Grunnatriði stýrikerfa
Forritunarverkefni 2 - röðun ferla
Assignment 2 - Process Scheduling

March 2018

Afurðin

Afurðin er forrit, byggt á grunninum sem gefinn er inni á vefsíðu verkefnisins á myschool, sem hermir eftir scheduling á prócessum. Sex mismunandi scheduling policies verða útfærðar. Grunnvirknin sér um að bæta við prócessum og herma eftir keyrslu þeirra, auk þess að láta vita hvaða policy eigi að nota við scheduling. Forritshluti nemenda þarf að geta tekið við þessum upplýsingum og séð um ákvarðanir um það hvaða process keyrir hverju sinni og hvenær þeim er skipt út í keyrslu.

Aðferðirnar sem á að útfæra eru:

- **25%** - First Come First Served (FCFS)
- **20%** - Round Robin (RR)
- **10%** - Shortest Process Next (SPN)
- **10%** - Shortest Remaining Time (SRT)
- **10%** - Highest Response Ratio Next (HRRN)
- **20%** - Feedback (FB)

Að auki á að skila PDF skjali með stuttri lýsingu á útfærslu hverrar aðferðar, með vísunum í kóða, t.d. hvernig time-slicing var útfært þar sem á því þarf að halda, hvernig ready queue og/eða aðrir listar voru útfærðir og hvaða hlutverki þeir gengdu. Í skýrslunni þurfa einnig að vera mæliniðurstöður þar sem kemur fram fyrir hverja aðferð:

- Average Response Time
- Average Turnaround Time

Mæliniðurstöðurnar gilda **5%** af einkunn en lýsingar á aðferðum eru teknar með í matið á hverri aðferð fyrir sig.

Að sækja og skila verkefni

Inni á síðu verkefnisins á myschool er skrá með Java projecti sem er grunnurinn að verkefninu. Sækið, opnið og endurnefnið verkefnið með nafni a.m.k. eins meðlims hópsins ykkar. Þegar forritið er tilbúið er því pakkað í zip/rar/7z skrá og skilað ásamt PDF skrá með skjalinu sem lýsir niðurstöðum keyrslanna. Verkefnið má vinna í allt að þriggja manna hópum.

Forritið

Athugið að nemendur ættu ekki að breyta neinum af klösunum í grunnverkefninu nema klasanum *Scheduler*. Að sjálfsögðu má bæta við klösum og forritshlutum að vild, sem og að breyta *SchedulingMainProgram* til að prófa ákveðna hluti. Allar aðrar skrá sem þegar eru í java projectinu ættu að vera látnar vera.

Athugið einnig að ekkert fall í *Scheduler* klasanum má hanga eða keyra endalausar lúppur, enda er kallað á þau föll úr aðallúppunni sem keyrir prócessana. Föllin sem kallað er á í *Scheduler* (*startScheduling*, *processAdded* og *processFinished*) eiga að stilla það sem þau þurfa, taka ákvörðun, kalla á *switchToProcess* ef þarf og svo klárast. Þannig getur heildarlúppan keyrt óhindrað og einungis notað *Scheduler* til að taka ákvarðanir um hvort skipta eigi um prócess eða setja nýjan í gang. Þannig virkar *Scheduler* að miklu leyti eins og dispatcher, sem er í rauninni bara stutt forrit sem sér um að skipta einum prócess út fyrir annan.

Þegar forritið er sett í gang býr það sjálfkrafa til tilvik af *Scheduler* og sendir því tilvik sem útfærir *ProcessExecution* skilin. Á *ProcessExecution* eru tvö föll sem þið getið kallað á, *getProcessInfo* og *switchToProcess*. Bæði föllin taka *processID* sem heiltölufæribreytu. *getProcessInfo* skilar tilviki af *ProcessInfo* sem inniheldur tölurnar *elapsedWaitingTime* (w), *elapsedExecutionTime* (e) og *totalServiceTime* (s). Útskýringar á þessum tölum og notkun á þeim er bæði í bókinni og fyrsta fyrirlestri um scheduling.

Einungis þegar allir prócessar í testinu hafa runnið sitt skeið hefst testið að nýju og biður nú um að næsta aðferð sé notuð.

Vinnufyrirkomulag

Prófið að breyta fallinu *processAdded* í Scheduler klasanum ykkar og kallið t.d. í fyrsta skiptið sem kallað er á fallið á *processExecution.switchToProcess(processID)*. Sjáið hvað gerist núna þegar fyrsta prócessnum er bætt við. Prófið ykkur áfram með að skipa á milli prócessa til að sjá hvað gerist þegar þeir klára og þegar skipt er á prócess þó að fyrri prócess sé ekki búinn.

Útfærið nú hverja aðferðina á fætur annarri, nýtið tölurnar í *processInfo* þegar þess þarf og útfærið þá lista og raðir sem nauðsynlegar eru á þann hátt sem ykkur hentar best. Þið komið til með að vinna einungis með *processID*, og látið *processExecution* kerfið um að vinna með prócessana sjálfa.

Grunnkerfið keyrir allt í einum þræði, en ykkur er velkomið að nýta þræði og fleira til að útbúa þá atburði sem ykkur vantar til að *Scheduler* geri það sem hann þarf að gera. Kerfið sjálft kallar á *processAdded* og *processFinished* en ykkur er velkomið að útbúa fleiri interruption föll og finna leiðir til að kalla á þau sjálf. Athugið bara að gera ekki *sleep*, endalausar lúppur eða annars konar *block* í aðalþræðinum því það blokkar þá allan herminn í leiðinni.

Góða skemmtun!

The Product

The product is a program, based on the base project provided on the project webpage on myschool, that simulates process scheduling. Six different scheduling policies will be implemented. The base functionality adds processes and simulates them running, as well as letting the scheduler know which policy to use each time. The student's part of the program must process this information and make decisions on which process should run next and when they should be switched out.

The methods to be implemented are:

- **25%** - First Come First Served (FCFS)
- **20%** - Round Robin (RR)
- **10%** - Shortest Process Next (SPN)
- **10%** - Shortest Remaining Time (SRT)
- **10%** - Highest Response Ratio Next (HRRN)
- **20%** - Feedback (FB)

In addition a PDF document should be returned, containing a short description of the implementation of each policy, with references to code, for example how time-slicing was implemented where that is needed, how ready-queues and/or other lists were implemented and what their purpose was. The report should also contain measurement data stating for each policy:

- Average Response Time
- Average Turnaround Time

The measurement data are **5%** of the final grade while the descriptions of the policies will be taken into the evaluation for each policy implemented.

Getting and returning the assignment

On the project website on myschool there is an archive with a Java project which serves as the base for your program. Fetch, open and rename the project with the name of at least one member of your group. Once the program is ready, archive it (zip/rar/7z) and return along with a PDF with your results. The project can be done in groups of three (or less).

The Program

Note that students should not change any of the classes in the base project except the class *Scheduler*. Of course it is allowed to add classes and objects to the heart's content, as well as change *SchedulingMainProgram* to test specific things. All other files that are in the original java project should be left alone.

Also note that no function in the *Scheduler* class can block or run endless loops, as it is the main loop that runs the processes which calls those functions. The operations called on *Scheduler* (*startScheduling*, *processAdded* and *processFinished*) should set and initialize what they need, make a decision, call *switchToProcess* if needed and then finish. That way *Scheduler*, for the most part, works like a dispatcher, a small program which simply switches one process out for another.

When the program is started, it automatically makes an instance of *Scheduler* and sends it an instance of a class implementing the *ProcessExecution* interface. *ProcessExecution* has two operations that you can call, *getProcessInfo* and *switchToProcess*. Both take *processID* as an integer parameter. *getProcessInfo* returns an instance of *ProcessInfo* which holds the numbers *elapsedWaitingTime* (w), *elapsedExecutionTime* (e) and *totalServiceTime* (s). Explanations of these numbers and their use are in the book and the first lecture on scheduling.

Only if all processes in a test case are run to completion does the test case start over, now asking for the next policy.

The Process

Try to change the function *processAdded* in your *Scheduler* class and, for example, for the first time it's called, call *processExecution.switchToProcess(processID)*. See what happens now when the first process is added. Experiment with switching between processes to see what happens when they finish and when a process is preempted before it finishes.

Now implement all the policies. Use the numbers from *ProcessInfo* when needed and implement any lists and queues you need any way you feel works best. You will only be working with *processID* and allowing the *processExecution* system to handle the processes themselves.

The base program runs in a single thread, but you are welcome to use threads or other means to make the events you need for Scheduler to work as intended. The system itself calls *processAdded* and *processFinished* but you are welcome to implement more interruption operations and find ways to call them yourselves. Just make sure you don't *sleep*, run endless loops or *block* in any way in the main thread, as that will block the entire simulation as well.

Have a good time!