Beispiel `01_shell1`

Dr. Günter Kolousek

3. September 2016

1 Allgemeines

- Drucke dieses Dokument **nicht** aus!
- Als Betriebssystem wird in diesem Jahr **ausschließlich** Linux verwendet.
- In diesem Beispiel findet die Maus **absolut keine** Verwendung!!
- Am Anfang verwenden wir ausschließlich den vorgegebenen Texteditor (keine IDE)!

2 Anleitung

1. Öffne ein Terminal. Wenn du das Terminal öffnest, dann wird zusätzlich ein weiterer Prozess, die Shell, gestartet. Auf unserem System ist das die `bash` (bourne-again shell). Das kannst du dir mit dem Befehl `ps` (processes) anzeigen lassen. Probiere es aus.

   Denke an das, das du bereits gelernt hast: Ein Prozess ist ein gestartetes Programm!

2. Bedenke, dass die bash dir Kommando- **und** Dateinamenvervollständigung mittels Drücken der Taste `TAB` bietet. Weiters kannst du auf die alten Befehle mittels der Cursor-Tasten `Up` und `Down` zugreifen.

   Innerhalb der Kommandozeile kommst du mittels `C-a` (”C” steht für die CTRL oder STRNG, also Steuerungstaste) an den Zeilenanfang (Merkhilfe: ”a” wie anfang) und mittels `CTRL-e` an das Zeilenende (”e” wie end). Ausprobieren!

   Navigieren kann man weiters innerhalb der Zeile zeichenweise mit `CTRL-f` (forward) und `CTRL-b` (backward). Natürlich kannst du dafür auch die Cursortasten verwenden, aber wenn du mit der Tastatur gut umgehen kannst, dann verlassen deine Finger nie die Grundhaltung und du bist flotter beim Tippen.

In gleicher Art und Weise kannst du wortweise mit `M-f` ("M" wie Meta, das ist die `Alt` Taste) bzw. mit `M-b`. Wieder ausprobieren.

Mit `CTRL-d` wird das dem Cursor folgende Zeichen gelöscht. Der Effekt diese Tastenkombination kann natürlich auch mit der `Del` (bzw. `Entf`) Taste erreicht werden.

`CTRL-k` löscht den Inhalt von der aktuellen Cursorposition bis zum Zeilenende und merkt sich den gelöschten Text. Mittels `C-y` kann der gelöschte Text wieder eingefügt werden. Nach dem Drücken von `C-y` kannst du mit `M-y` den gerade ersetzten durch den vorigen gelöschten Text ersetzen. Das probierst du besser einmal aus.

Ok, das wird so nicht oft zum Einsatz kommen, aber es gibt den Editor Emacs, bei dem du alle bis jetzt gelernten Tastenbefehle ganz gleich verwenden kannst. Und da macht `M-y` schon mehr Sinn.

`CTRL-c` bricht die aktuelle Eingabe bzw. das gerade laufende Programm ab.

Das sind die notwendigen Grundkenntnisse zum Arbeiten mit der Kommandozeile.

3. Bei `ps` handelt es sich also um ein Programm, das du gestartet hast.

   Wenn du wissen willst *wo* im Dateisystem sich dieses Programm befindet, dann kannst du dies mit dem Programm `which` herausfinden. Probiere es einfach mit `which ps` aus!

   D.h. `ps` liefert dir die Prozesse, die du gestartet hast, aber es kann auch alle anderen Prozesse im System anzeigen.

4. Gibst du den Befehl `id` ein, dann werden dir Informationen zu deinem Benutzerkonto angezeigt. Gleich am Anfang findest du deine "uid" und deinen Benutzernamen. Weitere Informationen wie die Hauptgruppe "gid" und die weiteren Gruppen folgen hinterher.

   Benutzer können also zu Gruppen zusammengefasst werden. Der Grund dafür ist, dass man Rechte — wie zum Beispiel dem Zugriff auf Dateien — nicht nur für einzelne Benutzer sondern eben auch für eine Gruppe von Benutzern festlegen will. Doch damit werden wir später noch einmal befassen.

5. Ermittle jetzt auch das Verzeichnis in dem sich das Programm `id` befindet. Tja, die systemeigenen Programme, die von den "normalen" Benutzer verwendet werden können, befinden sich viele in diesem Verzeichnis.

6. Der angemeldete Benutzer ist auch in der Umgebungsvariable `USER` gespeichert. Auf den Wert einer Umgebungsvariable wird in der bash mittels einem vorgestellten `$` zugegriffen, also hier `$USER`. Eine Ausgabe eines Wertes kann mit dem Befehl `echo` erreicht werden. Also: `echo $USER`.

7. Mittels Eingabe eines Programmnamens wird also ein Prozess gestartet. An welchem Ort sich das in Frage kommende Programm befindet, lässt sich — wie gelernt — mit dem Befehl `which` bestimmen. Okay, aber woher weiß dieses Kommando,

dass sich das Programm gerade dort befindet? Woher weiß die Shell welches Programm gestartet werden soll, wenn du einen Programmnamen eingibst?

Schauen wir uns dazu einmal den folgenden Befehl an:

```
$ echo $USER
ko
```

Hier muss die Shell das Programm finden, das den Namen "echo" hat und dieses mit einem Kommandozeilenargument, nämlich dem Wert der Umgebungsvariable `USER`, aufrufen. Aber *wie* findet die Shell das Programm `echo`?

Das liegt an einer weiteren Umgebungsvariable, nämlich `PATH`. Lasse dir einmal den Wert dieser Umgebungsvariable anzeigen.

Du wirst sehen, dass es sich um eine Sequenz von Verzeichnispfaden handelt, wobei zwei hintereinanderfolgende Pfade jeweils durch einen Doppelpunkt getrennt sind. Soll ein Programm gestartet werden, dann durchsucht die Shell diese Liste der Pfadangaben vom Beginn an und führt das Programm aus, das es als erstes mit dem angegebenen Namen findet.

8. Nach dem Starten eines Terminals befindest du dich in deinem Home-Verzeichnis (wenn nichts anderes konfiguriert wurde). D.h. bei deinem aktuellen Verzeichnis (auch Arbeitsverzeichnis oder working directory genannt) handelt es sich um dein Home-Verzeichnis. Überprüfe dies mit dem Befehl `pwd` (print working directory).

9. Das Home-Verzeichnis ist auch in der Umgebungsvariable `HOME` gespeichert. Lasse dir gleich diesen Wert anzeigen.

   Jetzt kennst du schon die drei wichtigsten Umgebungsvariable, die für einen Benutzer wichtig sind. Es gibt allerdings noch viele mehr und man kann sich auch eigene anlegen.

10. Erstelle in deinem aktuellen Verzeichnis ein Verzeichnis `shell_testing`. Verwende dazu den Befehl `mkdir`. "`shell_testing`" soll als Kommandozeilenargument (kurz Argument) an den Befehl `mkdir` übergeben werden.

11. Wechsle in dieses Verzeichnis. Dafür gibt es den Befehl `cd`. Verwende diesen mit einer relativer Pfadangabe. Überprüfe wiederum, dass du dich in dem gewünschten Verzeichnis befindest.

12. Wenn du allerdings `which cd` eingibst, dann… Probiere es aus. Als Erklärung: Es gibt eben Befehle, die als Programme vorliegen und solche, die in der Shell eingebaut sind (builtins). Diese werden — im Gegensatz zu den Programmen — nicht als eigener Prozess gestartet.

13. Erstelle in deinem aktuellen Verzeichnis darin eine leere Datei `test1.txt`. Dazu verwende den Befehl `touch` und übergebe den gewünschten Dateinamen als Argument.

14. Überprüfe, dass diese Datei erstellt worden. Verwende dazu den Befehl `ls` ohne Argumente als auch mit dem Namen der erstellten Datei als Argument.

15. Editiere diese Datei mit einem Editor und schreibe in diese Datei einen beliebigen Text deiner Wahl. Schließe den Editor und finde die Größe der angelegten Textdatei heraus.

    Dazu verwendest du wieder den Befehl `ls`, aber diesmal wird eine Option übergeben. Optionen sind mit einem (kurze Option) oder mit zwei Bindestrichen (lange Option) gekennzeichnet. Fürs erste verwende die Option `-l`, also `ls -l`, wobei "l" für "long" steht, also soviel wie lange Ausgabe. Es handelt sich hierbei also um eine kurze Option.

    - Woran erkennst du bei dieser Ausgabe, ob es sich um ein Verzeichnis oder um kein Verzeichnis handelt?
    - Weiters findest du hier auch eine Folge von Zeichen ("r", "w", "x"), deren Bedeutung wir uns später noch ansehen werden.
    - Außerdem findest du auch noch eine Zahl, derzeit meist 1 oder 2. Auch diese Bedeutung werden wir uns noch später zu Gemüte führen.
    - Danach kommt noch dein Benutzernamen und deine Gruppe zu der du gehörst.
    - Dann kommt noch das Modifikationsdatum und der Dateiname.

16. Den Inhalt der gerade angelegten Datei "test1.txt" kannst du dir mittels `cat test1.txt` ausgeben lassen. Probiere es aus.

17. Unter Umständen willst du den Inhalt von "test1.txt" am Bildschirm nicht mehr sehen, dann kannst du den sichtbaren Inhalt deines Terminals mittel `clear` löschen.

18. Handelt es sich allerdings um eine Datei, die größer ist als der verfügbare Platz im Terminal, dann hast du mit dieser Art von Ausgabe keine Freude, da der Anfang nicht mehr zu sehen ist. Dann kommt ein sogenannter "Pager" ins Spiel. Es handelt sich dabei um ein Programm, dass zum Betrachten von langen Textdateien gut geeignet ist.

    Wir verwenden `less`, bei dem es sich um das Nachfolgerprogramm zu einem Programm `more` handelt. `more` kann "mehr" von einer Textdatei anzeigen und deshalb der Name.

    `less` kann aber mehr als `more`, aber… Informatiker eben `;-)`.

    Die Bedienung von `less` ist einfach:

    - Mittels der Cursor-Tasten kannst du navigieren,
    - Suchen indem du einen Schrägstrich (engl. slash) und das Suchwort gefolgt von der Taste `Enter` eigibst. Mittels Drücken von `n` kommst du zum nächsten Suchergebnis.

- Beenden kannst du durch Drücken von `q`.

19. Eine andere Option für `ls` ist `-a`. Probiere es einfach aus. Was siehst du zusätzlich? Dateien, die mit einem Punkt beginnen, werden standardmäßig nicht angezeigt. Diese werden also "versteckt".

    Für `-a` versteht der Befehl auch eine lange Optionsangabe: `ls --all`. Dies hat den gleichen Effekt wie `ls -a`. Probiere es einfach wieder aus. Wie du unschwer erkennen kannst steht `-a` für "all" und bedeutet, dass alle Dateien angezeigt werden sollen, also auch die versteckten Dateien.

20. Du weißt schon, dass die Datei `.` das aktuelle Verzeichnis angibt. Es handelt sich dabei um eine relative Pfadangabe. Was passiert bei `cd .`?

21. Was passiert bei `cd ..`? Teste dies und stelle den vorhergehenden Zustand des Arbeitsverzeichnisses wieder her: `cd -` hilft dir sicher weiter.

    Sowohl `.` als auch `..` findest du auch bei der Ausgabe von `ls -a`

22. Wechsle in das Home-Verzeichnis. Das kannst du mittels des `cd` Befehls ohne weiteres Argument erreichen. Probiere es aus und wechsle wieder zurück in das vorhergehende Verzeichnis.

23. Wechsle nochmals in das Home-Verzeichnis. Diesmal verwende die Tilde (`~`) als Argument für den Befehl `cd`. Wechsle wieder zurück in das vorhergehende Verzeichnis. D.h. du befindest dich wieder in deinem angelegtem Verzeichnis.

24. Lege eine versteckte Datei an und teste, ob diese auch wirklich versteckt ist.

25. Lösche nun diese versteckte Datei wieder aus dem aktuellen Verzeichnis. Dies geht mit dem Befehl `rm` und Angabe des Dateinamens als Kommandozeilenargument.

26. Kopiere die Datei `test1.txt` auf die Datei `test2.txt` im selben Verzeichnis. Dazu ist der Befehl `cp` wie geschaffen. Dieser erwartet sich jetzt allerdings zwei Kommandozeilenargumente, nämlich die Quelle und das Ziel. Also `cp test1.txt test2.txt` sollte das gewünschte Ergebnis bringen.

27. Ändere den Text der Datei `test2.txt` mit dem Editor ab. Überprüfe wiederum die Größen dieser Dateien.

28. Kopiere `test2.txt` nach `data.txt`. Erstelle mit `ls` eine Directory-Anzeige aller Dateien, die mit `test` beginnen. Verwende dafür das Wildcard-Zeichen `*` in der Form `ls test*`.

29. Lege weiters ein Verzeichnis `shell_testing2` im gleichen Verzeichnis an in dem sich auch `shell_testing` befindet. Das aktuelle Arbeitsverzeichnis darf **nicht** gewechselt werden! Verwende dazu `..` in der Pfadangabe.

30. Danach sind alle Dateien, die mit `test` beginnen, danach genau ein Zeichen kommt und abschließend mit `.txt` enden, nach `shell_testing2` zu kopieren. Verwende dafür das Wildcard-Zeichen `?`!

31. Verschiebe die Datei `data.txt` in das Verzeichnis `shell_testing2` (spezifiziert durch einen absoluten Pfad). Verwende dazu den Befehl `mv` und das Zeichen `~`. Überprüfe, ob die Datei wirklich in das Zielverzeichnis verschoben wurde.

32. Wechsle in dein Home-Verzeichnis. Verwende dazu den Befehl `cd` (ohne weitere Argumente).

33. Wechsle wieder in das Verzeichnis, in das du vorher gewesen bist. Ok, du weißt schon wie das geht.

34. Wechsle wieder in dein Home-Verzeichnis. In diesem Fall allerdings, indem du die Tatsache ausnützt, dass es sich um das übergeordnete Verzeichnis handelt. Danach lösche das gesamte Verzeichnis `shell_testing`. Verwende dazu `rm` mit der Option `-r`. "-r" steht für "recursive", also rekursiv.

35. Benenne das Verzeichnis `shell_testing2` nach `testing` um. Umbennen wird durch Verschieben realisiert. Also verschieben in das gleiche Verzeichnis unter einen anderen Namen.

36. Kopiere das gesamte Verzeichnis `testing` in ein Verzeichnis `testing.bak`. Verwende dazu die Option `-r`. Auch hier steht "r" wieder für "recursive".

37. Zum Löschen von Verzeichnissen gibt es noch das Kommando `rmdir`. Probiere gleich das Verzeichnis `testing.bak` mittels `rmdir` zu löschen. Was passiert?

    Aha, ist ja gut, aber wie kommt man zu Informationen *was* ein Kommando eigentlich tut? Verwende dazu z.B. das Kommando `man rmdir`. Die Bedienung funktioniert so wie die Bedienung von `less`.

    Gut, jetzt weißt du, dass es mit `rmdir` nicht funktioniert und auch warum es nicht funktioniert. Lösche das Verzeichnis trotzdem!

38. Nehmen wir an, dass wir ein Archiv des Verzeichnis `testing` erstellen wollen. Wir verwenden dazu das Kommando `tar` (soviel wie "tape archiver"), das aus einem Verzeichnis ein Archiv erstellen kann: `tar -cf testing.tar testing` erstellt (`-c` wie "create") eine Archivdatei (`-f` wie "file") mit dem Namen "testing.tar" aus dem Verzeichnis "testing". Probiere das gleich aus.

    Du siehst hier, dass mehrere kurze Optionen zusammengefasst werden können und nicht `tar -c -f testing.tar testing` geschrieben werden muss.

39. So, jetzt gibt es eine Archivdatei und in unserem Fall ist diese auch nicht groß, aber unter Umständen enthält diese zum Beispiel große Bilder. Bilder lassen sich oft auch gut komprimieren. Zum Komprimieren gibt es mehrere Möglichkeiten wie

das bekannte `zip`. Unter Unix/Linux verwendet man allerdings eher `gzip` oder `bzip2`.

Wir wollen jetzt `gzip` verwenden, um die Archivdatei zu komprimieren. Dazu gehen wir folgendermaßen vor: `gzip testing.tar`. Dann wird automatisch eine komprimierte Datei `testing.tar.gz` erstellt.

Du kannst dir jetzt ansehen um wieviel kleiner die neue Archivdatei gegenüber der alten Archivdatei ist.

40. Lösche das Verzeichnis `testing`. Jetzt gibt es nur noch die komprimierte Archivdatei und die gilt es zu dekomprimieren und zu entpacken, um wieder an das Verzeichnis `testing` zu kommen:

    ```
    gunzip testing.tar.gz
    tar -xf testing.tar
    ```

    Die Option "`-x`" steht für "extract".

41. Ok, funktioniert ja ganz gut, aber ist auch ein bisschen umständlich. Lösche die Archivdatei und erstelle eine neue mit dem folgenden Befehl `tar -czf testing.tgz testing`. Die Option "`-z`" bewirkt, dass das nach dem Erstellen des Archives das Programm `gzip` aufgerufen wird, um das Archiv zu komprimieren. Beachte, dass kein unkomprimiertes Archiv im Dateisystem angelegt wird.

    So, lösche wiederum das Verzeichnis `testing`.

42. Wie weiß man eigentlich *was* sich in einem Verzeichnis befindet? Dazu gibt es die Option "`-t`": `tar -tf testing.tgz`. Für mich bedeutet die Option "table of contents" und damit kann ich mir dies auch leicht merken. Beachte, dass `tar` erkennt, dass es sich um ein mit `gzip` komprimiertes Archiv handelt und automatisch `gunzip` aufruft! D.h. die Option `-z` ist in diesem Fall nicht notwendig, kann allerdings sehr wohl angegeben werden.

    Willst du explizit klarstellen, dass es sich um ein mittels `gzip` komprimiertes Archiv handelt, dann kannst du natürlich gerne auch die längere Version `tar -tzf testing.tgz` verwenden.

43. Beachte, dass der Dateinamen und im speziellen die Erweiterung keine Bedeutung unter Unix/Linux haben. D.h. ein Archiv muss nicht mit `.tar` oder `.tar.gz` oder `.tgz` enden. Es handelt sich hier nur um die am meisten verwendeten Dateinamenerweiterungen.

    Die Information steht alleine in der Datei selber und diese kann das Programm `file` auslesen. Probiere `file testing.tgz`!

44. Will man mehr Informationen bei einer beliebigen Aktion von `tar`, dann kann man die Aktion auch mit der Option `-v` (wie "verbose", d.h. geschwätzig) kombinieren, wie z.B. bei `tar -tvf testing.tgz`.

45. Auch das Entpacken geht wie schon gelernt: `tar -xf testing.tgz`.

46. Als allerletztes Programm werden wir uns `scp` ansehen. Es handelt sich um ein Programm um Dateien über das Netzwerk in sicherer Art und Weise zu übertragen.

    Nehmen wir an, dass du dir die Datei `testing.tgz` als Backup am `edvossh.htlwrn.ac.at` ablegen willst, dann kannst du dies so erledigen: `scp testing.tgz edvossh.htlwrn.ac.at:Private`. Damit wird diese Datei in das Verzeichnis "Private" deines HOME-Verzeichnisses am Rechner `edvossh.htlwrn.ac.at` kopiert. Bei `Private` handelt es sich um einen relativen Pfad. Wolltest du einen absuluten Pfad verwenden, dann geht das in diesem Fall so: `scp testing.tgz edvossh.htlwrn.ac.at:/home/students/i99001/Private`, vorausgesetzt dir ist die Matrikelnummer zugeordnet.

    Du siehst, dass der Doppelpunkt absolut notwendig ist, denn nur daran erkennt `scp`, dass es sich bei `edvossh.htlwrn.ac.at` um einen Rechner handelt. Probiere aus was passiert, wenn du `scp testing.tgz edvossh.htlwrn.ac.at` eingibst. Was ist passiert? Nicht das was du wolltest?

    Wolltest du vielleicht `scp testing.tgz edvossh.htlwrn.ac.at:`? Was ist hier passiert?

Ende.

3 Übungszweck dieses Beispiels

- Terminal und Shell (`bash`) verwenden
- `C-a`, `C-e`, `C-k`, `C-d`, `M-d`, `C-f`, `C-b`, `M-f`, `M-b`, `M-y`
- `C-c`
- Shell-Befehle wiederholen und lernen
  - `ps`, `which`, `id`, `echo`
  - `pwd`, `mkdir`, `cd`
  - `touch`, `cat`, `clear`, `less`, `ls`
  - Pfadangaben: absolut, relativ, `.`, `..`, `~`
  - Kommandozeilenarguemente, Optionen (kurz, lang)
  - Wildcard-Zeichen `*` und `?`
  - `rm`, `cp`, `mv`, `rmdir`
  - `man`
  - `tar`, `gzip`, `gunzip`, `file`
  - `scp`
- Umgebungsvariable `HOME`, `USER`, `PATH`

- Linux-Bedienung üben
- Verwendung eines Texteditors