bw | HPC – C5

Einführungskurs bwHPC-C5

Grundlagen Linux Shell-Programmierung

Rafael Doros und Rainer Keller

Förderträger: Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

www.bwhpc-c5.de

Agenda

Agenda

Hochschule für Technik Stuttgart

- **Erste Schritte**
- Das Dateisystem
- Dateiverwaltung
- Kommandos
- Shell Skripte
- Module

Erste Schritte

Hochschule für Technik Stuttgart

Wissenswertes für Linux Einsteiger:

- Sicherheitsabfragen sind häufig nicht existent! Damit bleiben gelöschte Dateien auch gelöscht.
- Linux arbeitet case sensitive, das ist gerade am Anfang häufig etwas ungewohnt.
- Arbeit auf der Kommandozeile (Shell) ist üblich.
- Ein Kommando ist entweder ein (vorhandenes) Programm oder ein „eingebauter“ Befehl.
- Einem Kommando kann man Parameter übergeben.
- Spezielle Parameter, die der Steuerung von Kommandos dienen, nennt man auch Optionen.
- Angaben für Optionen nennt man Argumente.
- `Kommandoname` und zugehörige `Parameter` werden auf den Folien durch `eine andere Schriftart` hervorgehoben.

Erste Schritte

Hochschule für Technik Stuttgart

- Programme werden in einer sog. Shell gestartet.
- Eine Shell hat aber auch „eingebaute“ Kommandos.
- Die Shell ist die niedrigste, systemnahe Software, mit der ein User interagiert.
- Shells kann man programmieren!
- Es gibt unter Unix/Linux eine Menge an (alten und nicht ganz so alten) Shells.
- Beispiele
  - `Bourne Shell (sh)`
  - `C Shell (csh)`
  - `Korn Shell (ksh)`
  - `Z-Shell (zsh)`
  - `Bourne Again Shell (bash)`
- Die wohl am häufigsten benutzte ist vermutlich die `Bash`.
- Daher konzentrieren wir uns hier auf die `Bash` Shell.

Erste Schritte

Hochschule für Technik Stuttgart

- `echo` → Gibt eine Zeile Text aus.
  - `echo "Hallo Welt" : Hallo Welt`

- Echo kann nicht nur einfach Text ausgeben, sondern auch Variableninhalte.
  - `var="Hallo"` → Variable `var` auf den Wert `Hallo` setzten.

    `echo $var : Hallo` → Den Wert der Variable `var` ausgeben.

    oder

    `echo ${var} : Hallo`

Erste Schritte

- Unter Linux werden Variablen sehr häufig verwendet.
- Viele Variablen sind standardmässig vorhanden und belegt.

- Beispiel für Standardvariablen
  - `echo $USER : ?` → Die Variable USER.
  - `echo ${LOGNAME} : ?` → Und die Variable LOGNAME.

Erste Schritte

- Es gibt viele Kommandos und diese haben sehr häufig wiederum viele Parameter.
- Damit man sich diese nicht alle merken muss, existiert zu fast jedem Kommando eine Beschreibung, eine sog. ***man****ual page*.

- `man` → Gibt die Anleitung zu einem Kommando aus.
  - `man man` → Die Beschreibung von dem Kommando `man`.

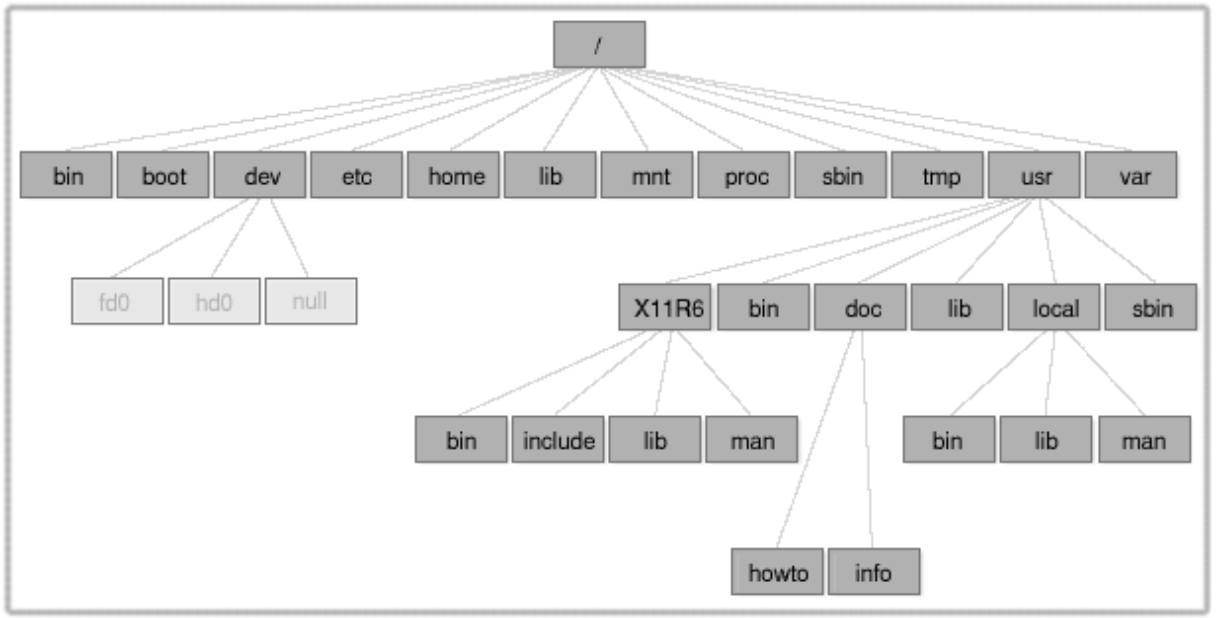
- Für eine schnelle Übersicht reicht häufig die Hilfsausgabe des Kommandos selbst. Diese ist meistens mit dem Parameter `--help` erreichbar.
  - `man --help`

Agenda

Hochschule für Technik Stuttgart

- Erste Schritte
- **Das Dateisystem**
- Dateiverwaltung
- Kommandos
- Shell Skripte
- Module

Das Dateisystem



Quelle: http://upload.wikimedia.org/wikibooks/de/c/cf/Filesystem.gif
GNU-Lizenz für freie Dokumentation

- Dateien und Verzeichnisse werden als Baum „organisiert“.
- Es existieren keine Laufwerke (wie z.B. in Windows). Diese sind Teil des sog. Verzeichnisbaums.
- Einige Verzeichnisse werden durch den *Filesystem Hierarchy Standard (FHS)* beschrieben.
- Das Trennzeichen für Pfade unter Linux ist das / Zeichen.

Das Dateisystem

Hochschule für Technik Stuttgart

- `tree` → gibt die Verzeichnisse und Dateien strukturiert auf der Konsole aus. (Achtung: nicht auf jedem System installiert!)
- `tree -d` → Beispiel Ausgabe

```
├── boot
│   ├── grub
├── dev
│   ├── block
│   ├── bus
│   │   └── usb
├── etc
│   ├── NetworkManager
│   │   ├── VPN
│   ├── acpi
│   │   └── events
│   ├── conf.d
├── home
│   ├── user1
```

Das Dateisystem

Hochschule für Technik Stuttgart

- `pwd` → ***p**rint **w**orking **d**irectory;* Zeigt das aktuelle Verzeichnis in dem man sich gerade „befindet“.
- `ls` → ***l**i**s**t;* Listet die Verzeichnisse und Dateien im einem Verzeichnis auf.
- `cd` → ***c**hange **d**irectory;* Damit kann man zwischen den Verzeichnissen wechseln.

- Beispiel
  - `pwd : /home/user1` → Der aktuelle Standort innerhalb des Verzeichnisbaumes.
  - `ls : Desktop bsp.txt` → Auflistung des Inhalts des aktuellen Verzeichnisses.
  - `cd Desktop :` → Wechseln in das Desktop Verzeichnis. Aber, wie wieder „hoch“ kommen?
  - `pwd : /home/user1/Desktop`

Das Dateisystem

Hochschule für Technik Stuttgart

- Jedes Verzeichnis enthält zwei besondere „Verzeichnisse“.
  - `.` → Ein Verweis auf das aktuelle Verzeichnis.
  - `..` → Ein Verweis auf das nächst höhere Verzeichnis.
  - Dadurch sind auch zum Standort relative Pfadangaben möglich.
- Beispiele für absolute / relative Pfadangabe
  - `pwd : /home/user1/Desktop` → Aktueller Standort.
  - Wechseln in das nächst höhere Verzeichnis
  - → durch absolute Pfadangabe
    - `cd /home/user1`
    - `pwd : /home/user1`
  - → oder durch relative Pfadangabe
    - `cd ../`
    - `pwd : /home/user1`
  - Was würde stattdessen `cd ./../` oder `cd ../../user1/` machen?

Das Dateisystem

- Variablen
  - `$HOME` und `~` → Benutzerverzeichnis.
  - `$PWD` → Aktuelles Verzeichnis.
  - `$OLDPWD` → Das vorherige Verzeichnis.
  - `$PATH` → Verzeichnisse die bei einem Programmaufruf ggf. durchsucht werden. In der angegeben Reihenfolge.

- Beispiel
  - `echo $HOME : /home/user1` → Benutzerverzeichnis.
  - `echo ~ : /home/user1` → Benutzerverzeichnis.

- `env` → ***env****ironment;* Zeigt eine Liste der Umgebungsvariablen. Kann zum Ändern der Umgebung benutzt werden. (Und ist zum Ausführen von Programmen mit abweichender Umgebung fähig.)

Das Dateisystem

Hochschule für Technik Stuttgart

- `file` → Erkennt den Typ einer Datei.
  - Die Identifikation wird anhand von sog. *„magic numbers“* durchgeführt,
  - d.h. Dateiendungen sind nicht von Bedeutung!
- `more` → „Seitenweise“ (kein zurückscrollen) Anzeige von Textdateien.
  - Heute häufig durch less „ersetzt“.
- `less` → Anzeige von Textdateien.
- `du` → Ermittelt den Platzbedarf von Verzeichnissen und Dateien.
  - `du -sh .` → Verbrauch des aktuellen Verzeichnisses.
- `cat` → Den Inhalt einer Datei ausgeben.

Agenda

Hochschule für Technik Stuttgart

- Erste Schritte
- Das Dateisystem
- **Dateiverwaltung**
- Kommandos
- Shell Skripte
- Module

Dateiverwaltung

Hochschule für Technik Stuttgart

- Dateinamen haben max. 256 Zeichen (und benötigen keine Erweiterung für den Dateityp).
- Es wird zwischen Groß,- und Kleinschreibung unterschieden (case sensitive), d.h. `Dateiname` ist nicht das gleiche wie `dateiname`!
- Dateien mit einem . am Anfang, zeigt `ls` nicht an, da diese als "versteckt" gelten (z.B. Konfig. Datei `~/.bashrc`).

- Namen können bestehen aus (Auswahl)
  - Großbuchstaben
  - Kleinbuchstaben
  - Ziffern
  - Bindestrich, Unterstrich, Punkt

Dateiverwaltung

Hochschule für Technik Stuttgart

- Sonderzeichen (eine Auswahl)
  - `/` → Verzeichnistrenner
  - `?` → steht für ein beliebiges Zeichen
  - `*` → beliebige Zeichenkette
  - `[ ]` → ein Zeichen in den Klammern kann ersetzt werden
  - `[A-Z]` → Bereiche die ersetzt werden
  - `\` → Backslash ist Escape Character für Sonderzeichen
  - `<` , `>` → Umlenkung von Ein- und Ausgabe
  - `|` , `||` , `;` , `&&` → Verkettung von Kommandos
  - `(` , `)` → Zusammenfassung von Kommandos zu Gruppen
  - `"` , `'` → Quotierung

Dateiverwaltung

Hochschule für Technik Stuttgart

- `touch` → Ändert den Zeitstempel einer Datei. Kann auch dazu benutzt werden um (leere) Dateien anzulegen.
  - `touch datei.txt`
  - `ls -lh :`

    ```
    -rw-r--r-- 1 user1 users   0 16. Nov 15:30 datei.txt
    ```

- `mkdir` → ***mak**e **dir**ectories;* Ein Verzeichnis anlegen.
  - `mkdir dirname`
  - `mkdir dir1/dir2` → Legt dir2 im Verzeichnis dir1 an.

- `diff` → Zeigt die Unterschiede zwischen zwei Dateien oder Verzeichnissen auf.
  - `diff datei1 datei2`
  - `diff dir1 dir2`

Dateiverwaltung

Hochschule für Technik Stuttgart

- `cp` → ***cop****y;* Kopieren von Dateien oder …
  - `cp file1 file2`
  - `cp file1 dir1/file1`
  - `cp file1 dir1/file2` → Unterschied?
  - `cp -r` → … Verzeichnissen.
    - `cp -r dir1 dir2/dir3` → auch klar?
- `mv` → ***mov****e;* Verschiebt Dateien oder Verzeichnisse …
  - `mv dir1 dir2/` → `dir1` in das Verzeichnis `dir2` verschieben.
- … oder benennt Dateien oder Verzeichnisse um.
  - `mv dir1 dir2/dir3` → Verschieben mit Umbenennen.
  - `mv file1 file2` → Umbenennen.
- Es existiert kein eigenes Kommando für das Umbenennen!

Dateiverwaltung

Hochschule für Technik Stuttgart

- `rm` → Löschen von Dateien oder …
  - `rm -r` → … Verzeichnissen.
  - `rm -rf` → Löscht auch Verzeichnisse. Unterschied?
- `rmdir` → Alternative zum Löschen von Verzeichnissen.

Agenda

Hochschule für Technik Stuttgart

- Erste Schritte
- Das Dateisystem
- Dateiverwaltung
- **Kommandos**
- Shell Skripte
- Module

Kommandos

Hochschule für Technik Stuttgart

- `apropos` → Durchsucht die Datenbank mit den Kurzbeschreibungen (whatis) nach dem gegebenem Stichwort.
  - `apropos password :`
    - `chage (1)      - change user password expiry information`
    - `chgpasswd (8) - update group passwords in batch mode`
    - `...`
- `which` → Zeigt den kompletten Pfad zu einem Programm.
  - `which vim : /usr/bin/vim`
- `tail` → Die letzten Zeilen einer Datei ausgeben.
  - `tail -n 2 .bash_history :`
    - `cd /usr/share/eclipse/dropins/`
    - `cd -`
- `head` → Die ersten Zeilen einer Datei ausgeben.
    - `head -n 1 .bash_history :`
    - `Passwd`

Kommandos

- `cut` → Teilt die Zeilen einer Datei in Bereiche auf und erlaubt Zugriff auf die einzelnen Teile.
  - `cut -d' ' -f1 .bash_history` → Teilt die Zeilen in einzelne Wörter auf und gibt das erste Wort aus.
- `wc` → Gibt die Anzahl der Zeilen, Wörter und Byte einer Datei aus.
  - `wc filename : 1695 4804 43904 filename` → 1695 Zeilen, 4804 Wörter und 43904 Byte.
- `sort` → Sortiert die Zeilen einer Textdatei.
- `uniq` → Doppelte Zeilen eliminieren oder berichten.
- `grep` → Gibt die Zeilen einer Dateien aus, die ein bestimmtes Muster enthalten. Kann auch Rekursiv alle Dateien einer Verzeichnishierarchie (mit einstellbarer Tiefe) durchsuchen.
  - `grep -i cd .bash_history :`
    - `cd Desktop`
    - `find $HOME -iname "*cdt*" -type f`

Kommandos

- `find` → Eine Verzeichnishierarchie anhand von Kriterien nach Dateien oder Verzeichnissen durchsuchen.
- Dabei werden viele Arten von Kriterien (Alter, Name, Typ, Änderungsdatum, Größe usw.) für Dateien unterstützt
- Es ist möglich auf die gefundenen Dateien ein Kommando auszuführen (z.B. löschen).
- Es können mehrere Kriterien verknüpft (UND, ODER) oder negiert werden. Standardmässig sind die Kriterien mit UND verknüpft
- Beispiel
  - `find $HOME -iname "*cdt*" -type f :`
    - `/home/user/Desktop/Downloads/eclipse-cdt-pkgbuild.tar.xz`

Verknüpfung von Kommandos

- Linux kennt drei Standardein- und ausgaben.
  - `stdout` → Standard Ausgabe (z.B. der Bildschirm).
  - `stdin` → Standard Eigabe (z.B. Tastatur oder eine Datei).
  - `stderr` → Standard Fehlerausgabe.
- Viele Linuxkommandos können diese nutzen.

- `|` → *pipe;* Damit ist es möglich die Ausgabe von einem Kommando an ein anderes weiterzuleiten.
  - `history | sort | less` → die Ausgabe von `history` an `sort` weitergeben. Dann die sortierte Ausgabe als Eingabe an less weiterleiten.

Verknüpfung von Kommandos

- Anstatt die Ausgabe an ein weiteres Kommando weiter zu geben, kann man diese auch in eine Datei umleiten.
- `>` → Umleiten von `stdout` in eine Datei. Dabei die Datei neu anlegen oder überschreiben.
- `2>` → Umleiten von `stderr` in eine Datei. Dabei die Datei neu anlegen oder überschreiben.
- `2>&1` → Umleiten von `stdout` und `stderr` in eine Datei. Dabei die Datei neu anlegen oder überschreiben.
- `>>` → Die Ausgabe umleiten und an die Datei anhängen.
- `<` → Die Eingabe umlenken.
- Beispiel
  - `history | sort >> filename.txt` → Sortierte Kommandohistorie an die Datei `filename.txt` anhängen.

    `nano -w filename.txt` → Die Datei `filename.txt` bearbeiten.

Verknüpfung von Kommandos

- Kommandos kann man, in Abhängigkeit vom Ausgang des vorhergehenden, nacheinander ausführen.
- `&&` → Das nachfolgende Kommando ausführen, wenn das vorherige erfolgreich war.
- `||` → Das nachfolgende Kommando ausführen, wenn das vorhergehende nicht erfolgreich war.
- `;` → Die Kommandos nacheinander ausführen, unabhängig vom Ausgang des vorherigen.

- Beispiel
  - `gcc -Wall -O2 -o app app.c && ./app` → Programm `./app` wird ausgeführt, wenn gcc erfolgreich (Rückgabewert 0) war.

Verknüpfung von Kommandos

- `tee` → Damit kann man die Ausgabe eines Kommandos gleichzeitig in eine Datei und auf die Standardausgabe lenken.

- Beispiel
  - `history | sort | tee sortiert.txt | less` → Kommandohistorie sortieren, in die Datei `sortiert.txt` schreiben und an `less` weitergeben.

Kommandos II

Hochschule für Technik Stuttgart

- `tar` → ***t**ape **ar**chiver;* Wurde ursprünglich entwickelt um Daten auf Bänder zu schreiben. Heute wird es benutzt, um Daten in einer Datei zu archivieren.
- Wichtig dabei ist, daß tar auch die Rechte sichert, weswegen es unter Linux z.B. gegenüber zip bevorzugt wird.
  - `tar -cvf tarfile.tar directory/` → Archiviere das Verzeichnis `direcory` in die Datei `tarfile.tar`.
- Linux kennt diverse (alle verbreiteten) komprimierungs Tools.
  - `zip` → Ein unter Windows sehr bekanntes Format.
  - `bzip2` → Bis vor kurzen das Standardformat unter Linux.
  - `xz` → Entwickelt sich gerade zum neuen Standardformat.
    - Es gibt einige abgeleitete Tools die `xz` mit häufig verwendeten Paramentern aufrufen.
    - `unxz` → Entspricht `xz --decompress`.
    - `xzcat` → Entspricht `xz --decompress -stdout`.
- uvm.

Kommandos II

Hochschule für Technik Stuttgart

- `ssh` → Damit kann man sich mit einem entfernten Computer zum arbeiten verbinden.
  - `ssh username@remote-server` → Verbindet sich mit dem Computer `remote-server` mit dem Benutzernamen `username`.

- `scp` → ***s**ecure **c**o**p**y;* Kommando um Dateien oder Verzeichnisse zu oder von einem entfernten Rechner zu übertragen.
  - `scp file user@remote-server:/home/user/` → Überträgt die Datei `file` auf `remote-server` in das home-Verzeichnis von user.
  - `scp -r directory user@remote-server:/home/user/` → Überträgt das Verzeichnis `direcory` auf `remote-server`.

Sonstiges

Hochschule für Technik Stuttgart

- `date` → Gibt Datum und Zeit aus.
  - `date -R : 2015-11-14` → Ausgabe im iso-8601 Format.
- `alias` → Es können zusätzliche Namen für (z.B. häufig benutzte) Kommandos (samt Parameter) angelegt werden.
  - `lll : Kommando nicht gefunden.`
  - `alias "lll=ls -lhA"` → Ein Alias anlegen.
  - `lll` → Was passiert?
  - `alias : alias lll='ls -lha'` → Ohne Parameter werden alle angelegten Aliase angezeigt.
- `export` → Damit wird veranlasst, dass eine Variable an die aufgerufenen Programme (Kindprozesse) weitergegeben wird.
  - `var1=foo`
  - `export var1`
  - `export var2=$var1`
- `$HOME/.bashrc` → Das Skript wird beim Einloggen ausgeführt.

Agenda

Hochschule für Technik Stuttgart

- Erste Schritte
- Das Dateisystem
- Dateiverwaltung
- Kommandos
- **Shell Skripte**
- Module

Shell Skripte

Hochschule für Technik Stuttgart

- Es gibt verschiedene Shells, die unterschiedliche Programmiersyntax verwenden.
- Dabei ist die `Bash` vermutlich am weitesten verbreitet.
- Ein Shellskript ist eine einfache Textdatei.
- Häufig mit der Endung `.sh` (um es für den Benutzer kenntlich zu machen, nicht für das Betriebssystem).
- Die Textdatei beginnt immer mit `#!/bin/bash` . Damit wird die Shell gesetzt, mit der das Skript ausgeführt wird.
- Um das Skript direkt ausführen zu können, müssen die Ausführungsrechte (e**x**ecute) gesetzt sein.

Shell Skripte

Hochschule für Technik Stuttgart

- `$var oder ${var}` → Variablenzugriff.
- `#` → Kommentar.
- `var=Wert` → Die Variable `var` wird auf `Wert` gesetzt (Zuweisung).
- Bei Verknüpfungen mit Variablen immer mit der `${…}` Schreibweise zugreifen.
  - `$varWert` → `${var}Wert`

- Ausgesuchte Standardvariablen
  - `$0` → Skriptname mitsamt dem kompletten Pfad.
  - `$#` → Anzahl der mitgegebenen Parameter.
  - `$1,$2,…,$n` → Die mitgegebenen Parameter.
  - `$*` → Alle Parameter als Zeichenkette.
  - `$?` → Enthält nach dem Beenden des Skripts den sog. Exit Status.

Shell Skripte

Hochschule für Technik Stuttgart

- `basename` → Entfernt bei einem Pfad den Verzeichnisanteil.
- `dirname` → Liefert bei einem Pfad den Verzeichnisanteil.
- `read` → Ermöglicht das Einlesen von Werten aus der Standardeingabe.

- `+,-,*,/` → Addition, Subtraktion, Multiplikation, Division
- `%` → Modulo
- `>>,<<` → Bitweise Verschiebung
- `&,^,|,~` → Bitweise UND, exkl. ODER (XOR), ODER, Negation

Shell Skripte

- `test` od. `[ ... ]` → Prüfung von Bedingungen (Zahlen, Zeichenketten und Zuständen einer Datei)

- Zustand einer Datei
  - `-e` → Datei existiert
  - `-d` → Datei ist ein Verzeichnis
  - `-f` → Datei ist ein sog. 'Plain File'
  - `-r` → Datei ist lesbar
  - `-s` → Datei ist größer 0 Byte
  - `-w` → Datei ist beschreibbar
  - `-x` → Datei ist ausführbar

Shell Skripte

- Zeichenketten
  - `-n` → `${var}` ist nicht leer (länger als 0 Zeichen).
  - `-z` → `${var}` ist leer (0 Zeichen lang).
  - `=` → Gleichheit
  - `!=` → Ungleichheit
- Empfehlung: Variablen und Zeichenketten in doppelte Anführungszeichen " setzen.

- Zahlen
  - `-eq` → **eq**`ual;` Gleichheit
  - `-ne` → **n**`ot` **e**`qual;` Ungleichheit
  - `-ge` → **g**`reater` **e**`qual;` Größer oder gleich
  - `-gt` → **g**`reater` **t**`han;` Größer
  - `-le` → **l**`ess` **e**`qual;` Kleiner oder gleich
  - `-lt` → **l**`ess` **t**`han;` Kleiner

Shell Skripte

Hochschule für Technik Stuttgart

- Beispiel für eine `if`-Bedingung.
  - ```
    #!/bin/bash
    if [ -e ${HOME}/.bashrc ];
    then
            echo ".bashrc exists!"
    else
             echo "no .bashrc!"
    fi
    ```
- Für innere Bedingungen wird `elif/then` verwendet.
    ```

Shell Skripte

- Beispiel für eine `for`-Schleife.
  - ```
    #!/bin/bash
    dummy=(one two three four)     # Array!

    for u in ${dummy[*]}
    do
            echo $u
    done
    ```

- Auch mal mit `$dummy[*]` versuchen anstatt mit `${dummy[*]}`. Was passiert dann?
    ```

Shell Skripte

- Es gibt auch die `while`- und die `until`-Schleife.
- Auch die `case` Bedingung existiert.

- Beispiel für eine `case`-Bedingung.
  - ```
    #!/bin/bash
    read answer

    case $answer in
        yes) result=0;;
        no)  result=1
             ;;
        *)   echo "Unknown input!"
             result=-1;;
    esac
    ```
    ```

Agenda

Hochschule für Technik Stuttgart

- Erste Schritte
- Das Dateisystem
- Dateiverwaltung
- Kommandos
- Shell Skripte
- **Module**

Module

Hochschule für Technik Stuttgart

- Das Modulesystem erlaubt das Bereithalten und die kontrollierte Nutzung von verschiedenen Versionen (sowohl Versionsnummer als auch bzgl. der Abhängigkeiten) einer Software.

- Die Umgebung wird für die benötigte Software dynamisch angepasst. Die dafür notwendigen Informationen sind in sog. `modulefiles` hinterlegt.

- Allgemeine Informationen zum Modulesystem

  → `module help`

Module

- `module avail` → Alle verfügbaren Module anzeigen lassen.

```
--------------------------- /opt/bwhpc/common/modulefiles ----------------------
bio/augustus/3.2.3        devel/python/3.4.1
devel/ddt/6.1.2           phys/alps/2.3.0-python_numpy-1.11.2-hdf5-1.8-intel-16.0
devel/forge/5.0.1         phys/meep/1.3-openmpi-1.8-gnu-4.9
```

- `module avail devel` → Alle verfügbaren Module in der Kategorie `devel` anzeigen lassen.

```
--------------------------- /opt/bwhpc/common/modulefiles ----------------------
devel/python/3.4.1        devel/ddt/6.1.2              devel/forge/5.0.1
```

- `module avail devel/ddt` → Alle verfügbaren Versionen von `ddt` in der Kategorie `devel`.

Module

- `module help modulename` → Die Hilfsseite für das angegebene Module.
- `module show modulename` → Genauere Informationen zum Module, wie z.B. Variablen und Pfade die gesetzt/geändert werden.
- `module load modulename` → Lädt die zum Module gehörende Software.
- `module list` → Zeigt alle geladene Module an.

Module

Hochschule für Technik Stuttgart

- `module unload modulename`

  `module remove modulename` → Entfernt die zum Module gehörende Software. Kann bei Abhängigkeiten zu Inkonsistenzen führen.

- `module swap modulename_new modulename_old` → Ist eine Kombination aus `unload` und `load`.

- `module purge` → Entfernt alle geladenen Module.

Module

- Moduledateien sind in Kategorien eingeteilt.
- Aus den Modulenamen kann man die Abhängigkeiten ableiten.

- Beispiel `mpi/openmpi/2.1-gnu-7.1`
  - Kategorie/Softwarename/Version-Attribute-Abhängigkeiten

    → `OpenMPI` Paket in der Version `2.1` kompiliert mit (und abhängig von) `gcc 7.1`.

    → Attribute können Angaben sein, wie z.B. einfache oder doppelte Genauigkeit.

Dokumentation

Hochschule für Technik Stuttgart

- Linux

  → http://openbook.rheinwerk-verlag.de/linux/

- Shell Programmierung

  → http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

  → http://www.tldp.org/LDP/Bash-Beginners-Guide/html/

  → http://openbook.rheinwerk-verlag.de/shell_programmierung/

- Module

  → http://www.bwhpc-c5.de/wiki/index.php/BwUniCluster_Environment_Modules

- Batch jobs

  → http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs

Geschafft!