Dokumentacja

Symulacja - Algorytmy przydziału czasu procesora

Autor:   
Rafał Szymanek, student Politechniki Wrocławskiej  
nr. Indeksu: 241436  
Wydział: Elektroniki  
Kierunek: Cyberbezpieczeństwo

# Dokumentacja użytkownika

## Założenia wstępne:

Napisany program jest częścią składową zaliczenia kursu „Systemy Operacyjne” u prowadzącego dr inż. Pawła Trajdosa.

Program służy do przeprowadzenia symulacji przydziału czasu procesora dla dwóch algorytmów **FCFS** oraz **SJF**. Dzięki owej symulacji będziemy mogli porównać ze sobą owe dwa algorytmy i stwierdzić, który z nich byłby wydajniejszy w realnym użytku.

**FCFS** (First-Come, First-Served scheduling) – jest to algorytm nie wywłaszczeniowy, który przydziela czas procesora na podstawie utworzenia wątku. Dzięki temu powstaje kolejka, która jest kolejką FIFO.   
<https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)>

**SJF** (Shortest Job First) – algorytm w wersji nie wywłaszczeniowej. Jego zadaniem jest uszeregowanie i wykonanie procesów w kolejności od procesu, który potrzebuje najkrótszego czasu użycia procesora, do tego, który wątek zajmie procesorowi najwięcej czasu.   
<https://en.wikipedia.org/wiki/Shortest_job_next>

**Uwaga!** Algorytm może zachować się różnie w zależności od danych wejściowych. Pomimo symulacji może okazać się, że w skrajnych sytuacjach wydajność algorytmów będzie różna. Dlatego też zaleca się jak największą ilość danych wejściowych, aby zmniejszyć prawdopodobieństwo uzyskania mało realistycznych wyników.

Program uwzględnia, że:  
- wątkimają**jednakowy czas przyjścia** (różnice w czasie przyjścia są pomijalne),  
- wątki nie **oczekują na udzielenie zasobów,**

- używamy tylko **jednego rdzenia** do obsłużenia wątków.

## Sposób oceny algorytmów:

Kryteriami jakimi program ocenia zaimplementowane algorytmy są:

- **średni czas oczekiwania na przydzielenie procesora** (czas, który wątek musiał odczekać w kolejce gotowych procesów)

- **średni czas cyklu przetwarzania** (czas od momentu przyjścia wątku, aż do jego pełnego zakończenia)

Program przetwarza wszystkie przypadki, które umieściliśmy na wejściu. Następnie nastąpi przetwarzanie danych, z których uzyskamy średnią arytmetyczną oraz medianę dla obu algorytmów. Dzięki temu zabiegowi, jesteśmy w stanie ocenić efektywność algorytmów przydziału czasu procesora.

# Dokumentacja techniczna

Język programowania: Python

Wersja: 3.7.2

Program pisany: Obiektowo

Język używany w kodzie: Angielski

Kontrola wersji: Git

Link: <https://github.com/rafalszymanek/Simulation_allocation_of_the_processor_time>  
  
Program został przetestowany na systemie operacyjnym: MacOS Mojave 10.14.2

## Struktura plików:

simulation.py – plik główny, który wykonuje cały program

LICENSE – licencja programu

README.md – Krótki poradnik jak uruchomić program

Katalogi:

data/:  
Katalog zawierający wszystkie pliki, na których pracuje program

docs:/

Katalog zawiera całą dokumentacje programu

sample:/

Katalog zawiera wszystkie autorskie moduły używane w programie

Pliki katalogu:

thread.py – plik zawierający klasę „Thread”

fcfsalgotythm.py – plik zawiera implementacje algorytmu FCSF na klasie „Thread”

# simulation.py

Plik, który jest odpowiedzialny za wykonanie całego programu.

## Import

Importuje całą zawartość modułu fcfsalgotythm.py

## Zmienne

listOfWaitingTime – LISTA - zmienna globalna pliku „fcfsalgotythm.py” -   
Przechowuje ona czasy oczekiwania przetworzonych wątków.

listOfProcessingTime– LISTA - zmienna globalna pliku „fcfsalgotythm.py” -   
Przechowuje ona czasy wykonania przetworzonych wątków.

## Działanie

Wykonaj algorytm FCFS dla danych

////// DO POPRAWY

# fcfsalgotythm.py

Plik, który zawiera implementacje algorytmu FCFS, na podstawie klasy „Thread”

## Import

Moduł importuje klasę „Thread” z pliku thread.py

## Zmienne globalne

**listOfWaitingTime** – Globalna LISTA   
Zawiera czasy oczekiwania przetworzonych wątków

**listOfProcessingTime** – Globalna LISTA

Zawiera czasy wykonania przetworzonych wątków

**listOfThread** – Globalna LISTA

Zawiera kolejkę wątków (klasy “Thread”) do wykonania

## Funkcje

#### fcfs(listOfThreadNotClass = [], \*args)

Funkcja, która zarządza całym algoytmem FCFS. Używa zmiennych globalnych (listOfWaitingTime**,** listOfProcessingTime**,** listOfThread).

Argumenty   
**listOfThreadNotClass – LISTA**  
Lista z danymi ile czasu mają trwać poszczególne procesy.

#### createArrayOfTheads(listOfThreadNotClass)

Funkcja odpowiedzialna za stworzenie listy obiektów klasy “Thread” na podstawie tablicy z danymi i włożenie jej do tablicy „listOfThread”.

Argumenty  
**listOfThreadNotClass – LISTA**  
Lista z danymi ile czasu mają trwać poszczególne procesy.

Zmienne   
**listOfThread** – Globalna LISTA

Zawiera kolejkę wątków (klasy “Thread”) do wykonania

#### fscsExecution()

Wykonanie algorytmu FCFS. Idzie wzdłuż kolejki, wykonuje aktualny proces, dodaje czas wykonania wątku do pozostałych wątków w kolejce.

Zmienne

**i** – INT – wartość domyślna = 0  
Zlicza numer kolejnego wątku. Zmienna potrzebna do dodania czasu wykonania aktualnego wątku, do pozostałych wątków w kolejce.

**listOfWaitingTime** – Globalna LISTA   
Zawiera czasy oczekiwania przetworzonych wątków

**listOfProcessingTime** – Globalna LISTA

Zawiera czasy wykonania przetworzonych wątków

**listOfThread** – Globalna LISTA

Zawiera kolejkę wątków (klasy “Thread”) do wykonania

# thread.py

Plik, który zawiera implementacje klasy „Thread”. Symuluje ona działanie wątku w systemie.

## Zmienne

**waitingTime** – INT – wartość domyślna = 0  
Zmienna przechowująca czas oczekiwania wątku.

**processingTime** – INT – wartość domyślna = 0

Zmienna przechowująca całkowity czas przetwarzania wątku przez program.

**allocationOfProcessorTime** – INT – wartość domyślna = 0

Zmienna przechowująca czas zajęcia procesora przez wątek.

**didFinished** – BOOL – wartość domyślna = FALSE

Zmienna, która zwraca nam czy proces został już wykonany czy też nie.

## Metody

\_\_init\_\_(self, allocationOfProcessorTime)  
Metoda inicjalizująca obiekt “Thread”, wraz z przypisaniem czasu który potrzebuje procesor na wykonanie tego wątku.

Argumenty   
**allocationOfProcessorTime** – Czas zajętości procesora przez wątek.

executeProcess(self)  
Metoda, która symuluje uruchomienie procesu. Jej zadaniem jest dodanie czasu przetwarzania wątku do całkowitego czasu przetwarzania danego wątku, a także ustawienie wątku jako wykonanego.

endingPreviousProcess (self, durationOfEndingProcess)Metoda, którą wywołujemy po wykonaniu procesu. Dodaje ona „allocationOfProcessorTime” aktualnego wątku do „waitingTime” i „processingTime” pozostałych wątków w kolejce.

Argumenty  
**durationOfEndingProcess** – czas zajęcia przez procesor aktualnie wykonanego wątku.

putResultsToTable(self, listOfWaitingTime = [], listOfProcessingTime = [], \*args)  
Metoda, która wkłada “waitingTime” oraz “processingTime” do tablicy wyników.

Argumenty  
**listOfWaitingTime** – LIST – Przechowuje czasy czekania wykonanych procesów.

**listOfProcessingTime** LIST – Przechowuje czasy wykonania przetworzonych procesów.