2 Infrastruktur: Netzwerk, SecureShell und Prozesse

GEMEINSAME ÜBUNG

2.1 Vernetzung

Alle LINUX-Rechner am Institut sind eng miteinander vernetzt. Praktische Konsequenzen für den Benutzer sind:

- man kann sich überall einloggen und hat immer die gleiche Umgebung (Home-Dir, etc.). Home-Directories aller Mitarbeiter stehen unter `'/homes/<user>'`

```
cip1 % ls /homes/schiel
cip1 % ls /homes/jmh          (= Jonathan Harrington)
...
```

- man kann von überall auf (fast) alle Daten zugreifen (virtuelles Filesystem). Die sogenannten Daten-Platten sind Bereiche, wo grössere Datenmengen für Projekte gespeichert und bearbeitet werden (das Home-Dir sollte damit nicht belastet werden!). Sämtliche Daten-Platten sind unter dem Pfad `'/data/data#'` oder `'/raid/tera#'` zu sehen (`#` = Nummer der Datenplatte).

```
cip1 % ls /data/data16
cip1 % ls /raid/tera5
...
```

Wo die Daten wirklich liegen, d.h. auf welchem Server oder Rechner, ist für den Benutzer nicht wichtig. Er muss nur den logischen Pfad und Dateinamen wissen.

Oft verwenden Projekte mit mehreren Mitarbeitern solche Bereiche für die gemeinsame Arbeit, z.B. `/raid/tera5/ALC`.

2.2 Email

Jeder Benutzer hat automatisch auch eine Mailbox mit der Adresse `<user>@phonetik.uni-muenchen.de` (Innerhalb des Instituts reicht es, nur an die Adresse '`<user>`' zu schicken, z.B. an 'schiel'.)

Das einfachste Email-Programm für die Benutzung in einem Terminal ist 'pine'. 'Pine' hat viele Features, die aber sehr gut im Programm selber erklärt werden (Online Hilfe).

```
cip1 % pine
```

'Pine' hat den Vorteil, dass es innerhalb eines Terminal-Windows läuft. Das bedeutet, dass man auch von außerhalb des Instituts sich per 'ssh' (s.u.) einloggen kann und seine Email bearbeiten.

2.3 Secure Shell, ssh, Rechnen auf anderen Computern

In der praktischen Arbeit ist es oft notwendig, sich auf einem anderen Rechner einzuloggen (z.B. um dort ein länger dauerndes Programm zu starten). Mit dem Befehl 'ssh' (secure shell) kann man sich jederzeit an einem beliebigen anderen Rechner einloggen.

```
cip1 % ssh linux31
...
linux31 % exit
cip1 %
```

Ein Rechner wird immer mit dem Befehl 'exit' verlassen. Dann landet man wieder an dem Rechner, von wo man den Befehl 'ssh' eingegeben hatte. Im Zweifelsfalle gibt man den Befehl 'hostname' ein, um zu sehen, auf welchem Rechner man sich gerade befindet.

'ssh' verschlüsselt sämtliche Daten, die zwischen den beiden Rechnern hin und her gehen. Dadurch kann man auch gefahrlos Passwörter eintippen, ohne befürchten zu müssen, dass diese 'abgehört' werden.

SSH gibt es auch für Windows:

```
http://putty.freemirror.de
```

Mit 'ssh' können auch Befehle auf einem anderen Rechner gestartet werden, ohne dort erst eine Shell zu starten, z.B.

```
cip1 % ssh linux31 who
...
```

gibt die Liste der auf dem Rechner linux31 gerade eingeloggten Benutzer aus.

Bevor man auf einem fremden Rechner ein rechenintensives Programm startet, sollte man sich vergewissern, dass man niemanden stört. Der Befehl who zeigt, ob jemand direkt an dem Rechner arbeitet:

```
cip1 % ssh linux31
...
linux31 % who
schiel   :0           2009-12-02 09:31 (console)
schiel   pts/0        2009-12-02 09:31
ulrike   :1           2010-01-20 15:07 (localhost)
ulrike   pts/8        2010-01-20 15:09
ekurs2   pts/15       2010-03-16 15:02 (cip1.phonetik.uni-muenchen.de)
linux31 % exit
cip1 %
```

In diesem Beispiel sind auf dem Rechner linux31 sogar zwei Benutzer, 'schiel' und 'ulrike', auf beiden Konsolen eingeloggt (:0 und :1). Auf so einem Rechner sollte man keine zusätzlichen Programme starten, weil sonst das Arbeiten an diesem Rechner zu langsam wird.

```
cip1 % ssh linux60
...
linux60 % who
ekurs2   pts/15       2010-03-16 15:02 (cip1.phonetik.uni-muenchen.de)
ema      pts/2        2010-03-15 13:04 (linux29.phonetik.uni-muenchen.de)
linux60 %
```

In diesem Beispiel ist kein Benutzer direkt am Rechner eingeloggt (kein :0 oder :1), aber ein anderer Benutzer 'ema' ist ebenfalls per ssh eingeloggt. Um nun zu prüfen, ob dieser Benutzer 'ema' den Rechner bereits auslastet, gibt man den Befehl 'top' ein.

```
linux60 % top
top - 15:09:15 up 42 days,  3:06,  4 users,  load average: 0.06, 0.04, 0.09
...
```

Der Befehl top zeigt in Echtzeit die Auslastung des Rechners an. Die 'load average' ist hier nahe bei Null, d.h. der Rechner wird nicht benutzt. Ist die load average bei ca. 1 oder höher, ist der Rechner bereits voll ausgelastet und sollte nicht zusätzlich mit einem Programm belastet werden.

Das Programm top beendet man mit der Eingabe von 'q'.

2.4 Secure Copy scp

Mit dem Befehl 'scp' (secure copy) kann man sehr effektiv Dateien von einem Rechner holen oder zu einem Rechner schicken ('remotehost' = anderer Rechner). Auch dazu muss man sich bei dem fremden Rechner als zugelassener User anmelden.

```
cip1 % scp <localefile> <remotehost>:<remote file>
```

z.B.:

```
cip1 % scp testfile1 linux31:/homes/ekurs2/TESTFILE1
```

'scp' fragt ev. nach dem Passwort, wenn der Benutzer nicht als 'trusted' eingetragen ist. Um das File unter einem anderen Benutzernamen zu kopieren, stellt man wie in einer Email-Adresse, den Namen vor den 'remotehost':

```
cip1 % scp testfile1 ekurs@linux31:/homes/ekurs2/TESTFILE1
Password:
```

SCP für Windows:

`http://winscp.net`

2.5 Starten von Programmen 'im Hintergrund'

Normalerweise ist ein Programm mit der Kommandozeile derart verbunden, dass die Shell wartet, bis das Kommando beendet wurde. Oft will man länger laufende Programme, die keine interaktive Bedienung von der Kommandozeile erfordern, starten und mit der Shell weiterarbeiten (oder sich ev. sogar ausloggen und nach Hause gehen), während das programm fleissig weitermacht.

Man nennt dies 'ein Programm in den Hintergrund starten' (background process). Dazu schließt man einfach die Kommandozeile mit dem Zeichen '&' ab:

```
cip1 % command &
[1] 2454
cip1 %
```

Die Shell gibt aus, dass es sich um den ersten Prozess (job) im Hintergrund handelt ([1]) und die Prozessnummer (2454). Dann meldet sie sich sofort wieder mit dem Prompt , egal wie lange das Programm command braucht, um fertig zu werden. Sobald das Programm beendet, gibt die Shell eine Meldung aus:

```
cip1 % [1]    Done                          command
```

Da ein Programm, das im Hintergrund läuft, nicht mehr von der Tastatur beendet werden oder unterbrochen werden kann (Strg-C), muss man den Befehl 'kill' verwendet, welcher ein Signal an den Prozess im Hintergrund schickt.

Entweder

```
cip1 % kill <prozess-nummer>
```

oder:

```
cip1 % kill %<job-nummer>
```

Beispiel:

```
cip1 % sleep 120 &
[1] 12826
cip1 % jobs
[1]  + Running                       sleep 120
kill %1
cip1 % kill %1
[1]    Terminated                    sleep 120
```