Systemy Operacyjne

Adrian Matkowski

Styczeń 2023

Spis treści

1 Wstęp

Celem projektu było wykonanie dwóch symulacji, w dowolnym języku programowania, dzięki którym będziemy mogli wykonać analizę danych oraz porównać rózne rozwiązania problemu. Pierwszą symulacją był "Process Scheduler", drugą natomiast "Page Manager" Jako język wybrałem Python w wersji 3.10.5. Algorytmy które postanowiłem zaimplementować do poszczególnych symulacji to:

- Dla Process Scheduler - First Come First Serve oraz Last Come First Serve,
- Dla Page Manager - First In First Out oraz Least Recently Used.

2 Zasada działania algorytmów

2.1 Process Scheduler

2.1.1 First Come First Serve

Kolejność wykonywania procesów można porównać do kolejki, pierwszy proces, który przyszedł, będzie wykonywany jako następny. Z uwagi na to, że nie jest to algorytm wywłaszczający, to kiedy Proces przyjdzie w traktcie wykonywania innego Procesu, będzie on musiał poczekać aż do momentu ukończenia wcześniejszego procesu.

2.1.2 Last Come First Serve

Analogicznie porównując do poprzedniego algorytmu, możemy sposób wykonywania porównać do stosu. Zamiast chronologicznego sposobu wykonywania procesu, kolejny wykonywany proces będzie ten, który ostatni przyszedł. LCFS również nie jest wywłaszczający, więc bedzie on czekał na "swoją kolej".

2.2 Page Manager

Należy zauważyć, że jeśli dana strona jest już w oknie, nie ma potrzeby dodawania jej ponownie.

2.2.1 First In First Out

Podczas, kiedy wszystkie okna będą już zapełnione i przyjdzie strona (Page), która nie jest w danym momencie w oknie, strona podlegające usunięciu będzie ta, która jako pierwsza została dodana to okna. Przedstawiając to na przykładzie: jeśli mamy 3 dostępne okna i mają one w sobie następujące strony [1] [2] [3] a przyjdzie strona o numerze 4, okno ulegające wyczyszczeniu będzie to z 1 w środku, ponieważ ono jako pierwsze zostało zapełnione. Aby zachować porządek, która strona przyszła jako pierwsza organizujemy okienka w następujący sposób: [2] [3] [4], aby najstarsza strona znajdowała się po lewej stronie.

2.2.2 Least Recently Used

LRU jest w zasadzie podobnie działającym algorytmem, jednak z jedną zmienioną regułą. Zamiast najstarszej strony, wylatuje strona najmniej używana. Aby zachować porządek i śledzić, która strona będzie nadawała się do wyrzucenia, to przy każdej powtarzającej się stronie, przerzucamy ją na prawą stronę. Dzięki temu, strony, które nie są używane znajdują się po lewej stronie, czyli strona z której "wylatują" z okien.

3 Poszukiwane dane

Dane, które będziemy wykorzystywać do porównywania algorytmów to:

3.1 Process Scheduler

- Czas całkowity wykonania algorytmu
- Średni czas oczekiwania na wykonanie procesu

3.2 Page Manager

- Ilość wykonanych podmian strony.

4 Sposób użycia programu

Obydwie symulacji składaję sie z 3 części:

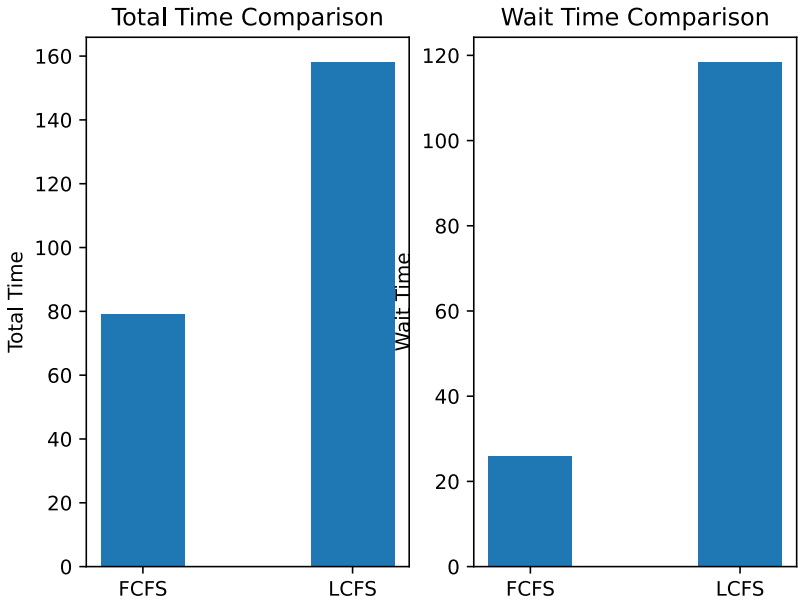
- Generowanie danych,
- Symulacjia,
- Narysowanie wykresu.

Aby wygenerować losowo dane, należy uruchomić odpowiedni skrypt generaotra w folderze *generators*. Następnie, po podaniu nazwy, np. *data* i wpisaniu pożądanych wartości, zostanie stworzony plik *data.json* w folderze *input*. Kolejnym krokiem jest uruchomienie symulacji z poziomu terminala i podanie w pierwszym argumencie ścieżkę do pliku wejściowego, np *data.json* (program automatycznie wybiera pliki z folderu input). Po uruchomieniu skryptu, w folderze *output* pojawią nam się pliki z nazwą odpowiedniego algorytmu. Ostatnim, opcjonalnym krokiem jest wykonanie wykresu porównującego z otrzymanych nam danych. Skrypty generujące znajdują się w folderze *plots*, a wykresy są generowane w *plots/output*.

5 Porównywanie wyników

5.1 Process Scheduler

Przy wykonywaniu badań użyłem podane przykładowe pliki znajdujące się w *input*. Wyniki końcowe przedstawione są na rysunku 1.

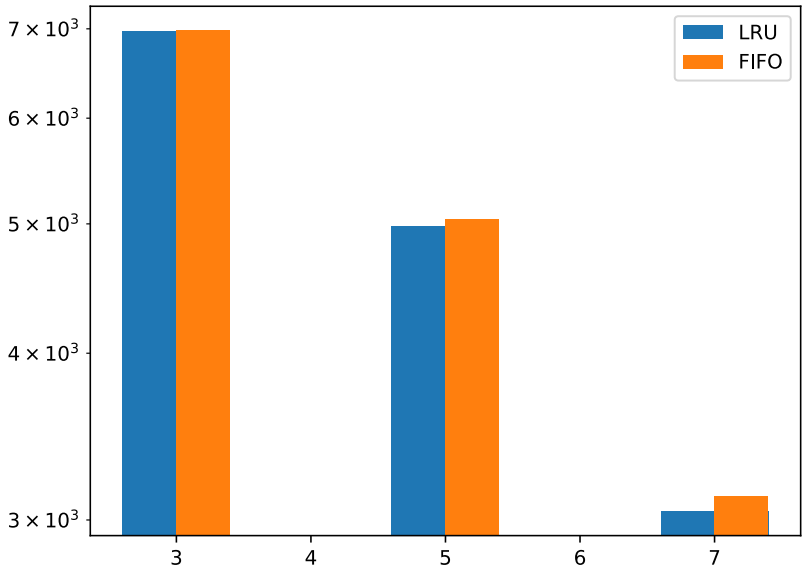


Rysunek 1: Porównanie sprawności dwóch algorytmów.

Dzięki wykresom, jesteśmy w stanie zauważyć, że algorytm $FCFS$ jest o wiele bardziej efektywny i sprawny. Przy użytym pliku wejściowym średni czas oczekiwania na wykonanie procesu wynosi zaledwie ok. 25 jednostek czasu, gdzie przy $LCFS$ jest to już w okolicach 120 jednostek. Czas wykonania jest również o wiele szybszy, w tym przypadku aż ok. 2 razy. Na podstawie tych danych, jesteśmy w stanie stwierdzić, że w tym przypadku algorytm $FCFS$ jest o wiele bardziej efektywny, niż $LCFS$.

5.2 Page Manager

W drugiej symulacji postanowiłem porównać sprawność dwóch algorytmów, poprzez analizę ilości wykonywanych zamian stron. Wykres porównawczy widoczny jest na rysunku 2.



Rysunek 2: Porównanie sprawności dwóch algorytmów.

Patrząc po wykresie, jesteśmy w stanie stwierdzić, że algorytm $FIFO$ wykonał mniejszą ilość zamian stron niż LRU, co może oznaczać większą wydajność. Warto jednak zauważyć, że przy zwiększającej się ilości wymian stron, różnica wydajności algorytmów zaczyna się robić minimalna.