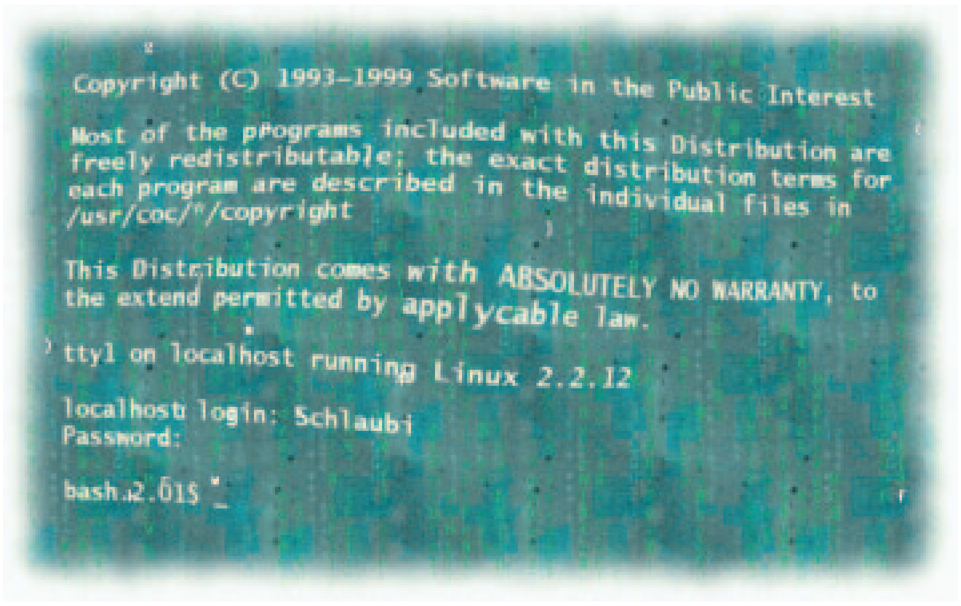
At your command
A TERMINAL CASE

HEIKE JURZIK

Even though lots of things can easily be controlled by means of graphical user interfaces such as KDE or GNOME – anyone really wanting to tax his Linux system cannot avoid using the command line. Apart from this, there are also many situations where it's good to know your way around a bit in the jungle of command lines.



```
Copyright (C) 1993-1999 Software in the Public Interest

Most of the programs included with this Distribution are
freely redistributable; the exact distribution terms for
each program are described in the individual files in
/usr/coc/*/copyright

This Distribution comes with ABSOLUTELY NO WARRANTY, to
the extend permitted by applycable law.

tty1 on localhost running Linux 2.2.12

localhost login: Schlaubi
Password:

bash.2.01$ _
```

Have you ever got fed up of switching back and forth between lots of terminals because you wanted to run several applications at once in the foreground? Or have you been downcast because a process which you had to start on a computer at the workstation was not ready in time to go to the ball, but you wanted to check the output of the program? *screen* is an extremely powerful tool which can make many tasks easier for you.

Before starting the program for the first time it's best to check to what value the environmental variable TERM (**terminal emulation**) has been set, since this will be evaluated immediately when the program is started:

```
huhn@asteroid:# echo $TERM
xterm
huhn@asteroid:# export TERM=vt100
huhn@asteroid:# echo $TERM
vt100
```

The programmers of *screen* point out explicitly in their documentation to the fact that this tool gets along best with *vt100* – so it's best to check first. With *screen* you can simulate up to ten virtual windows in a single Xterm (or on the console). You can then run programs in all these windows – with each of the virtual windows being independent of the others. Simply type *screen* – after a short greeting text there are instructions on what happens next:

```
[Press Space or Return to end.]
```

So using the space bar, you can now enter the realm of infinite expanses of terminals. You have access to a whole range of commands, all starting with [Ctrl-a]: Hold down the [Ctrl] key and type [a]. Now the program waits for the next command inputs: [Ctrl-a] [?] for example gives a complete overview of the key configuration (see Table 1).

Out of the blue – screen!

Apart from all the control commands within the window it is of course also possible to provide the

Terminal emulation: *The program responsible for screen output, appears to the system as a terminal. Linux consoles or an Xterm for example can use certain control sequences for highlighting, cursor positioning etc. Sometimes these emulate real hardware terminals, e.g. those of the type DEC vt100. If the environmental variable TERM is set to vt100, the program can be controlled like a vt100 terminal.*

program with various parameters at the start. In case you have started *screen* several times and no longer know how many and whether these are currently active, there is the option *-ls* (stands for: *-list*):

```
huhn@asteroid:# screen -list
There are screens on:
    1200.pts-10.asteroid    (Attached)
    1203.pts-14.asteroid    (Detached)
```

Here you can see the process ID (*pid*), then the virtual terminal (*tty*) in which the *screen* was started, the host (*asteroid*), and as the last piece of information whether it is currently active ("attached") or has been put to sleep ("detached"). Inactive *screens* can be brought back to life with *screen -r [pid.tty.host]*. It is only necessary to specify the process number and terminal if several *screens* are inactive. You can make this task much easier for yourself by giving the session a name right from the start: *screen -S petronella* names your *screen* "petronella". In the overview this is then called: *1364.petronella* – the name thus replaces terminal and host. By the way, if ever a *screen* process hangs, you can detect this in the overview from the status flag "dead". You can get rid of it elegantly with the parameter *screen -wipe*.

When you revive a *screen* which has been put to sleep, you may sometimes want to scroll back to look at the last outputs from current programs. 100 lines are standard for the buffer. You can alter this with the aid of the option *-h number of lines*. So with *screen -h 1000* you can now go back 1000 lines. To move around in this buffer there is a range of keyboard commands. To do this first go into Copy/Scrollback mode (see Table 1, [Ctrl-a] [Esc]). If you already know and use the editor *vi*, you will certainly be familiar with the commands for cursor movement. Otherwise you can find a short reference on the commands in Table 2.

Stars of the small *screen*

You can create a configuration file in your home directory, *.screenrc*, in which you can enter specific wishes for program behaviour. For example if you enter

```
startup_message off
```

the greeting message at the start of the program will be left out. Another practical option is that of defining your own commands. E.g. if you write in the *.screenrc*

```
bindkey ^f screen ssh marvin.cologne.de
```

(and not, as described in the Man page, *bind xy*!), when [Ctrl-f] is pressed in the *screen* window, a new screen will automatically be opened with an *ssh* connection to the computer *marvin.cologne.de*. In this way you can define lots of useful aliases. If you would like to have more than 100 lines as standard buffer, you can use the entry

```
defscrollback 1000
```

to define your own buffer size. Another nice feature is the so-called *vbell_msg*. For this you must first define: *vbell on*, then the desired message which is to appear when a window receives a "beep" ([Ctrl-g]), e.g.

```
vbell_msg "Hello! Here's a beep!"
```

There is a whole range of tips and tricks on this subject in the very comprehensive Man page. It is also worthwhile taking a look into the default configuration file */etc/screenrc*. If you would like to read more on this subject, most distributions have a very well-written *README* and an *FAQ*. The directory in which these files are located depends on the distribution. For Red Hat it's */usr/share/doc/screen-3.9.5/*, for Debian */usr/doc/screen/*, and for Mandrake */usr/doc/screen-3.9.5/*.

(Tip: you can read these files, if they end in *.gz*, thus are *gzip*-compressed, with the program *zless* – this is the case for example with Debian Linux.) Otherwise the following applies:

```
Send bugreports, fixes, enhancements,
t-shirts, money, beer & pizza to
screen@uni-erlangen.de
```

(hge) ■

The author

Heike Jurzik works at the computing centre of the University of Cologne as Administrator of the local news-server. She has been working on Linux systems since 1996. And because the computer keyboard is enough when it comes to keyboard instruments, instead of piano she prefers to play the violin in a symphony orchestra and when she gets the chance enjoys reading a good book.

Screen URLs

Homepage of the GNU Project screen is http://www.gnu.org/software/screen/
A nice collection of information can be found at http://www.math.fu-berlin.de/~guckes/screen/ .

Table 2: The most important commands for Movement command

h, j, k, l	move the cursor line by line or column by column, left, right, up, down.
0, $	go to extreme left or right end of line.
H, L, M	moves the cursor in the column on the far left to the top, bottom or middle.
+, -	line up or down.
G	jumps to end of buffer.
g	jumps to start of buffer.
w,b,e	jump word by word: back, forward and to end of word.

Table 1: Key combinations in *screen*

Keyboard shortcut	Command	Meaning
[Ctrl-a] [?]	help	Lists all key configurations.
[Ctrl-a] [c]	screen	Opens an additional virtual window.
[Ctrl-a] [space bar]	next	Changes to next window, and if the command is repeated, one can ”run through” all the windows.
[Ctrl-a] [Ctrl-a]	other	Constantly changes back and forth between two windows.
[Ctrl-a] [0...9]	select n	Changes to window with no. n.
[Ctrl-a] [w]	windows	Shows in a line on the lower edge for a short time how many windows have been started, the current one being highlighted with *.
[Ctrl-a] [a], [s] or [q]	meta/xoff/xon	Sends a [Ctrl-a], [Ctrl-s] or [Ctrl-q] direct to the window, needed for some programs (e.g. Emacs), which also have [Ctrl-a] control sequences
[Ctrl-a] [x]	lockscreen	Locks the screen – after entering a valid password you can carry on working
[Ctrl-a] [H]	log	Logs the standard output in a file, and depending on the number of the window (1-10) the logfile is called screenlog.n, calling up [Ctrl-a H] again ends the logging
[Ctrl-a] [Esc]	copy	Changes to copy mode: If there is no mouse to mark text, one can now go with the letters h, j, k, l to the desired point on the screen, make the marking with the space bar, then go to the next point, press the space bar again, to store it on the ”clipboard”. With [Ctrl-a] ”]” ([Ctrl-A] followed by a closing square bracket) incidentally one inserts the marked text, with [Esc] the action is interrupted.
[Ctrl-a] [d]	detach	”Releases” the screen, all processes started therein continue to run, but the program detaches itself from the terminal: Now you can log out. With screen -r the screen can be called up again (complete explanations follow in the text).
[Ctrl-a] [D] [D]	pow_detach	”Power Detach” – not only detaches the screen, but also immediately logs out of the terminal.
[Ctrl-a] [K]	kill	Destroys the whole screen – fortunately there is a safety challenge at this point: Really kill this window [y/n]

A series of tips on command line favourites.

AT YOUR COMMAND

RICHARD SMEDLEY

| less [pipe, less]

If you are fairly new to the command line, you may have been occasionally frustrated by a list command, such as ‘ls /etc’ scrolling off the screen past the information that you wanted. Perhaps you have used <RtShift><PgUp> and <RtShift><PgDn>, a very useful utility if your shell allows it – but annoying over several pages. The answer is:

```
ls /etc | less
```

The | [pipe] is the Unix tool for gluing together commands by sending the output of one to the input of the next. Piping several screens to the pager *less* enables easy back-and-forth scrolling with <b> and <f>. For more details of the scrolling commands available type

```
less –help
```

If you have never used less before, then you need to know that ‘q’ will exit the program (and this is also the command to leave man pages). ■