Übungen zu EDV für Physikerinnen und Physiker (physik131) WS 2010/2011

Jörg Pretz und Daniel Elsner

3. Übung Woche: 31.10.-4.11.2009

Anmerkungen

- Vor Beginn der Übungen sollen Sie die Vorbereitungsaufgaben bearbeitet haben. Falls Probleme aufgetreten sind, besprechen Sie diese mit Ihrem Tutor.
- Das Nacharbeiten der letzten Vorlesung mittels der Vorlesungsfolien und des Skripts ist selbstverständlich.
- Legen Sie sich ein neues Verzeichnis mit `mkdir uebung_03` an, in dem Sie heute arbeiten.

Vorbereitungsübungen

Lösen Sie die folgende Aufgaben um zu überprüfen ob den bisherigen Stoff im Griff haben:

1. Wie finden Sie heraus, in welchem Verzeichnis Sie sich gerade befinden?
2. Wie heisst Ihr Rechner?
3. Wie finden Sie heraus, wer gerade (ausser Ihnen selber) auf Ihrem Rechner angemeldet ist?
4. Was finden Sie heraus, wie gross eine bestimmte Datei ist?
5. Erstellen Sie mit einem Konsolenbefehl im Ordner `uebung_03` die leere Datei `test.txt`.
6. Sie wollen die Datei `test.txt` von `/home/someuser/uebung_03` nach `/home/someuser` kopieren. Sie befinden sich in `/home/someuser/uebung_03`. Geben Sie zwei `cp` Befehle an, die dies erledigen; einmal soll das Ziel als *relativer* und einmal als *absoluter* Pfad angegeben werden.
7. Ändern Sie die Rechte der Datei `test.txt` so, dass jeder Nutzer in die Datei schreiben kann.
8. Was bewirken die Zeichen `*` und `?` in Verbindung mit UNIX Kommandos?
9. Welches Programm erzeugt im Moment die höhste Speicherauslastung auf Ihrem Rechner?

Lernziele

- Eingabe- und Ausgabeumleitung
- Pipelines
- Komprimierung und Archivierung
- Verteiltes Arbeiten
- Shell-Skripte

Präsenzübungen

In dieser Übungseinheit behandeln wir weitere Unix-Befehle und zusammengesetzte Befehle, sog. Pipelines. Für alle neuen Befehle sollten Sie sich selbständig mit der Unix-Hilfe über die Benutzung der Befehle informieren. Ebenso sollen Sie weiter diese Befehle für Ihre eigene Referenz und den Abschlussbericht dokumentieren.

1. **Weitere Unix-Befehle**

   (a) Informationen über das Unix-System.
   - `ps` Auskunft über alle gestarteten Prozesse
   - `df` Belegung des Festspeichers (Festplatten, DVDs, USB-Sticks,..)
     `df /home` zeigt die Belegung der Homeverzeichnisse an
   - `du` Grösse von Dateien und Verzeichnissen
     du -k filename gibt die Grösse von filename in Kilobytes an

   (a) Befehle für den Umgang mit Dateien.
   - `less` Anzeigen von Dateien (mit Pfeiltasten navigieren, 'q' zum Beenden)
   - `cat` Kopieren von Dateien in die Standardausgabe (STDOUT)
   - `head` Anzeigen von Anfangszeilen einer ASCII-Datei
   - `tail` Anzeigen der letzten Zeiten einer ASCII-Datei
   - `grep` Filtert Zeilen in ASCII-Dateien (siehe extra Übung unten)
   - `sort` Sortiert Zeilen in ASCII-Dateien
   - `uniq` Filtert doppelte Zeilen in sortierten ASCII-Dateien
   - `find` Sucht nach Dateien (siehe extra Übung unten)

2. **Eingabe- und Ausgabeumleitung**

   Ausgaben von Unix-Befehlen (z.B. `ls`) werden in der Regel direkt auf den Bildschirm geschrieben. Technisch wird die Ausgabe auf der sog. Standardausgabe (kurz STDOUT) geschrieben, welche mit dem Bildschirm verbunden ist. Daneben gibt es weiter die sog. Standardeingabe (kurz STDIN), welche mit der Tastatur verknüpft ist. Für Fehlermeldungen gibt es noch die Fehlerausgabe STDERR. Diese Ein-/Ausgaben können auf Dateien umgelenkt werden; dazu werden die Operatoren „>“und „<“verwendet. So schreibt z.B. `ls > standardausgabe.txt` den Output in die Datei standardausgabe.txt. Probieren Sie mal diesen Befehl aus und schauen Sie sich das Ergebnis mit `less standardausgabe.txt` wieder an.

3. **Handling von Textdateien Dateien**

   Der grep Befehl kann mit Hilfe von regulären Ausdücken zum Filtern von Textdateien verwendet werden. Nach einem bestimmten Zeichen aus einer Liste lässt sich durch durch die Option [abc] realisieren (hier wird nach a oder b oder c gesucht). Ein Suchbereich für ein Zeichen lässt sich duch einen Bindestrich definieren, z.B: [a-e] oder [1-5]. Die Anzahl der Zeichen nach denen gesucht werden soll lässt sich durch {..} definieren. Z.B. sucht {n} nach genau n Zeichen und {n,m} nach mindestens n und maximal m Zeichen. Mit `grep -E [A-Z]{1,2} Datei.txt` suchen Sie nach Buchstabenfolgen mit mindestens einem und höchstens zwei Buchstaben zwischen A und Z in Folge.
   Für diese Übung laden Sie sich bitte die Datei `Autokennzeichen.txt` von der **eCampus** Webseite herunter (im Ordner Uebung 3). Filtern Sie nun mit dem grep Befehl und durch die richtige Kombination von Optionen nur die deutschen Autokennzeichen aus der Datei heraus. Schreiben Sie die Audgabe direkt in eine Datei mit Namen `Auto_out_D.txt`.

4. **Pipelines**

   In der Vorlesung wurden kurz zusammengesetzte Befehle oder sog. Pipelines angesprochen. In diesem Block sollen Sie einige Pipelines selber ausprobieren und so die Zusammenhänge erlernen. Pipelines verknüpfen Ausgaben und Eingaben von Programmen. So kann die Ausgabe von ls nicht nur in eine Datei umgelenkt werden, mit dem Befehl grep lässt sich auch ein Filter einbauen. Wenn Sie eine komplexe Pipeline erstellen können Sie Schritt für Schritt vorgehen und sich die Ausgabe jeden Befehls anschauen, bevor Sie diese in ein weiteres Programm ,,pipen"!
   Arbeiten Sie folgende Beispiele durch: Laden Sie die Dateien namen.dat und zahlen.dat von der **eCampus** Seite in ihr Arbeitsverzeichnis.

   - Sortieren Sie mit den Befehlen `cat` und `sort` den Inhalt der Datei `namen.dat` mittels einer Pipeline in der Konsole.
   - Sortieren Sie nun die Datei in umgekehrter alphabeticher Reihenfolge.
   - Sortieren Sie nur die Mitglieder der Familie Meier (Tipp: `grep`).
   - Sortieren Sie die Datei `zahlen.dat`. Warum sieht der Output nicht so aus wie erwartet? Schauen Sie sich nochmal das Manual für `sort` an und verwenden Sie die richtige Option.
   - Schreiben Sie die Ausgabe des letzten Befehles direkt in eine Datei mit Namen `zahlen_out.dat`. Überprüfen Sie den Inhalt der neuen Datei.

5. **Komprimierung und Archivierung**

   Unter Linux werden Ihnen oft Dateien mit den Endungen `.tar, .tgz, .tar.gz`, oder `.tar.bz2` begegnen. Hierbei handelt es sich meist um Dateienarchive. Das Pendant aus der Windows-Welt sind z. B. zip oder rar Dateien, die Sie sicherlich bereits kennen. Beachten Sie, dass zip Dateien Archive sind, d.h. eine Zusammenfassung mehrerer Dateien. Darüberhinaus komprimiert zip die Archive gleich mit. Unter Unix sind das Archivieren und das Komprimieren üblicherweise zwei getrennte Vorgänge mit denen Sie in dieser Übung vertraut gemacht werden.

   Die Befehle *gzip* und *bzip2* komprimieren Dateien. Überfliegen Sie die man-Dateien zu beiden Befehlen, um sich damit vertraut zu machen. *gzip* ist das Standardtool zur Komprimierung unter Unix. *bzip2* hat eine bessere Komprimierungsrate. Selbstverständlich

finden Sie bei gängigen Linuxdistributionen auch Frontendprogramme die meherere Formate beherschen, z. B. `ark` unter KDE.

Laden Sie die Datei `cc++.tar` von **eCampus** in Ihr Arbeitsverzeichnis. Komprimieren Sie die Datei und vergleichen Sie die Dateigröße vor und nach der Komprimierung mit *gzip* und *bzip2*. Der *gzip* Befehl erlaubt verschiedene Komprimierungsstufen. Vergleichen Sie den Default mit der Stufe 1 und 9. Testen Sie die Art der Komprimierung mit dem *file* Befehl. Die Option `-d` (Englisch: decompression) erlaubt das Dekomprimieren einer Datei. Alternativ gibt es die Befehle *gunzip* und *bunzip2*. Achten Sie auf die Dateiendungen, die die Komprimierungsprogramme an den Dateinamen anhängen. Protokollieren Sie Ihre Messung in der Datei `komprimierung.txt`.

Mit dem *tar* Befehl können Sie ganze Verzeichnisse und die enthaltenen Dateien archivieren. `tar -cf NAME.tar VERZEICHNIS` erstellt ein Archiv mit dem Namen `NAME.tar` vom Verzeichnis `VERZEICHNIS`. Mit `tar -tf NAME.tar` wird der Inhalt des Archivs `NAME.tar` angezeigt und mit `tar -xf NAME.tar` wird das Archiv im Arbeitsverzeichnis wieder entpackt. Bei der Archivierung können Sie auch Dateien komprimieren lassen. Lesen Sie in den manpages nach, wie die Option zur Komprimierung mit *gzip* und *bzip2* heißen und wie Sie das Archiv in einem anderen Verzeichnis als dem Arbeitsverzeichnis entpacken können.

Erstellen Sie ein Archiv von Ihrem Verzeichnis `uebung_03`. Benutzen Sie *tar*, um sich den Inhalt anzeigen zu lassen. Komprimieren Sie das Archiv mit dem *gzip* Befehl und entpacken das Archiv in das Verzeichnis `/tmp`. Erstellen Sie ein zweites Archiv von dem Verzeichnis und benutzen Sie die bzip2-Komprimierung von *tar*. Benutzen Sie `tar.bz2` als Dateiendung, um Ihnen selber eine Tipp zu geben, welche Komprimierung benutzt wurde. Vergleichen Sie die Dateigrößen vor und nach der Komprimierung und schreiben Sie die Messung in die Datei `komprimierung.txt`.

Am Schluß der Vorlesung werden Sie ein Archiv mit allen Berichtsdateien erstellen und bei den Kursleitern einreichen. Dokumentieren Sie daher den *tar* Befehl gut!

6. **Verteiltes Arbeiten** Listen Sie zunächst alle Verzeichnisse und Dateien in Ihrem Homeverzeichnis und speichern Sie diese in einer Datei: `ls -lR ~ > ~/ls-lR.host1`

   Benutzen Sie den Befehl *hostname* um den Namen Ihres Rechners in Erfahrung zu bringen. Loggen Sie sich mittels *ssh* auf dem Rechner Ihres Nachbarn ein: `ssh hostname`. Erfragen Sie hierzu den Namen des Rechners von Ihrem Nachbarn. Ist Ihr Username immer noch der gleiche (*whoami* Befehl)? Listen Sie alle eingeloggten User mit dem *who* Befehl auf.

   Listen Sie nun erneut alle Verzeichnisse und Dateien in Ihrem Homeverzeichnis und speichern Sie diese in einer Datei: `ls -lR ~ > ~/ls-lR.host2` Vergleichen Sie diese Datei mit der Datei von oben mittels des *diff* Befehls. Beide Dateien sollten bis auf einen Eintrag (welcher und warum ?) gleich sein. In unserem Fall teilen sich alle Rechner in CIP-Pool das gleiche Homeverzeichnis.

   Loggen Sie sich nun im CIP-Pool der Astronomie ein, wenn Sie im CIP-Pool der Physik sitzen `ssh cipserv1.astro.uni-bonn.de` und umgekehrt
   `ssh cipw01.physik.uni-bonn.de`.

   Wiederholen Sie den relevanten Teil der Übung (*hostname*, *whoami*, Erzeugen und Vergleich der Dateiliste). In diesem Fall sollen Sie die neu erstellte Datei mit der Dateiliste

aus der vorhergehenden Übung (*ls-lR.host1*) vergleichen. Diese müssen Sie vom jeweils anderen CIP-Pool mit Hilfe des `scp` Befehls kopieren:

```
scp cipw01.physik.uni-bonn.de: ~/ls-lR.host1 .
```

- wenn Sie sich vom Physik CIP-Pool im Astronomie CIP-Pool eingeloggt haben, kopieren Sie hiermit die Datei von Ihrem HOME-Verzeichnis in der Physik zur Astronomie.

```
scp cipserv1.astro.uni-bonn.de:~/ls-lR.host1 .
```

- wenn Sie sich vom CIP-Pool der Astronomie in den CIP-Pool der Physik eingeloggt haben, kopieren Sie hiermit die Datei von Ihrem HOME-Verzeichnis in der Astronomie zur Physik.

Jetzt sollten die Dateien sehr unterschiedlich sein, da die Homeverzeichnisse zwischen den beiden CIP-Pools nicht geteilt werden.

7. **Shell - Skript**

   In der Vorlesung wurden ein paar einfache Bespiele für die Progammierung eines sogenannten Shell-Skriptes angegeben.

   Erzeugen Sie mit Hilfe einer `for`-Schleife in einem Shell-Skript folgende Ausgabe auf der Konsole:

```
Meine Lieblingsfarbe ist blau. Also fahre ich blaue Fahrraeder.
Meine Lieblingsfarbe ist gelb. Also fahre ich gelbe Fahrraeder.
Meine Lieblingsfarbe ist gruen. Also fahre ich gruene Fahrraeder.
Meine Lieblingsfarbe ist rot. Also fahre ich rote Fahrraeder.
```

   Hiefür benötigen Sie den richtigen Syntax für die `for`-Schleife und den `echo` Befehl für die Ausgabe. (Tipp: Definieren in der Schleife eine Variable mit den verschiedenen Farben.)

   Innerhalb einer `for`-Schleife können Sie auch eine Aufzählung generieren. Zum Beispiel erzeugt folgendes Skript die Ausgabe:

```
1
2
3
```

```
#!/bin/bash
for[ Counter = 1; Counter< 4; Counter++ ] ; do
      echo $Counter
done
```

   Modifizieren Sie dieses Skript so, dass die Zahlen zwischen 2 und 20 ausgegeben werden. Zusätzlich soll Ihr Vorname ausgegeben werden, wenn der Zähler auf 10 steht (Tipp: `if`-Bedingung, siehe Vorlesung.).

Berichtsaufgaben

1. **Dokumentation**

   Sie sollen weiterhin Ihre Datei `commands.txt` für den Bericht erweitern. Diese Aufgabe wird nicht mehr explizit gestellt.

Alle in den Übungen erstellten Dateien und Lösungen gehören zu den Berichtsaufgaben und müssen mit dem Bericht am Ende des Semesters abgegeben werden.

2. **Analyse von Pipelines**

   Betrachten Sie die Pipeline:

   ```
   find ~/uebung_03 -type f -exec du -k {} \;| sort -n -r
   ```

   Lesen Sie die manpages zum *find* Befehl, um sich mit dem Befehl und den hier benutzten Optionen vertraut zu machen. Speziell für die Option `-exec` sollten sie sich den Abschnitt „ACTIONS“ anschauen. Wenden Sie die Pipeline auf verschiedene Verzeichnisse an. Analysieren Sie die Pipeline und erklären Sie schriftlich (Bericht) was in jedem Einzelschritt der Pipeline passiert.

   Konstruieren Sie folgende Befehle zu einer Pipeline:

   (a) Das Kommando `ps aux` gibt Ihnen eine lange Liste aller Programme, die gerade auf ihrem Rechner laufen. Verbinden Sie dieses Kommando über eine Pipeline mit `grep` damit **nur** Prozesszeilen erscheinen, in denen Ihr Benutzername enthalten ist.

   (b) Fügen Sie Ihrer Pipeline aus vorheriger Aufgabe noch einen *sort* Befehl an. Er soll Ihre Prozesse nach der PID (zweite Spalte der *ps* Ausgabe) numerisch sortieren.

   (c) Wiederholen Sie die letzte Teilaufgabe und speichern Sie die Ausgabe in einer Datei `myprocesses.txt` ab (`STDOUT` Umlenkung).