Vorlesung Betriebssystemarchitektur WS 2004/05

Aufgabenblatt 3 vom 18. November 2004

(Vorstellung der Lösungen bei den Tutoren bis zum 8. Dezember 2004)

Aufgabe 3.1: (20 Punkte)
Erklären Sie Ihrem Tutor die Begriffe "Virtueller Speicher", "Swapping", "interne/externe Fragmentierung", "Paging", "Page-Replacement Algorithm" und "Page-Fault".

Aufgabe 3.2: (30 Punkte)
Untersuchen Sie die Page-Replacement-Algorithmen FIFO und LRU. Angenommen, ein System habe nur vier physikalische Speicherseiten und diese seien zunächst nicht benutzt. Stellen Sie für beide Algorithmen graphisch dar, wie zu jedem Zeitpunkt die physikalische Speicherbelegung aussieht, wenn in folgender Reihenfolge auf virtuelle Speicherseiten zugegriffen wird und wann Page-Faults auftreten:

```
9 1 10 9 6 3 13 9 2 1 4 6 8 2 13 13 14 13 4 0 13 1 1 6 4
```

Bringen Sie die beiden Diagramme in Papierform zum Tutoriumstermin mit und erklären Sie die Funktion der Algorithmen.

Aufgabe 3.3: (25 Punkte)
Gegeben sei ein 32-Bit-Rechnersystem. Die Seitenrahmengröße (Page-Frame Size) betrage 1kByte. Der virtuelle Speicheradressraum sei 4GByte groß. Die virtuelle Speicherverwaltung unterstütze lediglich eine Ebene von Page-Tables, d.h. zu jedem Prozess gehört eine Tabelle, die den gesamten virtuellen Adressraum auf den physikalischen abbildet. Diese Tabellen können nicht ausgelagert werden. Der Rechner habe 64MByte physikalischen Speicher. Das Betriebssystem benötigt mindestens 3MByte Speicher, der nicht ausgelagert werden kann. Beantworten Sie Ihrem Tutor folgende Fragen:

- Wieviele Einträge hat ein Page-Table?
- Wieviel physikalischer Speicher wird von einem Page-Table belegt?
- Wenn zwei Prozesse in diesem System laufen, wie groß kann das Working Set eines Prozesses maximal werden?

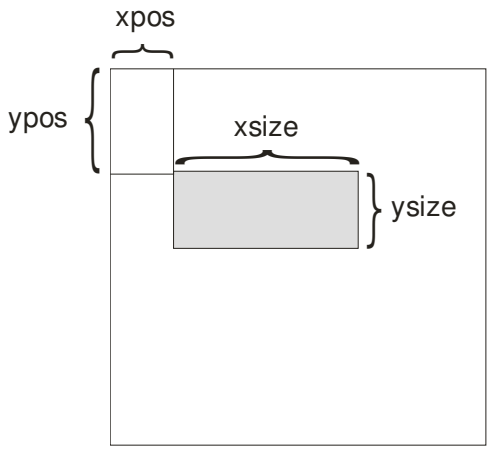
Aufgabe 3.4: (40 Punkte)
Das Programm `mandelbrot.c` (siehe: `http://www.dcl.hpi.uni-potsdam.de/uebung` bzw. `/usr/local/hpi/bs_uebungen/blatt3`) berechnet ein fraktales Bild, in diesem Fall die Mandelbrot-Menge. Dabei wird im komplexen Zahlenraum der Bereich {-1,1},{-i,i} auf ein Bild mit 500x500 Pixeln abgebildet. Die Farbe eines Pixels wird durch die Iterations-Tiefe an der jeweiligen Position bestimmt. Bei einer maximalen Iterations-Tiefe von 200 wird die Iteration abgebrochen und das entsprechende Pixel mit der Farbe schwarz versehen. Die Ausgabe des Programms lässt sich mit einem speziellen Anzeige-Programm anzeigen (siehe oben genannte Webseite bzw. `/usr/local/...`). Die Anzeige-Programme versteht folgende Eingabe:

- das Anzeige-Programm liest Anzeige-Blöcke (einen oder mehrere) von der Standardeingabe
- es werden zeilenweise einzelne Zahlen eingelesen (ASCII-Text)
- ein Anzeige-Block hat folgenden Aufbau:
  <xpos>\n x-Koordinate des oberen linken Punktes des Blocks (0= ganz links)

<ypos>\n y-Koordinate des oberen linken Punktes des Blocks (0= ganz oben)
<xsize>\n Anzahl der Spalten des Blocks
<ysize>\n Anzahl der Zeilen des Blocks
<R>\n Rot-Wert des ersten Pixels (0-255, 0=dunkel)
<G>\n Grün-Wert des ersten Pixels (0-255)
<B>\n Blau-Wert des ersten Pixels (0-255)
... weitere Pixel (xsize*ysize Stück) von links nach rechts, von oben nach unten

Beispiel:

```
130
140
2
2
255
0
0
255
255
0
255
0
0
255
255
0
```



malt zwei rote Pixel untereinander an die Position (130,140) bzw. (130,141) und zwei gelbe Pixel untereinander an die Position (131,140) bzw. (131,141)

Ändern Sie das Programm so ab, dass drei Threads zur Ausführung kommen. Zwei Threads sollen jeweils unterschiedliche Teile des Fraktals berechnen und der dritte Thread soll die Koordinierung übernehmen und die Berechnungsergebnisse in kleinen Stückchen (nicht mehr als hundert Stücke) in oben beschriebenem Format auf die Standardausgabe schreiben. Jedes dieser kleinen Fraktal-Stückchen soll sofort ausgegeben werden, damit der Fortschritt der Berechnung mit dem Anzeige-Programm verfolgt werden kann. Implementieren Sie das Programm entweder unter Windows 2000/XP (`CreateThread()` usw.) oder unter Linux mit der POSIX-Thread-Bibliothek (`pthread_create()` usw.). Achten Sie auf eine aussagekräftige und vollständige Fehlerbehandlung.Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei und schicken Sie diese ZIP-Datei als Attachment mindestens 24 Stunden werktags vor dem Tutoriumstermin an `bs@hpi.uni-potsdam.de` mit dem Betreff
`AUFGABE=3.4 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder
`AUFGABE=3.4 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.
Die Mail wird automatisch bearbeitet: die ZIP-Datei wird ausgepackt und daraufhin überprüft, ob die Datei **`Makefile`** im lokalen Verzeichnis enthalten ist. Danach wird der Befehl **`nmake aufg34.exe`** (Windows) bzw. **`make aufg34`** (Linux) ausgeführt, welcher das Hauptprogramm (**`aufg34.exe`** bzw. **`aufg34`**) im lokalen Verzeichnis und eventuelle Zusatzprogramme erzeugen muss.
Erklären Sie Ihrem Tutor das Programm und führen Sie es vor. Beantworten Sie dabei folgende Fragen:

- Wie haben Sie die Arbeitsaufteilung zwischen den beiden Arbeits-Threads realisiert?
- Wie koordinieren Sie die Ausgabe der berechneten Teil-Fraktale?