Aufgabe 1

Einführung in die Shell, SVN und das Unix-Hilfesystem und ein erstes eigenes kleines Skript

Aufgabenstellung

In dieser ersten Aufgabe wollen wir unsere ersten Gehversuche in der Unix-Shell machen.

Am Anfang steht sinnvollerweise das Hilfesystem, das auch bei den späteren Übungen und beim praktischen Umgang mit dem Unix-Betriebssystem eine wichtige Rolle spielt und das vetraut machen mit den grundlegenden Shell-Befehlen.

Ein weiterer wichtiger Punkt ist das Versionsverwaltungssystem SVN, das wir in den Übungen benutzen werden, um die Lösungen einer Gruppe allen Gruppenmitgliedern zur Verfügung zu stellen und um die automatisierten Tests auf dem Server durchzuführen.

Als letzten Punkt wollen wir ein einfaches erstes kleines Skript erstellen und damit beginnen uns mit dem systematischen Testen von Programmen zu befassen.

Die Aufgabe gliedert sich daher in mehrere Teilaufgaben.

Teil 1: Hilfesystem und Arbeiten auf der Unix-Shell

Macht euch mit der Shell vertraut. Probiert dazu mindestens alle Befehle aus, die sich in der **Übersicht wichtiger Shell-Befehle** befinden. Wir gehen in den folgenden Übungen davon aus, dass ihr hiermit dann keine Probleme mehr habt (nicht auswendig lernen, aber verstehen und wissen, wo ihr ggf. nachsehen könnt). Besonders wichtig für die Übung sind alle Kommandos, die irgendetwas mit Hilfe zu tun haben.

Zusätzlich zu den Hilfe-Funktionen solltet ihr euch grundlegend auch mit der Navigation im Dateisystem und dem Anlegen, Kopieren, Verschieben und Ausgeben von Dateien und insbesondere auch dem Dateirechtemanagement unter Unix vertraut machen, da ihr dieses auch für alle späteren Übungen benötigt.

Gewöhnt euch rechtzeitig an konsequent auf der Konsole zu arbeiten - dies wird euch im weiteren Übungsverlauf die Arbeit deutlich erleichtern.

Teil 2: Umgang mit SVN auf der Kommandozeile

Die grundsätzliche Funktionsweise von SVN sollte euch aus den Übungen PS1 (und ggfs. PS2) bereits bekannt sein. Macht euch daher nun auch mit dem SVN-System von der Kommandozeile aus vertraut.

Probiert dafür alle Schritte aus, die in den **Hinweisen zu SVN** aufgeführt sind.

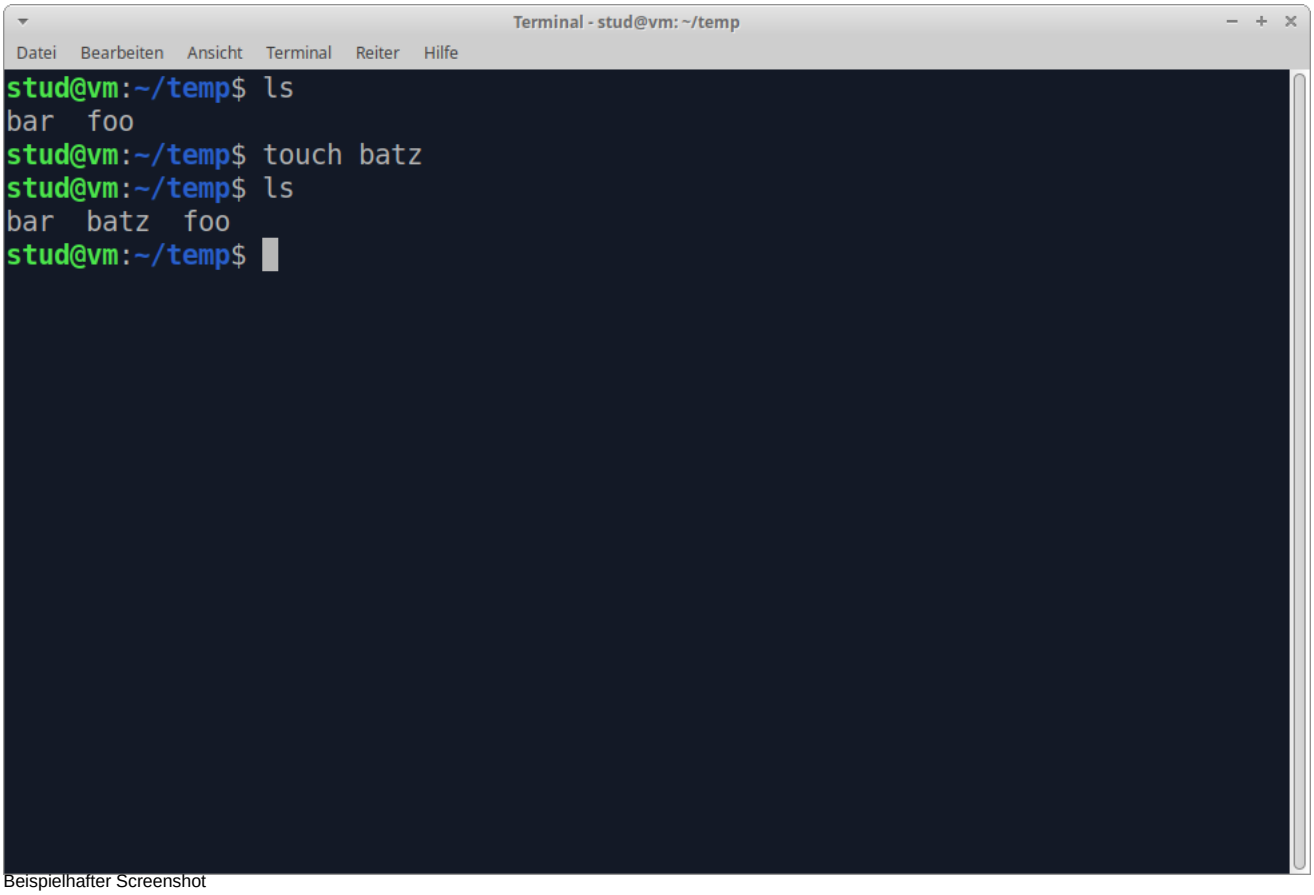
Benutzt das Verzeichnis `ueb01` für die erste Übung, das sich bereits in eurem persönlichen Repository befindet. Die Repositories wurden bereits eingerichtet. Die Einteilung und die entsprechenden Gruppennummern sind unter **Gruppeneinteilung** einsehbar.

Um nachvollziehen zu können, dass ihr euch mit SVN auf der Konsole ausreichend befasst habt und damit vertraut seid, ist es Aufgabenbestandteil, dass ihr zu den entsprechenden Schritten:

- Anlegen einer lokalen Arbeitskopie
- Hinzufügen und Übertragen einer bisher nur lokal vorhandenen Datei an den Server
- Aktualisierung der lokalen Arbeitskopie / der lokalen Dateien

einen oder mehrere entsprechende Screenshots der Konsole mit in das SVN-Repository eincheckt. Dabei müssen immer mindestens die entsprechenden Befehle und ihre Wirkung auf der Konsole erkennbar sein. (Überlegt euch hierbei auch, wie man zeigen kann, dass eine Datei bisher nicht auf dem Server vorhanden ist.)

Ein beispielhafter Screenshot (hier mit dem Nachweis, dass eine Datei namens `batz` zusätzlich zu bereits vorhandenen Dateien im Verzeichnis `temp` angelegt wurde), kann z.B. so aussehen:



Beispielhafter Screenshot

Legt zusätzlich zu dem/den Screenshot(s) eine Textdatei `svninfo.txt` bei, in der ihr kurz in eigenen Worten beschreibt was auf den Screenshots zu sehen ist und was dadurch auf dem SVN Server bzw. in der lokalen Arbeitskopie bewirkt wird, Ein beispielhafter Eintrag in dieser Datei kann wie folgt aussehen:

```
... die Datei xyz befindet sich nach dem Aufruf von
blafussel unter Versionskontrolle, d.h. ...
```

Legt die Screenshots und die ergänzende Textdatei in einen Unterordner mit dem Namen `svn` unter `ueb01` ab und achtet darauf den Ordner und alle Dateien auch mit ins SVN einzuchecken.

Dieser Aufgabenteil geht mit **30%** in die Bewertung der Aufgabe mit ein.

Teil 3: Das erste kleine Skript

Da ihr ja bereits zu Beginn dieser Aufgabe diverse Shell-Befehle auf der Konsole ausprobiert habt, wollen wir einige davon auch gleich zu einem kleinen Skript zusammenfassen.

Das Skript soll in dem Ordner in dem es sich befindet zunächst einen Unterordner mit dem Namen *info* anlegen. Sollte der Unterordner bereits existieren, so soll dies ignoriert werden und **nicht** zu einer Fehlermeldung führen. (Um Herauszufinden wie das geht, bietet sich ein Studium der manpage des passenden Befehls an). Danach soll das Skript alle Dateien im aktuellen Ordner, die die Dateiendung *txt* haben in der natürlichen Sortierungsreihenfolge für Dateinamen (alphabetisch aufsteigend sortiert) in die Datei *mytextfiles.txt* im Unterordner *info* schreiben. Sofern diese Datei noch nicht existiert, soll sie angelegt werden, falls sie bereits existiert, so soll sie überschrieben werden. Die Fehlermeldung, die entsteht, wenn sich keine passenden Dateien im aktuellen Ordner befinden, darf hierbei ignoriert werden.

Hinweis: Ein Aufruf vor `sort` ist für die Lösung der Aufgabe nicht notwendig.

Beispiel

Wenn wir also einen Ordner haben, in dem sich die Dateien: *foo.txt*, *bar.txt* und *hallo.pdf* befinden, sieht die entstehende Datei *mytextfiles.txt* wie folgt aus:

```
bar.txt
foo.txt
```

Um euch Textdateien fürs Testen anzulegen, solltet ihr euch auch damit befassen, wie ihr auf der Konsole auch ohne die Verwendung eines Editors Dateien (mal eben schnell) erzeugen könnt. Die Inhalte der Dateien sind für diese Aufgabe völlig unerheblich, d.h., dass die Dateien durchaus auch leer sein können.

Benennung

Nennt euer Skript `lister.sh` und achtet dabei unbedingt auf die richtige Benennung (inkl. Groß- und Kleinschreibung), da der Server euer Skript sonst nicht erkennt und dementsprechend auch nicht ausführen kann. Das Skript muss sich spätestens zum Servertestzeitpunkt direkt im Verzeichnis `ueb01` eures SVN-Repositories befinden. Ihr dürft dort für diese Aufgabe gern auch weitere Dateien mit einchecken, da wir diese ignorieren. Beachtet auch den “Hinweis zu SVN und Dateirechten” weiter unten im Text.

Fehlerbehandlung

Eine explizite Fehlerbehandlung ist nicht notwendig. Wir gehen einfach davon aus, dass der Benutzer unser erstes Skript richtig verwendet. Den Fall, dass ihr keine Schreibrechte für die Ausgabedatei oder den Ordner *info* habt, müsst ihr nicht behandeln.

Erlaubte Tools

Verwendet die bereits in der Vorlesung besprochen “Hausmittel” der `shell`. Fortgeschrittene Tools wie etwa `perl`, `sed`, `awk` und andere Skriptsprachen wie z. B. `ruby` sind weder erlaubt noch vonnöten.

Hinweise zur Entwicklung von Skripten

Es ist sinnvoll und arbeitssparend beim Entwickeln eines Skriptes die einzelnen Befehle / Verkettungen von Befehlen zunächst auf der Konsole zu testen und sie erst dann in das Skript einzubinden, wenn die das tun was sie sollen. Gewöhnt euch dies frühzeitig an, da es gerade bei späteren Aufgaben mit ggf. komplexen Filteraufrufen sehr hilfreich ist.

Zudem ist sehr hilfreich in einem Skript stets die Langform der Parameter der einzelnen Befehle zu verwenden, wie z.B. `grep --invert-match` statt `grep -v`. Dies erhöht zum einen die Lesbarkeit deutlich und kann zum anderen auch direkt als Kommentierung gesehen werden.

Dokumentation

Bitte beachtet unsere Dokumentationsrichtlinien. Mangelhafte Dokumentation führt zu Nachbesserungen. Eine sinnvolle Dokumentation besteht aus:

- Modulköpfen, mit einer aussagekräftigen Beschreibung was das entsprechende Skript tut
- Funktionsköpfen
- Inline-Dokumentation (insbesondere Beschreibung der Algorithmen)
- Bei `shell`-Skripten besonders wichtig: Beschreibung aller Parameter (Ein-/Ausgabe von Hauptprogramm & Funktionen)

Testen

Bei dieser Aufgabe und allen folgenden Aufgaben werden wir euer Skript automatisiert mit dem **Testserver** überprüfen. Ihr könnt diesen auch selbst nutzen um sicherzustellen, dass ihr alle Anforderungen richtig gelesen und umgesetzt habt. Beachtet hierzu die **Allgemeinen Hinweise zum Testen**.

Bitte denkt daran euer Programm **rechtzeitig** ins SVN einzuchecken und auf dem Server testen zu lassen. Probleme, die erst beim automatisierten Servertest erkannt werden (falscher Name, falscher Ordner, Skript tut nicht was es soll, etc.), führen im schlimmsten Fall zum Verlust aller Punkte für den Skriptteil diese Aufgabe.

Bedenkt, dass ihr bei allen folgenden Aufgaben immer **mindestens 50% im Servertest** erreichen müsst, damit wir die Lösung akzeptieren. Bei dieser Aufgabe gilt diese 50%-Regel aus Kulanzgründen noch nicht.

Teil 4: Systematisches Testen

Das Schreiben von eigenen automatisierten Tests erwarten wir bei dieser Aufgabe noch nicht. Allerdings sollt ihr euch Gedanken machen, welche unterschiedlichen Fälle bei eurem Programm auftreten können (keine Dateien, eine Textdatei, …) und wie sich euer Programm in diesen Fällen verhalten soll (und das selbstverständlich auf der Konsole auch überprüfen).

Schreibt diese Gedanken in eine extra Text-Datei und fügt sie eurem Repository im Ordner *ueb01* hinzu. Diese Datei ist Teil der Aufgabenstellung und muss zum Zeitpunkt des Servertest im Repository vorhanden sein.

Dieser Aufgabenteil geht mit **20%** in die Bewertung der Aufgabe mit ein.

Gewichtung

Das Ergebnis des automatischen Servertests wird bei dieser Aufgabe mit 50% in die Gesamtwertung eingehen, weitere 30 % der Punkte ergeben sich aus den Screenshots und der Textdatei zur SVN-Benutzung und die restlichen 20% aus der Datei mit den möglichen Testfällen für das Skript.

Denkt daran zusätzlich zu eurem Skript auch alle anderen benötigten Dateien einzuchecken. Alle Dateien müssen zum Abnahmezeitpunkt (Servertest) im Repository vorhanden sein um in die Bewertung mit einzugehen.

Hinweis zu SVN und Dateirechten

Es ist sinnvoll das Skript **vor** dem Hinzufügen (`svn add`) ausführbar zu machen.

Spätere Änderungen an den Dateirechten übernimmt SVN nicht mehr automatisch und das Skript bleibt auf dem Server nicht ausführbar (auch wenn es lokal ausführbar ist). Für den Servertest spielt dies keine Rolle, da wir die Ausführbarkeit auf dem Server für das Skript explizit sicherstellen.

Wenn das Ausführbarkeitsrecht nachträglich setzen wollt, kann dies über über den svn-Befehl **propset** erreicht werden:

```
$ svn propset svn:executable on <dateiname>
```

Hinweis zu Skriptdateien

Solltet ihr eure Skriptdateien nicht unter Linux erstellen, achtet darauf, dass darin nur Unix-Zeilenumbrüche und keine Unicode Byte Order Marks enthalten sein dürfen - dies würde die Ausführbarkeit auf dem Server verhindern.

Nicht-funktionale Mängel

Nachbesserungsfähige nicht-funktionale Mängel:

Nachbesserungsgrund	Abzug Prozentpunkte bei Nichterfüllung
Repository / Ordnerstruktur schlecht gepflegt	5
Interpreterangabe fehlt oder falsch	5
Uneinheitliche Formatierung oder Sprachwahl für Bezeichner oder Kommentierung	5
Unangemessene Bezeichner (z.B. hinsichtlich ihrer Länge oder Aussagekraft)	5
Mangelhafte Formatierung / schlecht lesbarer Code (z.B. Einrückung, Tabs und Leerzeichen gemischt)	5

Nachbesserungsgrund	Abzug Prozentpunkte bei Nichterfüllung
Mangelhafte Dokumentation (z.B. Funktionsköpfe / Inline-Kommentierung)	5
Mangelnde Codehygiene (z.B. toter Code)	5
Unnötige Codeverdopplung	10
Verwendung von unstrukturierter Programmierung (z.B. unangemessen viele returns oder exits)	10
Mangelnde Effizienz / umständliche Lösung (z.B. unnötige temporäre Dateien / Variablen)	10
Nicht aufgabenkonforme Implementierung	20

Editoren

In unseren Rechenzentren sind u.a. die folgenden Kommandozeilen-Editoren installiert:

- **vi** / **vim** ist der Standard-Systemeditor aller Unix-Systeme auf der Konsole. Für Geübte ist dies ein sehr schneller und mächtiger Texteditor. Er hat ein hervorragendes Syntaxhighlighting, kommt ohne X-Server (grafische Oberfläche) aus und ist zudem auf jedem Unix-System vorhanden.
- **nano** ist ein einfach zu bedienender Editor auf der Unix-Konsole (d.h. keine Menüs und keine Maus). Alle möglichen Kommandos sind im Fuß des Fensters aufgelistet. Der Zirkumflex (^) steht dabei immer für die `[Strg]`-Taste
- **emacs** ist ein sehr mächtiger Editor (wie man sagt die “Eier legende Wollmilchsau”). Er kann selbstständig Programmcode formatieren, ist auch geeignet als Mail- oder News-Reader. Die Benutzung ist jedoch etwas gewöhnungsbedürftig. Einarbeitung kann sich aber durchaus lohnen! Man findet zudem sehr viele Konfigurationsdateien im Netz, um z.B. intuitivere Keyboard Shortcuts zu definieren. Zu Beginn ist sicherlich der CUA-Mode sinnvoll, er sorgt dafür, dass Copy und Paste (und weitere) Funktionen auf den “normalen” Tasten liegen. Ihr findet ihn direkt im Options-Menüpunkt.

Zudem gibt es (mindestens)s die folgenden GUI-Editoren:

- **kate** ist ein recht einfach zu bedienender und recht komfortabler KDE Texteditor, mit dem man als Windows-Benutzer sicherlich auf Anhieb klarkommt. Er verfügt über eine Plugin-Schnittstelle, mit der man diverse Plugins, wie z.B. für Syntax-Highlighting, einbinden kann.
- **geany** ist ein mit kate vergleichbarer Texteditor aus der gnome-Welt, der auch gut als kleine Entwicklungsumgebung genutzt werden kann.

Siehe auch: **Standard-Editor konfigurieren**.

Newsgoup

Alle Infos zur Benutzung und Einrichtung der Newsgroup findet ihr in den **RZ-Informationen**. Die zur Übung gehörige Newsgroup heißt `fhw.unix-internet`.

Folien

Die **Folien** zur Aufgabenvorstellung (die ja in diesem Semester leider nicht stattfinden kann), geben einen groben Überblick über die Aufgabenstellung und sollen dem besseren Einfinden in die Aufgabenstellung dienen. Sie sind nicht als weitere Spezifikation gedachte und zu verstehen - im Zweifelsfall gelten immer die Angaben, so wie sie hier in der Aufgabenstellung stehen.

letzte Änderung: 06.05.2020 17:38