Opdracht JSF32, week 12

In deze opdracht experimenteren we met:

* mounten in Linux
* permissions in Linux
* verschillende filetypes en de tijdsverschillen hiertussen bij lezen en schrijven.

Voer de volgende stappen uit:

* Start Linux nog niet op, maar voeg eerst in VMWare een extra virtuele disk toe (grootte bv 8 GB).
* Start Linux nu wel op en installeer GParted: sudo apt-get install gparted
* Mount de nieuwe disk m.b.v. GParted op een geschikte folder.
* Zorg ook dat de disk permanent (dus ook automatisch na een herstart) gemount wordt.   
  Let goed op dat je hier geen fout maakt, omdat dan je hele VM onbruikbaar kan worden.   
  Geef hieronder aan hoe je dat gedaan hebt:

Sudo blkid

Gksudo gedit /etc/fstab

Hiereen een line toegevoegd : UUID=73e5f944-b1a4-48fa-b3b4-918a5a67c0ee /mountpoint ext4 defaults 0 2

Maak vervolgens de volgende 2 applicaties, gebaseerd op de klassen die in blok 1 gebruikt zijn:

* Een variant zonder GUI (dus een console applicatie) die de gebruiker vraagt voor welk level de edges gegenereerd moeten worden, en die edges vervolgens in een file zet (dus niet in een array, en ook niet tekenen).   
  Het level moet ook in de file staan.   
  Voor deze variant worden alleen de klassen KochFractal en Edge uit het vorige kwartaal gebruikt.   
  De betreffende file moet zich op de hierboven gemounte nieuwe disk bevinden.   
  Zet zonodig de permissions goed. Geef hieronder aan welke file (inclusief pad) er precies gebruikt is en welke permissions je aan moest passen.
* Een variant met een GUI die bij elke drawEdges() deze file uitleest (eerst het level, en dan de edges) en de edges in de GUI toont. De edges worden hier dus niet berekend, en ook niet uit een array gelezen, maar steeds uit de file gelezen. KochFractal wordt dus niet gebruikt.

Implementeer bovenstaande zowel met tekstfiles als met binaire files, (naar eigen keuze Data- of Object-streams) en ook met en zonder buffers. (Gebruik voor het inlezen van de tekstfiles de klasse Scanner <http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html> en <http://www.java-made-easy.com/java-scanner.html>).

Dit geeft in totaal dus 4 varianten. Geef in de tabellen hieronder voor elk van deze 4 varianten aan hoeveel tijd het lezen en schrijven van de files kost (doe dit voor 3 van de hogere levels, zodat het genereren van de files minstens enkele seconden duurt; voer elk experiment meerdere keren uit en neem dan een gemiddelde).

File schrijven:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level | Tekstfile  Geen buffer | Binaire file  Geen buffer | Tekstfile  Met buffer | Binaire file  Met buffer |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

File lezen:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level | Tekstfile  Geen buffer | Binaire file  Geen buffer | Tekstfile  Met buffer | Binaire file  Met buffer |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(optioneel)

Test je programma’s ook door in je Linux VM te schrijven, en in je Windows host te lezen en omgekeerd. Hiertoe kun je gebruik maken van een eigenschap van VMware, namelijk dat het mogelijk is, een directory van je host systeem te mounten in je Linux VM. Je gemounte host directory staat dan in /mnt/hgfs. Instellen van welke host directory gedeeld wordt met je VM, kun je instellen bij je VM settings, Options tab, shared folders.