19.01.2019, Wrocław

SPRAWOZDANIE

ALGORYTMY PRZYDZIAŁU CZASU PROCESORA

Rafał Szymanek Nr. Indeksu: 241436

Student Politechniki Wrocławskiej

Kierunek: Cyberbezpieczeństwo

Wydział: Elektroniki(W-4)

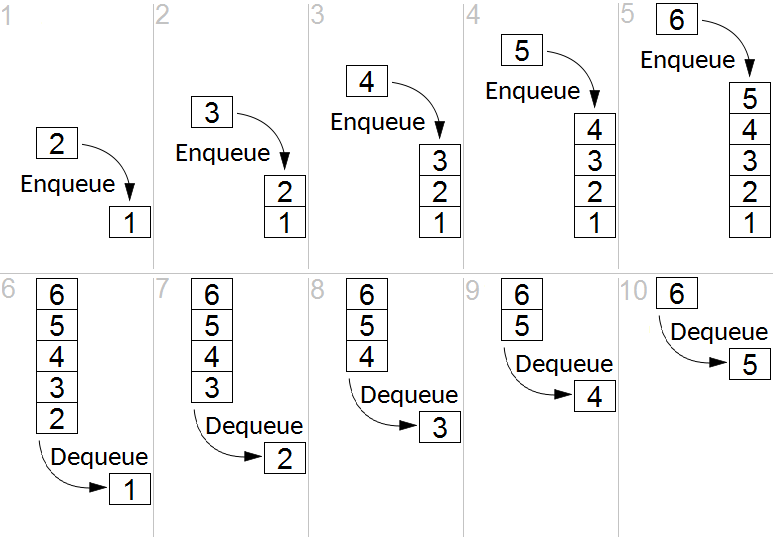
# Wprowadzanie

Test został przeprowadzony w ramach formy zaliczenia kursu „Systemy Operacyjne” u prowadzącego dr inż. Pawła Trajdosa na Politechnice Wrocławskiej.

W ramach sprawozdania mieliśmy porównać dwa wybrane przez siebie algorytmy przydziału czasu procesora, a następnie je porównać i wyciągnąć odpowiednie wnioski.

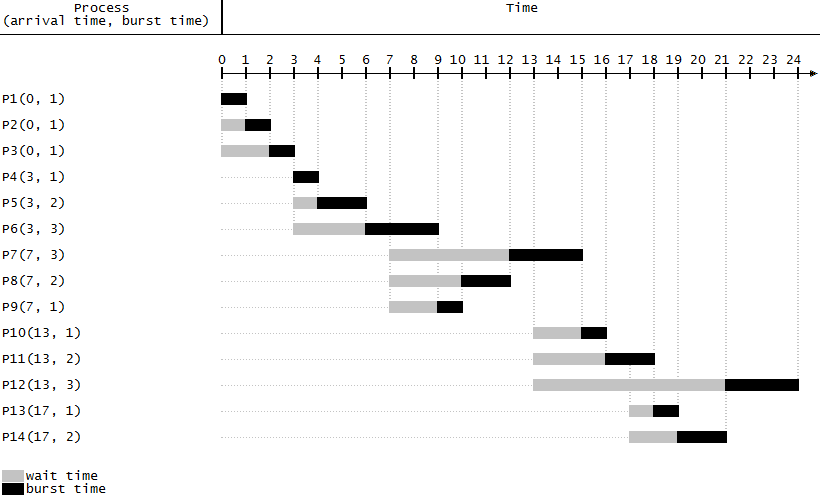
## Wybrane algorytmy przydziału czasu procesora

**FCFS** – Jest algorytmem niewywłaszczający opierającym się o kolejkę FIFO. Oznacza to, że procesy wykonywane są w kolejności takiej jakiej pojawiły się w systemie.



Wikipedia.org

**SJF** – Algorytm w wersji nie wywłaszczeniowej. Oznacza sortowanie kolejki względem procesu, który ma najkrótszy czas wykonania, do procesu o najdłuższym czasie wykonania.



Wikipedia.org

# Opis procedury testowania

Założeniem tego testu było napisanie dwóch osobnych programów, symulujących wykonanie dwóch algorytmów. Każdy z nich miał za zadanie obliczenie czasu oczekiwania, a także czasu przetwarzania procesu dla tych samych danych wejściowych.

## Założenia:

- Kryteria oceny algorytmów:

- Średni czas oczekiwania na przydzielenie procesora.

- Średni czas cyklu przetwarzania.

- Wszystkie procesy przychodzą w jednym czasie – różnice w czasie nadejścia są̨ pomijalnie

- Symulujemy działanie algorytmu dla jednego procesora

- Każdy proces opisuje pozostały czas obliczeń́ (1, 20) [s].

- Danymi testowymi jest 100 prób, które zawierają po 100 procesów do wykonania.

- Wyniki zostały uśrednione

## Programy symulacyjne

Programy użyte w teście są programami autorskimi stworzonymi specjalnie na potrzeby testu. Wykonują one tylko i wyłącznie działania matematyczne.

Ich zadaniem jest:

1. Wygenerowanie danych i zapis do pliku
2. Pobranie danych z pliku
3. Wykonanie algorytmu szeregowania procesów
4. Wykonanie algorytmów dla próby
5. Uśrednienie wyników i wyświetlenie ich na ekranie
6. Zapis wyników do pliku

# Wyniki testu

Wyniki testu są przeprowadzone dla konkretnej próby.

Dane wejściowe dla próby zapisane są w pliku: **/data/officialInputData.txt**

Dane wyjściowe zostały zapisane w pliku: **/data/officialResult.txt**

|  |  |  |
| --- | --- | --- |
| Nr. Testu | Czas oczekiwania FCFS [s] | Czas oczekiwania SJF [s] |
| 1 | 500.49 | 365.41 |
| 5 | 540.01 | 354.12 |
| 20 | 465.63 | 321.52 |
| 71 | 580.2 | 402.11 |
| 100 | 521.3 | 348.17 |
| Śr. Całk. | 517.2925 | 353.3185 |

|  |  |  |
| --- | --- | --- |
| Nr. Testu | Czas przetwarzania FCFS [s] | Czas przetwarzania SJF [s] |
| 1 | 511.18 | 376.1 |
| 5 | 550.59 | 364.7 |
| 20 | 475.3 | 331.19 |
| 71 | 591.71 | 413.62 |
| 100 | 531.85 | 358.72 |
| Śr. Całk. | 527.7456 | 363.7716 |

Powyższe tabele przedstawiają wyniki dla poszczególnych prób o wymienionych numerach. Ostatni wiersz zawiera średnią całkowitą ze wszystkich prób\*

\*Nie tylko wyników zawartych w tabeli, ale wszystkich uzyskanych wyników z danych, które wprowadziliśmy.

# Wnioski

Porównując oba algorytmy możemy zauważyć znaczącą przewagę algorytmu SJF nad algorytmem FCFS. W żadnej z prób algorytm FCFS nie uzyskał korzystniejszego czasu oczekiwania i przetwarzania.

Warto jednak nadmienić, że w naszym teście nie uwzględniamy czasu sortowania algorytmu SJF (uważamy go za pomijalnie mały), co wpływa na korzyść tego algorytmu. Dodatkowo wszystkie procesy przychodzą w jednakowym czasie (pomijalnie małym), dzięki czemu algorytm SJF może od razu posortować wszystkie procesy na raz. Wykonując w kolejności od tego, który wykona się najkrócej.

Dlaczego jeden algorytm ma nad drugim taką przewagę? Ponieważ algorytm SJF sortuje procesy czekające w kolejce i wykonuj od razu te, które wykonywane są najkrócej. Dzięki takiemu podejściu uzyskujemy najkrótsze czasy oczekiwania i przetwarzania. Jednakże ten algorytm ma znaczącą wadę jaką może być zagłodzenie\* dłużej trwających procesów. Aby tego uniknąć powinniśmy wprowadzić priorytety względem czasu oczekiwania.

\*głodzenie procesu – zachodzi wtedy, kiedy cały czas napływają do kolejki procesy o krótszym czasie zajętości procesora. Może zajść wtedy taki wyjątek, że dany proces nigdy się nie wykona.