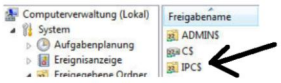
zurecht. Einfach, weil viele Befehle dort nicht funktionieren (Befehlszeilenmodus z. Bsp.). Der fehlende Wechsel auf die Kommandozeile fällt mir da auf Anhieb ein. Zugegeben, ich habe mich nie lange mit `kate` aufgehalten, weil ich sowieso mit `gvim` einen brauchbaren Editor hatte. Darum bleibe ich auch auf der Benutzeroberfläche bei `gvim`. Dort funktionieren uneingeschränkt alle Tastenkombis.

Es spielt auch keine Rolle, wenn man `vim` nicht perfekt beherrscht. Ist auch nicht notwendig. Nebenbei kenne ich auch niemanden, der das von sich behauptet.

Wenn Du beispielsweise ein neues Kommando lernst (`dd` zum Beispiel… lösche Zeile) so kannst Du dies sogleich mit Deinen Pfeiltasten verknüpfen. `d2j` bedeutet in diesem Fall, dass Du diese und die unteren zwei Zeilen löscht. `d` *(delete)* gibt an, dass gelöscht wird (`dd` würde gleich die Zeile löschen… wollen wir ja jetzt nicht), zwei wie viele Zeilen danach und `j` die Richtung. Also nach unten, wie wir ja eingangs mit den Pfeiltasten festgestellt haben. Ergo, drei Zeilen insgesamt gelöscht. Einfach. Für die Profis geht `3dd` natürlich auch.

Du willst von der aktuellen Stelle bis ans Zeilenende löschen? (`d$` oder `D`) `d$`, zumindest das $ Zeichen kennt man aus der Windowswelt für administrative Freigaben. Das Dollarzeichen wird immer am



Schluss angehängt. Also ans Ende. Somit ist der Befehl auch klar > `d…` und bis ans Ende `>$=d$…` bis am Anfang wäre `d0`. (Null). Das ist der selbe Schmäh wie bei den regulären Ausdrücken (`regex…` *regular expressions*) $ ist dort auch mit dem Zeilenende definiert.

Die Stärke von `vim` liegt in seiner schier unendlichen Kombinierbarkeit. Alleine mit den vier Pfeiltasten kannst Du schon zig Kommandos kombinieren. Obwohl Du in Wirklichkeit nur fünf Kommandos gelernt hast. vier für die Richtung und `d` für löschen. `y` steht für kopieren *(yank)*. Selbes Spiel. Ich will diese und die obere Zeile kopieren? `Y1k`. Fertig. Einfügen geht am einfachsten mit `p` *(paste)*.

Damit man sich beim Zeilen zählen nicht so schwer tut, kann man `vim` auch mit relativer Zeilennummerierung betreiben. Siehe **Bild 24e**.

Was siehst Du da? Der gelbe Pfeil zeigt die aktuelle Zeile an. 16. Von dort weg zählen die Zeilen relativ zur Position. Heißt, ich brauch nicht so viel Kopfrechnen und nur die relative Zeilennummer für Operationen ablesen. „`d5k`" würde alles von der aktuellen Zeile bis rauf zur Zeile „`UUID… `" löschen.

Das ist vielleicht bei kleinen Dateien nicht so tragisch mit dem Zählen, aber bei ein paar 100 oder 1000 Zeilen langen Dateien kann das schon hilfreich sein.

Das sind so Kommandos, die vergisst man nur sehr schwer, helfen aber ungemein. Nicht umsonst hat sich das Glumpat schon über 40 Jahre gehalten. Die Basics musst Du lernen. Das bleibt niemandem erspart. Aber es gibt Schöneres.

Sei gewarnt. Wenn Du mit Deinem eingefahrenen Habitus an diesen Editor herangehst, schreist Du spätestens nach 10 Sekunden gegen die Rauhfasertapete. Immer wenn Du `vim` startest, bist Du automatisch im Normalmodus.

Warum? Richtig, weil Du höchstwahrscheinlich erst zur richtigen Stelle navigieren willst und dort erst was machst. Vim startet immer im Command mode (Normalmodus). Man kann es nicht oft genug sagen. Heißt, dass Du nicht gleich drauf losschreiben kannst. Können schon, wird aber nix werden. Deshalb kommen dann die akustischen Angriffe auf die Tapete. Du willst schreiben? Drücke „`i`" für *insert*, das reicht fürs Erste.

Der einzige praktikable Fall, wo ich `vim` gleich im Einfügemodus starten wollte, ist wenn ich eine neue leere Datei anlege und etwas hinein schreibe. Wenn man die Datei danach öffnet, navigiert man zu 99% erst zu einer bestimmten Stelle, um dort zu editieren.

Die Wahrscheinlichkeit, dass Du eine Datei ausgerechnet in der ersten Zeile editieren willst, ist verschwindend gering. Darum der Normalmodus. Denk einmal darüber nach. Das Navigieren (und hauptsächlich Nachdenken, was ich eigentlich machen will) kostet die meiste Zeit. Darum auch die hilfreichen Tastenkombinationen zum Navigieren, Löschen, Markieren, Suchen, Blättern oder Kopieren.

Tastenkombinationen sind immer schneller als Mausbewegungen. Und vor allem genauer. `Go` (unterhalb der letzten Zeile etwas schreiben), `A` (ans Ende der Zeile anhängen… schaltet automatisch in den Einfügemodus um), `gg0` (oberhalb der ersten Zeile etwas schreiben).

`Ci"`… hmmm *(change in)* heißt, dass ich den Text innerhalb der Apostrophe lösche und gleich einen Neuen hineinschreiben kann. Ich bin dann also schon im Einfügemodus. Dabei kann ich irgendwo auf der entsprechenden Zeile stehen und muss auch gar nicht zwischen den Apostrophen sein.

Mit der Maus müsste ich erst hinfahren, genau zwischen den Apostrophen hineinzielen, markieren und dann reinschreiben.

Oder `ESC:wqENTER`. Damit speichert und schließt man eine Datei. Profis nehmen `ESC` und `ZZ`. Fertig. vier zu sechs Tastendrücke. Das scheint nicht nach viel Ersparnis zu klingen, summiert sich aber mit der Zeit auf.

Ja klar steigen da jetzt ein paar Leser aus und schauen wie ein Schwein in ein Schweizer Uhrwerk. `vimtutor` anwerfen und es wird klarer.

Ich könnte jetzt noch hundert praktische Beispiele anführen, was meiner Erfahrung nach aber nur noch mehr zur Verwirrung beiträgt. Mach den `vimtutor` fertig und vor allem verinnerliche Dir das Konzept und Du bist auf einem sehr guten Weg. Wenn Du den `vimtutor` beherrscht, kannst Du wunderbar arbeiten. Und das, obwohl der `vimtutor` vielleicht 3% von den Fähigkeiten eines `vim` darstellt.

Ich weiß, anfangs tendierst Du öfters zu `nano` oder `geany` zurück. Das ist nur menschlich. Ich habe Linux-Admins gekannt, die haben die ersten fünf Jahre auch nur `nano` verwendet, um schlussendlich doch bei `vim` zu landen. Ps: Nano ist ein einfacher, populärer Editor unter Linux.

Schreiben, editieren und Text in allen möglichen Formen zu bearbeiten sind die Domäne des `vim`. Seine Vielseitigkeit unterscheidet ihn aber entsprechend von den meisten anderen Editoren. Du willst von `vim` aus auf die Kommandozeile? `Escape:!command`. Damit springst Du auf die Kommandozeile (Kommandozeilenmodi) und der Befehl (`df -h`) gibt Dir den Festplattenplatz aus. Wohlgemerkt, das alles aus dem Editor heraus. Siehe **Bild 23a**.

Vim ist der Standardeditor des Sysadmins. Zumindest sollte man rudimentäre Kenntnisse darüber besitzen. Programmierer greifen auch gerne zu `vim`. Er ist einfach ein Profiwerkzeug.

Auch wenn man sich remote auf ein anderes unixoides System aufschaltet, so hat man doch immer zumindest einen `vi` am entfernten System zur Verfügung. Sehr schön ist auch die Möglichkeit, mittels `timeline` schon gelöschte Sachen zurück zu holen. Siehe **Bild 24**.

Eines der größten Features von `vim`. Man kann jederzeit auch gelöschte Sachen aus der Textdatei wieder zurückholen. Auch wenn man schon ein paar mal neu darüber geschrieben, die Datei schon abgespeichert oder dazwischen neu gebootet hat. Auch drei Tage später kann ich mir von einer gespeicherten Datei jederzeit sämtliche Änderungen anzeigen lassen.

„`u`" für *undo* ist die entsprechende Taste. Grafisch sieht man das schön, wie auf Bild 24 ein neuer „track" abzweigt. Dort wurde schon mal gelöscht. Normalerweise kommt man auf diesen neuen track mit einem gewöhnlichen Editor nicht mehr hin. Vim kann das. Egal, für welchen Editor Du dich schlussendlich entscheidest, schau dass dieser das beherrscht. Ich kann Dir nur sagen, viele können es nicht. Aber das sind Sachen, die Dir eines Tages Deinen Allerwertesten retten können. Alleine dafür lohnt sich schon die steile Lernkurve.

`vim` legt einen Ordner an (`undodir`) worin sämtliche Änderungen der Datei enthalten sind. Siehe **Bild 24a.**