Kurzanleitung zum Editor vi

Der vi ist ein klassischer Unix-Editor, der im Textmodus arbeitet. Anders als bei den „gewohnten" Editoren unterscheidet er zwischen einem **Befehlsmodus** und einem **Eingabemodus**. Der Editor startet (wenn Sie ihn z. B. mit `vi dateiname` aufrufen) im Befehlsmodus.

- Um den Editor ohne Änderungen wieder zu verlassen, drücken Sie [Esc] und geben dann `:q!` ein.
- Wenn Sie Inhalte im Editor geändert haben, können Sie mit [Esc] und `:wq` (write + quit) gleichzeitig speichern und den Editor verlassen.
- Um nur (zwischen-) zu speichern, drücken Sie [Esc] und geben `:w` (write) ein.
- In den Eingabemodus wechseln Sie mit `i` (Eingabe ab aktueller Cursorposition) oder `a` (Eingabe hinter dem Cursor); jetzt können Sie Text eingeben. Sie verlassen den Befehlsmodus mit [Esc].
- Im Befehlsmodus können Sie die geöffnete Datei nach Suchbegriffen durchsuchen. Dazu drücken Sie `/` und direkt dahinter den Suchbegriff (abgeschlossen mit [Eingabe]). Der Editor springt dann zum ersten Treffer; weitere Treffer finden Sie mit `n` (next), rückwärts springen Sie mit `N`. Alternativ zu `/` können Sie die Suche auch mit `?` starten – dann ist die Standard-Suchrichtung umgekehrt (`n` springt also nach oben, `N` nach unten).
- Löschen können Sie im Befehlsmodus mit verschiedenen Tasten: `x` löscht das Zeichen unter dem Cursor, `dw` löscht ab Cursorposition bis zum Anfang des nächsten Worts, `d$` löscht ab Cursorposition bis zum Zeilenende, `d^` bis zum Zeilenanfang. `dd` löscht eine ganze Zeile. Vor diesen Befehlen können Sie eine Zahl eingeben, um eine Aktion mehrfach ausführen zu lassen: So löscht z. B. `6x` sechs Buchstaben, und `3dd` löscht drei Zeilen.
- Mit `u` machen Sie im Befehlsmodus die letzte Aktion rückgängig, das geht auch mehrfach, und auch `u` unterstützt das Voranstellen einer Zahlenangabe. Eine Redo-Funktion (also Aufheben des letzten Rückgängigmachen) gibt es auch, das geht mit [Strg]+`r`.

Installation der virtuellen Maschine

Für die Installation der virtuellen Maschine mit Debian Linux benötigen Sie eine Installation des Virtualisierungsprogramms VirtualBox (oder alternativ VMware) und das Installationsimage von Debian Linux 6.0.10, das von u. a. von der Kurswebseite aus verlinkt ist.

`http://virtualbox.org/`

`http://cdimage.debian.org/mirror/cdimage/archive/6.0.10/i386/iso-dvd/debian-6.0.10-i386-DVD-1.iso`

Erzeugen Sie in VirtualBox eine neue virtuelle Maschine, wählen Sie als Betriebssystem Linux (Debian, 32-bit) und legen Sie eine 4 GByte große virtuelle Platte an. Wählen Sie außerdem für das virtuelle DVD-Laufwerk die Debian-Imagedatei aus. Dann können Sie die virtuelle Maschine starten und die Installationsroutine durchlaufen, wie ich sie in der Vorlesung am 07.04. demonstriert habe.

Übungen zur Shell und zum Editor vi

1. Starten Sie Linux (in der virtuellen Maschine) im Textmodus. (Die offizielle Linux-VM für diese Veranstaltung bootet in den Textmodus.)
2. Melden Sie sich als normaler Anwender (nicht als *root*) mit Ihrem Benutzernamen und dem Passwort an.
3. Überprüfen Sie mit `pwd`, dass Sie in Ihrem Home-Verzeichnis „sind“.
4. Erstellen Sie mit `mkdir` zwei neue Unterverzeichnisse `texte` und `beispiele` in diesem Ordner.
5. Wechseln Sie mit `cd` in das Verzeichnis `beispiele` und überprüfen Sie nach dem Wechsel, dass Sie wirklich dort gelandet sind.
6. Kopieren Sie mit `cp` die Dateien `/etc/fstab` und `/etc/passwd` in das aktuelle Verzeichnis.
7. Öffnen Sie die Datei `fstab` (die im Arbeitsverzeichnis, nicht die in `/etc`) im Editor `vi`.
8. Benutzen Sie die Suchfunktion von `vi`, um nacheinander alle Vorkommen des Wortes `dev` (wie: device) in der Datei zu finden. Beobachten Sie, was passiert, wenn Sie nach dem letzten Treffer in der Datei weiter suchen.
9. Testen Sie auch die Rückwärtssuche; diesmal können Sie z. B. nach `sys` suchen.
10. Was passiert, wenn Sie einen Suchbegriff angeben, der im Text gar nicht vorkommt? Suchen Sie z. B. nach `qwertzuiop`.
11. Die Datei ist in Spalten sortiert. Löschen Sie möglichst effizient in jeder Zeile die komplette dritte Spalte (mit den Dateisystem-Eintrag, im Beispiel `ext3`), so dass z. B. aus
    ```
    /dev/sda1   /   ext3   errors=remount-ro   0   1
    ```
    der Eintrag
    ```
    /dev/sda1   /   errors=remount-ro   0   1
    ```
    wird. Hinweis: Einzelne Buchstaben zu löschen, ist nicht möglichst effizient; löschen Sie wortweise.
12. Speichern Sie die geänderte Datei und verlassen Sie den Editor.
13. Zeigen Sie den Inhalt das Arbeitsverzeichnisses an und überprüfen Sie mit `ls -l`, dass die Datei `fstab` das aktuelle Datum und die aktuelle Uhrzeit trägt. (Falls die Uhr in der VM nicht richtig eingestellt ist, vergleichen Sie mit der Zeit, die in der VM eingestellt ist; dazu geben Sie `date` ein.)
14. Jede Datei besitzt unter Linux eine eindeutige Kennung, die so genannte *Inode-Nummer*. Schlagen Sie mit `man ls` in der **Manpage** zu `ls` nach, wie Sie diese Inodes anzeigen können. (Sie verlassen die Anzeige der Manpage mit `q`.) Zeigen Sie danach die Dateien im Arbeitsverzeichnis mit Inode-Nummern an.
15. Wechseln Sie eine Verzeichnisebene nach oben (also zurück in Ihr Home-Verzeichnis).
16. Lesen Sie erneut die Manpage zu `ls` – diesmal geht es darum, wie Sie rekursiv nicht nur den Inhalt des Arbeitsverzeichnisses, sondern auch die Inhalte aller Unterordner anzeigen können. Testen Sie diese Option, wobei Sie sowohl die einfache als auch die lange Darstellung (`-l`) verwenden. Beachten Sie, dass Sie mehrere Optionen auch kombinieren können (`-abc` statt `-a -b -c`).
17. Legen Sie mit `mkdir` ein Verzeichnis `kopien` im Arbeitsverzeichnis an und kopieren Sie alle Dateien aus dem Ordner `beispiele` dort hinein. Wechseln Sie anschließend in den Ordner `kopien` und überprüfen Sie, dass sich dort nun ebenfalls die beiden Dateien `fstab` und `passwd` befinden.
18. Mit `ls` können Sie nicht nur den Inhalt des Arbeitsverzeichnisses ausgeben; wenn Sie als Argument einen Verzeichnisnamen ergänzen, zeigt `ls` den Inhalt dieses Ordners an. Wechseln Sie eine Ebene nach oben (ins Home-Verzeichnis) und lassen Sie `ls` die Inhalte der beiden Verzeichnisse `beispiele` und `kopien` in ausführlicher Darstellung anzeigen – dabei soll `ls` auch die Inode-Nummern ausgeben. Vergleichen Sie die Inode-Nummern der gleichnamigen Dateien (jeweils `fstab` und `passwd` in beiden Ordnern) und überzeugen Sie sich so davon, dass es verschiedene Dateien sind.

19. Auch Verzeichnisse haben eine Inode-Nummer. Wenn Sie versuchen, diese Nummern zu den Verzeichnissen `beispiele` und `kopien` ausgeben zu lassen, erscheinen aber die Inhalte dieser Ordner. Es gibt eine `ls`-Option, mit der Sie dies unterdrücken können und stattdessen nur die Verzeichniseinträge selbst ausgeben können. Finden Sie diese (wieder in der Manpage) und wenden Sie sie an.
20. Versuchen Sie, das (nicht leere) Verzeichnis `kopien` mit `rmdir` zu löschen; der Versuch wird fehlschlagen.
21. Verwenden Sie einen geeigneten Befehl, um das Verzeichnis `kopien` mit allen enthaltenen Dateien zu löschen.
22. Erzeugen Sie eine ganze Verzeichnishierarchie `a/b/c/d/e/f/g` im Arbeitsverzeichnis. Wenn Sie versuchen, das mit `mkdir a/b/c/d/e/f/g` zu tun, erhalten Sie eine Fehlermeldung. Der Ansatz, der Reihe nach `mkdir a`, `mkdir a/b` und `mkdir a/b/c` auszuführen (was funktionieren würde), ist zu aufwendig – es geht auch schneller. Lesen Sie dazu in der Manpage zu `mkdir` nach, wie Sie die Aufgabe mit einem einzigen Kommando erledigen können. Prüfen Sie, dass es geklappt hat, indem Sie in das unterste Verzeichnis (`g`) wechseln.
23. Lassen Sie sich den vollen Pfad des Arbeitsverzeichnisses ausgeben; das Ergebnis sollte die Form
`/home/user/a/b/c/d/e/f/g`
haben.
24. Wechseln Sie in Ihr Home-Verzeichnis zurück – Sie könnten das über einen relativen Pfad mit `cd ../../../../../../..` tun, aber es gibt auch einen direkten Weg ins Home-Verzeichnis.
25. Um das Verzeichnis `a` mit allen Unterordnern zu löschen, könnten Sie wie in Aufgabe 21 vorgehen. Da `a` aber nur einen Unterordner (mit weiteren Unterordnern) und keine Dateien enthält, können Sie auch `rmdir` verwenden. Schlagen Sie in der Manpage zu `rmdir` nach, welche Option Sie dafür verwenden müssen, und testen Sie das Kommando. Wenn es nicht auf Anhieb klappt, haben Sie evtl. ein falsches Argument angegeben.
26. Wechseln Sie ins Wurzelverzeichnis `/` und von dort aus direkt in den Unterordner `beispiele` Ihres Home-Verzeichnisses. Wie sieht für den letzten Schritt der bei `cd` anzugebende Pfad aus? Falls Sie diesen Pfad mit `/home/...` zusammenbauen: Es geht auch kürzer, aber wie?
27. Um die Datei `passwd` im Arbeitsverzeichnis (`beispiele`) anzuzeigen, könnten Sie den Editor `vi` verwenden. Es gibt aber auch zwei Alternativen dazu: Lesen Sie die Manpages zu den Kommandos `cat` und `less` und probieren Sie mit beiden aus, die Datei `passwd` zu betrachten. Hinweis: `less` beenden Sie, indem Sie `q` drücken.
28. In `less` können Sie genauso nach einem Suchbegriff suchen wie im Editor `vi`. Testen Sie Vor- und Rückwärtssuche und den Wechsel zwischen den Suchrichtungen (ohne Ändern des Suchbegriffs).
29. Das Kommando `grep` hilft beim Suchen in Dateien; die Syntax ist `grep Begriff Dateiname`. Geben Sie mit Hilfe dieses Tools alle Zeilen der Datei `passwd` aus, in denen der Dateiname `/bin/sh` (das ist die **Sh**ell) auftaucht. Geben Sie anschließend alle Zeilen aus, in denen dieser Dateiname *nicht* auftaucht – wie das geht, verrät die Manpage zu `grep`.
30. Mit den beiden Tools `head` und `tail` (deutsch: Kopf und Schwanz) können Sie die ersten bzw. letzten Zeilen einer Textdatei ausgeben lassen; standardmäßig sind es die ersten/letzten zehn. Finden Sie heraus, wie Sie die ersten bzw. letzten vier Zeilen erhalten können; wenden Sie das auf die Datei `passwd` an.
31. Fahren Sie den virtuellen Rechner herunter: Dazu geben Sie zunächst `su` ein, dann das Passwort des Benutzers *root*, und dann den Befehl `shutdown -h now`. Fertig :)

Übungen zur Job- und Prozess-Verwaltung

1. Starten Sie Linux (in der virtuellen Maschine).
2. Melden Sie sich als normaler Anwender (nicht als *root*) mit Ihrem Benutzernamen und dem Passwort an.
3. Erzeugen Sie im Home-Verzeichnis mit dem Befehl `touch` drei Dateien `1.txt`, `2.txt` und `3.txt`. (Sie können also drei Dateinamen gleichzeitig als Argumente für `touch` vergeben.)
4. Öffnen Sie die erste der drei Dateien im Editor `vi`. Drücken Sie dann (im Kommandomodus von `vi`) [Strg-Z], um den Editor zu unterbrechen (aber nicht zu beenden). Prüfen Sie mit `jobs`, dass er nun als „stopped“ in der Job-Liste auftaucht.
5. Wiederholen Sie Schritt 4 mit den Dateien `2.txt` und `3.txt`; nun gibt es also drei suspendierte Editoren (und entsprechend drei Jobs in der Job-Liste).
6. Das Kommando `jobs` können Sie auch mit der Option `-l` aufrufen: Dann zeigt es neben den Job-Nummern auch die Prozess-IDs an. Probieren Sie das aus.
7. Geben Sie `ps` ein und identifizieren Sie die drei Editor-Jobs in der (größeren) Prozessliste. Geben Sie `ps auxw` ein, um die Liste aller Prozesse in Langform zu sehen – finden Sie auch hier wieder die Prozesse. Nutzen Sie z. B. `grep`, um die Ausgabe von `ps` zu filtern.
8. In der Vorlesung haben Sie gesehen, dass man die Spalten in der `ps`-Ausgabe über die Option `-o` und geeignete Argumente anpassen kann (z. B. `ps -o user,pid,cmd` für die Ausgabe von Benutzername, Prozess-ID und Kommando). Suchen Sie in der Manpage zu `ps` (`man ps`) die richtigen Argumente für `-o`, um a) die Prozess-ID, b) die „parent process id“ (Prozess-ID des Vaterprozesses), c) den Nice-Level, d) die Anzahl der Threads (in Linux-Notation: „light weight processes“) und e) das Kommando anzuzeigen.
9. Mit welcher Option für `ps` können Sie die Ausgabe auf Prozesse beschränken, die einen bestimmten Namen haben? Geben Sie probeweise eine Liste aller Shell-Prozesse (Name: `bash`) aus.
10. Starten Sie durch dreimalige Eingabe von `sleep 1000 &` im Hintergrund drei Jobs, die einfach 1000 Sekunden schlafen und sich dann beenden. Beenden Sie den ersten Job mit `kill`, wobei Sie die Job-ID dieses Jobs verwenden. Achten Sie dabei auf die richtige Syntax für die Angabe der Job-ID. Prüfen Sie mit `jobs`, dass der Prozess verschwunden ist. Beenden Sie danach den zweiten Prozess, diesmal aber über seine Prozess-ID. (In Aufgabe 6 haben Sie gesehen, wie Sie die Zuordnung Job-ID → Prozess-ID herausfinden.) Prüfen Sie erneut, dass nun nur noch der dritte `sleep`-Prozess übrig ist.
11. Die Editor-Prozesse werden Sie mit `kill` nicht so einfach los, weil sie suspendiert sind und das Signal erst nach dem „Aufwecken“ (mit `fg`) verarbeiten würden. Sie können aber das KILL-Signal (Nr. 9, also `kill -9 …` oder `kill -KILL …`) an die Editor-Prozesse schicken, das auch suspendierte Prozesse erfolgreich beendet. Schießen Sie auf diese Weise die ersten beiden Editor-Prozesse ab (wahlweise über die Job- oder Prozess-IDs). Holen Sie dann den verbleibenden Editor-Prozess in den Vordergrund: Sie sollten dann wieder im Editor arbeiten können (in der Datei `3.txt`). Verlassen Sie den Editor auf „normale“ Weise, z. B. über [Esc] und `:wq`.
12. Geben Sie den Befehl `top` ein, um eine sich regelmäßig aktualisierende Liste der laufenden Prozesse zu erhalten. Nach welcher Spalte ist die Ausgabe sortiert? Ändern Sie die Sortierspalte und beobachten Sie, wie dadurch andere Prozesse „nach oben“ kommen. Tipp: Die Hilfe in `top` erreichen Sie über `?` oder `h`. Verlassen Sie das Programm (`q`).

    Übrigens: Eine hübsche Alternative zu `top`, die auch das Scrollen zu weiteren Prozessen (die bei `top` aus der Anzeige rausfallen) erlaubt, ist `htop`. Sie können es mit `sudo apt-get install htop` nachinstallieren.

Verständnisfragen

13. Was ist der Unterschied zwischen Jobs und Prozessen?
14. Sie können Jobs und Prozesse beenden, indem Sie ihnen eines der beiden Signale `TERM` und `KILL` schicken. Welches Signal verschickt das `kill`-Kommando, wenn Sie kein Signal explizit angeben? Wie unterscheidet sich die Prozessbeendung via `TERM` und `KILL`?
15. Welche Signale kennt Ihr Linux-System? (Tipp: Die Manpage von `kill` erklärt, wie Sie eine Liste aller bekannten Signale anzeigen lassen können.)
16. Warum ist es (bei Linux/Unix) eine gute Eselsbrücke, von Nice-Levels statt von Prioritäten zu sprechen? Welche Nice-Levels kennt ein Linux-System?
17. Was meinen Sie: Warum kann ein normaler Anwender (nicht *root*) Prozessen zwar mit `nice`/`renice` eine niedrigere als die Standardpriorität zuweisen, aber keine höhere?
18. Um einen Prozess auch über die Abmeldung vom System hinaus weiter arbeiten zu lassen, können Sie `nohup` oder `disown` verwenden. Erklären Sie den Unterschied.

Aufgaben zur Programmierung und zur Theorie

19. **fork im C-Programm:** Betrachten Sie das kleine C-Programm `forktest.c`, das die Systemfunktion `fork()` aufruft, um einen neuen Prozess zu erzeugen:

```
#include <stdio.h>
main() {
  printf ("vor dem Fork-Aufruf \n");
  int pid = fork ();
  printf ("nach dem Fork-Aufruf \n");
}
```

Das Programm können Sie mit `gcc -o forktest forktest.c` kompilieren und dann mit `./forktest` (wichtig: mit führendem `./`) aufrufen.

a) Welche Zeilen gibt das Programm aus? Erklären Sie die Ausgabe und nennen Sie die Anzahl der Prozesse, die liefen.

b) Wie viele Prozesse laufen insgesamt, wenn Sie einen zweiten `fork()`-Aufruf unmittelbar nach dem ersten einbauen (`int pid2 = fork ();`)?

20. **Prozesse und Signale:** Öffnen Sie (in der grafischen Oberfläche; Start mit `startx`) ein Kommandozeilenfenster (Konsole, `xterm` etc.) und rufen Sie darin einen Editor auf, z. B. nedit:

```
nedit &
```

(Mit `&` starten Sie den Editor im Hintergrund und können in der Shell weiterarbeiten. Wenn nedit nicht installiert ist, müssen Sie ein anderes Programm starten, z. B. xeyes.)

Suchen Sie nun mit `ps auxw | grep nedit` nach dem Prozess und finden Sie seine Prozess-ID heraus. Mit dem Befehl `kill` können Sie dem Prozess Signale senden, z. B.

- `SIGSTOP` zum Anhalten des Prozesses (`kill -STOP` *`ID`*)
- `SIGCONT` zum Fortsetzen des Prozesses (`kill -CONT` *`ID`*)
- `SIGTERM` zum Beenden des Prozesses (`kill -TERM` *`ID`* oder `kill` *`ID`*)

Probieren Sie zunächst die ersten beiden Signale aus: Schicken Sie dem Editor das `STOP`-Signal und versuchen Sie dann, im Editor-Fenster Text einzugeben. Wechseln Sie danach zurück in die Konsole und schicken Sie dem Editor das `CONT`-Signal. Sehen Sie nun den Text, den Sie im gestoppten Zustand eingegeben haben?

Beenden Sie schließlich den Prozess, indem Sie ihm das `TERM`-Signal schicken.

a) Suchen Sie nach einem Prozess, der Ihnen nicht gehört. Was passiert, wenn Sie diesem ein Signal (z. B. SIGSTOP) schicken?

b) Lesen Sie die Manpage zu `killall` durch. Was würde passieren (nicht ausprobieren...), wenn Sie den Befehl `killall tcsh` bzw. `killall bash` (je nach Standardshell auf Ihrem Linux- System) eingeben?

21. **Prozesszustände:** Warum gibt es die Zustandsübergänge **a)** blockiert → laufend und **b)** bereit → blockiert nicht?

22. **Prozesse und Speicher:** Bei der Definition des Prozesses haben wir als wichtig erkannt, dass zu einem Prozess immer ein separater Speicherbereich gehört, die Schnittmenge der Speicherbereiche von zwei Prozessen ist immer leer. Wie genau eine Aufteilung des physikalischen Hauptspeichers erfolgt, haben wir noch nicht besprochen, und das ist für diese Aufgabe auch irrelevant.

    Gehen Sie mal von folgender alternativen „Definition“ aus, die nur im Rahmen dieser Übungsaufgabe gültig sein soll:

    > Ein Betriebssystem unterteilt den Hauptspeicher in mehrere paarweise disjunkte (sich nicht überschneidende) Speicherbereiche. Ferner gibt es mehrere „Aktivitätsstränge“ (ausführbarer Code), und es gilt: Das Betriebssystem ordnet jedem Speicherbereich keinen, einen oder mehrere dieser Aktivitätsstränge zu.
    >
    > Der Scheduler eines solchen Betriebssystems wählt (nach einem nicht näher zu bestimmenden Verfahren) einen Speicherbereich aus und aktiviert dann einen der Aktivitätsstränge, die diesem Speicherbereich zugeordnet sind. Wir sprechen nicht von „Prozesswechsel“, sondern von „Speicherbereichswechsel“.

    a) Wenn einem Speicherbereich 0 Aktivitätsstränge zugeordnet sind, was bedeutet das?

    b) Einem Speicherbereich ist ein Aktivitätsstrang zugeordnet. Haben wir es hier mit einem Thread oder einem Prozess (bei klassischer Betrachtung) zu tun?

    c) Einem Speicherbereich sind drei Aktivitätsstränge zugeordnet. Handelt es sich dabei (bei klassischer Betrachtung) um Threads oder Prozesse?