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 procesu. 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 process 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:
- procesy mają **jednakowy czas przyjścia** (różnice w czasie przyjścia są pomijalne),
- procesy nie **oczekują na udzielenie zasobów,**
- używamy tylko **jednego rdzenia** do obsłużenia procesów.

Sposób oceny algorytmów:

Kryteriami jakimi program ocenia zaimplementowane algorytmy są:

- **średni czas oczekiwania na przydzielenie procesora** (czas, który proces musiał odczekać w kolejce gotowych procesów)
- **średni czas cyklu przetwarzania** (czas od momentu przyjścia procesu, 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.

Użycie programu:

Jest to program konsolowy, dlatego wymagana jest podstawowa znajomość wiersza poleceń/terminala. Wymagana jest także podstawowa znajomość programowania.

Aby algorytm prawidłowo zadziałał musimy przesłać mu dane wejściowe (są ustawione domyślne dane).

W terminalu uruchamiamy główny plik: simulation.py
Program wykona się dla domyślnych danych.

Tworzenie własnych danych:

Aby utworzyć własne dane musimy uruchomić *sample/generate_random_values.py*
Po uruchomieniu program zapyta nas o wymiary:
- Ile procesów ma mieścić się w próbie [wiersze]
- Ile prób ma zostać wykonanych [kolumny]
- Do jakiego zakresu mają zostać rozlosowane liczby (1-?)

Następnie skrypt się wykona, a wynik zostanie zapisany w *data/random_values.txt*

Użycie nowych danych:

Aby użyć nowych danych w symulacji musimy jedynie podmienić zmienną „*path*” w pliku *sample/globals.py.* Algorytm automatycznie dostosuje się do nowych danych.

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 dokumentację programu

sample:/
Katalog zawiera wszystkie autorskie moduły używane w programie

simulation.py

Plik, który jest odpowiedzialny za wykonanie całego programu.

Import

Importuje całą zawartość modułu sample/fcfsalgotythm.py
Importuje całą zawartość modułu sample/sjfalgorythm.py
Importuje całą zawartość modułu sample/open_data.py
Importuje całą zawartość modułu sample/statistics.py
Importuje zmienną path z modułu sample/globals.py

Zmienne

matrix – Tablica 2D – Przechowuje ona pobrane dane z pliku. Każda komórka przedstawia czas potrzebny na wykonanie procesu przez procesor. Każdy wiersz przedstawia jedną próbę. Kolumny zaś ilość procesów w próbie.

listOfAllWaitingTime – LISTA – przechowuje ona wszystkie czasy oczekiwania procesu na przejęcie przez procesor.

listOfAllProcessingTime – LISTA – przechowuje ona wszystkie czasy przetwarzania procesu przez system (od przyjścia procesu do jego zakończenia).

averageValue – INT – przechowuje aktualnie liczoną wartość średnią.

medianValue – INT – przechowuje aktualnie liczoną wartość mediany.

Działanie

Plik odpowiedzialny jest za wywołanie funkcji, które wykonają:
1. Zapis danych z pliku do zmiennej matrix
2. Wykonanie algorytmu FCFS na danych ze zmiennej matrix
3. Wyświetlenie na ekranie wyników algorytmu FCFS
4. Wykonanie algorytmu SJF na danych ze zmiennej matrix
5. Wyświetlenie na ekranie wyników algorytmu SJF

globals.py

Plik przechowuje zmienne globalne używane w programie.

Zmienne

path – STRING – Zmienna przechowuje ścieżkę do pliku, z którego pobieramy dane.

process.py

Plik, który zawiera implementacje klasy „Process”. Symuluje ona działanie procesu w systemie.

Zmienne

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

processingTime – INT – wartość domyślna = 0
Zmienna przechowująca całkowity czas przetwarzania procesu przez program.

allocationOfProcessorTime – INT – wartość domyślna = 0
Zmienna przechowująca czas, który procesor potrzebuje na wykonanie procesu.

didFinished – BOOL – wartość domyślna = FALSE
Zmienna, która zwraca nam czy proces został już wykonany czy też nie.

Metody

__init__(self, allocationOfProcessorTime)

Metoda inicjalizująca obiekt “Process”, wraz z przypisaniem czasu który potrzebuje procesor na wykonanie tego procesu.

Argumenty

allocationOfProcessorTime – Czas zajętości procesora przez proces.

executeProcess(self)

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

endingPreviousProcess (self, durationOfEndingProcess)

Metoda, którą wywołujemy po wykonaniu procesu. Dodaje ona „allocationOfProcessorTime" aktualnego procesu do „waitingTime" i „processingTime" pozostałych procesu w kolejce.

Argumenty

durationOfEndingProcess – czas zajęcia przez procesor wykonanego procesu.

putResultsToTable(self)

Metoda, która zwraca nam czas oczekiwania danego procesu oraz czas przetwarzania danego procesu.

open_data.py

Plik odpowiedzialny jest za operacje na plikach w tym zapis danych z pliku do naszej tablicy 2D.

Funkcje

openFileAndPutIntoMatrix(path)

Funkcja jest odpowiedzialna za otworzenie pliku i włożenie danych do tablicy 2D.

Argumenty

path – STRING - przechowuje scieżkę do pliku

Zmienne

width – INT – Przechowuje rozmiar pliku w szerz (ile danych mieści się w jednej próbie)

height - INT – Przechowuje rozmiar pliku wzdłuż (ile jest prób)

matrix – Tablica 2D – To w niej zapisujemy wszystkie dane z pliku.

fileWithData – FILE – Zmienna odpowiadajaca otwartemu plikowi

actualAttempt – INT – Zlicza, którą linijkę aktualnie obsługujemy.

checkWidthAndHeightOfFile(path)

Funkcja odpowiedzialna za sprawdzenie szerokości i wysokości pliku i zwrócenie ich (szerokość – ile procesów jest w próbie. Wysokość – Ile jest prób)

Argumenty

path – STRING - przechowuje scieżkę do pliku

Zmienne

FileWithData – FILE – Zmienna odpowiadajaca otwartemu plikowi.

width – INT – Przechowuje szerokość pliku

height – INT – Przechowuje wysokość pliku

fcfsalgotythm.py

Plik, który zawiera implementacje algorytmu FCFS, na podstawie klasy „Process”. Wykorzystuje również funkcje pliku doalgorythm.py

Import

Importuj zmienna „path” z pliku sample/globals.py
Importuj klasę „Process” z pliku sample/proces.py
Importuj funkcję „checkWidthAndHeightOfFile” z pliku sample/open_data.py
Importuj wszystko z pliku sample/doalgorythm.py

Zmienne globalne

width – INT – Przechowuje szerokość pliku

height – INT – Przechowuje wysokość pliku

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

actualAttempt – INT – Zawiera informacje, która próba będzie aktualnie wykonywana.

Funkcje

fcfs(matrix)

Funkcja, która zarządza całym algorytmem FCFS. Używa zmiennych globalnych (listOfWaitingTime**,** listOfProcessingTime**,** listOfProcesses).

Argumenty

listOfProcessesNotClass – LISTA - Lista z danymi, ile czasu mają trwać poszczególne procesy.

Działanie

Dla każdego wiersza wykonaj fsfsProcess. Zwróć listOfAllWaitingTime oraz listOfAllProcessingTime.

fcfsProcess(listOfProcessNotClass = [], *args)

Odpowiedzialna jest za sam proces wykonania algorytmu FCFS dla jednej próby.

Agumenty:

listOfProcessNotClass – INT Tablica – Tablica przechowująca czasy zajętości procesora całej jednej próby (są to czyste dane z pliku nie klasa)

Operuje na zmiennych globalnych:

width – INT – Przechowuje szerokość pliku

height – INT – Przechowuje wysokość pliku

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

actualAttempt – INT – Zawiera informacje, która próba będzie aktualnie wykonywana.

Działanie:

1. Do listy procesów włóż obiekty klasy „Process”.
2. Wykonaj algorytm dla danych i zapisz je do tablic listOfAllWaitingTime oraz listOfAllProcessingTime.
3. Zwiększ wykonywa próbę o jeden.

clearAllLists()

Jej zadaniem jest wyczyszczenie wszystkich globalnych tablic.

Zmienne globalne:

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

Działanie:

1. Wstaw „0” do wszystkich pół listy listOfAllWaitingTime
2. Wstaw „0” do wszystkich pół listy listOfAllProcessingTime
3. Wyczyść tablicę listOfProcess

sjfalgorythm.py

Plik, który zawiera implementacje algorytmu SJF, na podstawie klasy „Process”. Wykorzystuje również funkcje pliku doalgorythm.py

Import

Importuj zmienna „path” z pliku sample/globals.py
Importuj klasę „Process” z pliku sample/proces.py
Importuj funkcję „checkWidthAndHeightOfFile” z pliku sample/open_data.py
Importuj wszystko z pliku sample/doalgorythm.py

Zmienne globalne

width – INT – Przechowuje szerokość pliku

height – INT – Przechowuje wysokość pliku

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

actualAttempt – INT – Zawiera informacje, która próba będzie aktualnie wykonywana.

Funkcje

sjf(matrix)

Funkcja, która zarządza całym algorytmem SJF. Używa zmiennych globalnych (listOfWaitingTime**,** listOfProcessingTime**,** listOfProcesses).

Argumenty

listOfProcessesNotClass – LISTA
Lista z danymi ile czasu mają trwać poszczególne procesy.

Działanie

Dla każdego wiersza wykonaj sjfProcess. Zwróć listOfAllWaitingTime oraz listOfAllProcessingTime.

SjfProcess(listOfProcessNotClass = [], *args)

Odpowiedzialna jest za sam proces wykonania algorytmu FCFS dla jednej próby.

Agumenty:

listOfProcessNotClass – INT Tablica – Tablica przechowująca czasy zajętości procesora całej jednej próby (są to czyste dane z pliku nie klasa)
Operuje na zmiennych globalnych:
width – INT – Przechowuje szerokość pliku

height – INT – Przechowuje wysokość pliku

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

actualAttempt – INT – Zawiera informacje, która próba będzie aktualnie wykonywana.

Działanie:

1. Posortuj tablice czasów zajętości procesora (nie klas)
2. Do listy procesów włóż obiekty klasy „Process”.
2. Wykonaj algorytm dla danych i zapisz je do tablic listOfAllWaitingTime oraz listOfAllProcessingTime.
3. Zwiększ wykonywa próbę o jeden.

clearAllLists()

Jej zadaniem jest wyczyszczenie wszystkich globalnych tablic.

Zmienne globalne:

listOfAllWaitingTime – Globalna LISTA - Zawiera czasy oczekiwania przetworzonych procesów

listOfAllProcessingTime – Globalna LISTA - Zawiera czasy wykonania przetworzonych procesów

listOfProcesses– Globalna LISTA - Zawiera kolejkę procesów (klasy “Process”) do wykonania

Działanie:

1. Wstaw „0” do wszystkich pół listy listOfAllWaitingTime
2. Wstaw „0” do wszystkich pół listy listOfAllProcessingTime
3. Wyczyść tablicę listOfProcess

doalgorythm.py

Plik, który wspiera działanie plików fcfsalgorythm.py oraz sjfalgorythm.py. Znajdują się tutaj wspólne funkcje używane przez algorytmy FCFS oraz SJF.

Import

Importuj klasę „Process” z pliku sample/proces.py

Funkcje

createArrayOfProcesses(listOfProcessNotClass)

Funkcja tworzy tablice z danych wejściowych, a następnie ją zwraca.

Argumenty

listOfProcessNotClass – LISTA - Przechowuje pobrane dane z pliku, które oznaczają czas zajętości danego procesu przez procesor.

Zmienne

listOfProcess – LISTA – Przechowuje obiekty klasy Process, które zostały utworzone na podstawie danych wejściowch.

Działanie

Dla każdej danej z tablicy listOfProcessNotClass, utwórz obiekt klasy Process, a następnie włóż go do tablicy wynikowej. Zwróć tablice listOfProcess.

algorythmExecution (listOfProcess, listOfAllWaitingTime, listOfAllProcessingTime, actualAttempt)

Główny algorytm odpowiedzialny za symulowanie wykonania procesów. Symuluje uruchomienie procesu.

Argumenty:

listOfProcess – LIST – Lista obiektów klasy Process.

listOfAllWaitingTime – LIST – Lista czasów oczekiwania poszczególnych procesów.

listOfAllProcessingTime – LIST – Lista czasów przetwarzania poszczególnych procesów.

actualAttempt – INT – Zlicza, która próba jest aktualnie wykonywana.

Zmienne:

i – INT – Indeks aktualnie obsługiwanego procesu

Działanie:

Przejdź przez całą listę procesów znajdujących się w próbie. Dla każdego z nich wykonaj proces. Zapisz wynik tego procesu do tablicy z wynikami. Przesuń indeks na kolejny proces. Dla pozostałych procesów dodaj czas wykonania zakończonego procesu. Pętle powtórz do końca próby. Zwróć listę czasów oczekiwania oraz czasów przetwarzania.

statistics.py

Plik, który jest odpowiedzialny za wykonanie wszelkich przekształceń statystycznych.

Import

Importuj moduł statystyki

Funkcje

averageOfArray(array)

Funkcja odpowiedzialna jest za sprawdzenie jaka tablica została przyjęta w argumencie, a następnie wywołanie funkcji, która obliczy średnią arytmetyczną.

Argumenty:

array – Lista 1D/Lista 2D – Zmienna przechowuje tablicę albo w wymiarze 1D albo w 2D.

Działanie:

Sprawdź jaka tablica została przyjęta w argumencie, wykonaj odpowiednie funkcje liczącą średnią, a następnie zwróć wynik.

averageForList2D(array)

Liczy wartość średnią dla tablicy 2D. Zwraca uśrednione dane w tablicy 1D.

Argumenty:

array – Lista 2D – Przechowuje dane do uśrednienia.

averageForList1D(array)

Liczy wartość średnią dla tablicy 1D. Zwraca uśrednioną wartość.

Argumenty:

array – Lista 2D – Przechowuje dane do uśrednienia.

medianOfArray(array)

Funkcja odpowiedzialna jest za sprawdzenie jaka tablica została przyjęta w argumencie, a następnie wywołanie funkcji, która obliczy mediane.

Argumenty:

array – Lista 1D/Lista 2D – Zmienna przechowuje tablicę albo w wymiarze 1D albo w 2D.

Działanie:

Sprawdź jaka tablica została przyjęta w argumencie, wykonaj odpowiednie funkcje liczącą mediane, a następnie zwróć wynik.

medianForList2D(array)

Liczy wartość średnią dla tablicy 2D. Zwraca mediany w tablicy 1D.

Argumenty:

array – Lista 2D – Przechowuje dane do obliczenia mediany.

medianForList1D(array)

Liczy wartość mediany dla tablicy 1D. Zwraca wartość mediany.

Argumenty:

array – Lista 2D – Przechowuje dane do uśrednienia.

arytythmeticStuff(listOfTime)

Wykonuje obliczenia mediany i średniej arytmetycznej dla tablicy wejściowej 2D tak aby danymi wyjściowymi były pojedyncze wartości średniej arytmetycznej i mediany z wszystkich prób.

Argumenty

listOfTime – TABLICA 2D – Tablica 2D, która przechowuje wyniki wszystkich prób.

Zmienne

listAverageOfEachAttemptTime – Lista 1D – przechowuje średni czas dla każdej z prób.

listMedianOfEachAttemptTime- Lista 1D – przechowuje mediane dla każdej z prób.

Działanie

Wykonaj liczenie średniej i mediany dla tablicy 2D zapisane do tablicy 1D, następnie policz i zwróć średnią i medianę z tablic 1D.

generate_random_values.py

Plik, dzięki któremu możemy wygenerować dane testowe o dowolny rozmiarze.

Import

Importuj moduł liczb losowych „random”
Importuj moduł „sys”

Zmienne

processesInAttempt – INT – Przechowuje wartość liczby kolumn. Ile procesów ma być w próbie.

manyOfAttempt – INT - Przechowuje wartość liczby wierzy. Ile ma być prób.

toNumber – INT – Przechowuje zawartość maksymalnej liczby, która może zostać wylosowana.

fromNumber – INT – Przechowuje zawartość, od jakiej liczby maja zostać losowane liczby. Najmniejsza wartość, którą może wylosować algorytm.

Działanie

Zapytaj użytkownika o wymiary danych oraz o zakres losowania. Otwórz plik, a następnie zapisz do niego losowe liczby. Plik będzie o wymiarach podanych przez użytkownika.

Dane wejściowe

Algorytm obsługuje dane wejściowe w konkretnym schemacie, który używany jest przez skrypt *generate_random_values.py.*

Każdy wiersz symbolizuje daną próbę.
Każda kolumna w wierszu symbolizuje czas przetwarzania procesu przez procesor.
Każdy czas przetwarzania procesu przez procesor przedzielony jest spacją.
Preferowane rozszerzenie pliku: „txt”
Przykładowy format pliku:

1 2 3 4 5 6 7 8 9
7 6 2 2 1 3 4 1 4
1 2 2 3 3 4 1 2 1

Plik oznacza:
3 próby po 9 procesów.