1. Erste Erfahrungen mit der bash – Shell

a) Machen Sie sich mit der bash Shell vertraut, indem Sie die fett gedruckten Befehle ausprobieren. Protokollieren Sie alle Ihre Tests und halten Sie die Protokolldatei zur Abgabe bereit.

strg-C	Laufendes Programm abbrechen
alias *name=value*	Zeichenersetzung („Alias“) definieren (ohne Leerzeichen!)
bg	Programm im Hintergrund laufen lassen (ohne Benutzereingaben)
cat *file*	**Textdatei file auf Standardausgabe (stdout) ausgeben**
cd *dir*	**Verzeichnis wechseln**
chmod [ugoa][+/-][rwx] *file*	Zugriffsrechte bzgl. Datei file ändern (x: ausführbar)
cp [–i] *file1 file2*	**Kopiere Datei file1 nach file2**
date	**Datum und Zeit anzeigen**
df	Information über Dateisysteme anzeigen
diff *file1 file2*	Unterschiede zwischen Datei file1 und Datei file2 anzeigen
echo *string*	**Zeichkette string auf Standardausgabe (stdout) ausgeben**
emacs *file* | vi *file*	Editiere Datei file
env	**Umgebungsvariablen anzeigen**
exit	**Shell oder Skript beenden**
export *var*	**Shell-Variable var an alle Kindprozesse vererben**
fg	Programm in den Vordergrund holen
find *dir* -name *file* -print	**Finde eine Datei namens file beginnend im Verzeichnis dir**
grep [-r] *string file*	**Suche in der Datei file nach der Zeichnkette string**
jobs	Information über Hintergrund-Programme der aktuellen Shell
kill [–9] *pid*	Prozess (laufendes Programm) mit der Prozessnr. pid beenden
In [-s] *file1 file2*	[symbolischen] Link file1 –> file2 erzeugen
lpq	Drucke Warteschlangen-Status
lpr [–P *queue*] *file*	Drucke file auf Drucker-Queue queue
ls [-l] [*file*]	**Aktuellen Verzeichnis-Inhalt als Liste von Dateinamen ausgeben**
man *prog*	**Beschreibung des Programms prog**
mkdir *dir*	**Verzeichnis erzeugen**
more *file*	**Textdatei file seitenweise anzeigen**
mv *Quelle Ziel*	**Datei Quelle umbenennen bzw. verschieben**
passwd	Ändert das Passwort des aktuellen Benutzers
prog	Ausführbares Programm *prog* starten (wird in den in $PATH angegebenen Verzeichnissen gesucht)
prog&	Programm prog direkt als Hintergrundprozess starten (ohne Benutzereingaben)
ps	**Prozess-Informationen anzeigen**
pstree	Prozess-Informationen als Baumstruktur anzeigen (Eltern/Kinder)
pwd	**Name des aktuellen Verzeichnisses ausgeben**
rm [-i] *file*	Datei file löschen
rmdir *dir*	Verzeichnis dir löschen
sleep *sec*	Hält die aktuelle Shell-Ausführung um sec Sekunden an
time *prog*	Programm prog starten und verbrauchte CPU-Zeit ausgeben
var=value	**Shell-Variable var den Wert value zuweisen**
who	Aktuelle Benutzer dieses Systems anzeigen
> file	**Standardausgabe (stdout) auf file umlenken, file ggf. neu erzeugen oder überschreiben**

>> *file*	Standardausgabe (stdout) auf file umlenken, file ggf. neu erzeugen oder Ausgaben an file anhängen
$var	**Die Zeichenkette $var durch den aktuellen Wert der Variablen var ersetzen**
$1 $2 $3 …	Zeichenkette $1, $2, $3, .. durch jeweils 1., 2., 3. .. Parameter der Befehlszeile ersetzen ($0: Programmname)
$#	Zeichenkette $# durch Anzahl der Parameter der Befehlszeile ersetzen (Dezimalzahl)
$?	Zeichenkette $? durch return value des zuletzt aufgerufenen Programms (Vordergrundprozesses) ersetzen
$(*prog*)	Das Programm prog starten und die Zeichenkette $(prog) durch die Ausgaben des Programms ersetzen
.	Zeiger auf das aktuelle Verzeichnis
..	Zeiger auf das direkt übergeordnete Verzeichnis
*	Metazeichen: Platzhalter für beliebig viele Zeichen

Tipp: Informationen zur bash und Unix-Befehlen finden Sie auf einem Linux-Rechner (Befehl „man“), in Büchern (Bibliothek) und im Internet. Unten finden Sie außerdem Tipps, wie Sie auf ein Linux-System außerhalb des Labors zugreifen können.

b) Beantworten Sie die folgenden Fragen:

- Was enthalten die folgenden Umgebungsvariablen (Environment Variables)?
  $HOME, $PATH, $UID, $USER
- Was bewirkt der Befehl "cd $HOME" ? Gibt es eine einfachere Alternative?
- Was für eine Funktion haben die folgenden Eingaben?
  [↑] [↓] [strg-d] (in leerer Zeile)
- Was ist die Funktion der .bashrc Datei im Verzeichnis $HOME?

2. Shell-Skripte

Vorab ein kleines bash-Skript `example.sh` als Beispiel (liegt auch im Pub-Verzeichnis):

```
#!/bin/bash
# <Description of this shell script>
# <Your name> # <Date>

# ------------------------------------------------------------
# This function asks the user for his name
ask_for_name() {
    echo "Please enter your name: "
    read user_name
}
```

```
# This function shows the help text for this bash script
usage() {
      echo "
      $0 [OPTIONS] [<user name>]
      Ask the user for her or his name and display a greeting
      OPTIONS:
            -h  : Display this help text"
}


# ---------------------- main --------------------------------
# check parameters
if [ $# -gt 1 ]; then
    usage
    exit 1
fi

case $1 in
    "-h")
        usage
        exit 0
        ;;
    "")
        ask_for_name
        ;;
    *)
        user_name=$1
esac

# print greetings
echo "
####################
 Hello $user_name,
 nice to meet you!
####################
"
exit 0
# ---------------------- end ---------------------------------
```

2. Shell-Skripte

Erstellen Sie folgende Shell-Skripte (bash) und testen Sie diese. Für das Protokoll reicht der kommentierte Source Code des Skripts. Während der Abnahme sollten Sie den Code erklären können.

a) `frename.sh <string>`
Hängt für alle Dateien im aktuellen Verzeichnis die Zeichenkette *string* an den aktuellen Dateinamen an (Umbenennung).

b) `try_host.sh [-h|-s <sec>] <hostname>|<IP-Address>`
Der in der Befehlszeile angegebene Rechner (Hostname oder IP-Adresse) soll auf Erreichbarkeit hin überwacht werden. Dazu sendet das Skript in regelmäßigen Zeitabständen ein "ping" an den angegebenen Rechner (nur ein ping Paket) und wertet den return value aus (siehe `man ping`). War der ping Befehl erfolgreich, wird der Rechnername mit einem OK- Vermerk ausgegeben, andernfalls wird er mit einem FAILED-Vermerk ausgegeben.

Das Skript unterstützt folgende Optionen (es darf aber nur eine Option gleichzeitig angegeben werden):
`-h` : Nur Ausgabe der „Usage Message“
`-s <sec>` : Der ping wird zyklisch alle <sec> Sekunden ausgeführt.
Fehlt die –s Option, wird der ping alle 10 Sekunden ausgeführt.

Beispiel: Der Aufruf
```
bash try_host.sh -s 5 google.de
```
erzeugt alle 5 Sekunden eine Ausgabe der Art:
```
google.de OK
```
falls der Host google.de erreichbar ist, anderenfalls
```
google.de FAILED
```

c) Ändern Sie den Status jedes Skripts auf „ausführbar“ und starten sie beide Skripte jeweils als Programm (ohne `bash`-Aufruf, aber mit Angabe des aktuellen Verzeichnisses, z.B. durch Voranstellen von `./`)

d) Erweitern Sie den Inhalt der Umgebungsvariablen PATH so, dass immer das momentan aktuelle Verzeichnis enthalten ist.

3. C-Programm mit Systemaufrufen

Als Startpunkt vorab ein kleines C Programm:
```
/*
* <Description of this C program>
*
* <Your name> # <Date>
*
*/

#include <stdio.h>

int main (void) {
      printf("Willkommen im BSP!\n");
      return 0;
}
```

Laden Sie das Programm `hello.c` aus dem Pub-Verzeichnis herunter. Es wird mit dem Befehl `gcc –o hello hello.c` übersetzt, so dass anschließend das ausführbare Programm `hello` vorhanden ist.

Das zu erstellende C-Programm `mkfile` soll eine Eingabeaufforderung anzeigen, danach maximal 30 Zeichen von der Tastatur lesen (einen Dateinamen), daraufhin eine leere Datei mit diesem Namen und den Zugriffsrechten 0700 (Zugriff nur für Besitzer) erzeugen sowie eine entsprechende Meldung auf dem Bildschirm ausgeben. Falls ein Fehler aufgetreten ist, soll eine allgemeine Fehlermeldung ausgegeben werden.
Bei der Abgabe sollte das Programm getestet und sinnvoll kommentiert sein.

Beispiel (Ausgaben sind *kursiv* dargestellt):

```
$ mkfile
Name der neuen Datei: bsp1
Die Datei bsp1 wurde erfolgreich angelegt!
```

Benutzen Sie zur Realisierung der Systemaufrufe folgende C-Bibliotheksfunktionen (Dokumentation auch über „man"-Befehl erhältlich):

- `fgets` : Liest eine Zeichenkette von stdin ein (Achtung: inkl. Newline!)
- `creat` : Legt eine Datei an und öffnet sie
- `close` : Schließt eine Datei
- `printf` : Erzeugt eine Ausgabe

Hinweise:

- Ggf. muss die C-Bibliotheksfunktion explizit in Ihren Programmcode eingebunden werden (#`include`)
- Eine char-Feld-Variable (z.B. `char name[20]`), auch „String" genannt, hat in C als Wert die Adresse des ersten Elements, ist also ein Zeiger auf einen Speicherplatz vom Typ `char`.
- Die Funktion `int strlen(char *string)` liefert als Rückgabewert die Länge eines Strings.
- Ein String wird in C durch den ASCII-Wert 0 (Konstante: `0x0`) nach dem letzten Zeichen abgeschlossen (unabhängig vom Speicherplatz, der durch das char-Array belegt wird). Daher wird z.B. durch eine Zuweisung `name[5] = 0x0` der String name auf die ersten fünf Zeichen verkürzt (0 – 4).
- Beschreibung von `creat`:
  Prototyp: `int creat(char *pathname, int mode);`
  Effekt: Erzeugt eine neue Datei. Falls die Datei bereits existiert, wird ihr Inhalt gelöscht.
  Parameter:
  `char *pathname` Name oder Pfad der neuen Datei.
  `int mode` Bitmuster, das die Zugriffsrechte für die neue Datei festlegt. Die Positionen und Bedeutungen der Bits sind dieselben wie in der Ausgabe des Kommandos `ls -l` (Rechte für Besitzer, Gruppe und andere Benutzer).
  Rückgabe: Dateideskriptor (`int`) für folgende Dateizugriffe oder -1 bei Fehler.

Tipp: Für den kostenlosen Zugriff auf ein Linux-System gibt es u.a. folgende Möglichkeiten:

- PC-Pool der Informatik nutzen (Linux booten)
- Knoppix (http://www.knopper.net/knoppix) oder ubuntu (www.ubuntu.com) downloaden und auf CD brennen oder bootfähigen USB-Stick kopieren
  Anschließend von CD/DVD oder USB-Stick Linux booten (ohne Installation auf Platte)
- Linux downloaden und auf Partition des eigenen Rechners installieren
  (z.B. http://de.opensuse.org)
- Virtuelle-Maschine-Monitor (Typ2-Hypervisor) unter Windows installieren und Linux in der virtuellen Maschine installieren/starten
  - http://www.vmware.com/de/products/player/ (VMWare kostenlos – empfohlen) oder:
  - http://www.virtualbox.org/ (SUN / Oracle – OpenSource)
- Zugriff auf Informatik-Server unter Windows mittels putty (ssh/telnet-Client)
  - Putty.exe downloaden http://www.putty.org und starten
  - SSH-Verbindung aufbauen mit ssh.informatik.haw-hamburg.de unter Port 22 (Benutzername und Passwort: HAW-Account)