Praktikum Betriebssysteme – Übung 2

Auch in der heutigen Übung geht es um die Arbeit mit der UNIX- bzw. GNU/Linux Kommandozeile. Die UNIX-Philosophie wird bei der Arbeit auf der Kommandozeile besonders deutlich. Ein wenig verkürzt besagt diese Philosophie:

- *Write programs that do one thing and do it well*
- *Write programs to work together*
- *Write programs to handle text streams, because that is a universal interface.*

Wie in der Vorlesung erwähnt kommt diese Idee daher, dass man viele kleine hilfreiche Programme für die Kommandozeile geschrieben hatte und diese durch geschickte Nutzung (die Ausgabe des einen ist die Eingabe des nächsten) auch zur Lösung komplexer Probleme nutzen konnte.

Beispiele für Kommandozeilenprogramme

Neben den Beispielen, die Sie letzte Woche bereits kennengelernt haben (`ls`, `rm`, `chmod`, usw.) finden Sie im folgenden weitere Beispiele für kleine aber wichtige Programme, die auf der Kommandozeile sehr häufig benutzt werden. Falls Sie die Programme noch nicht kennen, sehen Sie sich die Manpage an (oder benutzen Sie das Programm `whatis`) um herauszufinden, was diese tun.

Programme	**Anwendungsbereich**
`ls`, `cd`, `pwd`, `cp`, `mv`, `rm`, `mkdir`, `touch`	Datei- und Verzeichnismanipulation
`cat`, `head`, `tail`, `less`, `grep`	Zeigt Inhalt von Dateien an
`cut`, `sort`, `uniq`, `tr`, `wc`	Textmanipulation (filtern, sortieren, ...)
`ps`, `kill`, `pkill`	Prozesse anzeigen oder beenden
`wget`, `curl`	Inhalte aus dem Internet laden
`su`, `sudo`	Programme als Administrator (root) bzw. anderer Benutzer ausführen

Eine schöne Übersicht finden Sie im Wiki der Webseite ubuntuusers.de.

Aufgabe 1 – Ausgabe umleiten

Eine grundlegende Philosophie UNIXoider Systeme besteht darin, Programme so zu gestaltet, dass diese jeweils nur eine konkrete Aufgabe erfüllen und die Ausgabe eines Programms die Eingabe eines anderen sein kann (das gilt natürlich vorwiegend für kleine Kommandozeilentools).

Jedes Programm besitzt drei Eingabe/Ausgabe-Kanäle, die im Normalfall an die Tastatur (Eingabe) und den Bildschirm (Ausgabe und Fehlerausgabe) gebunden sind. Mit Hilfe von *Umleitungen* (Operator `<` für die Eingabe und `>` für die Ausgabe) kann dies verändert werden.

- Leiten Sie die Ausgabe des Kommandos `ls -al /usr/bin` in eine Datei namens `dirlist.txt` um
- Geben Sie den Inhalt der Datei `dirlist.txt` auf der Kommandozeile aus
- Geben Sie nur solche Zeilen der Datei `dirlist.txt` auf der Kommandozeile aus, die den Text `zip` enthalten

Eine schöne Übersicht zu Umleitungen finden Sie im Wiki der Webseite ubuntuusers.de.

Aufgabe 2 – Pipes

Das Konzept der *Pipes* (`|`) hat die Arbeit auf UNIXoiden Systemen grundlegend verändert bzw. vereinfacht. Pipes kombinieren die zuvor kennengelernten Umleitungsoperatoren für Ein- und Ausgabe und erlauben es somit, Programme so zu verbinden, dass die Ausgabe des einen Programms direkt als Eingabe des anderen Programms verwendet wird.

1. Geben Sie alle Zeilen des Kommandos `ls -al /usr/bin` aus, die den Text `zip` enthalten. Nutzen Sie nun aber den Pipe-Operator, statt die Ausgabe von `ls` zuvor in eine Datei zu schreiben.
2. Geben Sie anschließend lediglich die Anzahl der gefundenen Zeilen aus, die den Text `zip` enthalten. (Tip: `man wc`)

Aufgabe 3 – Advanced Pipes

Auf dem Gitlab-Server finden Sie eine CSV-Datei. Diese enthält eine unsortierte Liste von Ländern und deren imaginäre Einwohnerzahl in unterschiedlichen Jahren.

Geben Sie die Liste der Länder in alphabethischer Reihenfolge auf dem Bildschirm aus. Es sollen nur die Länder angezeigt werden (nicht die Jahreszahlen oder Einwohner) und es soll kein Land mehrfach in der Ausgabe vorkommen.

Nutzen Sie dazu eine einzelne Eingabe in der Shell (“Oneliner”), also eine durch Pipes verbundene Aneinanderreihung von Tools.

Ursprüngliche Datei:

```
year,population,country
2021,7483742,Schweiz
1992,75922510,Deutschland
[...]
```

Resultat ihres “Oneliner”:

```
Afghanistan
Albania
Algeria
[...]
```

Aufgabe 4 – Shell Scripting (“oneliner”)

Shells unterstützen verhältnissmäßig mächtige Skriptinterpreter (vgl. IEEE Std 1003.1-2017). So können Sie in `bash` zum Beispiel mit Hilfe einer einfachen Schleife über den Verzeichnisbaum iterieren:

```
for i in /*; do echo "Verzeichnis: $i"; done
```

Schönder formatiert (und dadurch ohne die Zeilentrennzeichen (;) sieht das ganze so aus:

```
for i in /*
    do
    echo "Verzeichnis: $i"
    done
```

Eine Einführung in den Skriptinterpreter der Shell finden Sie **hier**.

Nutzen Sie `mkdir`, `touch` und `rm` zum Erstellen und Löschen von Verzeichnissen und Dateien.

- Erstellen Sie ein Verzeichnis namens `OSTemp` in Ihrem Heimatverzeichnis.
- Wechseln sie indas Verzeichnis `OSTemp` und erzeugen sie dort mit dem Befehl `touch` eine Datei mit dem Namen `TestFile.txt`.
- Erstellen Sie 100 Unterverzeichnisse `TestFolderXY` (`TestFolder0` – `TestFolder99`).
  (Tipp: Verwenden Sie eine Schleife.)
- Erstellen Sie in jedem der 100 Verzeichnisse `TestFolderXY` jeweils eine Datei `TestFileXY.txt`.
- Löschen Sie die Verzeichnisse und die Datei `TestFile.txt` mit dem Befehl `rm`

Aufgabe 5 – Shellscript (Datei)

Erstellen Sie ein einfaches Shell-Skript, dem beim Aufruf ein Parameter übergeben wird.

- Falls der Parameter fehlt, geben Sie eine kurze Hilfe aus und beenden das Programm mit dem Rückgabewert 1.
- Ist der Parameter vorhanden, ermitteln Sie dessen Länge (Anzahl der Zeichen), geben diese auf dem Bildschirm aus und beenden das Programm mit dem Rückgabewert 0.

Optional, falls Sie bereits vor der Zeit fertig sind:
Modifizieren Sie das Programm so, dass ihm beliebig viele Parameter übergeben werden können. Das Programm ermittelt den längsten Parameter und gibt dessen Länge aus.

Aufgabe 6 – Optional: Advanced Shellscript

Schreiben Sie ein Shellscript, das Informationen über die Hardware und das Betriebssystem ausgibt. Es gibt eine ganze Menge Informationen, die Sie anzeigen können (CPU, RAM, HDD, Geräte, CPU-Temperatur, Lüfterdrehzahl, Betriebssystem, IP-Adressen, Routen, usw., usw.).

Die folgenden Tools sind vielleicht interessant, falls Sie diese noch nicht kennen:

`lscpu`, `lsblk`, `lsusb`, `lspci`, `lsscsi`, `dmidecode`, `lshw`, `fdisk`, `hdparm`, Prozessinfos (`/proc` filesystem), ...

Hier ein Minimal-Beispiel:

```
Hardware Information
--------------------
* CPU model : AMD Ryzen 7 PRO 4750U with Radeon Graphics
* Memory    : 31315 MB (2777 MB free)

Operating system information
---------------------------
* Hostname  : azrael
* OS Kernel : Linux 6.6.28-1-lts x86_64
* Date/Time : Thu Apr 25 02:45:23 PM CEST 2024
* Uptime    : 6 days, 18 hours, 51 minutes
```

Sie können das Programm auch interaktiv gestalten, indem Sie den Benutzern eine Auswahl anbieten.

- `Drücke 1 für Hardwareinformationen`
- `Drücke 2 für Betriebssysteminformationen`
- `Drücke 3 für Netzwerkinformationen`
- `Drücke 9 um das Programm zu beenden`

Nach Auswahl und Anzeige der Informationen sollte der Benutzer wieder im Auswahlmenü landen. Es macht Sinn Funktionen zu nutzen, die dann im Falle der entsprechenden Auswahl aufgerufen werden.