Praktikum 6 : Arbeiten mit Shell-Skripts

1 Vorbemerkung

1.1 Allgemeines

Um eine Makroansammlung von Befehlssequenzen durch einen einzigen externen Befehl darstellen zu können, ist es möglich, diese Befehlssequenzen ähnlich Programmanweisungen in einer Textdatei zu hinterlegen und an den Befehlsinterpreter zu binden.
Diese Methodik bezeichnet man als „Shell-Scripting“. Ein solches Shell-Skript ermöglicht den Aufbau komplexer Befehle auf Basis der elementar gegebenen Befehle.

1.2 Erstellen von Shell-Skripts

Für die Erstellung von Shell-Skripts eignet sich praktisch jeder Editor, der den eingegebenen Text als reinen ASCII-Text abspeichert. Unter Linux gibt es eine Reihe von Programmen (vi, nano), die für das Schreiben von Shell-Skripts verwendet werden können.

1.3 Grundsätzliche Funktionen von Shell-Skripts

Zu den wichtigsten Funktionen in Shell-Skripts gehören verschiedene Kontrollstrukturen (if-then-else, case), mit denen die Ausführung von Kommandos in Abhängigkeit von definierbaren Bedingungen steuerbar ist
Für die Überprüfung der Bedingungen steht mit dem Kommando „test“ eine Vielzahl von leistungsfähigen Vergleichsoperationen zur Verfügung. Dazu gehören numerische Vergleiche, Vergleiche von Strings und logischen Kombinationen dieser Bedingungen.

Zur wiederholten Ausführung von Befehlssequenzen gibt es verschiedene Formen von Schleifen (for, while, until), die mehrfach mit gleichen oder veränderten Parametern durchlaufen werden können.
Die interaktive Steuerung von Shell-Skripts stehen Funktionen zur Eingabe von Daten (read) und zur Ausgabe von Daten, Eingabeaufforderungen und Statusmeldungen (echo) zur Verfügung.

Anweisungen, die innerhalb einer Zeile hinter dem Kommentar-Zeichen (#) stehen, werden nicht ausgeführt. Auf diese Weise können Skripts dokumentiert oder einzelne Zeilen zu Testzwecken deaktiviert werden. Eine besondere Funktion erfüllt dabei jedoch ein Pseudo-Kommentar (#!) am Anfang eines Shell-Skripts.
Näheres dazu im Abschnitt zur Ausführung von Shell-Skripts.

1.4 Ausführen von Shell-Skripts

Bei der Ausführung eines Shell-Skripts wird das Skript durch den Befehlsinterpreter (Shell) zeilenweise eingelesen, die jeweiligen Kommandos interpretiert und anschließend ausgeführt. Als Befehlsinterpreter dient dabei die aktuell verwendete Shell oder die beim Aufruf explizit angegebene Shell.

Skripte können folgendermaßen von der Kommandozeile aus aufgerufen werden:

<scriptname> (z.B. `script01`) bzw. **<shell> <scriptname>** (z.B. `bash script01`)

Zusätzlich können beim Aufruf eines Shell-Skripts Parameter übergeben werden, die das Skript intern als Text oder als numerische Werte beliebig weiter verarbeiten kann.

Aufruf: **<scriptname> <par1> <par2>** … **<par[n]>** (z.B. `script01 10 mytext /home`)

2 Übungsaufgaben

Info: Die Ausgabe relevanter Daten erfolgt nur in Verbindung mit den angegebenen Skripts. Nutzen Sie daher für alle Aufgaben die im Tabellenkopf angegebenen Skripte.

Kommando	Aufgabe des Kommandos
`pr06-watch.sh`	Ausgabe zusätzlicher Informationen

2.1 Varianten und Voraussetzungen zum Ausführen von Shell-Skripts

Erklären Sie die jeweiligen Aufgaben und die Ergebnisse der durchgeführten Kommandos. Achten Sie dabei besonders auf das Ergebnis der grau hinterlegten Shell-Skript-Aufrufe.

`pr06-2.1.sh`	Aufgabe/Ergebnis des Skripts bzw. Kommandos
`echo "echo OK" > s01.sh`	
`cat s01.sh`	
`chmod 0 s01.sh`	
`bash s01.sh`	
`./s01.sh`	
`s01.sh`	
`chmod +r s01.sh`	
`bash s01.sh`	
`./s01.sh`	
`s01.sh`	
`chmod +x s01.sh`	
`bash s01.sh`	
`./s01.sh`	
`s01.sh`	
`V1=$(pwd)`	
`PATH=$PATH:$V1`	
`bash s01.sh`	
`./s01.sh`	
`s01.sh`	

2.2 Besondere Variablen in Shell-Skripts (Interne Shell-Variablen)

Analysieren Sie den Inhalt der Shell-Skripts, führen Sie das Shell-Skript in den vorgegebenen Varianten au und ermitteln Sie die Werte der angegebenen Shell-Variablen für jeden Aufruf.

Erklären Sie anschließend die Werte dieser Variablen innerhalb von ausgeführten Shell-Skripts.

`pr06-2.2.sh`	`$0`	`$1`	`$2`	`$3`	`$*`	`$#`
`s02.sh`						
`s02.sh Hello World !`						
`s02.sh "Hello World" !`						

2.3 Kontrolle von erfüllten bzw. nicht erfüllten Bedingungen in Shell-Skripts

Analysieren Sie die Funktion vom Befehl ***"test"*** und die Ergebnisse der ausgeführten Kommandos.

pr06-2.3.sh	**Aufgabe des Kommandos**
`read X`	Interaktives Einlesen von Variablen
`if [ "$X" = "10" ]` `then` `  echo "Bedingung wahr"` `else` `  echo "Bedingung falsch"` `fi`	Testen auf erfüllte bzw. nicht erfüllte Bedingungen **Hinweis:** Eckige Klammern sind die Kurzform vom Befehl "***test***"
`[ $X = "10" ] && Y=1 || Y=2`	Kurzform vom obigen Befehl

2.4 Kontrollstrukturen und Ablaufsteuerungen in Shell-Skripts

Analysieren Sie die den Code der Shell-Skripts in Verbindung mit den jeweiligen Ergebnissen unter Berücksichtigung der jeweiligen Syntax in Verbindung mit den verwendeten Operatoren.

Ermitteln Sie Eingaben, damit die Bedingungen in den Skripts mindestens einmal erfüllt wurden. Informationen zu den verwendeten Operatoren sind in der man-Page zum Befehl ***"test"*** verfügbar.

Skript-Aufruf	**Bedingungen mit**	**Syntax**	**Eingaben**
`s03.sh 1`	Numerische Variablen	`[ $N1 <op> $N2 ]`	
`s03.sh 2`	Zeichenketten	`[ $S1 <op> $S2 ]`	
`s03.sh 3`	Dateinamen	`[ <op> $F1 ]`	
`s03.sh 4`	Logischer Verknüpfung	`[ <B1> <op> <B2> ]`	

Ermitteln Sie die Eingaben, damit jede der folgenden Bedingung mindestens einmal erfüllt wurde. Informationen zu den verwendeten Schleifen sind in der man-Page zum Befehl ***"bash"*** verfügbar.

Skript-Aufruf	**Steuerung durch**	**Syntax**	**Eingaben**
`s03.sh 5`	Mehrfachauswahl	`case $X in` `1) echo "$X = 1"` `  ;;` `A|b) echo "$X = A\|b"` `  ;;` `*) echo "$X != 1|A|b"` `  ;;` `esac`	
`s03.sh 6`	Bedingte Schleife	`while [ "$X" != "J" ]` `do` `  echo -n "Ende? "` `  read X` `done`	

2.5 Weitere Schleifen-Varianten in der Shell (<u>für MIMEB optional</u>)

Informationen zu den verwendeten Schleifen sind in der man-Page zum Befehl ***"bash"*** verfügbar.

Skript-Aufruf	**Steuerung durch**	**Syntax**	**Eingaben**
`s03.sh 7`	Numerische Schleife	`for((i=1; i<$X; i++))` `do` `  echo "$i von $X"` `done`	
`s03.sh 8`	Listengesteuerte Schleife	`for i in $X` `do` `  echo "$i aus $X"` `done`	

2.6 Besonderheiten bei der Ausführung von Shell-Skripts (<u>für MIMEB optional)</u>

Zur Demonstration einiger Besonderheiten bei Shell-Skripts wird folgendes Skript verwendet:

Kommando	Relevanter Inhalt des Skripts
cat $(which s04.sh)	`echo "Start: V1 = $V1"` `echo -n "Schleife"` `for((V1;$V1<=5;V1++))` `do` `  echo "V1 = $V"` `done` `echo "Ende: V1 = $V1"`

2.6.1 Gültigkeit von Shell-Variablen

Erklären Sie die aktuellen Werte der Variable **„V1“** im bzw. außerhalb vom verwendeten Skript.

pr06-2.6.1.sh	**Ergebnis des Skripts**
export V1=2	
echo "Check01: $V1"	
s04.sh	
echo "Check02: $V1"	
. s04.sh	
echo "Check03: $V1"	

2.6.2 Festlegung vom aktuellen Shell-Interpreter

Je nach verwendeter Shell kann es beim selben Shell-Skript zu unterschiedlichen Ergebnissen kommen, da sich die Syntax der Funktionen und der Funktionsumfang der einzelnen Shells unterscheiden können. Daher wird oft in der ersten Zeile ein Pseudo-Kommentar mit der zu verwendenden Shell angegeben.

<u>Syntax:</u> **`#!<pfad_zum_interpreter>`** (z.B. `#!/bin/bash`)

Erklären Sie die Auswirkung vom Pseudo-Kommentar **„#!/bin/bash“** im betroffenen Skript.

pr06-2.6.2.sh	**Ergebnis des Skripts**
cat $(which s04.sh) > s05.sh	
echo '#!/bin/bash' > s06.sh	
cat s05.sh >> s06.sh	
chmod +rx s0[56].*	
export V1=2	
echo $V1	
./s05.sh	
./s06.sh	
sh	In die "sh"-Shell wechseln
echo $V1	
./s05.sh	
./s06.sh	

Name: Marschall, Kubernath **Studiengang:** SMI, SMS **Gruppe:** 25

3 Praxisaufgabe

Geben Sie diesen Aufgabenzettel und das Skript **elektronisch (!) ausgefüllt/erstellt** auf Ilias ab.

3.1 Analyse von Shell-Skripts

Ermitteln Sie den Wert der Variable "R" **am Ende der einzelnen Code-Segmente (!)**, wenn ein Skript laut Tabellenkopf aufgerufen wird und den betreffenden Code ausführt hat.

s07.sh 1 2 10 A B C	Wert von "R"	inkl. kurzer Begründung
`R=$*`	1 2 10 A B C	alle Argumente durch Wildcard
`if [ "$#" -gt "$1" ]` `then` `  R=wahr` `else` `  R=falsch` `fi`	wahr	Falls Anzahl der Argumente größer als Argument 1
`case $4 in` `  a) R=$2 ;;` `  b) R=$3 ;;` `  *) R="$2+$3" ;;` `esac`	2+10	$4 ist nicht a und nicht b, also tritt * ein
`while [ "$R" != "$6" ]` `do` `  read R` `done`		fragt nach Wert für R bis dieser als C eingegeben wird
`if [ ! -e "$0" ]` `then` `  R=1` `else` `  R=2` `fi`	2	Wenn das File mit Namen des ausgeführten Skriptes nicht existitert 1, sonst 2
Erweiterte Aufgabenstellung		
`for i in $*` `do` `  R="$i+$R"` `done`	C+B+A+10+2+1+	R ="" zu Beginn und jede Iteration schiebt das fortlaufende Argument von links an
`for((i=$1;i<=$3;i++))` `do` `  R=$i` `done`	10	For Schleife läuft bis 10

3.2 Erstellen von komplexen Shell-Skripts (<u>für MIMEB optional</u>)

Erstellen und **testen (!!!)** Sie ein Shell-Skript, das durch den Aufruf **"s08.sh <name> <count>"** gestartet wird, dabei einen Verzeichnisbaum erstellt und dazu folgende Aufgaben übernimmt:

- Verzeichnisname und Anzahl der Unterverzeichnisse aus Kommandozeile übernehmen
- Überprüfen, ob Zielverzeichnis bereits existiert und ggf. neuen Verzeichnisnamen abfragen
- Interaktives Einlesen der Namen für die einzelnen Unterverzeichnisse
- Verzeichnisstruktur anlegen, wobei das Hauptverzeichnis für alle Benutzer beschreibbar ist, die Unterverzeichnisse für sie jedoch nur lesbar sein sollen
- Für alle Schritte aussagekräftige Status-Meldungen ausgeben