Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Betriebssysteme im Wintersemester 2018/2019
Übungsblatt 2

Abgabetermin: 05.11.2018, 18:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 29. Oktober – 02. November 2018
Besprechung der H-Aufgaben in den Tutorien vom 05. – 09. November 2018

Aufgabe 6: (T) Betriebssystem-Schichten

(– Pkt.)

In dieser Aufgabe sollen Sie sich klar machen, wie das Betriebssystem mit den restlichen Komponenten eines Rechners verbunden ist. Finden Sie für die folgenden Begriffe eine sinnvolle Kategorisierung, und stellen Sie die sich ergebenen Rechner-Schichten grafisch dar:

Web-Browser, CPU, Dateisystem, Office-Programme, Ein- und Ausgaberoutinen, Festplatte, Hauptspeicher, Gerätemanagement, Benutzer, Scheduler, Shell, Speicherverwaltung, Unix-Compiler, Drucker, Windows-Systemsteuerung.

Aufgabe 7: (T) Unterprogramme und Stack

(– Pkt.)

In dieser Aufgabe soll das Zusammenspiel von Unterprogrammen und dem Stack untersucht werden. Ein Unterprogramm ist ein Programmstück, welches nur durch den Sprung an seine Anfangsadresse betreten und durch einen Rücksprung an die aufrufende Stelle beendet wird. Diese Art der Programmtechnik wird unter anderem zur Umsetzung rekursiver Aufrufe verwendet. Gegeben das folgende linear rekursive Unterprogramm `mult`:

```
1 int mult(int a, int b) {
2    if(a == 0) {
3       return 0;
4    } else {
5       a = a - 1;
6       return b + mult(a, b);
7    }
8 }
```

Damit man Unterprogramme korrekt ausführen kann muss das Unterprogramm sowie das aufrufende Hauptprogramm bestimmte organisatorische Bedingungen erfüllen. Dazu kann unter anderem ein Stack verwendet werden.
Bearbeiten Sie in diesem Zusammenhang die folgenden Aufgaben:

a. Geben Sie eine Abfolge aller Unterprogrammaufrufe mit den entsprechenden Parametern an, die sich für den Aufruf von `mult(2,3)` ergeben.

b. Welche Zustandsinformationen müssen auf dem Stack gespeichert werden, damit das Hauptprogramm nach dem Unterprogrammaufruf korrekt fortgesetzt werden kann.

c. Geben Sie die aus der Vorlesung *Rechnerarchitekturen* bekannte vierstufige Aufrufkonvention an, die ein korrektes Zusammenarbeiten von Hauptprogramm und Unterprogramm mit Hilfe des Stacks gewährleistet. Geben Sie zu jedem der vier Aufrufe an, welche der in Aufgabe b) genannten Zustandsinformationen jeweils verarbeitet werden.

d. Gehen Sie im Folgenden davon aus, dass der CALL-Befehl für den Aufruf der Funktion `mult` an der Speicheradresse 1000 erfolgt. Die Prozedur `mult` beginnt an der Adresse 4000. Der rekursive Aufruf von `mult` erfolgt an Adresse (4000+x). Skizzieren Sie nun den Stack mit den dazugehörigen Zustandsinformationen unmittelbar vor einem CALL, unmittelbar nach einem CALL und unmittelbar nach einem RET für den Aufruf von `mult(2,3)`. Geben Sie zudem an, welchen Teil der in Aufgabe c) erläuterten Aufrufkonventionen der gegenwärtige Stackzustand repräsentiert. Sie können davon ausgehen, dass die kleinste adressierbare Einheit auf dem Stack einem Wort (4 Byte) entspricht. Dies entspricht auch der Breite eines Befehls samt seiner Parameter. Geben Sie in jedem Schritt stets den gesamten Inhalt des Stacks an.

Aufgabe 8: (T) Unix Prozesse

(– Pkt.)

In dieser Aufgabe sollen Sie sich mit der Verwendung der Systemfunktion `fork()` [1] vertraut machen. Betrachten Sie dazu das folgende C-Programm

```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4
5  int main(void){
6    pid_t kind = fork();
7
8    execl("./hallowelt"," ",NULL);
9    return 0;
10 }
```

Das Bash-Skript `hallowelt` beinhalte folgenden Code:

```
1 #!/bin/bash
2 echo Hallo Welt
```

Beantworten Sie nun die folgenden Fragen.

a. Welche Ausgabe liefert obiges C-Programm auf einem Unix-System? Erklären Sie die Ausgabe.

b. Sie werden beauftragt ein Terminal für Unix zu programmieren, in welches der Benutzer den Namen eines Programmes eingeben kann, welches dann ausgeführt wird. Wird das Programm gestartet, soll der Benutzer weiter Eingaben machen können. Erklären Sie, welche wesentlichen Systemaufrufe Sie wie verwenden müssen.

c. Wie hängt Ihr Programm mit der Baumstruktur von Prozessen in Unix zusammen?

d. Was passiert, wenn der Terminalprozess stirbt?

[1] Informationen zur Systemfunktion `fork()` erhalten Sie in der entsprechenden Manualpage (`man fork`).

Aufgabe 9: (T) Dispatching von Prozessen

(– Pkt.)

Angenommen es existieren drei Benutzer-Prozesse, deren Spuren (Traces) wie folgt aussehen:

Trace des Prozesses A	Trace des Prozesses B	Trace des Prozesses C
$\alpha + 0$	$\beta + 0$	$\gamma + 0$
$\alpha + 1$	$\beta + 1$	$\gamma + 1$
$\alpha + 2$	$\beta + 2$	$\gamma + 2$
(E/A-Op.)	$\beta + 3$	$\gamma + 3$
	$\beta + 4$	(E/A-Op.)
	$\beta + 5$	
	$\beta + 6$	
	$\beta + 7$	
	$\beta + 8$	
	$\beta + 9$	
	$\beta + 10$	

Dabei sind α, β und γ die Anfangsadressen der Prozesse A bzw. B und C. Bei Prozess A soll angenommen werden, dass der dritte Befehl (also $\alpha + 2$) eine E/A-Operation ist, die ein Warten des Prozesses bedingt. Ähnliches gilt für Prozess C und dessen vierten Befehl $\gamma + 3$.

Welche Sequenz abzuarbeitender Befehle ergibt sich aus Sicht des Prozessors, wenn die Ausführung eines Prozesses nach maximal sechs Befehlszyklen durch ein Timeout unterbrochen wird und darauffolgend ein Dispatcher-Prozess für weitere sechs Befehlszyklen aktiv wird, der die Kontrolle an den nächsten Benutzer-Prozess übergibt? Die Befehle des Dispatcher-Prozesses befinden sich im Speicher an den Adressen $\delta + 0$ bis $\delta + 5$.

Aufgabe 10: (H) Multiprogramming

(11 Pkt.)

Beantworten Sie folgende Fragen zum Thema Multiprogramming:

a. Was versteht man unter Multiprogramming?

b. Was ist der Hauptvorteil von Multiprogramming?

c. Nennen Sie jeweils einen Vor- und einen Nachteil eines Multiprozessor-Systems.

d. Betrachten Sie nun die unten stehenden zwei Programme P und Q. A, B, C, D und E sind beliebige atomare (d.h. nicht unterbrechbare) Anweisungen. Die beiden Programme werden jeweils in einem eigenen Prozess pseudo-parallel ausgeführt:

```
1  PROGRAMM P;
2  BEGIN
3    A;
4    B;
5    C;
6  END;
```

```
1  PROGRAMM Q;
2  BEGIN
3    D;
4    E;
5  END;
```

(i) Geben Sie alle möglichen Abläufe dieser Programme (in Form der Reihenfolgen der atomaren Anweisungen) an. Beispiel: Ein möglicher Ablauf wäre A B C D E (zuerst Prozess P komplett, dann Q komplett). Beachten Sie, dass die Anweisungsreihenfolge innerhalb eines Programms nicht verändert werden darf (z.B. darf B nicht vor A kommen).

(ii) Nehmen Sie an, dass die Anweisung B Daten (z.B. Berechnungsergebnisse) erzeugt, die von der Anweisung E gelesen und weiterverarbeitet werden. Welches Problem kann sich aus dieser Abhängigkeit ergeben? Wann tritt es auf?

Aufgabe 11: (H) Einfachauswahlaufgabe: Programme und Unterprogramme

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe explizit die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Was stellt allgemein eine Schnittstelle zwischen Anwendungsprogrammen und Betriebssystem dar, durch die ein Anwendungsprogramm auf eine Ressource zugreifen kann, auf die es keinen direkten Zugriff hat?

(i) Systemaufrufe	(ii) Shared Memory	(iii) Sockets	(iv) Pipes

b) Welche Aussage zu offenen Unterprogrammen ist falsch?

(i) Der entsprechende Programmtext wird an den erforderlichen Stellen ins Hauptprogramm hineinkopiert.
(ii) Sie sind vor allem bei großen/langen Unterprogrammen effizient.
(iii) Nachträgliche Modifikationen am Unterprogramm müssen an jedem Vorkommen des Unterprogramms vorgenommen werden.
(iv) Die Speicheradressen z.B. für Sprungbefehle im Unterprogramm können bei jedem Vorkommen des Unterprogramms verschieden sein.

c) Welche Information wird zur Realisierung eines geschlossen Unterprogramms nicht explizit benötigt?

(i) Anfangsadresse des Unterprogramms	(ii) Aufrufparameter	(iii) Rücksprung-adresse zum Haupt-programm	(iv) Endadresse des Unterprogramms

d) Wie bezeichnet man die sequenzielle, vollständige und unterbrechungsfreie Ausführung von Prozessen?

(i) Multi-programming	(ii) Multi-processoring	(iii) Uniprogram-ming	(iv) Broadprogram-ming

e) Wie ist die mittlere Antwortzeit bei Prozessen mit folgender Ressourcennutzung unter Anwendung von Multiprogramming?

Job	durchschnittliche CPU-Auslastung	Dauer	benötigter Speicher	Platte	Terminal	Drucker
1	50%	5 min.	50 KBytes	-	-	-
2	25%	20 min.	100 KBytes	-	ja	-
3	5%	15 min.	80 KBytes	ja	-	ja

(i) 5 min.	(ii) 13,3 min.	(iii) 20 min.	(iv) 23,3 min.