Tema 2 Structuri de Date-CB
Planificator de procese

Responsabili tema:	**Vali Ghita, Cosmin Oprea**
Data publicarii:	**16.04.2015**
Termenul de predare:	**30.04.2015 ora 23:55** **Se accepta teme trimise cu penalizare 10 puncte /zi (din max 100 puncte) până la data 3.05.2015 ora 23:55.**

I. Introducere

Orice sistem de operare modern conține un subsistem care se ocupa cu administrarea proceselor, numit planificator de procese, al cărui rol este de a determina la fiecare moment de timp procesul care se execută pe procesor.

În cadrul unui astfel de planificator, orice proces este caracterizat de două valori:

- **id** - un identificator unic al procesului în cadrul sistemului;
- **prioritate** - un număr întreg pozitiv, care determină ordinea în care sunt planificate procesele.

Un proces se poate afla la un anumit moment în una din următoarele stări:

- **ready** - așteaptă să fie planificat, deoarece exista un proces cu o prioritate mai bună care rulează în acel moment;
- **waiting** - procesul așteaptă terminarea unui eveniment extern (de exemplu de intrare/ieșire). Chiar daca procesul are prioritatea cea mai mare, el nu va fi planificat pentru execuție până când evenimentul pe care îl așteaptă nu se produce;
- **running** - procesul este cel ales pentru a rula, deoarece este procesul cu cea mai bună prioritate dintre cele care nu așteaptă după niciun eveniment.

În concluzie, la un anumit moment de timp va fi ales pentru execuție procesul care are cea mai mare prioritate și nu așteaptă producerea unui eveniment.

II. Enunț

Scopul temei este de a simula execuția unui planificator de procese, care va funcționa conform descrierii de mai sus.

Pentru simplificare, va exista un număr fix de evenimente care pot fi așteptate de procese, iar fiecare proces va putea aștepta după maxim un eveniment. Fiecare eveniment este identificat printr-un numar între 0 și numarul total de evenimente - 1. Numărul total de evenimente va fi citit din fișierul de intrare.

Fiecare proces va fi reprezentat printr-o structură în care se reține id-ul, prioritarea și momentul de timp la care a fost pornit (pasul).

Informatiile asociate proceselor vor fi păstrate în urmatoarele structuri de date:

- procesele în starea ready și procesul în starea running vor fi memorate într-o coadă de priorități, în care ordinea este dată de prioritatea proceselor; primul proces din această coadă va fi întotdeauna procesul care este planificat pentru execuție.
- pentru fiecare eveniment extern va exista câte o stivă, în care vor fi memorate procesele care așteaptă producerea evenimentului respectiv. Deci, dacă numărul total de evenimente este 10, vor exista 10 stive, câte una pentru fiecare eveniment.

În momentul în care un proces intră în așteptare după un anumit eveniment, el va fi scos din coada de priorități și va fi introdus în stiva asociată evenimentului pe care îl așteaptă.

Exemplu: Să presupunem ca numărul total de evenimente este 3 și în sistem există procesele următoare [1, 5] (id 1 și prioritate 5), [2, 7], [3, 2], [4, 8], [5, 1], [6, 4], [7,5]. Aceste procese au intrat în sistem în ordinea în care au fost enumerate.

Dacă niciun proces nu este în așteptare, coada de priorități va conține procesele în următoarea ordine: 4, 2, 1, 7, 6, 3, 5. În acest caz, stivele asociate celor trei evenimente sunt vide, iar procesul planificat este cel cu id-ul 4.

Dacă, pe rând, procesesele 3, 4 și 1 intră în așteptare după evenimentul 1, coada devine 2, 7, 6, 5, iar stiva asociată evenimentului 1 va conține procesele 1, 4, 3. În vârful stivei se află procesul 1, pentru că a fost adăugat ultimul.

Dacă procesele 5 și 6 intră în așteptare după evenimentul 2, în această ordine, coada devine 2, 7, iar stiva asociată evenimentului 2 conține procesele 6, 5.

În momentul apariției evenimentului 1, toate evenimentele din stiva asociată sunt mutate în coadă, deci coada devine 4, 2, 1, 7, 3

Dacă apare și evenimentul 2, coada va avea conținutul inițial, iar cele 2 stive asociate evenimentelor vor fi vide.

Execuția planificatorului va consta în mai mulți pași. La fiecare pas se efectuează următoarele operații:

1. Se citește o acțiune din fișierul de intrare
2. Se efectuează prelucrări asupra structurilor de date în care se păstrează procesele
3. Se scriu rezultatele modificărilor în fișierul de ieșire.

O acțiune se va citi dintr-un fiser de intrare si poate fi una din următoarele tipuri:

- start X Y - pornire proces cu id X și prioritate Y.
- wait E X- procesul X intră în așteptare după evenimentul E. Procesul va exista întoteaduna în sistem înainte de această acțiune și va fi în starea ready sau running, adică nu așteaptă dupa alt eveniment.
- event E - evenimentul E se produce, deci toate procesele care așteptau după el vor ieși din starea waiting, deci vor fi scoase din stiva corespunzătoare evenimentului și introduse în coada de priorități.
- end X - procesul cu id X își încheie execuția și trebuie eliminat din planificator.

III. Fișierele de intrare și ieșire

Informatiile necesare planificatorului vor fi preluate dintr-un fisier de intrare. rezultatele se vor genera intr-un fisier de iesire.

Structura fisierului de intrare este urmatoarea:

- pe prima linie se găsește numărul total de evenimente
- următoarele linii contin câte o acțiune, conform formatelor de mai sus (fiecare actiune este specificata pe o linie separata).

După fiecare pas, în fișierul de ieșire se vor genera următoarele informații, pe linii distincte (separate prin \n):

- numărul pasului (începând cu 1);
- conținutul cozii de priorități cu procese în starea ready și running (inclusiv primul proces, care este cel planificat); vor fi afișate id-urile proceselor, separate prin spațiu; În cazul în care coada este goală, se va afișa o linie goală.
- conținutul stivelor asociate evenimentelor, doar stivele care nu sunt goale, câte una pe fiecare linie. Formatul de afișare este E: X1 X2 .. Xn , unde E este numărul evenimentului și X1 .. Xn sunt id-urile proceselor care așteaptă după evenimentul respectiv. Procesele vor fi afișate în ordinea în care sunt memorate în stivă, adică ultimul element adăugat va fi afișat primul.
- o linie goală.

Exemplu de fișier de intrare:

```
5
start 1 5
start 4 10
start 2 7
wait 3 1
end 2
```

Exemplu de fișier de ieșire:

```
1
1

2
4 1

3
4 2 1

4
4 2
3: 1

5
4
3: 1
```

IV. Detalii de implementare

Tema va fi implementată în limbajul C. Structurile pentru stivă și coadă vor fi implementate folosind liste simplu inlantuite generice. Fișierele sursă vor fi compilate folosind gcc, cu ajutorul unui Makefile ce trebuie inclus în arhiva finală.

Executabilul generat trebuie sa se numească tema2. Numele fișierelor de intrare și ieșire vor fi primite ca parametrii in linia de comanda.

Exemplu de rulare:

./tema2 in.txt out.txt

V. Reguli de trimitere a temelor

A se vedea și Regulile generale de trimitere și punctare a temelor [2]
Temele vor trebui incarcate atat pe vmchecker (in secțiunea Structuri de Date (CB): **SD-CB**) cat si pe cs.curs.pub.ro, in sectiunea aferenta Temei 2.
Arhiva cu rezolvarea temei trebuie să fie .zip și să conțină:

- fisiere surse (fiecare fisier sursa creat sau modificat va trebui sa inceapa cu un comentariu de forma:

/* NUME Prenume - grupa */

- fisier README care sa contina detalii despre implementarea temei și eventualele probleme întâlnite.
- fișier Makefile cu doua reguli: Fișierul pentru make trebuie denumit obligatoriu Makefile și trebuie sa contina urmatoarele reguli:
  - build, care va compila sursele si va obtine executabilul, cu numele tema2.
  - clean, care va sterge executabilele generate.
- arhiva nu trebuie să contina decat fisierele sursa (nu se accepta fișiere executabile sau obiect)
- daca arhiva nu respecta specificațiile de mai sus nu va fi acceptata la upload și tema nu va fi luata in considerare

VI. Precizări suplimentare

a. Dacă există mai multe procese cu aceeași prioritate, ordinea este dată de momentul în care acestea au fost pornite. Procesul cel mai prioritar va fi cel care a fost pornit cel mai devreme.
b. Un proces poate fi terminat numai dacă este în starea ready sau running, adică dacă este memorat în coada de priorități. Daca este în așteptare, acțiunea de terminare va fi ignorata, iar procesul va rămâne în starea în care se află.
c. Operatiile permise pe structurile de date sunt urmatoarele:

- pentru coada de prioritati sunt permise operatiile de inserare ordonata si de eliminare a elementului cu prioritatea cea mai mare (primul element din coada).
- pentru stiva sunt permise operatiile specifice (push si pop);
- pentru ambele structuri este permisa parcurgerea pentru afisare.

VII. Notare

- 80 puncte obținute pe testele de pe vmchecker (punctajul pentru fiecare test este obtinut din rezultatul efectiv plus existenta sau nu a memory leak-urilor)
- 10 puncte: coding style, codul trebuie sa fie comentat, consistent si usor de citit (a se vedea [1]). De exemplu, tema nu trebuie sa contina:
    - warninguri la compilare;
    - linii mai lungi de 80 de caractere
    - tab-uri amestecate cu spatii; denumire neadecvata a functiilor sau a variabilelor
- 10 puncte: README + comentarii + alte eventuale penalizari
- **Bonus 20 puncte** ultimele 2 teste (inclusiv testul pentru memory leak).
- **O tema care nu va compila se va nota automat cu 0.**

Referințe:

[1] http://ocw.cs.pub.ro/courses/programare/coding-style
[2] http://cs.curs.pub.ro/2014/mod/page/view.php?id=4309