Kacper Leśniański

Systemy Operacyjne Projekt

Symulacja algorytmów planowania czasu procesora

**Opis programu**

Program symuluje działanie algorytmów przydziału czasu procesora.

**Dane:**  
Dane wejściowe są przyjmowane w postaci wartości oddzielonych przecinkami w pliku *txt*.

* ID – identyfikator procesu   
  (l. całkowita >= 0)
* Czas przybycia (Arrival time) – Czas, w którym proces zmienia stan na Gotowy  
   (l. całkowita >= 0)
* Czas wykonania (Burst time)– Czas potrzebny procesowi na wykonanie   
  (l. całkowita > 0)
* Priorytet – Priorytet procesu (używany przez algorytm priorytetowy)  
  (l. zmiennoprzecinkowa >= 0)

Format:  
Jedna linia dla każdego procesu

<id>,<czas\_przybycia>,<czas\_wykonania>,<priorytet>

Dodatkowo po uruchomieniu programu w konsoli należy podać długość kwantu czasu dla algorytmu Round Robin, oraz tempo postarzania dla algorytmu priorytetowego.

**Działanie:**

Program wykonuje symulację działania algorytmów i rysuje diagramy przedstawiające kolejność wykonywania procesów.

Kwant czasu symulacji to 1ms. Przyjęte zostało 0 jako czas przełączenia procesu (po zakończeniu wykonywania jednego natychmiast jest uruchamiany następny).

Domyślnie program działa w trybie „manualnym” – Naciśnięcie klawisza **spacja** powoduje przejście do następnego kroku symulacji.

Naciśnięcie klawisza **enter** włącza tryb automatyczny. Klawiszami **+** i **–** można regulować szybkość symulacji.

W dowolnej chwili trwania symulacji można dodać nowy proces. Naciśnięcie klawisza **A** wyświetla konsolę i pozwala na dodanie nowego procesu.

Po zakończeniu symulacji dla każdego algorytmu obliczony zostaje średni czas oczekiwania oraz zapisany raport.  
W raporcie zapisywane są dane każdego procesu. Są zapisane w postaci pliku *csv* w kolejności zakończenia wykonywania. Tabela zawiera wszystkie informacje podane w danych i dodatkowo:

* Czas zakończenia (completion time) – Czas, w którym proces zakończył wykonywanie
* Czas oczekiwania (waiting time) - Długość czasu, przez który proces gotowy do wykonania musiał czekać na czas procesora

**Algorytmy**

* **FCFS**  
  Procesy są szeregowane na podstawie czasu przybycia – im wcześniej proces przybędzie, tym wcześniej zostanie wykonany.  
  W przypadku gdy kilka procesów ma ten sam czas przybycia, najpierw wykonywany jest ten, który ma wyższą pozycję na liście.
* **SJF (Niewywłaszczający)**Procesy są szeregowane na podstawie czasu wykonania – najkrótszy proces jest wykonywany pierwszy.  
  W przypadku gdy kilka procesów ma ten sam czas wykonania, najpierw wykonywany jest ten, który ma wyższą pozycję na liście.
* **Round Robin**Każdy proces ma przydzielony kwant czasu, po którym zostaje wywłaszczony, a czas procesora jest przekazywany następnemu.  
  Czas zostaje po kolei przydzielony każdemu procesowi na liście.
* **SJF Priorytetowy**Procesy są szeregowane na podstawie priorytetu – im wyższy priorytet, tym wcześniej zostanie wykonany proces.  
  Najwyższy priorytet ma wartość 0 (im mniejsza wartość, tym wyższy priorytet)  
  W każdym cyklu symulacji priorytet oczekujących procesów jest zwiększany o pewną ustaloną wartość (postarzanie).  
  W przypadku, gdy kilka procesów ma ten sam priorytet, szeregowanie jest przeprowadzane wg zasad algorytmu SJF.

**Testy**

**Test 1: Weryfikacja poprawności działania programu i porównanie algorytmów**

Dane testowe:

Kwant czasu RR: 2ms

Tempo postarzania: 0.5

Procesy:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Czas przybycia | Czas wykonania | Priorytet |
| 0 | 0 | 3 | 0 |
| 1 | 0 | 2 | 0 |
| 2 | 5 | 5 | 1 |
| 3 | 5 | 4 | 7 |
| 4 | 10 | 3 | 6 |
| 5 | 20 | 2 | 3 |
| 6 | 21 | 1 | 1 |

Uzasadnienie wyboru tego zestawu danych:

* Czas przybycia dwóch pierwszych procesów jest równy. Pozwala to sprawdzić, jak algorytm FCFS uszereguje je w tej sytuacji.
* Priorytet dwóch pierwszych procesów jest równy. Pozwala to na sprawdzenie, czy algorytm priorytetowy prawidłowo je uszereguje wg. zasad SJF.
* Przetestowanie mechanizmu postarzania – proces 4 ma wyższy priorytet niż proces 3 ale przez postarzanie powinien wykonać się po nim.
* Sprawdzenie, czy program prawidłowo zachowa się podczas okresu bezczynności procesora między 17 a 20 milisekundą.
* Niewielka ilość procesów pozwala na stosunkowo łatwą weryfikację prawidłowego działania programu.

Procesy zostały najpierw poszeregowane „ręcznie” a następnie wyniki porównano z danymi wyjściowymi programu.

*Dane wyjściowe znajdują się w folderze raporty\_data1*

**Przewidywane wyniki:**

Obraz zawierający zegar, niebo

Opis wygenerowany automatycznie

**Wyniki działania programu:**

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Wniosek:** Program działa poprawnie

**Test 2: Porównanie algorytmów**

Dane testowe:

Kwant czasu RR: 2ms

Tempo postarzania: 0.5

Dane:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Czas Przybycia | Czas Wykonania | Priorytet |
| 0 | 0 | 4 | 4 |
| 1 | 7 | 5 | 7 |
| 2 | 6 | 6 | 6 |
| 3 | 9 | 6 | 0 |
| 4 | 8 | 7 | 8 |
| 5 | 6 | 9 | 7 |
| 6 | 3 | 7 | 2 |
| 7 | 0 | 7 | 7 |
| 8 | 9 | 3 | 0 |
| 9 | 7 | 5 | 4 |
| 10 | 4 | 7 | 9 |
| 11 | 6 | 6 | 8 |
| 12 | 3 | 4 | 9 |
| 13 | 3 | 6 | 8 |
| 14 | 6 | 8 | 9 |

Dane zostały dobrane losowo.

**Wyniki:**

*Diagram znajduje się w dołączonym pliku diagram2.png*

*Dane wyjściowe znajdują się w folderze raporty\_data2*

W obu testach najniższy średni czas oczekiwania miał algorytm SJF (1,57ms i 30,67ms).  
Najdłuższy średni czas oczekiwania miał algorytm Round Robin (2,71ms i 61,27ms).

W obu testach algorytm Priorytetowy SJF miał niższy średni czas oczekiwania niż algorytm FCFS. Są to jednak różnice dość niewielkie.

Test 1: FCFS: 1,86ms; PSJF: 1,71ms

Test 2: FCFS: 37,53ms; PSJF: 34ms

**Wnioski:**

Dla wybranych zestawów danych algorytm SJF ma najniższy czas oczekiwania, najgorzej wypada pod tym względem Round Robin.

Przewaga PSJF nad FCFS jest niewielka. Możliwe, że odpowiednie dobranie priorytetów dla procesów umożliwiłoby osiągnięcie krótszego czasu oczekiwania niż w przypadku FCFS.