Utilities

- find — Suchen nach Dateien
- grep — Filtern nach einem Suchbegriff
- file — Dateiarten auflisten
- cmp, comm — Vergleichen von Dateien
- du — Anzeige des belegten Festplattenplatzes
- join, split — Aufteilen und Zusammenführen von Dateien
- crypt — Verschlüsseln von Dateien
- compress — Komprimieren von Dateien
- tar — Archivieren von Dateien

Utilities

- finger — Aktive User anzeigen lassen
- ftp — Übertragen von Dateien zwischen Rechnern
- telnet — Terminal auf anderem Rechner öffnen
- mail — Elektronische Mails verschicken und empfangen
- talk — Telefonieren über Terminal
- write — Ausgabe von Nachrichten auf Terminals
- time — Ermittlung von Zeitverbrauch für Kommandos
- at, cron, crontab — Zeitgesteuerter automatischer Start von Kommandos

B5 Utilities - find (1)

- $ find *path bedingung(en)*

  `find` durchsucht den angegebenen Pfad *path* nach Dateien, für die die *bedingung(en)* erfüllt sind. **`find`** aufgerufen ohne Parameter listet, vom aktuellen Verzeichnis ausgehend, alle Dateien auf (einschließlich Unterverzeichnisse). Das Durchsuchen erfolgt rekursiv für alle (Sub-)Directories

- Abkürzungen für numerische Argumente:

`n`	genau n
`+n`	mehr als n
`-n`	weniger als n

B5 Utilities - find (2)

- Optionen (Bedingungen)

  `-print`
  Ausgabe auf Standard Output.

  `-name` ***`datei_name`***
  Bedingung erfüllt, wenn *datei_name* gefunden wird.

  `-user` ***`login-kennung`***
  Bedingungen erfüllt, wenn Datei gefunden wird, die dem Benutzer *login-kennung* gehört.

  `-size` ***`n`***`[c]`
  Bedingung erfüllt für Dateien deren Größe n Blöcke bzw. n Bytes ist (für *n*c).

  `-mtime` ***`n`***
  Bedingung erfüllt für Datei, die vor n Tagen zum letzten Mal modifiziert wurde.

  `-newer` ***`dateiname`***
  Bedingung erfüllt für Datei, die jünger ist als das Modifikationsdatum von ***`dateiname`***

- Beispiele:

```
$ find . -mtime -3 -print
```

gibt alle Dateien aus, die vor weniger als 3 Tagen geändert wurden (ausgehend vom aktuellen Verzeichnis (.))

```
$ find . -name 't*' -print
```

gibt alle Dateien aus, die mit „t“ beginnen (ausgehend vom aktuellen Verzeichnis (.))

- $ grep *options pattern files*

grep durchsucht die angegebenen Dateien *files* nach dem Muster *pattern*.

- Wichtige Optionen:

-f file
Das Muster aus der Datei file nehmen.
-h
Keine Ausgabe von Dateinamen.
-i
Keine Unterscheidung von Groß- und Kleinbuchstaben.

- Beispiel:

```
$ grep -i Banane -h obst
```

Durchsucht die Datei *obst* nach dem Wort *Banane,* ignoriert Groß-/Kleinschreibung und unterdrückt den Dateinamen bei der Ausgabe.

- $ file *options files*

  `file` gibt den dateityp von *files* an.

- Synopsis:

  If file appears to be a text file, file examines the first 512 bytes and tries to determine its programming language. If file is an executable a.out, file prints the version stamp, provided it is greater than 0. If file is a symbolic link, by default the link is followed and file tests the file to which the symbolic link refers.

  By default, file uses /etc/magic to identify files that have a magic number. A magic number is a numeric or string constant that indicates the file type. See magic(4) for an explanation of the format of /etc/magic.

- Wichtige Optionen:

  `-c`
  Überprüfen des magic file auf Formatfehler. Dies wird aus Effizienzgründen normalerweise nicht gemacht.

  `-h`
  Symbolische Links werden nicht überprüft.

  `-f ffile`
  Die Datei **`ffile`** enthält eine Liste der zu untersuchenden Dateien.

  `-m mfile`
  Verwenden der alternativen Datei **`mfile`** anstelle der Datei **`/etc/magic`**.

- Beispiel:

```
$ file *
```

  Bestimmt die Dateitypen aller Dateien im aktuellen Verzeichnis.

- $ cmp *options file1 file2*

  cmp vergleicht **allgemeine** Dateien *file1* und *file2* byteweise und gibt die Unterschiede aus.

- Wichtige Optionen:

  -i Keine Unterscheidung von Groß- und Kleinbuchstaben.

- $ comm *options file1 file2*

  comm vergleicht **zeilenweise** zwei sortierte Dateien.

  With no options, **comm** produces three column output. Column one contains lines unique to file1, column two contains lines unique to file2, and column three contains lines common to both files.

- Wichtige Optionen:

  -1, -2, -3 Suppress printing of the corresponding columns.

- $ du *options*

  du gibt die Speicherbelegung auf der Festplatte ab dem aktuellen Verzeichnis aus.

- Wichtige Optionen:

  -a, --all
  Angabe des belegten Speicherplatzes für alle Dateien und nicht nur für Verzeichnisse.

  -k, --kilobytes
  Ausgabe der Dateigröße in Kilobytes.

  -s, --summarize
  Nur die Summe ausgeben.

- Beispiel:

  $ du

  mit unterschiedlichen Optionen vom Homeverzeichnis aufrufen.

B5 Utilities - join

- $ join *options file1 file2*

  `join` gibt Zeilen mit **gemeinsamen** Feldern der Dateien *file1* und *file2* aus. Die Dateien müssen nach den Felder sortiert sein.

- Beispiel:

  `$ join obst obst`
  da beide Dateien identisch sind, gibt ein **`join`** den Inhalt der Dateien aus.

  `$ join obst gemuese`
  da beide Dateien in keiner Zeile übereinstimmen, ist die Ausgabe leer.

B5 Utilities - split

- $ split *options infile outfileprefix*

  `split` teilt die Datei **`infile`** in gleichgroße Teile und erzeugt neue Dateien mit dem Prefix **`outfileprefix`**. Dieser Prefix ist optional.

- Wichtige Optionen:

  `-lines, -l lines, --lines=lines`
  Anzahl **`lines`** Zeilen der Inputdatei in die Outputdatei schreiben.

  `-b bytes[bkm], --bytes=bytes[bkm]`
  Anzahl **`bytes`** Bytes der Inputdatei in die Outputdatei schreiben. Bytes ist eine natürliche Zahl, der von einem Zeichen für die Größenangabe folgen kann.
  - b 512-byte blocks.
  - k 1-kilobyte blocks.
  - m 1-megabyte blocks.

- Beispiel:
  `$ split -l 2 obst`
  teilt die Datei obst in mehrere kleine Dateien mit jeweils zwei Zeilen.

B5 Utilities - crypt

- $ crypt *password infile oufile*

  `crypt` verschlüsselt die Datei **`infile`** mit Hilfe des Paßworts **`password`** und schreibt das Ergebnis nach **`outfile`**. Ein nochmalige Aufrufen mit demselben Paßwort entschlüsselt die Datei wieder (Achtung: Paßwort nicht vergessen!).

- Beispiel:

```
$ crypt test <obst >obst.crpt
```

  verschlüsselt die Datei **`obst`** und schreibt das Ergebnis nach **`obst.crpt`**.

B5 Utilities - compress, uncompress

- $ compress *options files*
  `compress` komprimiert die Datei **`files.`** Die komprimierten Dateien erhalten die Endung **`.Z`**.

  $ uncompress *options files*
  `uncompress` dekomprimiert die Datei **`files.`** Die Endungen **`.Z`** entfallen.

- Wichtige Optionen:

  `-c`
  Ausgabe auf die Standardausgabe; keine Dateien werden verändert und keine komprimierten Dateien werden erzeugt.

  `-v`
  Verbose. Ausgabe der Kompressionsrate auf Standarderror.

- Beispiel:

```
$ compress *
$ uncompress *.Z
```

  komprimieren und dekomprimieren der Dateien des aktuellen Verzeichnisses.

- $ tar *options tarfile files*

  tar erzeugt das Archiv **tarfile** unter Verwendung der Dateien **files**.

- Wichtige Optionen:

  c
  Create. Erzeugen des Archivs **tarfile**.

  x
  Extract or restore. Die Dateien des Archivs **tarfile** werden ins aktuelle Verzeichnis unter Beibehaltung des relativen Pfades geschrieben.

  f
  File. Verwenden von **tarfile** als Namen für das Archiv. Wenn **f** spezifiziert ist, wird nicht nach dem Defaultnamen in /etc/default/tar gesucht.

  h
  Aufnehmen von symbolischen Links ins Archiv als wären Sie Dateien. Normalerweise werden symbolische Links nicht ins Archiv aufgenommen.

  v
  Verbose. Ausgabe jeder Datei des Archivs, auf die zugegriffen wird.

- Beispiel:

```
$ tar cvf alles.tar *
```

erzeugt das Archiv **alles.tar**, das alle Dateien (inklusive Unterverzeichnisbäumen) des aktuellen Verzeichnisses enthält.

```
$ tar xvf alles.tar
```

extrahiert alle Dateien des Archivs **alles.tar** (inklusive Unterverzeichnisbäumen).

- $ finger *user@hostname*

  oder

- $ finger *@hostname*

  Mit dem finger-Kommando kann man sich die aktiven Benutzer auf einem System ansehen. Aufgrund von Sicherheitslücken bei der Implementierung ist es auf einigen UNIX-Systemen deaktiviert.

- Beispiel:

```
$ finger @rossini
```


- $ ftp *options hostname*

  `ftp` stellt die Verbindung zum Rechner **`hostname`** für den Zweck eines Dateiaustausches her. Die Angabe des Zielrchners ist optional.

- Wichtige Befehle im ftp-Modus:

`open`
Öffnet die Verbindung zum Zielrechner, falls dieser nicht beim Aufruf von ftp angegeben war.

`user`
Ändert den User, der auf dem Zielrechner eingelogged ist.

`get file, mget files`
Kopieren von Dateien vom Zielrechner auf den Ausgangsrechner.

`put file, mput files`
Kopieren von Dateien vom Ausgangsrechner auf den Zielrechner.

- Fortsetzung Wichtige Befehle im ftp-Modus:

binary
Wird benötigt zum Übertrage von Binärdateien.

bye
quit
Verlassen von ftp.

case
Konvertiert alle Buchstaben zu Großbuchstaben bei den Dateinamen.

cd remote-directory
Navigieren im Dateibaum auf dem Zielrechner.

cdup
Im Dateibaum des Zielrechners ein Verzeichnis nach oben gehen.

close
Beenden der ftp Session und zurückkehren in den ftp-Kommando Modus.
(Gegenoperation zu open)

- Fortsetzung Wichtige Befehle im ftp-Modus:

delete remote-file
mdelete remote-files
Löschen von Dateien auf dem Zielrechner.

lcd [directory]
Navigieren im Verzeichnisbaum des lokalen Rechners (Ausgangsrechners).

ls [remote-directory | -al] [files]
Inhalte des Verzeichnisses des Zielrechners ausgeben

mkdir directory-name
rmdir directory-name
Verzeichnis auf dem Zielrechner anlegen, bzw. Verzeichnis auf dem Zielrechner löschen.

- Beispiel:

```
$ ftp kirk
```

- $ telnet *options host*

  `telnet` öffnet eine Terminalverbindung zum Rechner **`host.`**

- Beispiel:

```
$ telnet kirk
```


- $ mail *options user@hostname*

  `mail` verschickt Nachrichten an **`user`** auf dem Rechner **`hostname`** und empfängt Nachrichten.

- Beispiel:

```
$ mail unix??@kirk
$ mail unix??
```

  Senden einer Nachricht an den Benutzer **`unix??`**

- $ talk *user@terminal*

  `talk` baut bidirektionale Verbindung zum Bildschirm **`terminal`** zwecks Informationsaustausch mit dem Benutzer **`user`** auf.

- Beispiel:

```
$ talk unix??@kirk
$ talk unix??
```

  Aufbau einer Talkverbindung zum Benutzer **`unix??`**.

- $ write *user@terminal*

  `write` baut unidirektionale Terminalverbindung zwecks Ausgabe auf den Bildschirm **`terminal`** des Benutzers **`user`** auf.

- Beispiel:

```
$ write unix??@kirk
$ write unix??
```

  Ausgabe einer Nachricht auf den Bildschirm des Benutzers **`unix??`**.

- $ time *Kommando*

  `time` ermittelt die Zeit, die für die Ausführung eines Kommandos benötigt wird. Die Ausgabe zeigt die Verweildauer (**`real`**), die verbrauchte Zeit im Benutzermodus (**`system`**) und die verbrauchte Zeit im Systemmodus (**`sys`**) in Sekunden an. Die CPU-Zeit ergibt sich aus **`user+sys`**.

- Beispiel:

```
$ time gcc –o hello hello.c
```


- $ at *options –f Kommandodatei Zeitangabe*

  `at` startet eine *Kommandodatei* zu der durch die *Zeitangabe* festgelegten Zeit.

- Options:
  - -c Ausführen der *Kommandodatei* in einer C-Shell
  - -k Ausführen der *Kommandodatei* in einer Kornhell
  - -s Ausführen der *Kommandodatei* in einer Bourne-Shell

  Mögliche Zeitangaben sind:
  now, noon, midnight, tomorrow
  oder relative Angaben:
  + 2 hours, + 3 days

- Beispiel:

```
$ at –f hello now
(Achtung: die Ausgabe des Kommandos hello wird umgelenkt)
```

- $ cron
  $ crontab *options*

  `crontab` erstellt eine Tabelle, die Zeitangaben und dazugehörige Kommandos enthält. **`cron`** startet den clock deamon, der für die zeitgerechte Ausführung der durch **`crontab`** definierten Tabelle sorgt.

- Options:

  -l Auflisten der crontab Einträge
  -r Löschen der crontab Datei

  Zeitformat für crontab:

  minute(0-59) hour(1-24) day(1-31) month(1-12) day_of_the_week(0=sun – 6)

  wildcards(*) sind an jeder Stelle möglich

- Beispiele:

```
$ crontab
    > 10 10 26 11 * kommandodatei
```

Die **`kommandodatei`** wird am 26.11 um 10:10 Uhr ausgeführt

```
$ crontab
    > 10 5 1,15 * 1 kommandodatei
```

Die **`kommandodatei`** wird an jedem 1. und 15. eines Monats sowie an jedem Montag um 5:10 Uhr ausgeführt