Programy Symulacyjne – Sprawozdanie

Mateusz Mędykowski

241423

Środa 7:30-9:00

Cyberbezpieczeństwo

Spis treści

1. Algorytmy przydziału czasu procesora

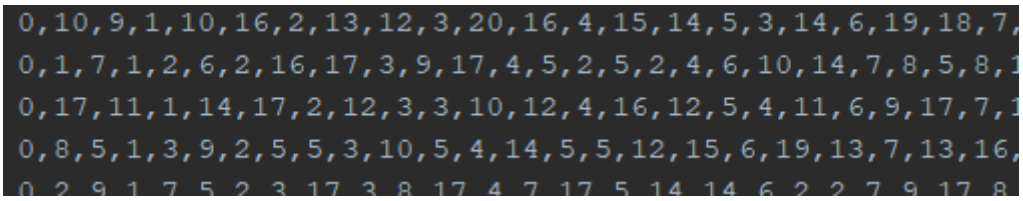
Do generowania procesów został stworzony program „generowanie_procesow2.py”. Dane o procesach zapisane są w pliku „procesy.txt”.

Zapisane są w formacie:

-każda linia pliku odpowiada jednej próbie

-dane o procesach są zapisane po przecinku w kolejności: „id procesu 1, czas wykonania procesu 1, czas przybycia procesu 1 do procesora, id procesu 2, czas wykonania procesu 2, czas przybycia procesu 2 do procesora”

-id są numerowane inkrementacyjnie od 0, długość wykonania procesu trwa od 1 do 20ms, czas przyjścia jest ustawiony od 0 do 20ms.



1.1 FCFS

1.1.1 Opis działania algorytmu

FCFS - pierwszy zgłoszony, pierwszy obsłużony. Procesy są obsłużone w kolejności w jakiej zostały zgłoszone do procesora

1.1.2 Odczytywanie danych z pliku

Każda linia w pliku jest danymi z jednej próby. Dane dla pojedynczego procesu są umieszczane w liście „proces”, a następnie taka lista jest dodawana do listy „procesy”

1.1.3 Działanie programu

Zakładamy, że do programy od razu przychodzą wszystkie procesy. Funkcja „szeregowanie_procesów()” szereguje je według działania algorytmu FCFS, dodając je do listy „kolejka”.

Funkcja „obliczenia()” oblicza czas oczekiwania w kolejce i czas cyklu. Dane o pojedynczych procesach przyjmują wygląd listy w formacie „[id procesu, czas wykonania procesu, czas przybycia procesu do procesora, czas oczekiwania, czas cyklu]”

```
[2, 8, 1, 0, 8]
```

Funkcja „statystyka()” oblicza czas średniego czasu oczekiwania i średniego czasu cyklu z danej próby i umieszcza dane w listach „czasy_oczekiwan” i „czasy_cykli”.

1.1.4 Wyniki

Wyniki średniego czasu oczekiwania i średniego czasu cyklu ze wszystkich prób są zapisywane do pliku „FCFS_wyniki.txt”

1.2 SJF

1.2.1 Opis działania algorytmu

SJF- najpierw najkrótsze zadanie. W pierwszej kolejności, procesor obsługuje zadanie o najniższym czasie wykonania.

1.2.2 Dane wejściowe

Każda linia w pliku jest danymi z jednej próby. Dane dla pojedynczego procesu są umieszczane w liście „proces”, a następnie taka lista jest dodawana do listy „procesy”

1.2.3 Działanie programu

Procesy są umieszczone w liście „procesy”. W ustawianiu procesów w kolejce kluczowa jest zmienna „czas_pracy” która zakłada czas pracy programu. W funkcji „obliczenia()”, zmienna „czas_pracy” czeka aż trafi się proces z identycznym czasem przybycia. Taki proces jest umieszczany w liście „kolejka”. Na podstawie danych o czasie pracy programu i danych procesu, są obliczane czasy oczekiwania i czasy cyklu analogicznie jak w symulacji FCFS. Wykonane procesy są umieszczane w liście „wyniki”.

Funkcja „statystyka()” analogicznie jak w FCFS oblicza średni czas oczekiwania i czas cyklu z próby.

1.2.4 Wyniki

Wyniki średniego czasu oczekiwania i średniego czasu cyklu ze wszystkich prób są zapisywane do pliku „FCFS_wyniki.txt”

1.3 Podsumowanie działania pracy algorytmów przydziału czasu procesora i wnioski

Do plików wyjściowych trafiają dane o średnim czasie oczekiwania i cyklu procesu ze wszystkich prób.

FCFS:

```
Sredni czas oczekiwania w kolejce:  514.982  ms
Sredni czas cyklu:  525.579  ms
```

SJF:

```
Sredni czas oczekiwania w kolejce:  351.896  ms
Sredni czas cyklu:  362.493  ms
```

Po obliczaniach jednego i drugiego algorytmu możemy wywnioskować, że algorytm SJF sprawuje się znacznie lepiej. Jeżeli przemyślimy proces działania danych algorytmów dla tak dużych prób to:

-FCFS wypada gorzej ponieważ na początku mogło przyjść kilka procesów o długim czasie realizacji, a później procesy, które miały krótki czas realizacji. Przez to procesom o krótkim czasie realizacji bardzo wydłużył się czas oczekiwania oraz czas cyklu. Warto wspomnieć, że procesy miały, bardzo ograniczony czas przyjścia bo od 1 do 20 ms. To też wpłynęło na wydłużenie czasów oczekiwania i cyklu.

SJF wypadł lepiej, do procesora musiało przyjść dużo procesów o krótkim czasie realizacji, które średnią czasów oczekiwania i cykli zaniżyli.

SJF nie posiadał mechanizmu przeciwdziałającemu zagłodzeniu procesu. Była ograniczona liczba procesów wiec nie było potrzebny stosowania takiego mechanizmu. W normalnej pracy procesora, mógłby przyjść proces o bardzo długim czasie realizacji, który by był odkładany w kolejce i nigdy by się nie wykonał. Doszłoby do zagłodzenia procesu.

2. Algorytmy stronicowania

Do generowania stron został stworzony program „generowanie_stron.py”. Dane o procesach zapisane są w pliku „strony.txt”.

Zapisane są w formacie:

-każda linia pliku odpowiada jednej próbie

-numery stron są wypisane po przecinku

-strony przyjmują wartości od 1 do 30

```
15,17,24,14,21,5,29,13,11,30,4,9,21,30,26,21,28,
27,2,24,7,14,21,18,30,13,14,9,11,28,21,13,19,19,
28,14,2,15,1,9,2,2,8,18,8,20,30,10,11,27,16,29,5
```

2.1 Algorytm FIFO

2.1.1 Opis działania algorytmu

FIFO-pierwszy przyszedł, pierwszy wyszedł. Algorytm obsługuje strony według kolejności przybycia.

2.1.2 Dane wejściowe

Dane wejściowe są odczytywane z pliku i umieszczane w liście „kolejka”

2.1.3 Działanie programu

Na początku działania algorytmu jest sprawdzana zmienna „potrzebny” zawierająca informacje czy jakaś strona potrzebuje procesora oraz zmienna „brakujący” informująca czy dana strona jest już w pamięci procesora, a w tym programie z liście „bufor”. Bufor ma pojemność 20 stron.

Jeśli jest strona, która jest potrzebna to sprawdzamy czy jest w buforze. Jeśli jej nie ma to sprawdzamy czy jest miejsce w buforze. Jeśli jest umieszczamy na końcu. Jeśli nie ma to usuwamy pierwszą stronę z bufora i wstawiamy aktualną stronę na koniec.

W dalszej części cyklu jest sprawdza strona, która następnie będzie potrzebna oraz czy jest ona w buforze.

Po każdym cyklu możemy sprawdzić stan „migawki". Zawiera ona informacje o zawartości bufora oraz potrzebnej i brakującej stronie. Jeżeli brakująca strona = 0 to oznacza, że potrzebna strona jest w buforze.

```
[[5, 1, 4], 5, 0]
```

W buforze są strony: 5,1,4. Następna w kolejce to strona 5, jest ona w buforze wiec równa zmienna brakującej stron jest równa 0.

2.1.4 Wyniki

Wyniki o ilości brakujących stron są zapisane w pliku „FIFO_wyniki.txt"

2.2 Algorytm LFU

2.2.1 Opis działania algorytmu

LFU-najwcześniej używany zostaje zastąpiony. Jeżeli przychodzi nowa strona, to strona która była najwcześniej używana w pamięci, zostaje zastąpiona nową stroną.

2.2.2 Dane wejściowe

Dane wejściowe są odczytywane z pliku i umieszczane w liście „strony"

2.2.3 Działanie programu

W pierwszej kolejności strony są numerowane, pojedyncza strona przyjmuje format listy zawierającej informacje o numerze strony i kolejności przybycia

```
[5, 1]
```

Takie listy są umieszczane w liście „kolejka"

```
[[5, 1], [1, 2], [4, 3], [5, 4], [1, 5], [2, 6], [4, 7], [1, 8], [4, 9], [5, 10]]
```

Przykładowa kolejka

Na początku działania algorytmu jest sprawdzana zmienna „potrzebny" zawierająca informacje czy jakaś strona potrzebuje procesora oraz zmienna „brakujący" informująca czy dana strona jest już w pamięci procesora, a w tym programie z liście „bufor". Bufor ma pojemność 20 stron.

Jeśli jest strona, która jest potrzebna to sprawdzamy czy jest w buforze. Jeśli jej nie ma to sprawdzamy czy jest miejsce w buforze. Jeśli jest umieszczamy na końcu. Jeśli nie ma to sprawdzamy która strona była najwcześniej używana, usuwamy stronę z bufora i wstawiamy aktualną stronę na koniec.

W dalszej części cyklu jest sprawdza strona, która następnie będzie potrzebna oraz czy jest ona w buforze.

Po każdym cyklu możemy sprawdzić stan „migawki”. Zawiera ona informacje o zawartości bufora oraz potrzebnej i brakującej stronie. Jeżeli brakująca strona = 0 to oznacza, że potrzebna strona jest w buforze.

```
[[[5, 4], [1, 2], [4, 3]], [1, 5], 0]
[[[5, 4], [1, 5], [4, 3]], [2, 6], [2, 6]]
[[[5, 4], [1, 5], [2, 6]], [4, 7], [4, 7]]
```

Krok1: W buforze są strony: 5,1,4. Następna w kolejce to strona 1, jest w buforze wiec zmienna brakującej stron jest równa 0.

Krok2: W buforze są strony: 5,1,4. Została zmieniona dana przy stronie 1 (z 2 na 5) opisująca informacje kiedy strona została użyta. Następna w kolejce jest strona 2, brakuje jej w buforze.

Krok3: Strona 4 była najwcześniej używana wiec zostaje zastąpiona nową stroną.

2.2.4 Wyniki

Wyniki o ilości brakujących stron są zapisane w pliku „LFU_wyniki.txt”

2.3 Podsumowanie pracy algorytmów stronicowania i wnioski

FIFO:

```
srednia ilosc brakujących stron:  42.27
```

LRU:

```
srednia ilosc brakujących stron:  42.86
```

Oba algorytmy wypadły bardzo podobnie. Nieco mniej stron brakowało w algorytmie FIFO. Ciężko jest ocenić, który algorytm jest lepszy. Wszystko zależy od kolejności w jakiej przyjdą strony. Osobiście uważam, że najwydajniejszy jest algorytm FIFO ponieważ tylko przesuwa strony, nie dokonując dodatkowych obliczeń.