Vorlesung Betriebssystemarchitektur WS 2004/05

Aufgabenblatt 2 vom 4. November 2004

(Vorstellung der Lösungen bei den Tutoren bis zum 17. November 2004)

Aufgabe 2.1: (10 Punkte)
Erklären Sie Ihrem Tutor die Begriffe "Kernelmode" und "Usermode" und ihre Verwendung in Betriebssystemen! Welche Vor- bzw. Nachteile ergeben sich durch die Trennung von Kernel- und Usermode?

Aufgabe 2.2: (30 Punkte)
Gegeben sei ein Einprozessor-System, das einen Round-Robin-Scheduling-Algorithmus mit 16 Prioritätsstufen (0-15, 0 = niedrigste, 15 = höchste Priorität) benutzt. Die Länge eines Quantums beträgt 20ms. Die Umschaltzeit zwischen zwei Prozessen ist vernachlässigbar klein. Laufende Prozesse sollen nicht unterbrochen werden können, d.h., sie laufen immer für die Dauer eines Quantums bis eine neue Scheduling-Entscheidung getroffen wird. Es gibt drei Prozesse mit folgenden Parametern:

Prozess	Gesamtausführungsdauer	ausführbereit ab
P1	70ms	t = 0ms
P2	90ms	t = 15ms
P3	80ms	t = 30ms

- Ermitteln Sie die zeitliche Abfolge, in der die drei Prozesse ausgeführt werden, wenn alle die gleiche Priorität 8 haben.
- Wie ändert sich die Abfolge, wenn P3 mit Priorität 9, P1 mit Priorität 7 läuft und nach 16ms Ausführungszeit in einen I/O-Wartezustand gelangt. Der Rest des Quantums soll verfallen. Durch die I/O-Operation wird die Priorität von P1 um 3 Prioritätsstufen erhöht (Prioritäts-Boost). Der Wartezustand von P1 wird bei t = 50ms beendet. Die angehobene Priorität von P1 erniedrigt sich nach jedem abgelaufenen Quantum um eine Stufe bis sie bei der ursprünglichen Basis-Priorität 7 angelangt ist.

Stellen Sie die zeitliche Abfolge und die Prioritäten der drei Prozesse für beide Fälle graphisch in geeigneter Weise dar. Erklären Sie Ihrem Tutor die Diagramme. Bitte die Diagramme in Papierform zum Tutoriumstermin mitbringen und abgeben.

Aufgabe 2.3: (30 Punkte)
Diese Aufgabe beschäftigt sich mit Windows 2000/XP. Machen Sie sich mit der MSDN-Dokumentation vertraut. Lesen Sie nach, wie die Funktionen *CreateProcess()*, *WaitForSingleObject()*, *CloseHandle()*, *GetLastError()*, *GetExitCodeProcess()*, *sprintf()* und *FormatMessage()* funktionieren.
Schreiben Sie in der Programmiersprache C ein Programm, das alle Primzahlen von 2 bis zu einer gewünschten Obergrenze (erster Übergabe-Parameter) ermittelt. Diese Primzahlen sollen aufsummiert werden; das Programm soll lediglich die beiden letzten Stellen dieser Summe als Prüfsumme ausgeben. Die Ermittlung dieser Prüfsumme soll auf mehrere Sub-Prozesse verteilt werden, die Sie im Hauptprogramm mit *CreateProcess()* erzeugen. Die Anzahl der nötigen Rechenschritte soll möglichst gleichmäßig auf die Sub-Prozesse verteilt werden. Die Anzahl der erzeugten Sub-Prozesse soll wählbar sein (zweiter Übergabe-Parameter). Ihre Lösung wird also vermutlich aus zwei Teilen bestehen: dem Hauptprogramm (dieses muss den Namen **`aufg23.exe`** haben) und dem Berechnungsprogramm, das Teil-Prüfsummen berechnet. Die Kommunikation zwischen diesen beiden Programmteilen soll mit Übergabeparametern bzw. Rückgabewert erfolgen. Werten Sie die Fehlerinformationen der benutzten Funktionen aus und

geben Sie ggf. eine entsprechende Fehlermeldung aus.

Auf der Webseite **http://www.dcl.hpi.uni-potsdam.de/uebung/** finden Sie Hilfsfunktionen, die Sie verwenden können. Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei (Name soll keine Umlaute oder Leerzeichen enthalten) und schicken Sie diese ZIP-Datei als Attachment mindestens 24 Stunden vor dem Tutoriumstermin an **bs@hpi.uni-potsdam.de** mit dem Betreff **AUFGABE=2.3 GRUPPE=**<Ihre Übungsgruppennummer> . Die Mail wird automatisch bearbeitet: die ZIP-Datei wird ausgepackt und daraufhin überprüft, ob die Datei **Makefile** enthalten ist. Danach wird der Befehl **nmake aufg23.exe** ausgeführt, welcher das Hauptprogramm und eventuelle Zusatzprogramme erzeugen muss. Die Ausgabe Ihres Programms muss folgendermaßen aussehen - Beispiel: 17 Prozesse berechnen die Prüfsumme der Primzahlen, die kleiner oder gleich 5000 sind:

```
C:\> aufg23.exe 5000 17
36
C:\>
```

Erklären Sie Ihrem Tutor das Programm und führen Sie es vor. Ermitteln Sie für Ihren Rechner die Obergrenze der zu berechnenden Primzahlen (erster Übergabe-Parameter), bei der unter Verwendung **eines** Sub-Prozesses etwa 30 Sekunden Rechenzeit benötigt werden. Wieviel Rechenzeit wird bei der Verwendung von 2, 5, 10, 20, bzw. 50 Sub-Prozessen benötigt?

Aufgabe 2.4: (30 Punkte)

Diese Aufgabe beschäftigt sich mit Unix/Linux. Schreiben Sie ein Programm in der Programmiersprache C, das die gleichen Anforderungen wie in Aufgabe 2.3 erfüllt. Hier sollen Sie jedoch *fork()* und eventuell *exec()* benutzen. Das Hauptprogramm muss den Namen **aufg24** haben und sollte sich durch den Aufruf des Kommandos **make aufg24** übersetzen lassen. Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei
(Name soll keine Umlaute oder Leerzeichen enthalten) und schicken Sie diese ZIP-Datei als Attachment mindestens 24 Stunden vor dem Tutoriumstermin an **bs@hpi.uni-potsdam.de** mit dem Betreff
AUFGABE=2.4 GRUPPE=<Ihre Übungsgruppennummer> .

Erklären Sie Ihrem Tutor das Programm und führen Sie es vor. Ermitteln Sie für Ihren Rechner die Obergrenze der zu berechnenden Primzahlen (erster Übergabe-Parameter), bei der unter Verwendung **eines** Sub-Prozesses etwa 30 Sekunden Rechenzeit benötigt werden. Wieviel Rechenzeit wird bei der Verwendung von 2, 5, 10, 20, bzw. 50 Sub-Prozessen benötigt?