THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Prof. Dr.-Ing. Guido Bartsch
Professur für Informatik

Übungsblatt

Betriebssysteme

Wintersemester 2024/2025

Woche 4

Aufgaben

1. In der Vorlesung wurden wichtige Systemdateien und -Befehle sowie die Nutzung von Pipes ausführlich besprochen. Diese Aufgabe beziehen sich hierauf.

3 P

(a) (3 P) Beschreiben Sie, ggf. unter Zuhilfenahme der *Manualpage* zu dem Befehl `cut`, welche Bildschirmausgabe durch

```
cat /etc/passwd | cut -d: -f6
```

inhaltlich zu erwarten ist und welche Verarbeitungsschritte zu dieser Ausgabe führen.

"cat /etc/passwd" schreibt den inhalt der /etc/passwd Datei in die Standard Ausgabe.
Über die Pipe wird aber der Inhalt der Standard Ausgabe an die Standardeingabe des Befehls "cut -d: -f6" weitergeleitet. "cut" schreibt nur die gewählten Teile der Standardeingabe oder einer angegebenen Datei in die Standard Ausgabe.
Die Option "-d:" bewirkt, dass das Trennzeichen zwischen Feldern (Zeichenketten) ein Doppelpunkt ist. Die Option "-f6" wählt das 6. Feld für den Befehl "cut", so dass nur dieses erhalten bleibt. Insgesamt wird das 6. Feld (home directory) jeder Zeile von der passwd Datei untereinander in der Ausgabe angezeigt.

2. Auch wurden in der Vorlesung Ein-/Ausgabeumlenkungen ausführlich besprochen. Folgende Fragen beziehen sich hierauf.

6 P

Ausgangspunkt dieser Aufgabe sei, dass Sie in Ihrem Home-Verzeichnis stehen, in dem Sie die das `read`-, `write`- und `execute`-Recht besitzen. Gehen Sie davon aus, dass ggf. angegebenen Kommandos von Ihnen ausgeführt werden können.

Folgende Zeile, die in einer Bourne-Shell eingegeben wird, soll betrachtet werden:

```
a 2> b | c 2> d | e > f 2> g
```

(a) (2 P) Geben Sie an, welche Dateien (Dateiname angeben) entstehen.

b, d, f, g

(b) (2 P) Beschreiben Sie, welches Programm respektive welcher Prozess x in welche der von Ihnen genannten Datei y unmittelbar Inhalte erzeugt. Nutzen Sie hierfür folgendes Format: `<x>` → `<y>`

<a> -> <b>
<c> -> <d>
<e> -> <f>
<e> -> <g>

(c) (2 P) Geben Sie an, welchen Inhalte i in jeder der von Ihnen genannten Datei d jeweils abgelegt ist. Nutzen Sie hierfür folgendes Format: `<d>: <i>`

<b>: <zsh: command not found: a>
<d>: <zsh: command not found: c>
<f>: <>
<g>: <zsh: command not found: e>

3. Auf einem fiktiven Multi-Tasking-System (Single-Core, kein Multithreading) gebe es nur die Prozess-Zustände „Ready“ und „Running“. Vereinfachend soll angenommen werden:

9 P

- Ein Prozesswechsel benötige keine Zeit.
- Prozessen, denen die CPU zugewiesen wurde, wird diese nach zwei Zeiteinheiten (2 ZE) wieder entzogen.
- Die Dauer einer Zeiteinheit sei um mehrere Größenordnungen kleiner als alle erdenklichen Programmlaufzeiten.
- Die CPU werde allen Prozessen auf dem System abwechselnd zugeteilt.

Auf dem o.g. System befinde sich zu Beginn nur ein Prozess D, von dem angenommen werden soll, dass dieser niemals terminiert.

Nun wird ein Programm A gestartet, das nach einer festen Anzahl an Operationen ordnungsgemäß terminiert. Die Zeit zwischen Start und Ende dieses Programms (Laufzeit) betrage T_A.

Nach dem Terminieren von A wird ein Programm B gestartet, das ebenfalls nach einer festen Anzahl an Operationen ordnungsgemäß terminiert. Die Zeit zwischen Start und Ende dieses Programms betrage T_B.

(a) (4 P) Für diese Unteraufgaben soll angenommen werden, dass die o.g. Laufzeit $T_A = 100\,\text{s}$ und $T_B = 100\,\text{s}$ betrage.

Programm A und B werden nun zeitgleich gestartet. Nach welcher Zeit (in Sekunden) sind die Prozesse A und B beide nicht mehr aktiv? Fügen Sie der Lösung (bitte entsprechend kennzeichnen) auch eine Skizze/Erläuterung Ihres Rechenwegs bei!

300s, da die 3 Prozesse gleichberechtigt jeweils 2 ZE CPU-Zeit erhalten und A und B jeweils 100s CPU-Zeit brauchen. Die Zuteilung würde etwa so aussehen: A-B-D-A-B-D-A-B-D... bis A-B-D-B-D-D-D. Die Zeiteinheiten sind so klein, dass es in Sekunden keinen Unterschied macht, dass entweder A oder B zuerst fertig sind. Dadurch dass D auch die gleiche Laufzeit bekommt, brauchen A und B 100s mehr bis sie zuende sind.

(b) (5 P) Für diese Unteraufgaben soll angenommen werden, dass die ganz oben angegebene Laufzeit $T_A = 200\,\text{s}$ und $T_B = 100\,\text{s}$ betrage.

Programm A und B werden wieder zeitgleich gestartet. Nach welcher Zeit (in Sekunden) terminiert A? Nach welcher Zeit (in Sekunden) terminiert B? Fügen Sie der Lösung (bitte entsprechend kennzeichnen) auch eine Skizze/Erläuterung Ihres Rechenwegs bei!

300s für B
500s für A

Erklärung siehe a), nur dass noch A und D nochmal jeweils 100s laufen:
A-B-D-A-B-D ... (300s) A-B-D-A-D-A-D... (200s) A-D-D-D

4. Nachfolgend wird das Zusammenspiel von Prozessen und *Pipes* betrachtet.

6 P

(a) (2 P) Schreiben Sie ein C-Programm `printNumbers.c`, welches in aufsteigender Reihenfolge jede natürliche Zahl $n \in [1, 10]$ in einer eigenen Zeile auf `stdout` ausgibt.

Das nach Kompilation aus `printNumbers.c` entstehende *Executable* soll `printNumbers` heißen und mit den zur Ausführung erforderlichen Rechten versehen werden.

```
#include <stdio.h>


int main() {

  for (int i = 1; i <= 10; i++) printf("%d \n", i);
```

(b) (2 P) Schreiben Sie ein zweites C-Programm `processNumber.c`, welches fortwährend via `stdin` eine ganze positive Zahl $i \in \mathbb{Z}^0$ entgegennimmt, diese Zahl verdoppelt und sodann das Ergebnis in einer Zeile, die mit dem eigenen Aufrufnamen beginnt, in folgendem Format auf `stdout` ausgibt: `<aufrufname>: <zahl>`

Das nach Kompilation aus `processNumber.c` entstehende *Executable* soll `processNumber` heißen und die Datei mit den zur Ausführung erforderlichen Rechten versehen werden.

(c) (2 P) Geben Sie die ersten drei Zeilen an, die durch folgende in der Shell eingegebene Kommandozeile entstehen:

```
./printNumbers | ./processNumber
```