Voraussetzungen

Ziele

Inhalt

- Zugang, Passwortschutz und Verlassen des Systems
- Kommandosyntax
- Hilfe, Online-Manual
- Dateien: Namen, Ausgabe, Sortieren, Umbenennen, Kopieren, Löschen
- Verzeichnisse und Dateiattribute
- Zugriffsrechte
- Navigation und Links
- Prozesse: Kenndaten, Beenden, Hintergrundprozesse, Pipes

- **Anmeldung** nach dem Login - Prompt:

  `Login: Benutzername <CR>`

- Passwort

  `password: xxxxx..... <CR>`

- Eingabeaufforderung durch **UNIX-Prompt**, z. B.

  `$` (für die Korn-Shell)

- UNIX-Prompt `$` kann geändert werden (z. B. Ausgabe von Arbeitsverzeichnis, Kommandonummer, Rechnerkennung)

- Beispiel: `$ PS1='kirk:${PWD##$HOME?(/)}[\!]$ '#`

  erzeugt einen Prompt inkl. Arbeitsverzeichnis und Befehlsnummer.

M2 Passwortschutz

- Änderung des Passwords durch Benutzer oder Systemadministrator (Superuser)
- `$ passwd`
- ≥ 6 Zeichen, 1. Zeichen = Buchstabe
- SVR4: mind. 1 Ziffer oder Sonderzeichen

M2 Verlassen des UNIX-Systems

- `$ exit` oder
- `CTRL D`
- evtl. vorher: `$ sync`

  (schreibt die Platten-Cache-Bereiche aus dem Hauptspeicher auf Platte: flush)
- Befindet man sich nicht auf UNIX-Kommandoebene ($), muss die laufende Anwendung vorher verlassen werden, notfalls mit `CTRL C`.

M2 Shell-Kommandos

- Das UNIX-System startet nach dem Hochfahren einen Kommando-Interpreter, eine sog. **Shell**. Sie meldet sich mit dem UNIX-Prompt ($).
- Kommandosyntax

  `$ kommando -optionen` ***`parameter`*** **`<CR>`**
- **Optionen** sind Kürzel, die die genaue Abarbeitung des Kommandos steuern.
- **Parameter** sind häufig Dateinamen.

M2 Abfrage eingeloggter Benutzer

- **`$ who`**

  gibt eine Liste aller momentan eingeloggten Benutzer aus
- **`$ who am i`**

```
falbala!cms pts/1 Dec 21 14:18
```

M2 Banner und Datum

- Ausgabe von Zeichenketten in Spruchbandform

  $ banner *text*

- Ausgabe von Datum und Uhrzeit

  $ date

  $ date +%a%j (a: Wochentag, j: Tag des Jahres)

M2 Zeilenfortsetzung

- Soll ein Kommando auf 2 Zeilen aufgeteilt werden, dann ist die Vorgängerzeile mit \ (Fortsetzungszeichen) abzuschließen.
- Beispiel (etwas exotisch)

```
$ bann\ <CR>
er ab\ <CR>
c <CR>
```

vergleiche hierzu: Quoting (Abschnitt M3)

M2 Online-Manual

- Genaue Informationen zu den Kommandos erhält man durch das **Online-Manual**

  `$ man kommando`

- Beispiel: Information zum Kommando `ls`:

  ```
  $ man ls
  ```

- Man kann auch die Korrekturaufforderung der Shell missbrauchen, wenn man die genaue Syntax vergessen hat.

- **Suche** im man-Kommando:

  / keyword

  N (für next)

M2 Aufbau einer Kommandobeschreibung (1)

NAME	Name des Kommandos (der Funktion) - Kurzbeschreibung
SYNOPSIS	Syntaxbeschreibung
DESCRIPTION	ausführliche Beschreibung des Kommandos
OPTIONS	Liste aller verwendbaren Optionen mit einer kurzen Beschreibung der Wirkung
COMMANDS	Bei manchen interaktiven Programmen kann man während der Ausführung Anweisungen absetzen, die hier erklärt werden.

Forts. -->

M2 Aufbau einer Kommandobeschreibung (2)

FILES	Dateien, die mit dem Kommando zusammenhängen
SEE ALSO	Hinweise auf verwandte Kommandos und Manualeinträge an anderen Stellen
DIAGNOSTICS	Liste von Fehlermeldungen, die unter Umständen erzeugt wird
EXAMPLE	Beispiele zum Aufruf des Kommandos
BUGS	Bekannte Fehler und Schwierigkeiten bei der Verwendung des Kommandos

M2 Kommandosequenzen

- Abschluss (Abschicken) eines Kommandos mit ***<CR>***
- Trennung auch mit ;
- Beispiel:

```
$ cat datei2 datei1 >datei3; cp datei3 datei2; rm datei3 <CR>
```

M2 Kommandogruppen

- Ziel: Gesamtausgabe von **ps** und **who** in datei1

  (1) **$ ps; who >*datei1***

  (2) **$ ps >*datei1*; who >*datei1***

  (3) **$ ps >*datei1*; who >>*datei1***

  Nur (3) ist korrekt.

- Alternative: **Gruppierung**

  $(ps; who) >*datei1*

  Dabei werden die geklammerten Kommandos in einem eigenen Prozess ausgeführt.

M2 Programmschnittstelle

- Systemdienste können mit Shell-Kommandos oder aus Programmen heraus aufgerufen werden.
- Namen und Funktion vielfach ähnlich, Syntax unterschiedlich.
- **C-Syntax**

  s = systemcall (a, b, c)

  s Rückgabeparameter

  Fehler: **s = -1**

  a, b, c Parameter

M2 Dateinamen

- max. Länge je nach UNIX-Variante, Empfehlung ≤ 14 Zeichen

  Groß- /Kleinbuchstaben, Ziffern, . _ - (Punkt, Underscore, Bindestrich)

- Namenskonventionen

.a	für Objektbibliotheken,
.c	für C-Quelltextdateien,
.f	für FORTRAN77-Quelltextdateien,
.f90	für Fortran 90-Quelltextdateien,
.o	für Objektdateien,
.p	für PASCAL-Quelltextdateien.

M2 Ausgabe einer Datei

- Ausgabe kurzer Dateien auf einmal:

  `$ cat dateiname` (concatenate)

- Ausgabe langer Dateien seitenweise:

  `$ pg dateiname` (nächste Seite: <CR>)

  `$ more dateiname` (nächste Seite: <Leertaste>)

- Dateianfang, -ende

  `$ head -n dateiname`

  `$ tail -n dateiname`

  n = auszugebende Zeilenzahl

M2 Dateiausgabe, Drucker

- Ausgabe auf den Bildschirm ähnlich **`cat`**, aber in druckeraufbereiteter Form (66 Zeilen / Seite):
  `$ pr -n` *dateiname* (Option -n: Zeilen numerieren)
- **Ausgabe** am Drucker
  `$ lp -d` ***`printer_name datei_name`*** (line printer)
  `request id is` ***`kennung`***
- Druckauftrag **abbrechen**
  `$ cancel` *kennung*
  `$ cancel` *printer-name* (alle Aufträge!)
- **Druckerstatus**
  `$ lpstat`
  Ausgabe des Status aller Druckaufträge des Users

M2 Sortieren (1)

- **`$ sort`** **sort**iert den Eingabetext alphabetisch
- **`Option -r`** absteigende Sortierung
- Beispiel

`$ sort` Aufruf des Kommandos **`sort`** ohne Angabe von Optionen

```
Birnen <CR>
Aprikosen <CR>
Kirschen <CR>
```

Eingabe des zu sortierenden Textes

`<CTRL D>` Texteingabe mit **`<CTRL D>`** abschließen

Text wird alphabetisch aufsteigend sortiert am Bildschirm ausgegeben: (→)

M2 Sortieren (2)

```
Aprikosen
Birnen
Kirschen
```

`$ sort -r`	Aufruf des Kommandos **sort** mit Option **-r** (absteigend sortieren)
`Birnen <CR>`	Eingabe des zu sortierenden Textes
`Aprikosen <CR>`	
`Kirschen <CR>`	
`<CTRL D>`	Texteingabe mit **<CTRL D>** abschließen

`Kirschen`
`Birnen`
`Aprikosen`
	Text wird alphabetisch absteigend **sortiert** am Bildschirm ausgegeben

- Eingabe jeweils abzuschließen durch EOF (end of file) = **<CTRL D>**

M2 Ausschneiden

- **$ cut -c *n-nn dateiname***
- Aus der Datei ***dateiname*** werden die Spalten n bis nn ausgeschnitten und ausgegeben.
- Beispiel

```
$ ls >dliste
$ cut -c2-5 dliste
```

M2 Dateien umbenennen, kopieren, löschen

- Umbenennen oder verschieben

  `$ mv alternname neuername` (move)

- Kopieren

  `$ cp alternname neuername` (copy)

- Löschen

  `$ rm dateiname` (remove)

  VORSICHT: endgültige Löschung ohne Warnung !!!

- Löschen mit Warnung

  `$ rm -i dateiname` (inform)

- Kopieren inkl. Unterverzeichnissen

  `$ cp -r directory_name_alt directory_name_neu`

M2 Verzeichnis (directory) anzeigen

- Ausgabe des Inhalts des momentanen Arbeitsverzeichnisses

  `$ ls` (list)

  Es werden alle Dateinamen außer den mit einem Punkt beginnenden (Punktdateien) ausgegeben.

- Ausgabe inkl. Punktdateien

  `$ ls -a`

- Ausgabe mit Attributen

  -i inode-Nummer

  -F Angabe des Dateityps: * Executables, / Directory, @ Symbolic Link, ohne Angabe: reguläre Dateien

  -l long

M2 Dateiattribute bei `ls -l`

- **Dateityp** (1. Spalte)
  - d Directory (Verzeichnis)
  - l Symbolic link
  - - normale Datei
  - b block special file
  - c character special file
  - p pipe
- **Protection Bits** rwx rwx rwx
- Anzahl **Hardlinks** auf die Datei
- **Eigentümer** (oder uid)
- **Gruppe** (oder gid)
- **Dateilänge** in bytes
- **Datum/Uhrzeit** letzte Änderung
- **Dateiname**

M2 Beispiel Dateiattribute

- `$ ls -l`
- **Ausdruck:**

```
total 1661
drwxr-x---   2 cms      staff         512 Aug 13  1993 ALife
-rwxr-xr-x   1 cms      staff         501 Sep 17  1997 ILEAF
-rwxr-xr-x   1 cms      staff         501 Sep 17  1997 NETSCAPE
drwxr-xr-x   5 cms      staff         512 Aug 21  1992 News
-rw-r--r--   1 cms      staff      330823 Jun 27  1996 Sandman.AIFF
-rwxr-xr-x   1 cms      staff         501 Oct  1  1997 WINGZ
-rwxr-xr-x   1 root     other         523 Apr 30  1996 Wingz
drwxr-xr-x   4 cms      staff         512 Jul  9  1992 addman
drwxr-x---   2 cms      staff         512 Aug  5  1993 alife
-rw-r--r--   1 cms      staff          80 Mar  1  1991 bikes
drwx------   2 cms      staff         512 Nov 14  1996 bin
-rw-r--r--   1 cms      staff         109 Mar  1  1991 books
drwx------   2 cms      staff         512 Jan  7  1998 cms
drwxrwxrwx   2 cms      staff        1024 Dec 22 11:14 decode
```

M2 Erzeugen und Löschen von Verzeichnissen

- $ mkdir *directory_name* (make directory)
- $ rmdir *directory_name* (remove directory, Dir. muss leer sein!)
- Jedes Verzeichnis enthält automatisch einen Verweis auf sich selbst (.) und auf das Parent-Directory (..) .

```
$ cd texte
$ ls
gedicht        test          text.1        text.1%
$ mkdir newdir
$ ls -F
gedicht        newdir/       test          text.1        text.1%
$ cd newdir
$ pwd
/home/meier/texte/newdir
$ ls -a
.              ..
$ cd ..
$ rmdir newdir
$ ls -aF
./       ../    gedicht       test          text.1        text.1%
```


M2 Navigieren im Dateibaum

- Jeweils ein Verzeichnis ist das aktuelle Arbeitsverzeichnis (**Working Directory** WD).

  Anzeige über

  $ pwd (print working directory)

- Änderung des Working Directory

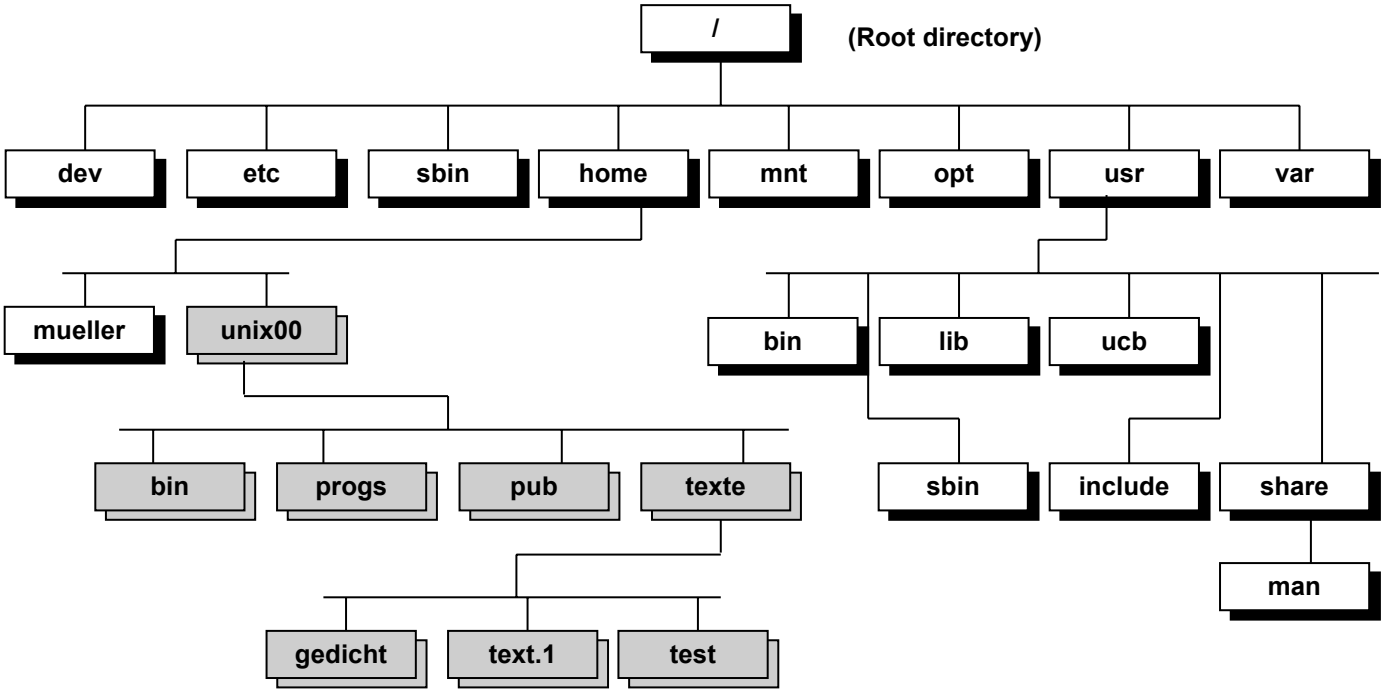
  $ cd *directory_name* (change directory)

- Das nach dem Login automatisch als WD eingestellte Verzeichnis heißt

  Home Directory

- cd ohne Argument: → Home Directory; cd ~username
- Abkürzungen
  - ~ Home Directory
  - • selbst
  - •• Parent Directory

M2 Beispiel zur Navigation (1)



M2 Beispiel zur Navigation (2)

```
cd
$ pwd
/home/unix00
$ cd ..
$ pwd
/home
$ cd unix00
$ pwd
/home/unix00
$ mkdir texte
$ cd texte
$ pwd
/home/unix00/texte
```

M2 Systemverzeichnisse (1)

- Vordefinierte und installierte Verzeichnisse

`/dev` (devices): In diesem Directory stehen alle Geräteeinträge (`special files`) sowie die Mülleimerdatei `/dev/null` (leere Datei).

`/etc` (etcetera): In diesem Directory liegen Systemkonfigurations- und -verwaltungsprogramme sowie die meisten Systeminformationsdateien.

`/home` Dieser Zweig des Dateisystems enthält die HOME-Directories der Benutzer.

`/mnt` Ein i. A. leeres Verzeichnis, welches zum Einhängen weiterer Massenspeicher (mittels `mount`-Kommando, daher der Name `/mnt`) vorgesehen ist.

`/sbin` Enthält (ähnlich `/etc`) ausführbare Dateien zur Systemverwaltung (im Unterschied zu `/etc` i.A. Binärversionen).

M2 Systemverzeichnisse (2)

`/tmp` (temporaries): Dieses Directory wird vom System in regelmäßigen Abständen gelöscht. Es kann von jedem Benutzer und Benutzerprogrammen als temporärer Speicherbereich benutzt werden.

`/usr` (unix system ressources) enthält weitere Systemverzeichnisse:

`/usr/bin` (binaries): In diesem Directory liegen die am häufigsten benutzten Dienstprogramme, z.B. `ls` und `rm`. Alle Dienstprogramme können einfach durch Eingabe ihres Namens (= Kommandoaufruf) ausgeführt werden. Sie bestehen im allgemeinen aus übersetzten C-Programmen.

M2 Systemverzeichnisse (3)

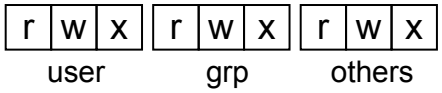
`/usr/lib`	Systembibliotheken,
`/usr/sbin`	ähnlich `/sbin`, seltener benutzte Kommandos,
`/usr/ucb`	aus BSD-Unix übernommene Kommandos,
`/var`	(variabel) enthält etliche Systemdirectories mit temporär veränderlichen Dateien, wie z.B. news und mail,
`/opt`	(optional) optionale, nützliche Ergänzungen des Systems.

M2 Zugriffsrechte

- 3 Benutzerklassen
  - u user
  - g group
  - o others
- jeweils 3 Rechte
  - r Lesen (read)
  - w Schreiben (write)
  - x Ausführen (execute)
- 9 Schutzbits (Protection Bits)
- Vergabe von Rechten für normale Dateien (auch Programme), Special Files, Verzeichnisse

M2 Schutzbits (Protection Bits)

- Abfrage mit `$ ls -l`



- Beispiel

```
rwx   r-x   --x

111   101   001
```

Eigentümer darf lesen, schreiben, ausführen.

Gruppe darf lesen, ausführen.

Alle anderen dürfen nur ausführen.

M2 Zugriffsrechte auf Verzeichnisse

- Schutzbits wie für normale Dateien.
- Zum Lesen und Durchsuchen sind **`r-`** und **`x-`**Bit nötig:

```
r-x
```

- Schreibrecht **`w`** auf ein Verzeichnis: Die Klasse darf neue Dateien anlegen (Eigentümer = anlegender der Benutzer) und Dateien löschen.

  Vorsicht! Es können auch Dateien gelöscht werden, für die der Benutzer keine Zugriffsrechte besitzt!

M2 Verändern von Zugriffsrechten (1)

- Eigentümer kann die Schutzbits setzen/rücksetzen mittels

  `$ chmod` (changemode)

- Format1: $ **chmod** *permission_list dateiname*

  wobei *permission_list* = *benutzerklasse* ± *rechte*

  benutzerklasse = u, g, o, a (a = alle)

  rechte = **r, w, x**

M2 Verändern von Zugriffsrechten (2)

- Beispiel

```
$ ls -l
- rwx r-- r--  1 gast3 gast 113 Sep 4 11:24 upro
$ chmod a+wx upro
$ ls -l
- rwx rwx      rwx  1 gast3 gast 113 Sep 4 11:24 upro
```

- `$ chmod g-r upro`

  Der Gruppe gast wird das Leserecht entzogen.

```
 $ ls -l
- rwx -wx      rwx  1 gast3 gast 113 Sep 4 11:24 upro
```

M2 Verändern von Zugriffsrechten (3)

- Format 2

  $ **chmod** *ooo dateiname*

  Schutzbits werden neu gesetzt.

  ooo = 3-stellige Oktalzahl entsprechend den 9 Schutzbits

  ooo = 764 = 111 110 100 = rwx rw- r--

M2 Verändern von Zugriffsrechten (4)

- Beispiel

```
$ ls -l

- rw- --- ---  1 gast3 gast 386 Aug 4 9:12 brief

$ chmod 640 brief

 $ ls -l

- rw- r-- ---  1 gast3 gast 386 Aug 4 9:12 brief
```

Erläuterung: **640 (oktal) = 110 100 000 (dual)**

M2 Voreinstellung von Zugriffsrechten (1)

- Beim **Anlegen** einer Datei werden bestimmte Schutzbits gemäß Voreinstellung automatisch gesetzt.

  Diese Voreinstellung kann mittels

  ```
  $ umask ooo
  ```

  verändert werden.

- `ooo` ist wieder als Oktalzahl zu lesen, wirkt aber **umgekehrt** wie bei **`chmod` (Maske!)**:

  `0` bewirkt Setzen des Rechts.

  `1` bewirkt Rücksetzen des Rechts.

- **`umask`** gilt bis Sitzungsende oder bis zum nächsten **`umask`**.

M2 Voreinstellung von Zugriffsrechten (2)

- Beispiel

  ```
  $ umask 077
  ```

  `rwx --- ---` für neu angelegte Datei

- Besonderheit

  `x` ist nur sinnvoll für Verzeichnisse und ausführbare Dateien.

  Das **`x`**-Bit wird (automatisch) nur für Verzeichnisse erzeugt oder für Ergebnisse von Compile-Läufen (executables).

M2 Voreinstellung von Zugriffsrechten (3)

- Automatisch vergebene Rechte beim Kopieren von Dateien:

  $ cp *datei1 datei2*

  a) ***datei2*** wird neu erzeugt:

  Schutzbits der ***datei2*** werden von ***datei1*** übernommen, soweit nicht von **umask-Maske** verdeckt.

  b) ***datei2*** wird überschrieben:

  Alte Schutzbits bleiben gesetzt.

- Ausgabe der momentan gültigen Maske:

```
$ umask
027
```


M2 Beispiel umask

-
```
$ ls -l
- rw- --- ---  1 gast3 386 Aug 4 9:12      brief
- rw- rw- r--  1 gast3 267 Jan 24 10:56    pbrief
-rwx rwx rwx   1 gast3 113 Sep 4 11:24     upro
$ umask  27
$ cp brief nbrief
$ cp brief pbrief
$ mkdir priv
$ ls -l
```

Forts. -->

M2 Beispiel umask (Forts.)

```
- rw- --- --- 1 gast3 386 Aug4 9:12     brief
- rw- --- --- 1 gast3 386 Nov18 16:43    nbrief
- rw- rw- r-- 1 gast3 386 Sep18 16:43    pbrief
d rwx r-x --- 2 gast3 512 Nov18 16:43    priv
- rwx rwx rwx 1 gast3 113 Sep4 11:24     upro
```


M2 Besitzrechte

- Eigentümer kann nur vom Superuser geändert werden, Gruppenzugehörigkeit auch vom Eigentümer:

  `$ chown` *neuer-user dateiname*

  `$ chgrp` *neue-gruppe dateiname*

- auch mit -R Option: rekursive Anwendung auf Unterverzeichnisse

M2 Links (1)

- Problem: mehrere Benutzer (oder auch 1 Benutzer) wollen auf 1 Datei unter unterschiedlichen Namen zugreifen.
  Lösung: (Hard-) Link: Ein **neuer_name** verweist zusätzlich auf die Datei **alter_name**.
- Datei existiert nur 1 x !
- `$ ln alter_name neuer_name` (link)
- Beispiel:

```
$ pwd
/home/meier/texte
$ ln text.1 newdir/text.ln
$ cd newdir
$ ls
text.ln
```


M2 Links (2)

- Einschränkung: beide Einträge müssen sich auf demselben Datenträger befinden!
- Alternative: Soft-Link (Verknüpfung über absoluten Pfadnamen)
  `$ ln -s alter_name neuer_name`

M2 Prozesse (1)

- Anzeige laufender Prozesse

  `$ ps` (process status) oder `ps -u username`

- Anzeige mit Prozessattributen

  `$ ps -eaf`

- Beispiel:

```
$ ps -eaf
UID     PID   PPID  C     STIME      TTY   TIME        COMMAND
root    0     0     0     Jul 13     ?     0:01         sched
root    1     0     0     Jul 13     ?      6:44        /sbin/init
root          2     0     Jul 13     ?     0:01        pageout
root          3     0     Jul 13     ?     1:13        fsflush
root    108   1     0     Jul 13   console 0:01         -ksh
```


M2 Prozesse (2)

Die Angaben bedeuten im einzelnen:

`UID`	Username des Prozesseigentümers
`PID`	Prozess-Nummer,
`PPID`	Eltern-Prozess-Nummer (Parent PID)
`C`	systeminterner Scheduling-Parameter,
`TTY`	Name der Dialogstation, von der dieser Prozess gestartet wurde (? = keiner Dialogstation zugeordnet),
`TIME`	bisher vom Prozess verbrauchte CPU-Zeit,
`Command`	Name des Kommandos, welches diesen Prozess gestartet hat.

M2 Prozesskenndaten mit ps (1)

- Erweiterte Zustandsinformation über Prozesse erhält man mittels

```
$ ps -ael
```

- Beispiel:

```
F   UID   PID  PPID CP PRI NI   SZ  RSS    WCHAN S TTY       TIME COMMAND
8   103   215     1  0  40 20 1856  672 lm_block S console   0:02 -tcsh HOME=/  ....
8   103  2970   215  0  45 20  836  468 lm_block S console   0:00 /bin/sh /usr
8   103  2974  2970  0  48 20 1728  864 lm_block S console   0:00 /usr/openwin
8   103  2975  2974  2  56 2010616 4628 block_lo S console  224:21 /usr/openwin
8   103  2976  2974  0  55 20  828  468 lm_block S console   0:00 sh /homes/cm
8   103  2981     1  0  59 20 1720  920 block_lo S console   0:00 fbconsole AP
8   103  2987     1  0  59 20 3380 1484 block_lo S console   0:00 vkbd -nopopu
8   103  2990     1  0  10 20 3476 1032 block_lo S console   0:01 ttsession -s
8   103  2993  2976  0  59 20 2456 1560 block_lo S console   0:04 olvwm -syncp
8   103  2995  2993  0  30 20 3344 1364 block_lo S console   0:00 olwmslave AP
8   103  3003     1  0  60 20 2808  728 block_lo S console   0:00 /interleaf/i
8   103  3012     1  1  48 20 4816 2816 block_lo S console  57:32 /usr/openwin
8     0  1107 29047  1  10 20 1044  844           O pts/1     0:00 ps -ael _=/u
```


M2 Prozesskenndaten mit ps (2)

Erläuterung der Spalten:

`F` Systemintern benutzte Flags; z. B. bedeutet 10, dass dieser Prozess im Arbeitsspeicher liegt.

`S` bezeichnet den Zustand des Prozesses:

- `O` Prozess ist aktiv,
- `I` Prozess wird gerade kreiert,
- `T` Prozess wurde angehalten,
- `S` Prozess wurde suspendiert,
- `R` Prozess ist im lauffähigen Zustand in der Warteschlange,
- `Z` Prozess ist fehlerhaft terminiert (Zombie),
- `X` Prozess wächst und wartet auf mehr Speicher.

M2 Prozesskenndaten mit ps (3)

Erläuterung der Spalten:

`UID`	UID-Nummer des Prozesseigentümers,
`PRI`	aktuelle Priorität des Prozesses (je kleiner PRI, desto größer die Priorität),
`NI`	Nice-Größe für die Prioritätsberechnung,
`ADDR`	Hauptspeicher- oder Plattenadresse des Prozesses,
`SZ`	Größe des Prozesses im Hauptspeicher in kbyte,
`RSS`	Resident set size in kbyte
`WCHAN`	Adresse eines Events, auf den der Prozess wartet.

M2 Beenden (eigener!) Prozesse

- Abbruchsignal (nicht maskierbar)

  ```
  $ kill -9 pid
  ```

- Das Kommando **`kill`** schickt ein Signal s an den Prozess ***`pid`***, welches nicht unbedingt zum Abbruch führen muss (falls $s \neq 9$).
- Fremde Prozesse (andere uid) können nur vom Superuser beendet werden.
- Prozesse können Signale **abfangen** (außer s = 9) mittels Signal Handler.

M2 Hintergrundprozesse (1)

- Nach Abschicken eines Kommandos (= Erzeugen eines Kindprozesses) wartet der Vaterprozess (hier: die Shell) auf das Ende des Kindprozesses: der Shell-Prompt erscheint erst dann wieder.
- Will man mit der Shell weiterarbeiten, muss das Kommando im Hintergrund laufen.

  Schreibweise:

  `$ kommando [argumente] &`

  `&` (Ampersand) muss am Zeilenende stehen!
- Das System quittiert mit **`[jobnummer] pid`**
- Beispiel

  `$ man sh | col -b >shell.man &`

  [2] 12905

  (**`col -b`** entfernt die nicht-druckbaren Zeichen.)

M2 Hintergrundprozesse (2)

- Meldung bei Beendigung des Hintergrundprozesses
  - Bourne-Shell: keine Meldung
  - Korn-Shell: explizite Meldung
- Tastatur ist mit std-in des Vordergrundprozesses verbunden.

  Für den Hintergrundprozess ist die Standard-Eingabe = /dev/null
- Wechsel zwischen Vorder- und Hintergrundprozess

  `$ fg` (foreground)

  `$ bg` (background)
- Ein Vordergrundprozess muss gestoppt werden, bevor er mit **`bg`** in den Hintergrund geschickt werden kann (Suspension Character **CTRL Z**, abh. von Voreinstellung)

M2 Hintergrundprozesse (3)

- Beispiel

```
$ myprog &        # Ausführung von myprog im Hintergrund
  [1]  4631       # pid=4631, JobID= 1 (dazu weiter unten mehr)
$ fg              # bewirkt Zurückholen auf den Bildschirm
                  # bis zum Programmende, aber: die Eingabe
CTRL z            # kann dieses Programm wiederum stoppen
                  # (sofern CTRL z der Suspension-Character ist) und
$ bg              # schickt es wiederum in den Hintergrund
```


M2 Hintergrundprozesse (4)

- Übersicht über alle im Hintergrund laufenden Jobs:

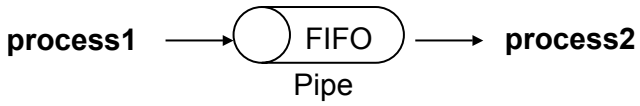
```
$ jobs
[4] + Running pro4 &
[3] - Running pro3 &
[2]   Running pro2 &
[1]   Running pro1 &
```

- Ansprechen eines bestimmten Jobs

```
$ fg %3
$ kill %2          (wie mit pid)
```

M2 Pipes (1)

- Eine Pipe wirkt wie eine Datei, in welche vom ***process1*** geschrieben und aus welcher vom ***process2*** gelesen wird. Reihenfolge: **FIFO**



- Schreibweise:

```
$ kommando1 | kommando2
```

- Verkettung beliebig vieler Prozesse.
- Standard-Ausgabe einer Pipe ist Standard-Ausgabe des letzten Prozesses.

M2 Pipes (2)

- Beispiel: Kommando **wc** (word count) zählt

  -l die Zeilen
  -w die Wörter
  -c die Zeichen

  einer Eingabedatei ***datei1***

```
$ wc -l datei1
```

  ohne Dateiangabe: Standard Input

- Verkettung mittels Pipe

```
$ ls /bin | wc -w
```

  Ergebnis: Anzahl der Dateien in ***/bin***.

- Seitenweise Ausgabe

```
$ ls -al /bin | more
```

- Pipe-Problem: Zwischenergebnis geht verloren
- Abhilfe: Einbau einer **Abzweigung** (“T-Stück“) in die Pipe

```
$ ls | tee dliste | wc -w
```