflicts?

If several people are working on one document there is not a CVS server in the world which can prevent amendments which one person checks in colliding with amendments which the other wants to commit somewhat later:

```
[trish@lunar linuxcourse]$ cvs commit
cvs commit: Examining .
cvs commit: Examining slides
cvs commit: Examining script
cvs commit: Up-to-date check failed for `script/unixcourse.tex'
cvs [commit aborted]: correct above errors first!
```

Obviously here someone has also tinkered with *script/unixcourse.tex*, and the depot contains a version which is newer than *trish*'s working version. This calls for an update:

```
[trish@lunar linuxcourse]$ cvs update
cvs update: Updating .
cvs update: Updating slides
cvs update: Updating script
RCS file:➥
/home/trish/cvs/linuxcourse/coursedocuments/linuxcourse/script/➥
unixcourse.tex,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into unixcourse.tex
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in script/unixcourse.tex
C script/unixcourse.tex
```

cvs tries to get the amendment made in the meantime and *trish*'s new amendments under one roof (to *merge*). If this works successfully, there is nothing else to worry about, but if however as here it goes awry, *trish* must set to work in person and load the conflicting file *unixcourse.tex* into the editor again. This file has now been amended in the meantime by CVS so that the conflict is visible and easily found for manual editing:

```
<<<<<<< unixcourse.tex
      Summer course Information Oxford Uni
=======
      Summer 2001 Oxford University
>>>> 1.3
```

Above is her own version, and below the current depot version. All that is left to do with this is to remove the <-, =- and > lines and to merge the contradictory lines into exactly the text which is now to be checked in, e.g.

```
      Summer course Information Oxford Uni
```

Then all that remains is to check it in.

Attic: In directories with this name CVS stores the content and the history of files deleted from the depot.

PGP: "Pretty Good Privacy", which is certainly the commonest program for encryption and signing of emails and other data.

```
[trish@lunar course]$ mv concept/concep➥
t.tex script/unixcourse.tex
[trish@lunar course]$ cvs remove conce➥
pt/concept.tex
cvs remove: scheduling `concept/conce➥
pt.tex' for removal
cvs remove: use 'cvs commit' to remove th➥
is file permanently
[trish@lunar course]$ cvs ci
```

There are now two challenges during check-in: The new content of *unixcourse.tex* wants to be commented (for example with *concept.tex now unixslide.tex*), and in the documentation of the removal of *concept.tex*, *cvs* kindly specifies the same comment:

```
concept.tex now unixslide.tex
CVS: ----------------------------------------
CVS: Enter Log.  Lines beginning with `CV➥
S:' are removed automatically
CVS:
CVS: Committing in concept
CVS:
CVS: Removed Files:
CVS:   concept.tex
CVS: ----------------------------------------
```

Once saved, we want witnesses to the amendments in the depot:

```
cvs commit: Examining .
cvs commit: Examining slides
cvs commit: Examining concept
cvs commit: Examining script
Checking in slides/unixslide.tex;
/home/trish/cvs/linuxcourse/coursedocume➥
nts/linuxcourse/slides/unixslide.te➥
x,v  <--  unixslide.tex
new revision: 1.3; previous revision: 1.2
```

No Shell Access?

An account on a computer does not mean that it should also be used for all colleagues and friends to poke around in the system, for the use of Internet services or for filing data. On the contrary, there are lots of good reason to limit access to the CVS, without giving up the security offered by the SecureShell.

All the administrator of the CVS server needs to do is to put the public SSH-key of the CVS user onto their workstation. In a similar way to **PGP** these create a key pair on their work account, of which the *public key* can be given out as you like, but the *private key* must be kept a secret. This is done with the command which comes with the SSH packages *ssh-keygen* and was discussed at length in the Answer-Girl in issue 9.

ssh-keygen places the secret private key in the pre-set under *~/.ssh/identity*. This must not be passed on! What the CVS sysadmin would like is a text file named *~/.ssh/identity.pub* and looks something like this:

```
1024 35 165043668525388007503675301831634125912119991502526700029105958161542269846546772572229108798152992529758074045707003573273020044380873112356724249904219939995856241718046388628225862962791292865950083481899332539835181290112611354730215142417376960062146599043065554089684980963002106747241282736545822186999 fred@fredsbox.fred.co.uk
```

This one long line is transferred by the sysadmin into the file *.ssh/authorized_keys* in the home directory of the user. *fred*'s public key thus ends up in *~fred/.ssh/authorized_keys* on the CVS server.

The server admin has in the meantime poked around in the manpage on SecureShell servers *sshd* and, under the heading of *AUTHORIZED_KEYS FILE FORMAT* has stumbled across the fact that at the start of a key line, it is possible to specify a command, which instead of a login shell is executed whenever the corresponding user tries to log in with the appropriate private key via *ssh*.

So when *fred* comes via *ssh* with his *cvs* request, the CVS server should simply start and grant access only to the depot in *~trish/cvs/linuxcourse/coursedocuments*. An appropriate *CVS* server is started with *cvs server —allow-root=/home/trish/cvs/linuxcourse/coursedocuments*, whereby the sysadmin only has to place a *command="cvs server —allow-root=/home/trish/cvs/linuxcourse/coursedocuments"* at the start of the key line of *fred*'s public key:

```
command="cvs server –allow-root=/home/trish/cvs/linuxcourse/coursedocuments" 1024 35 16504366852538800750367530183163412591211999150252670002910595816154226984654677257222910879815299252975807404570700357327302004438087311235672424990421993999585624171804638862822586296279129286595008348189933253983518129011261135473021514241737696006214659904306555408968498096300210674724128273654582218699 9 fred@fredsbox.fred.co.uk
```

What matters here is that the whole rat's nest must be kept as a single line.

```
done
Removing concept/concept.tex;
/home/trish/cvs/linuxcourse/coursedocu➥
ments/linuxcourse/concept/concept.te➥
x,v  <--  concept.tex
new revision: delete; previous revisio➥
n: 1.1.1.1
done
```

Now all that needs to be done is to delete the now-empty *concept* directory, and this is done for us by a

```
[trish@lunar course]$ cvs update -P
```

("*p*urge" - "cleanse").

Nothing is forever

So the concept no longer exists. But here comes the unexpected question: "Have you perhaps got a concept for me?" We haven't any more, but *cvs* has. All we need to do is remember at what point in time the concept was a concept. Fortunately, CVS keeps painstaking books. And one can inspect these, using *cvs log*. The best way to do this is if we don't let everything rush by us, but by sending *less*:

```
[trish@lunar course]$ cvs log | less
```

With the *less* search command */concept* we find therein the comment on the renaming:

```
RCS file:
/home/trish/cvs/linuxcourse/coursedocume➥
nts/linuxcourse/slides/unixslide.tex,v
[...]
----------------------------
revision 1.3
date: 2001/10/15 21:57:53;  author: tri➥
sh;  state: Exp;  lines: +1 -1
concept.tex now unixslide.tex
----------------------------
```

Unfortunately, we are given no information as to the former existence of the file *concept.tex* itself: Our fault – had we not deleted the empty *concept* directory with its information in the *CVS* subdirectory with the *-P* flag of *update*, we would not be so helpless now.

In the worst case, you will now have to rifle through the file tree of the CVS depot with *less* and *ls*, until the appropriate ***Attic*** file to *concept.tex* (*~/cvs/linuxcourse/coursedocuments/linuxcourse/concept/Attic/concept.tex,v*) is found. In any case, we establish that *concept.tex* must still have been in existence on 15.10.2001 at about 21:55.

So if we adapt our database to that which was current at that time:

```
[trish@lunar course]$ cvs update -D "200➥
1-10-15 21:55:53"
cvs update: Updating .
cvs update: Updating slides
U slides/unixslide.tex
cvs update: Updating script
U script/unixcourse.tex
```

Unfortunately, *concept.tex* is not there.

```
[trish@lunar course]$ cvs update – –help
update: invalid option – -  –
Usage: cvs update [-APdflRp] [-k kopt] [-➥
r rev|-D date] [-j rev]
 [-I ign] [-W spec] [files...]
   -A     Reset any sticky tags/date/kopts.
   -P     Prune empty directories.
   -d     Build directories, like checkou➥
t does.
[...]
```

... was not all that wrong: Although there is no such option as **– –***help*, *cvs* spits out precisely what we need: aid for the *cvs* command *update*. The option *-d* looks promising and that's also how it turns out: *update does in fact normally only update files already checked out, but when one says update -d*, it is a bit more willing to co-operate and also creates directories which are missing from the working data directory.

```
[trish@lunar course]$ cvs update -dD "200➥
1-10-15 21:55:53"
cvs update: Updating .
cvs update: Updating slides
cvs update: Updating concept
U concept/concept.tex
cvs update: Updating script
```

Now we have got *concept.tex* back and can print out the concept. So that you do not first have to consult the CVS book mentioned in the box "More documentation on *cvs*", I can reveal to you now that the ominous *-A* flag in the *update* command is the only option for getting an up-to-date working copy without *concept.tex*:

```
[trish@lunar course]$ cvs update -A
cvs update: Updating .
cvs update: Updating slides

U slides/unixslide.tex
cvs update: Updating concept
cvs update: warning: concept/concept.te➥
x is not (any longer) pertinent
cvs update: Updating script

U script/unixcourse.tex
```

Whether you now wish to delete the *concept* directory, using *-P*, is entirely up to you. ■

More about *cvs*

Anyone who is in utter despair with the documentation supplied with *CVS*, should not give up. At *http://cvsbook.red-bean.com/* can be found the GPLed section of Karl Fogel's CVS book for online browsing or downloading. Since this Answer Girl can in no way provide exhaustive information on *CVS*, it is recommended as further reading for all those who've tasted blood.