Aufgabe 1

(a) Erzeugen Sie mit einem Editor eine Datei `hallo` mit folgendem Inhalt:

```
#!/bin/bash
echo Guten Morgen!
```

Hierzu kann ein beliebiger Texteditor verwendet werden. Zum Beispiel `vim` oder `nano` in der Konsole oder `kedit` bzw. `gedit` auf einer grafischen Oberfläche.

(b) `bash hallo`
`chmod +x hallo`

Das Setzen der Ausführungsrechte mit „chmod“ ermöglicht das direkte Ausführen der Datei (ohne vorangestelltes „bash“.)
Falls das Skript in einem Verzeichnis im Suchpfad liegt: `hallo`
Falls das Skript im aktuellen Verzeichnis liegt, das aktuelle Verzeichnis aber nicht im Suchpfad steht, muss `./` vorangestellt werden: `./hallo`

(c) Erweitern Sie die Datei hallo, **so dass**

- **alle übergebenen Argumente,**
  `echo $@`
  „$@“ ist eine Liste aller übergebenen Argumente

  Alternativ:

```
#!/bin/bash
for arg in $@
do
     echo $arg
done
```

  Beispiel: `./skript argument1 argument2 argument3`

  „arg“ ist ein frei wählbarer Variablenname, der im i-ten Schleifendurchlauf das i-te Kommandozeilenargument beinhaltet.

- **die Anzahl der Argumente,**
  `echo $#`
  „$#“ ist einer der vielen speziellen/reservierten Variablennamen der Bash. Er steht für die Anzahl der übergebenen Argumente.

- **der Shellskript-Inhalt**
  `cat $0`
  Wird diese Zeile dem Skript hinzugefügt, dann gibt es sich selber aus. In „$0“ steht immer der Aufruf des Skriptes. (Beispielsweise „./skript“ wenn es auf der Kommandozeile über „./skipt“ aufgerufen wurde.) Da dies dem Pfad zum Skript entspricht kann man das Skript einfach mit cat ausgeben.

  auf dem Bildschirm angezeigt werden.

(d) Ändern Sie das Skript `laenge1.sh` aus dem Vortrag ab, so dass `myfile` als Parameter mit übergeben werden kann.

```
#!/bin/bash
echo Die Anzahl Woerter in $1 ist
cat $1 | wc -w
```

(e) Was passiert bei folgenden Befehlen:

```
set $(seq 1 15)
echo $@
echo $1 $9
shift 3
echo $1 $9
```

Zeile 1: `set` setzt der Reihe nach die Paramter $1 bis $n mit den übergebenen Werten. (Hier die Zahlen 1 bis 15.) Eventuell übergebene Kommandozeilenparamter werden dabei überschrieben.
Zeile 2: Die Paramter $1 bis $n (alle) werden ausgegeben.
Zeile 3: Parameter an Position 1 und 9 werden ausgegeben. (Werte: $1=1, $9=9)
Zeile 4: positionale Parameter werden um 3 nach links geschoben.
$4 ist jetzt $1, $5 ist jetzt $2, ...
Zeile 5: Parameter an der Position 1 und 9 werden ausgegeben.
(Werte: $1=4, $9=12)

(f) Erstellen Sie ein Skript, welches die Anzahl Dateien und Ordner im übergebenen Verzeichnis zählt. Falls kein Verzeichnis übergeben wird, soll das Home-Verzeichnis des Benutzers genutzt werden.

Variablensubstitution durch `${Variable:-Wert}`:

```
#!/bin/bash
ls -1 ${1:-~} | wc -l
```

Alternative Lösung, nachdem wir if und [ ] geübt haben:

```
#!/bin/bash
if [ ! -z $1 ] && [ -d $1 ]
then
     dir=$1
else
     dir=~
fi
ls -1 $dir|wc l
```

(g) Erstellen Sie ein Skript, das die übergebenen Dateien als .tar packt.
`./pack-skript hallo.txt welt.txt 123.pdf ...`
→ Ergebnis: hallo.txt.tar, welt.txt.tar, 123.pdf.tar, …

```
tar -cfv [ARCHIVNAME].tar [Datei]
```

„`tar`“ ist zum packen und entpacken von .tar Dateien.
„c“ steht für create. Es wird ein neues Archiv angelegt
„f“ steht für file. Der Archivname wird angegeben.
„v“ steht für verbose. Dieser Parameter ist optional. Er sorgt dafür das „tar“ ausführliche Informationen über das Packen ausgibt.

```
#!/bin/bash
for file in $@
do
     tar -cfv "$file.tar" $file
done
```

Auf „in $@“ kann auch verzichtet werden, da (wenn nichts angegeben wird) automatisch die Liste der positionalen Parameter genommen wird.

(h) Schreibe ein interaktives Skript satz.sh, das den Benutzer hintereinander nach drei Wörtern fragt, und diese hinterher in einem Text ausgibt. Automatisiere die Eingabe für dieses Skript in einem weiteren Skript mit „Here Document“.

```
#!/bin/bash
echo "Hallo, ich bin ein interaktives Skript."
echo "Gib eine Farbe ein:"
read color
echo "Gib einen Tiernamen ein:"
read animal
echo "Gib etwas Essbares ein:"
read food

echo "Heute in der Bild:"
echo "$animal hatte ${color}e Mütze auf!"
echo "Zur Strafe zu $food verarbeitet."
```

Über das „Here Document“ bekommt das Skript seine mehrzeilige Eingabe.

```
#!/bin/bash
sh satz.sh << TRENNER
grün
Schnecke
Spaghetti mit Tomatensoße
TRENNER
```

oder noch besser in einer Datei:

```
#!/bin/bash
{
echo "Hallo, ich bin ein interaktives Skript."
echo "Gib eine Farbe ein:"
read color
echo "Gib einen Tiernamen ein:"
read animal
echo "Gib etwas Essbares ein:"
read food

echo "Heute in der Bild:"
echo "$animal hatte ${color}e Mütze auf!"
echo "Zur Strafe zu $food verarbeitet."
} << TRENNER
grün
Schnecke
Spaghetti mit Tomatensoße
TRENNER
```