Vorlesung Betriebssystemarchitektur SS 2006
Aufgabenblatt 3 vom 8. Juni 2006
(Vorstellung der Lösungen bei den Tutoren bis zum 21. Juni 2006)

Aufgabe 3.1: (10 Punkte)
Erläutern Sie Ihrem Tutor den Unterschied zwischen preemptiven und nicht-preemptiven Scheduling. Welcher Ansatz ist zu bevorzugen, wenn der Systemdurchsatz maximiert werden soll? Wie versuchen Windows-Serversysteme den Durchsatz zu maximieren?

Aufgabe 3.2: (30 Punkte)
Gegeben sei ein Einprozessor-System welches Round-Robin-Scheduling mit 16 Prioritätsstufen verwendet (0-15, 0 = niedrigste, 15 = höchste Priorität). Die Quantumslänge beträgt 20ms. Die Zeit für einen Kontextwechsel ist vernachlässigbar. Der Scheduler verwaltet laufende Threads und entscheidet ausschließlich nach dem Ablauf eines Quantums welcher Thread als nächstes laufen soll (d.h. es gibt keine Unterbrechungen).

Es sollen drei Threads mit den folgenden Eigenschaften ausgeführt werden:

Thread ID	**Startzeit**	**Ausführungszeit**
Th1	t = 0ms	70ms
Th2	t = 15ms	90ms
Th3	t = 30ms	80ms

- Zeichnen Sie ein Gantt-Diagramm unter der Annahme, dass alle drei Threads mit einer (statischen) Priorität von 8 ausgeführt werden.
- Jetzt soll Th3 mit einer Priorität von 9 ausgeführt werden. Weiterhin erhält Th1 eine Priorität von 7 und tritt nach 16ms in einen I/O-Wartezustand. Die Priorität von Th1 soll nach Beendigung der I/O-Operation um drei erhöht werden („*Boost*“). Th1 verlässt den Wartezustand bei t=50ms. Die Priorität von Th1 wird um eine Stufe am Quantumsende reduziert, bis wieder die Basispriorität erreicht wird.
  Zeichnen Sie ein Gantt-Diagramm für den beschriebenen Vorgang!

Erklären Sie Ihrem Tutor die Diagramme! Bitte bringen Sie die Diagramme in Papierform zum Tutoriumstermin mit.

Aufgabe 3.3: (30 Punkte)
Schreiben Sie in der Programmiersprache C eine Kommando-Shell für Windows. Die Shell soll Kommandos aus einer Datei (1. Übergabe-Parameter) oder von der Standardeingabe zeilenweise einlesen. Jede Zeile enthält genau ein Kommando: ein ausführbares Programm oder ein Shellkommando. Ein ausführbares Programm zeichnet sich durch einen Pfad zum Programm oder durch einen Pfad zum Programm gefolgt von mit einem Leerzeichen getrennten „Und“ (&) aus:

- `C:\windows\system32\notepad.exe` startet das Programm notepad.exe als neuen Prozess und wartet blockierend auf dessen Beendigung.
- `c:\windows\system32\calc.exe &` startet das Programm calc.exe als neuen Prozess und steht danach für weitere Kommandos bereit.

Ein Shellkommando beginnt mit einem Doppelpunkt gefolgt vom eigentlichen Kommando. Es sollen folgende Kommandos unterstützt werden:

- `:wait` wartet blockierend auf die Beendigung aller nebenläufig (also mit &) gestarteten Prozesse
- `:exit` beendet die Shell

Beispiel:

```
> type skript.txt
C:\windows\system32\notepad.exe
C:\windows\system32\calc.exe &
C:\windows\system32\notepad.exe &
C:\windows\system32\charmap.exe &
:wait
C:\windows\system32\sol.exe
:exit
> min_shell.exe script.txt
```

Verwenden Sie zur Implementierung die folgenden API-Funktionen: *CreateProcess(), WaitForSingleObject(), WaitForMultipleObjects(), GetLastError(), FormatMessage(), fgets(), fopen(), fclose(), ferror(), perror(), strerror(), strlen()* und *strcmp()*.

Auftretende Fehler sollen durch die Ausgabe der Systemfehlermeldung, wenn verfügbar, auf die Standardfehlerausgabe dem Benutzer angezeigt werden.

Erläutern Sie Ihrem Tutor Ihre Implementierung und führen Sie Ihr Programm vor!

Erstellen Sie ein Makefile für **`nmake`** mit dem Ziel **`min_shell.exe`**. Verpacken Sie alle Source-Code Dateien (inklusive Makefile) in eine ZIP-Datei (blatt3win.zip). Diese ZIP-Datei muss mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden.

Aufgabe 3.4: (30 Punkte)
Schreiben Sie in der Programmiersprache C eine Kommando-Shell für Linux! Die Shell soll die gleichen Anforderungen aus Aufgabe 3.3 erfüllen.
Verwenden Sie hier jedoch die Systemaufrufe *fork(), exec()* und *waitpid()*.

Erläutern Sie Ihrem Tutor Ihre Implementierung und führen Sie Ihr Programm vor!

Erstellen Sie ein Makefile für **`make`** mit dem Ziel **`min_shell`**. Verpacken Sie alle Source-Code Dateien (inklusive Makefile) in eine ZIP-Datei (blatt3linux.zip). Diese ZIP-Datei muss mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden.