Simulation eines virtuellen Speichers (JAVA)

In der Datei **VirtualMemory.zip** befinden sich JAVA-Programme zur Simulation eines Hauptspeicherverwaltungssystems auf Basis eines "virtuellen Speichers" mit lokaler Seitenersetzungsstrategie und fester Hauptspeicherzuteilung pro Prozess.

Folgende Klassen stehen zur Verfügung:

`SimulationEnv`

- Initialisieren des Betriebssystems inkl. Parametern für den aktuellen Simulationslauf, Start/Ende der Simulation durch Erzeugen / Beenden von beliebig vielen Prozessen und Ausgabe einer statistischen Auswertung.

`Process`

- Simulation eines Anwendungsprogramms:
  Read-Operationen ausführen (pseudo-zufallsgesteuert) mit mehreren Operationen im selben Seitenbereich. Der „Lokalitätsfaktor“ bestimmt in der Simulation das Lokalitätsverhalten eines Prozesses, indem er festlegt, wieviele aufeinander folgende Zugriffsoperationen stark benachbarte Adressen verwenden, d.h. wie lange der aktuelle „Working Set“ nicht verlassen wird.

`OperatingSystem`

- Basisfunktionen eines Betriebssystems mit System Calls
  - **`createProcess`**: Prozess-Objekt erzeugen, in Prozesstabelle eintragen, Laden des Programmtextes in den Hauptspeicher und initialisieren der Datenbereiche (durch write-Operationen).
  - **`killAll`**: Alle aktiven Prozesse in der Prozesstabelle beenden
  - **`write`**: Datenwort auf eine virtuelle Adresse im virtuellen Speicher schreiben und ggf. neue Seite erzeugen (→ neuen Seitenrahmen anfordern, neuer Eintrag in der Seitentabelle)
  - **`read`**: Datenwort von einer Adresse im virtuellen Speicher lesen

`PageTable`

- Eine Seitentabelle eines Prozesses sowie eine zirkulare Liste (**`pteRAMlist`**) aller Seitentabelleneinträge, die sich zur Zeit im Hauptspeicher befinden.

`PageTableEntry`

- Datenstruktur eines Seitentabelleneintrags

`FreeListBlock`

- Datenstruktur eines Freibereichslisteneintrags

`Statistics`

- Sammlung und Auswertung statistischer Daten eines Simulationslaufs

1. Ergänzen Sie den Code für die Methoden

```
private int getVirtualPageNum(int virtAdr)    und
private int getOffset(int virtAdr)
```

in der Klasse `OperatingSystem`, die für eine übergebene virtuelle Adresse die virtuelle Seitennummer bzw. den berechneten Offset zurückgeben.

2. Ergänzen Sie den Code für die Methode

```
public synchronized int read(int pid, int virtAdr)
```

in der Klasse `OperatingSystem`, die für einen Prozess mit der angegebenen Prozess-ID (`pid`) ein Datenwort (`int`) von einer Adresse im virtuellen Speicher (`virtAdr`) liest und zurückgibt. Verwenden Sie dafür die vorgegebenen Methoden.

3. Fügen Sie den Code für den Seitenersetzungsalgorithmus RANDOM hinzu

Der Seitenersetzungsalgorithmus RANDOM soll *zufällig* eine beliebige Seite des Prozesses, die sich im Hauptspeicher befindet, zum Verdrängen auswählen. Fügen Sie in der Klasse `PageTable` den entsprechenden Code hinzu:

```
private PageTableEntry randomAlgorithm (PageTableEntry newPte)
```

`randomAlgorithm` erhält als Parameter einen Zeiger auf die „neue“ Seite (als Seitentabelleneintrag), soll eine zu verdrängende Seite auswählen und als Ergebnis zurückgeben. Außerdem muss eine Aktualisierung der internen Liste der Hauptspeicher-Seiten vorgenommen werden.

4. Testen Sie Ihren Code durch Ausführung der Klasse SimulationEnv

Tipps:

- *Über einen Aufruf von* **`setTestMode(boolean testMode)`** *in der Klasse OperatingSystem können Sie eingebaute Testausgaben aktivieren (true) und abschalten (false). Ein Aufruf ist bereits in der Main-Methode von SimulationEnv enthalten.*
- *Sie können unter Eclipse den Consolenpuffer für Ausgaben auf folgende Weise vergrößern: Window – Preferences – Run/Debug – Console – Console buffer size*

5. Führen Sie folgende Simulationsexperimente durch

Verwenden Sie das nun fertige Simulationssystem, um den Einfluss verschiedener Parameter auf die Seitenfehlerrate (Anzahl Seitenfehler / Anzahl Zugriffe auf den virtuellen Speicher) zu untersuchen. Die Parameter können als Werte in der Main-Methode von SimulationEnv gesetzt werden (hier mit Beispielwerten):

- Max. Anzahl Seiten pro Prozess im Hauptspeicher (sonst Verdrängung eigener Seiten)
  - `os.setMAX_RAM_PAGES_PER_PROCESS(10);`
- Seitenersetzungsalgorithmus (hier: RANDOM oder CLOCK oder FIFO)
  - `os.setREPLACEMENT_ALGORITHM(`
    `OperatingSystem.ImplementedReplacementAlgorithms.CLOCK);`
- Lokalitätsfaktor zur Steuerung des Verhaltens von Programmen (Prozessen):
  Anzahl aufeinander folgender Operationen, bei denen sich der Zugriff auf den virtuellen Speicher nur innerhalb weniger Seiten bewegt (nur kleiner Teil des Adressraums wird genutzt).
  - `os.setDEFAULT_LOCALITY_FACTOR(100);`

Ermitteln Sie die Seitenfehlerrate für folgende Parameterkombinationen, wobei Sie die Dauer der Simulation so wählen sollten, dass mindestens 50.000 Zugriffe gezählt werden können. Die Werte für die Programmgröße (5.120 Byte, entspricht 20 Seiten) und die maschinenabhängigen Parameter sollen dabei nicht verändert werden.

MAX_RAM_PAGES_ PER_PROCESS (Hauptspeicherzuteilung)	DEFAULT_ LOCALITY_FACTOR (Lokalitätsfaktor)	Seitenfehlerrate RANDOM	Seitenfehlerrate CLOCK	Seitenfehlerrate FIFO
10	1			
10	10			
10	100			
10	1000			
15	10			
20	10			

Hinweis: ***Schalten*** *Sie für die Simulationsexperimente die* ***Testausgaben ab:*** **`setTestMode(false)`** ***!***

5. Beantworten Sie folgende Fragen:

a) Ist der Wert bei absolut zufälligen Zugriffsfolgen (Lokalitätsfaktor = 1) Ihrer Ansicht nach plausibel? Wenn ja, aufgrund welcher Überlegung?
Tipp: Berücksichtigen Sie die Hauptspeicherzuteilung und die Programmgröße!

b) In welcher Größenordnung liegt (bei diesem einfachen Simulationsmodell) der Leistungsunterschied zwischen CLOCK-, FIFO- und RANDOM-Algorithmus (in %)?

c) Welche Maßnahme zur Leistungssteigerung Ihres Computers können Sie ergreifen, wenn Sie große Programme mit schlechtem Lokalitätsverhalten ablaufen lassen wollen?

Viel Spaß!