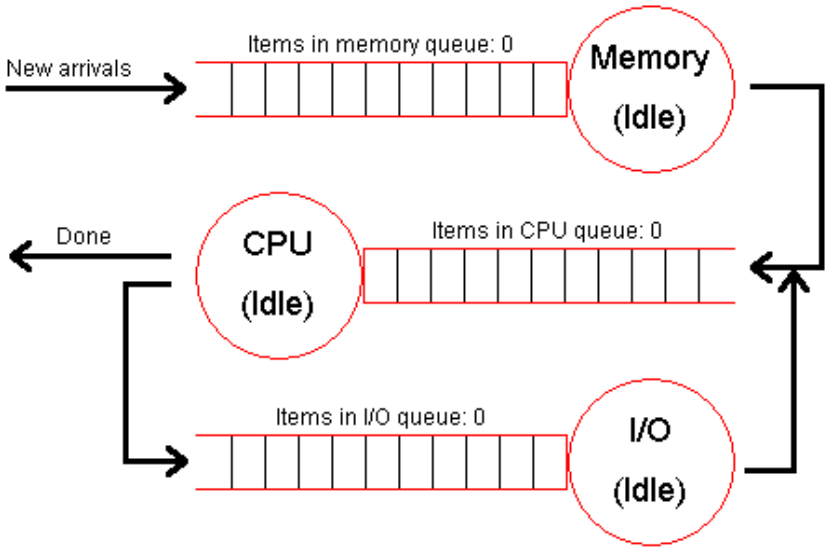
P3 – Tidsstyring av prosesser

Overordnet formål

Her skal dere implementere en form for tidsstyring av prosesser ved å simulere bruk av Round-Robin algoritmen i et forenklet datamaskinsystem. Oppgaven omfatter problemstillinger tilsvarende lærebokas kapittel 2.4.

Betrakt en enkel datamaskin med et enkelt CPU, et lite lager samt en I/O enhet. Datamaskinen behandler prosesser som utfører batchjobber med statisk minnebehov. Det finnes bare en type lager (minne), altså ingen form for virtuelt minne. Datamaskinen er modellert som et køsystem som vist i figuren under, der prosesser ankommer med jevne mellomrom og forsvinner ut av systemet når de er ferdig med jobben sin.



Det er tre køer i systemet:

- En kø for prosesser som venter på å få tildelt minne.
- En kø for prosesser som venter på prosessortid.
- En kø for prosesser som venter på å få utføre I/O.

Utnyttelsen av CPU-kraften styres med Round Robin-algoritmen. De andre køene opereres etter FIFO-prinsippet. Jobbene og deres behov genereres tilfeldig.

Detaljert beskrivelse

Programmet som skal lages, må først etterspørre en del parametere fra brukeren for å kontrollere simuleringen. Dette gjelder:

- Tilgjengelig lagerstørrelse (for eksempel 2048 KB)
- Tidskvant for Round Robin (for eksempel 500 ms)
- Gjennomsnittlig I/O operasjontid (for eksempel 225 ms)

- Total simuleringstid (for eksempel 250.000 ms)
- Gjennomsnittlig tid mellom ankomst for nye jobber (5.000 ms)

For hver ny jobb som skal kjøres, må systemet videre generere tilfeldige parameterverdier for de tre karakteristikaene:

- Prosessens lagerbehov (for eksempel 100 KB). Lagerbehov skal variere mellom 100 KB og 25% av

tilgjengelig lagerstørrelse.
- Prosessens eksekveringstid (for eksempel 5.000 ms). Eksekveringstid skal variere mellom 100 ms og 10.000 ms.
- Prosessens gjennomsnittlige eksekveringstid mellom I/O-behov (for eksempel 250 ms). Gjennomsnittlig tid mellom I/O-behov skal variere mellom 1% og 25% av prosessens eksekveringstid.

Den tilfeldige tallgeneratoren må også benyttes for følgende tre andre forhold: Tid fram til neste I/O-behov hver enkelt gang (variert rundt tilsvarende gjennomsnittlige tid), tid for neste I/O-behov hver enkelt gang (variert rundt tilsvarende gjennomsnittlige tid), samt tid til neste jobb-ankomst hver enkelt gang (variert rundt tilsvarende gjennomsnittlige tid).

En nyankommet jobb plasseres i lagerkøen. Her venter prosesser på tilstrekkelig lagerplass. Køen administreres ut fra FIFO prinsippet. En gitt prosess trenger ikke å vente på at dens lagerbehov kan dekkes som en enkelt minneblokk, bare på at dens lagerbehov kan dekkes totalt sett. Virtuelt lager eller swapping brukes ikke. Med andre ord trenger lageret kun å holde rede på hvor mye minne som er ledig til enhver tid.

Når en prosess får tildelt lagerplass, flyttes den direkte over i CPU-køen. Selve minnetildelingen tar ingen tid, så minne-enheten vil til enhver tid vises som ”Idle”, men minnekøen kan være lang. I CPU-køen venter prosesser på CPU-kraft. Denne køen administreres altså ut fra Round Robin prinsippet.

Når en prosess får tildelt CPU-en, beholder den kontrollen inntil ett av tre forhold skjer: Når tidskvantet spesifisert av RR-algoritmen er oppbrukt – hvorpå prosessen flyttes tilbake i CPU-køen, når et I/O-behov oppstår – hvorpå prosessen flyttes over til en egen I/O-kø, eller når prosessen er ferdig – hvorpå prosessens lagerplass tilbakeleveres og den forlater systemet.

En prosess som står i den egne I/O-køen, flyttes derfra og over til CPU-køen igjen etter tilhørende I/O-avslutning.

Når simuleringen er avsluttet – dvs. når simuleringstiden er ute selv om det fortsatt kan være prosesser som ikke er ferdige, skal programmet rapportere følgende:

- Antall ferdige prosesser
- Antall opprettede prosesser
- Antall prosessskifter (som skyldes oppbrukt tidskvant)
- Antall utførte I/O-operasjoner
- Gjennomsnittlig gjennomstrømning (ferdige prosesser per sekund)

- Total tid CPU har brukt på å prosessere
- Total tid CPU har vært ledig
- Prosentvis hvor mye tid CPU har brukt på å prosessere (utilisasjon)
- Prosentvis hvor lenge CPU har vært ledig
- Størst forekommende samt gjennomsnittlig lengde på alle køer
- Hvor mange ganger en ferdige prosess gjennomsnittlig har blitt plassert i hver enkelt kø

- Gjennomsnittlig tid tilbrakt i systemet per ferdige prosess
- Gjennomsnittlig tid ventende på tilstrekkelig lagerplass per ferdige prosess
- Gjennomsnittlig tid ventende på CPU-kraft per ferdige prosess
- Gjennomsnittlig tid brukt i CPU per ferdige prosess
- Gjennomsnittlig tid ventende på I/O-kapasitet per ferdige prosess
- Gjennomsnittlig tid brukt i I/O per ferdige prosess

Slik rapportering krever at ulike verdier tas vare på underveis i simuleringen. Resultatene avhenger mye av hvilke parametere som brukes i simuleringen, og dette åpner opp for ulik eksperimentering for å forbedre systemet.

Nyttige tips

Denne oppgaven er et eksempel på såkalt diskret hendelsessimulering. To hovedkomponenter vil være en klokke og en hendelseskø. Klokken vil holde oversikt over forløpt tid i simuleringen, mens hendelseskøen vil holde styr på framtidige hendelser i simuleringen. Eksempler på hendelser i denne sammenhengen er: Ankomst av en ny jobb, utløp av et tidskvant, oppstart av en I/O-operasjon, avslutning av en I/O-operasjon og terminering av en eksisterende jobb. Framtidige hendelser er vanligvis sortert på tid i hendelseskøen.

Hovedløkken i simuleringen tar hele tiden første hendelse ut av hendelseskøen, oppdaterer tiden tilsvarende den nye hendelsen og behandler hendelsen i seg selv. Således hopper tiden hele tiden framover i diskrete steg. Hvis flere hendelser skal skje på samme tid, vil de håndteres i rekkefølge uten at tiden forandrer seg. Behandlingen av en hendelse kan medføre at nye hendelser skapes og legges inn i hendelseskøen på riktig sted.

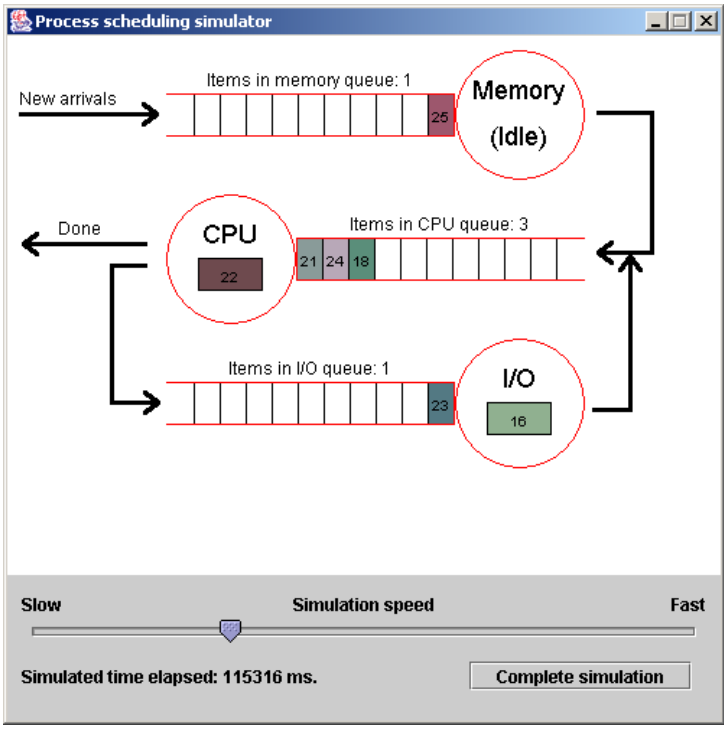
Når en prosess er ferdig før tidskvanten er over, vil ikke prosessoren (CPUen) starte med neste prosess.Prosessoren bare venter ("Idle state") til tidskvanten er over og bare da kan en ny prosess bli tildelt prosessoren.

Som en liten pekepinn er kjøreutskriften fra et korrekt fungerende program vist under:

```
Please input system parameters:
Memory size (KB): 2048
Maximum uninterrupted cpu time for a process (ms): 500
Average I/O operation time (ms): 225
Simulation length (ms): 250000
Average time between process arrivals (ms): 5000
Simulating.....done.

Simulation statistics:

Number of completed processes:                                39
Number of created processes:                                  48
Number of (forced) process switches:                          207
Number of processed I/O operations:                           753
Average throughput (processes per second):                    0.156

Total CPU time spent processing:                              199213 ms
Fraction of CPU time spent processing:                        79.6852%
Total CPU time spent waiting:                                 50787 ms
Fraction of CPU time spent waiting:                           20.3148%

Largest occuring memory queue length:                         2
Average memory queue length:                                  0.13544
Largest occuring cpu queue length:                            6
Average cpu queue length:                                     1.753956
Largest occuring I/O queue length:                            6
Average I/O queue length:                                     1.121616
Average # of times a process has been placed in memory queue: 1
Average # of times a process has been placed in cpu queue:    20.333334
Average # of times a process has been placed in I/O queue:    14.256411

Average time spent in system per process:                     22772 ms
Average time spent waiting for memory per process:            657 ms
Average time spent waiting for cpu per process:               10104 ms
Average time spent processing per process:                    4465 ms
Average time spent waiting for I/O per process:               4329 ms
Average time spent in I/O per process:                        3215 ms
```

Tilgjengelig delløsning

For å gjøre arbeidsomfanget mindre, vil vi her ferdigstille utvalgte moduler av det som etterspørres. Vi har laget en GUI som viser køsystemmodellen og hvor i systemet prosessene befinner seg. Et skjermbilde fra GUI'en er vist under.



GUI'en har et par metoder som må kalles på passende tidspunkter for at simulasjonens oppførsel skal vises i GUI'en. Disse metodene er deklarert i grensesnittet Gui, som deles ut. GUI'en er implementert av de utdelte klassene SimulationGui, PicturePanel, Resource og Constants. Når det gjelder selve simuleringen er klassene Event, EventQueue og Memory ferdig implementerte, mens oppstartsklassen Simulator og klassen Process er delvis implementert. Dere må ferdigstille resten av systemet, inkludert en klasse som simulerer CPU'en og en som simulerer I/O.

For å gjøre koden mest mulig ryddig anbefaler vi å samle flest mulig av variablene som fører statistikk med kjøringen på ett sted, gjerne i en separat klasse.

Kode for, utfyllende beskrivelse av og kommentar til de utleverte modulene vil bli gjort tilgjengelig dels via hjemmesiden, dels som papirkopier og dels i øvingstimene.

Eksakt oppgave

Følgende forventes som resultat av denne øvingen:

- Innlevering av kildekode til programmet – inkludert en kommentarseksjon som klart angir hvordan du vil eksperimentere med Round Robin algoritmen for å forbedre systemet.
- Demonstrasjon av programmet for undervisningsassistent / studentassistent.
- Innlevering av utskrift fra kjøringen – inkluderende en resultatseksjon som detaljert angir effektene av Round Robin eksperimenteringen. Prøv å identifisere nøkkelparametere som har stor innvirkning på systemets totale ytelse.
- Innlevering av en report (1 side) som skal beskriver hva skjer i køene hvis vi endre lagerstørrelse, tidskvant og gjennomsnittlig I/O operasjontid.