5.Übung

Systemsoftware (SYS)

Christian Baun
cray@unix-ag.uni-kl.de

Hochschule Mannheim – Fakultät für Informatik
Institut für Robotik

2.11.2007

Wiederholung vom letzten Mal

- Zugriffsrechte voreinstellen (`umask`)
- Hard Links und Symbolische Links (`ln`)
- Dateien durchsuchen (`grep`)
- Verzeichnisse packen und entpacken (`zip`, `rar`, `tar`, `gzip`,...)
- Editoren: Joe's Own Editor, vi(m), Emacs
- Prozesse anzeigen (`ps`)
- Prozesse in den Hintergrund schicken (`bg`)
- Prozesse in den Vordergrund holen (`fg`)
- Prozesse beenden (`kill`, `killall`)
- Prozessprioritäten festlegen und ändern (`nice`, `renice`)
- Prozessvererbung anzeigen (`pstree`)
- Prozesse/Kommandos verknüpfen mit Pipes (|)
- Wildcards (`?`, `*`, `[]`, `!` und `^`)

Heute

- Einführung für Linux/UNIX-Anwender (Teil 4)
  - Systemzeit und Systemdatum ausgeben oder ändern (`date`)
  - Umleiten von Ein- und Ausgaben (< und >)
  - Bytes, Zeichen, Wörter und Zeilen zählen (`wc`)
  - Abarbeitungsgeschwindigkeiten messen (`time`)
  - Der Alias-Mechanismus
  - Dateien suchen und finden (`find`, `locate`, `whereis`, `which`)

Die Systemzeit ausgeben und ändern mit `date`

`date` *`[Option] ... [+Format] [Systemzeit]`*

- Das Kommando `date` ermöglicht es, die Systemzeit und das Systemdatum auszugeben und zu ändern.
- Das Ausgabeformat kann nahezu beliebig angepasst werden.
- Nur der Benutzer `root` kann die Systemzeit und das Systemdatum ändern.
- Das Kommando ohne Optionen und Formatangaben aufrufen:

```
$ date
Do 25. Okt 09:36:19 CEST 2007
```

- Die Voreinstellung von `date` ist: `%a %e %b %T %Z %Y`

```
$ date +"%a %e. %b %T %Z %Y"
Do 25. Okt 09:36:19 CEST 2007
```

Formatangaben von `date` (1)

`%a` **Wochentag** in abgekürzter Schreibweise (Son..Sam)

`%b` **Monatsname** in abgekürzter Schreibweise (Jan..Dez)

`%c` **Datum und Uhrzeit** (z.B. Do 25 Okt 2007 09:50:36 CEST)

`%d` **Tag des Monats** (01..31)

`%j` **Tag des Jahres** (001..366)

`%k` **Stunde** im 24-Stunden-Format ohne führene Null (0..23)

`%l` **Stunde** im 12-Stunden-Format ohne führene Null (1..12)

`%m` **Monat** (01..12)

`%n` **Zeilenwechsel** (*newline*)

`%p` **Vor- oder Nachmittag** als String ausgeben (am/pm)

`%r` **Zeit im 12-Stunden-Format** (hh:mm:ss am/pm)

`%s` **UNIX-Zeit**: Anzahl der Sekunden seit dem 1.1.1970 00:00:00

`%t` **Horizontaler Tabulatorstop** (*tabulator*)

`%w` **Wochentag**: 0 entspricht dem Sonntag (0..6)

`%x` **Datum** nach landesüblicher Einstellung (z.B. 25.10.2007)

`%y` **Jahr** in abgekürzter Schreibweise (00..99)

`%z` **Zeitzone** als numerische Angabe im Stil von RFC-822

Formatangaben von `date` (2)

- `%A` **Wochentag** in voller Länge (Sonntag..Samstag)
- `%B` **Monatsname** in voller Länge (Januar..Dezember)
- `%D` **Datum/Monat/Jahr** mit jeweils zwei Ziffern (z.B: 10/25/07)
- `%H` **Stunde** im 24-Stunden-Format mit führener Null (00..23)
- `%I` **Stunde** im 12-Stunden-Format ohne führener Null (01..12)
- `%M` **Minuten** (00..59)
- `%S` **Sekunden** (00..59)
- `%T` **Zeit im 24-Stunden-Format** (hh:m:ss)
- `%U` **Woche**: Nummer der Woche im laufenden Jahr mit Wochenbeginn am Sonntag (00..53)
- `%V` **Woche**: Nummer der Woche im laufenden Jahr mit Wochenbeginn am Montag (01..52)
- `%W` **Woche**: Nummer der Woche im laufenden Jahr mit Wochenbeginn am Montag (00..53)
- `%X` **Zeit** nach landesüblicher Einstellung (z.B. 10:35:41)
- `%Y` **Jahr** mit vier Stellen (z.B. 2007)
- `%Z` **Zeitzone** mit ausgeschriebenem Namen.

Umleiten von Ein- und Ausgaben – < und >

- Programme kommunizieren über drei Kanäle mit der Außenwelt:
  - Daten werden von der Standardeingabe (**STDIN**) oder aus einer Datei gelesen.
  - Ausgaben werden auf der Standardausgabe (**STDOUT**) geschrieben.
  - Fehlermeldungen werden auf die Fehlerausgabe (**STDERR**) geschrieben.
- Ein- und Ausgaben können auf der Shell mit < und > umgeleitet werden.
- Ausgabe in eine andere Datei leiten (überschreiben bzw. neu anlegen):

```
$ cat folien_bts_uebung3.tex | grep itemize > ausgabe.txt
```

- Ausgabe in eine andere Datei leiten (anhängen bzw. neu anlegen):

```
$ cat folien_bts_uebung3.tex | grep itemize >> ausgabe.txt
```

Bytes, Zeichen, Worte, Zeilen zählen – wc

`wc [Option] ...[Datei] ...`

- Das Kommando `wc` ist in der Lage, die Anzahl der Bytes, Zeichen, Wörter und Zeilen einer Datei bzw. aus der Standardeingabe zu zählen.
  - `-c` Gibt die Anzahl der Bytes aus.
  - `-m` Gibt die Anzahl der Zeichen aus.
  - `-l` Gibt die Anzahl der Zeilen aus.
  - `-L` Gibt die Länge der längsten Zeile aus.
  - `-w` Gibt die Anzahl der Wörter aus.

```
$ wc -l folien_bts_uebung3.tex
601 folien_bts_uebung3.tex
```

```
$ cat folien_bts_uebung3.tex | wc -m
19471
```

Abarbeitungsgeschwindigkeiten messen mit `time`

- Mit dem Kommando `time` kann die Zeit gemessen werden, die ein Prozess verbraucht.

```
$ dd if=/dev/zero of=/tmp/testdatei bs=1024 count=512000
512000+0 Datensätze ein
512000+0 Datensätze aus
524288000 Bytes (524 MB) kopiert, 15,7041 Sekunden, 33,4 MB/s

$ time cp testdatei kopie

real    0m55.774s
user    0m0.006s
sys     0m3.142s
```

Die Ausgabe von `time`

- Von `time` werden drei Zeitwerte ausgegeben:
  - **Realzeit**: Zeit zwischen Prozessstart und Prozessende.
  - **Userzeit**: Zeit, die die CPU für die Anweisungen des Prozesses aufwenden mußte.
  - **Systemzeit**: Zeit, die die CPU für die Anweisungen des Betriebssystems (System Calls) aufwenden mußte. Die System Calls sind Anweisungen des Betriebssystems, die vom Prozess aufgerufen wurden.

Der Alias-Mechanismus

- Der Alias-Mechanismus ermöglicht es, Abkürzungen für Kommandos oder einen beliebigen Text auf der Shell zu definieren.

```
$ alias textalias=Text
```

- Wird `alias` ohne Optionen aufgerufen, wird eine Übersicht aller vorhandenen Aliase ausgegeben.

```
$ alias
alias dir='ls --color=auto --format=vertical'
alias ls='ls --color=auto'
alias vdir='ls --color=auto --format=long'
```

- Mit dem Kommando `unalias` kann ein Alias wieder entfernt werden.

```
$ unalias textalias
```

Dateien suchen und finden – `find` (1)

`find [Verzeichnis] [Option] ...[Aktion] ...`

- Das Kommando `find` sucht Dateien in Verzeichnisbäumen.
- `find` kennt sehr viele Suchbedingungen, um die Suche zu verfeinern.
- Beim Aufruf von `find` ohne Argumente werden alle Dateien in allen Unterverzeichnissen gefunden und ausgegeben.
- Suchbedingungen können u.a. sein: Dateiname, Dateigröße, Zugriffsrechte, Besitzer, das Datum der Erstellung, Dateityp, usw.
- Sucht im aktuellen Verzeichnis und seinen Unterverzeichnissen nach der Datei mit dem Namen `index.tex`:

```
find . -name index.tex
```

Dateien suchen und finden – `find` (2)

- Sucht im Verzeichnis `/usr/local/` und seinen Unterverzeichnissen nach der Datei `index.tex`. Ignoriert dabei Groß- und Kleinschreibung:

```
find /usr/local/ -iname index.tex
```

- Nach Dateien mit einer bestimmten Dateigröße kann mit der Option `-size` gesucht werden. `c` steht für Byte und `k` für kByte. + oder – legt fest, ob `find` Dateien suchen soll, die größer oder kleiner als der angegebene Wert sind.

```
find . -size +100k dokument.ps
```

- Auf gefundenen Dateien Befehle ausführen
  `-exec befehl "{}" ";"` $\Longleftarrow$ Ohne Rückfrage
  oder
  `-ok befehl "{}" ";"` $\Longleftarrow$ Mit Rückfrage

Dateien suchen und finden – `find` (3)

```
find . -name "*.txt" -user student -atime -7 -ok cat "{}" ";"
```

- Sucht alle Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen, die die Endung .txt haben, dem Benutzer `student` gehören und höchstens sieben Tage alt sind. Der Inhalt der gefundenen Dateien wird nach Rückfrage mit dem Kommando `cat` ausgegeben.
- Einige Suchbedingungen von `find`:

`-name dateiname`	Sucht Dateien mit dem Namen `dateiname`.
`-iname dateiname`	Ignoriert Groß- und Kleinschreibung.
`-perm 0000`	Sucht Dateien, die die Zugriffsrechte 0000 besitzen.
`-amin [+-]minuten`	Letzte Änderung vor mehr bzw. weniger `minuten` Minuten.
`-mtime [+-]tage`	Letzte Änderung vor mehr bzw. weniger `tage` Tagen.
`-user benutzername`	Dateien, die dem Benutzer `benutzername` gehören.

Dateien suchen und finden – `locate`

```
locate [Option] ...[Suchmuster] ...
```

- `locate` sucht nicht direkt in den Verzeichnissen, sondern in einer zuvor angelegten Datenbank. Diese enthält alle Dateien auf dem Computer mit ihren Pfaden.
- Ein Suchlauf mit `locate` dauert in der Regel nur wenige Sekunden.
- Im Gegensatz zu `find` sehr eingeschränkte Möglichkeiten, Suchkriterien anzugeben.
- Informationen zu Dateigröße, Zugriffsrechten, Besitzer usw. hält die Datenbank von locate nicht vor.
- Suchanfragen können aber Wildcards der Shell enthalten.
- Die Datenbank von `locate` muss regelmäßig aktualisiert werden, sonst sind die Einträge veraltet. Aktualisierung der Datenbank $\Longrightarrow$ `updatedb`.

Beispiel zu `locate`

- Beispiel für einen Suchlauf mit `locate`:

```
user@server:~$ locate *texte*index.t[eo]?
/home/user/texte/Diplomarbeit/DA_bibtech/index.tex
/home/user/texte/Diplomarbeit/DA_bibtech/index.toc
/home/user/texte/Diplomarbeit/DA/index.tex
/home/user/texte/Diplomarbeit/DA/index.toc
/home/user/texte/Diplomarbeit/index.tex
/home/user/texte/Master-Thesis/index.tex
/home/user/texte/Master-Thesis/index.toc
/home/user/texte/MMS-Abgabe/index.tex
/home/user/texte/MMS-Abgabe/index.toc
```

Weitere Suchprogramme – `whereis` und `which`

- whereis sucht in den Standardverzeichnissen nach ausführbaren Dateien, Konfigurationsdateien, Quellcode und Hilfeseiten (man-Pages).
- Mit den Optionen `-b`, `-m` und `-s` kann festgelegt werden, dass nur nach ausführbaren Dateien, Hilfeseiten bzw. Quellcode gesucht wird.

```
user@server:~$ whereis -bm top
top: /usr/bin/top /usr/share/man/man1/top.1.gz
```

- Das Kommando `which` sucht nach Programmen in den Verzeichnissen, die sich in der Umgebungsvariable `$PATH` befinden.

```
user@server:~$ which firefox
/usr/bin/firefox
```

Nächste Übung:

9.11.2007