UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 3
Musterlösung

Vorlesung Betriebssysteme und nebenläufige Programmierung
Sommersemester 2025

Aufgabe 1: Anmeldung zur Studienleistung

Aufgabe 2: Shellskript, Verwendung der Kommandos: date, sleep

```
#!/bin/bash
# The output of date is controlled by a format string:
#  %d–day of month (01..31), %m– month (01..12),
#  %y–last two digits of year (00..99), %n –a newline,
#  %H – hour (00..23), %M – minute (00..59)
#
m1=$((`date +%M` -1))   # Previous minute
while true
do
 mi=`date +%M`   # Minute
 if [ ! $mi -eq $m1 ]   # when the minute changed (NOT current minute == previous minute)
 then
   clear   # clear the screen
   # display the time (new minute)
   date '+Heute ist der %d.%m.%y%nAktuelle Zeit ist: %H:%M'
   m1=$mi  # Previous minute
 fi
 sleep 1
done
```

Erweiterung:

```
#!/bin/bash
m1=$((`date +%M` -1))   # Previous minute
while true
do
  mi=`date +%M`   # Minute
  if [ ! $mi -eq $m1 ]   # when the minute changed (NOT current minute == previous minute)
  then
    # display the time (new minute)
    date '+Heute ist der %d.%m.%y%nAktuelle Zeit ist: %H:%M'
    m1=$mi  # Previous minute
    echo -e "\033[3F" # move cursor 3 lines up
  fi
  sleep 1
done
```

Aufgabe 3: System- und Benutzermodus (Pflichtaufgabe für die Studienleistung! Abgabe bis So., 04.05., 23:59 über moodle)

Aufgabe 4: Bibliotheksfunktionen und Systemaufrufe

In Bezug auf die Programmlogik spielt es keine Rolle, ob ein Aufruf einer Bibliotheksfunktion zu einem Systemaufruf führt. Aber wenn man die Leistung betrachtet, läuft ein Programm schneller, wenn eine Aufgabe ohne Systemaufruf ausgeführt werden kann. Jeder Systemaufruf bedeutet einen Overhead beim Wechsel vom Benutzerkontext zum Betriebssystemkontext. Darüber hinaus kann das Betriebssystem auf einem Mehrbenutzersystem einen anderen Prozess bzw. Thread einplanen, der ausgeführt wird, wenn der Systemaufruf abgeschlossen ist, wodurch der Fortschritt des aufrufenden Prozesses / Threads in Echtzeit weiter verlangsamt wird.

Aufgabe 5: Logischer Adressraum

Dadurch kann ein ausführbares Programm bei jeder Ausfuhrung in einen unterschiedlichen Teil des Hauptspeichers geladen werden. Außerdem wird es dadurch ermöglicht, daß der Adressraum eines Prozesses die Größe des Hauptspeichers überschreiten kann. Schließlich kann auch diese Weise auch der Speicherschutz einfach sichergestellt werden.

Aufgabe 6: Systemaufrufe zum Prozessmanagement

a) Beschreibung der Routinen:

- **`fork`** erzeugt einen neuen Prozess (Adressraum + Thread), der mit dem erzeugenden Prozess bis auf die Prozess-ID (*pid*) und Parent Prozess-ID (*ppid*) identisch ist.
- **`execve`** lädt ein Programm zur Ausführung in den aktuellen Prozess. D.h., ersetzt den Adressraum (Text-, Daten-, und Stacksegment) des laufenden Prozesses durch den des neu zu startenden Programms.
- **`execl`**, **`execlp`**, **`execle`**, **`execv`**, **`execvp`** sind verschiedene, teilweise einfachere oder komfortablere Schnittstellen zu **`execve`**.
- **`wait`**, **`waitpid`** wartet auf die Beendigung eines anderen Prozesses.
- **`system`** führt ein Shell-Kommando aus und kehrt ins aufrufende Programm zurück.

b) Bibliotheksfunktionen laufen im Adressraum eines Anwendungsprogramms ab. Die Zugehörigkeit zu Betriebssystemkern oder Bibliothek lässt sich auch an der Kapitel-Nummer auf der jeweiligen Manual-Seite erkennen. **Kapitel 2** listet die Kern-Funktionen und **Kapitel 3** die Bibliotheksfunktionen.

Beide Sorten von Routinen führen ihre essentiellen Anteile im priviligierten Modus im Kern aus. Die Bibliotheksfunktionen bieten jedoch meist ein komfortableres Interface zur Kern-Funktionalität. Ein deutliches Beispiel sind die **`exec`** Funktionen: Der Kern stellt nur die essentielle Variante **`execve`** zur Verfügung, während die Bibliotheksvarianten verschiedene Parameter-Kombinationen und die Suche des ausführbaren Programms über den Systempfad ermöglichen.

Systemaufrufe	Bibliotheksfunktionen
`execve`	`execl`, `execlp`, `execle`, `execv`, `execvp`
`fork`	`system`
`wait`	
`waitpid`	

Siehe auch https://man7.org/linux/man-pages/dir_section_2.html