– Praktikumsaufgabe 7 –

Thema: *Netzoperationen, fork()*

Zielstellung: Erlernen typischer Unix-Kommandos zum Umgang mit entfernten Rechnern. Kennenlernen der Nutzung von `fork()` im C-Programm.

Spielen Sie mit den im folgenden aufgeführten Kommandos:

1. Interaktives Arbeiten mit entfernten Rechnern:
   - `ssh` (*Secure Shell)*)
     - Kommandoausführung
     - X-Forwarding
   - `rdesktop` (*Remote Desktop*)
   - `telnet`
2. Für die Übertragung von Dateien kommen andere Kommandos zum Einsatz:
   - `scp`
   - `ftp` (z. B. ftp.tu-chemnitz.de)
   - `wget`
3. Nutzerinformationen erhalten Sie mittels
   - `who` (eingeloggte Nutzer)
   - `w` (eingeloggte Nutzer, was diese tun)
   - `last` - Liste der zuletzt eingeloggten Nutzer
   - `finger` - Zuordnung Nutzerkennzeichen → Klarnamen
4. allgemeine Netzinformationen:
   - `ping`
   - `nslookup`
   - `/usr/sbin/traceroute`
   - `netstat` – zeigt geöffnete Verbindungen des Rechners an

5. Schreiben Sie ein Skript, das eine Pfadangabe entgegennimmt und dann per `wget` alle Praktikumsaufgaben in diesen Pfad kopiert!

6. Konfigurieren und aktivieren Sie `ssh-agent`, um sich per ssh ohne Passwortabfrage auf allen Unix-Rechnern der Fakultät einloggen zu können.

   Dazu müssen Sie:

   a) Ein OpenSSH-Schlüsselpaar generieren, das von einer (guten) Passphrase geschützt ist:

   ```
   $ ssh-keygen
   ```

   Es entstehen ein öffentlicher Schlüssel, `id_rsa.pub` sowie ein privater Schlüssel `id_rsa`, abgelegt normalerweise im Verzeichnis `~/.ssh`. (Sie sollten die voreingestellten Dateinamen nicht ändern!)

   b) Den öffentlichen Schlüssel müssen Sie auf alle Rechner verteilen, auf die Sie sich verbinden wollen, und zwar muss er an die Datei `~/.ssh/authorized_keys` *angehangen* werden, etwa so:

   ```
   $ cat ~/.ssh/id_rsa.pub | ssh nkz@zielhost "cat - >> ~/.ssh/ ↩
      authorized_keys"
   ```

   oder mittels des Kommandos `ssh-copy-id`.

   c) Der private Schlüssel verbleibt auf dem Rechner, von dem aus Ihre Verbindung abgeht.

   d) Sie müssen den so genannten `ssh-agent` beim grafischen Einloggen starten. Auf den Rechnern im Labor 136c ist dies bereits erfolgt, wie ein `ps`-Aufruf zeigt.

   e) Sie müssen dem `ssh-agent` Ihre Identität mitteilen, dies geschieht gewöhnlich beim (grafischen) Einloggen mittels des Programmes `ssh-askpass`. Nutzen Sie KDE, dann können Sie dessen Autostart-Fähigkeit ausnutzen, indem Sie ein Skript in `~/.kde4/Autostart` plazieren mit folgendem Inhalt:

   ```
   #!/bin/bash
   ssh-add </dev/null
   ```

   Wenn alles reibungslos funktioniert, können Sie sich (nach einem Aus- und Einloggvorgang) auf dem Zielrechner einloggen, ohne eine Passphrase einzugeben:

   ```
   robge@isys121:~$ ssh ilux150
   Last login: Wed Nov 25 16:25:51 2009 from
   isys121.informatik.htw-dresden.de
   Have a lot of fun...
   robge@ilux150:~$
   ```

   Wenn Sie keine grafische Oberfläche nutzen, sondern z. B. mittels eines Terminalemulators auf einem Linux-Rechner wie `ilux150` arbeiten, ist der `ssh-agent` ebenfalls

von Nutzen:

```
robge@ilux150:~> exec ssh-agent bash
robge@ilux150:~> ssh-add -l
The agent has no identities.
robge@ilux150:~> ssh-add
Enter passphrase for /u/robge/.ssh/id_rsa:
Identity added: /u/robge/.ssh/id_rsa (/u/robge/.ssh/id_rsa)
robge@ilux150:~> ssh-add -l
2048 40:7e:af:e6:e7:6c:9a:24:df:a0:8f:27:c2:fb:32:13 /u/robge/.ssh ↩
   /id_rsa (RSA)
robge@ilux150:~> ssh ilpro122
Linux ilpro122 2.6.32-5-amd64 #1 SMP Thu Nov 3 03:41:26 UTC 2011
[...]
robge@ilpro122:~$
```

Mehr Informationen finden Sie hier (u. a.) hier:
http://www.ibm.com/developerworks/library/l-keyc.html

7. Schreiben Sie ein C-Programm, das *genau* drei Prozesse erzeugt. Die Prozesse sollen ihren PID ausgeben und sich dann beenden.

   Hinweis: der Systemruf `pid_t getpid(void);` liefert die PID des rufenden Prozesses zurück. Sie benötigen darüber hinaus `fork()` und `exit()`.