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 2015/2016
Übungsblatt 2

Abgabetermin: 23.11.2015, 16:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 26. – 30. Oktober
Besprechung der H-Aufgaben in den Tutorien vom 23. – 27. November

Aufgabe 6: (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 7: (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 8: (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  function int mult(int a, int b){
2     if(a == 0) {
3        return 0;
4     }
5     else {
6        a = a - 1;
7        int c = mult(a, b);
8        int d = b + c;
9        return d;
10    }
11 }
```

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 und die Prozedur `mult` selbst an der Adresse 4000 beginnt. 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. Geben Sie in jedem Schritt stets den gesamten Inhalt des Stacks an.

Aufgabe 9: (H) Multiprogramming

(4 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.

Aufgabe 10: (H) Einfachauswahlaufgabe: Programme und Unterprogramme

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

<table>
<tr><td colspan="4">a) Durch welchen Mechanismus kann ein Anwendungsprogramm allgemein auf eine Ressource zugreifen, auf die es keinen direkten Zugriff hat?</td></tr>
<tr><td>(i) Sockets</td><td>(ii) Shared Memory</td><td>(iii) Systemaufrufe</td><td>(iv) Pipes</td></tr>
<tr><td colspan="4">b) Wie heißt ein Unterprogramm, das sich selbst aufruft?</td></tr>
<tr><td>(i) reflexiv</td><td>(ii) regressiv</td><td>(iii) relativ</td><td>(iv) rekursiv</td></tr>
<tr><td colspan="4">c) Welche Information wird zur Realisierung eines geschlossen Unterprogramms nicht explizit benötigt?</td></tr>
<tr><td>(i) Anfangsadresse des Unterprogramms</td><td>(ii) Endadresse des Unterprogramms</td><td>(iii) Rücksprung-adresse zum Haupt-programm</td><td>(iv) Aufrufparameter</td></tr>
<tr><td colspan="4">d) Welche der folgenden Antworten zeigt eine korrekte Implementierung des <code>CALL</code>-Befehls?</td></tr>
<tr><td>(i)
<pre>COMMAND CALL addr
BEGIN
  RA := PC + 1;
  PC := addr;
END</pre></td><td>(ii)
<pre>COMMAND CALL addr
BEGIN
  PUSH (RA);
  PC := POP;
END</pre></td><td>(iii)
<pre>COMMAND CALL addr
BEGIN
  PUSH (PC);
  PC := addr;
END</pre></td><td>(iv)
<pre>COMMAND CALL addr
BEGIN
  RA := PC + 1;
  PC := addr + 1;
END</pre></td></tr>
<tr><td colspan="4">e) Was ist keine allgemeine Komponente, die anderen Komponenten einen Dienst bereitstellt (Modul)?</td></tr>
<tr><td>(i) Unterprogramm</td><td>(ii) Betriebssystem als Ganzes</td><td>(iii) Prozess</td><td>(iv) Client-Anwendung</td></tr>
</table>