НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп‘ютерних систем

**Лабораторна робота №1  
*З дисципліни: «Операційні системи»***

«Планування процесів»

Студентка групи КВ-42

Брикалова Вікторія

Перевірив(ла):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2017

**Завдання**

1. Написати програму, що моделює процедуру обслуговування процесів, які знаходяться у черзі готових до виконання, за певним алгоритмом планування (перелік варіантів представлений нижче).

Вхідні дані (студент задає самостійно з урахуванням особливостей заданого алгоритму):

* послідовність процесів, які надходять до черги готових процесів;
* час надходження процесів до черги;
* очікуваний час виконання;
* пріоритет процесів та/або інщі додаткові дані, що необхідні для реалізації алгоритму планування згідно завдання.

Програма повинна обчислювати:

* час початку виконання для кожного процесу;
* час завершення для кожного процесу;
* час затримки (очікування) в черзі для кожного процесу;
* повний час виконання (з урахуванням затримки) для кожного процесу;
* середній час очікування процесів у черзі;
* середній повний час виконання процесів (з урахуванням затримки);
* інші результати, необхідні для оцінки алгоритму планування.

**2.** Зробити візуалізацію роботи програми у часі (на різних наборах вхідних даних, що ілюструють особливості алгоритму), а також представити кінцеві результати роботи на екрані у вигляді таблиці (як наприклад, у табл. 1.5), діаграми, графіку, малюнку тощо.

**Варіант 2:**

Розробити окремі програми для кожного з нижчезазначених алгоритмів та порівняти результати їх роботи:

1) алгоритм **FCFS**

2)алгоритм **SJF**

**Текст програми**

**Process.py**

class Process:

def \_\_init\_\_(self, executive\_time, appear\_time, name):

self.name = name;

self.executive\_time = executive\_time

self.wait\_time = 0

self.time\_to\_finish = executive\_time

self.appear\_time = appear\_time

self.start\_time = -1

self.finish\_time = -1

def wait(self):

self.wait\_time += 1

def execute(self):

self.time\_to\_finish -= 1

**FCFS.py**

import process as pr

import random as rnd

global PROCESS\_APPEAR\_PROBABILITY

global PROCESS\_LIST

global FINISHED\_PROCESS\_LIST

global CURRENT\_ITERATION

global CURRENT\_PROCESS

global FREE\_PROCESSOR

global TASK\_NUMBER

global FINISHED\_TASK\_NUMBER

global CURRENT\_TASK\_NUMBER

def planning():

global CURRENT\_ITERATION

global TASK\_NUMBER

global FINISHED\_TASK\_NUMBER

global FREE\_PROCESSOR

global CURRENT\_PROCESS

global CURRENT\_TASK\_NUMBER

global FINISHED\_PROCESS\_LIST

while True:

if TASK\_NUMBER == FINISHED\_TASK\_NUMBER:

break

print("-------------------------------------------------------------------------------")

print("Iteration ", CURRENT\_ITERATION)

print()

if CURRENT\_TASK\_NUMBER < TASK\_NUMBER:

if rnd.random() < PROCESS\_APPEAR\_PROBABILITY:

CURRENT\_TASK\_NUMBER += 1

PROCESS\_LIST.append(pr.Process(rnd.randint(3, 10), CURRENT\_ITERATION, CURRENT\_TASK\_NUMBER))

print("Task ", CURRENT\_TASK\_NUMBER, " appeared")

else:

print("Task didn't appear")

else:

print("All tasks have already appeared. There will not be other tasks")

if FREE\_PROCESSOR and PROCESS\_LIST == []:

print ("Processor is free. There are no task to compute")

if (FREE\_PROCESSOR and PROCESS\_LIST):

FREE\_PROCESSOR = False

CURRENT\_PROCESS = PROCESS\_LIST.pop(0)

CURRENT\_PROCESS.start\_time = CURRENT\_ITERATION

if CURRENT\_PROCESS != []:

CURRENT\_PROCESS.execute();

print("Task ", CURRENT\_PROCESS.name, " is computing. Time to finish ", CURRENT\_PROCESS.time\_to\_finish)

for element in PROCESS\_LIST:

element.wait();

if CURRENT\_PROCESS.time\_to\_finish == 0:

FREE\_PROCESSOR = True

FINISHED\_TASK\_NUMBER +=1

CURRENT\_PROCESS.finish\_time = CURRENT\_ITERATION

FINISHED\_PROCESS\_LIST.append(CURRENT\_PROCESS)

CURRENT\_PROCESS = []

CURRENT\_ITERATION += 1

print("-------------------------------------------------------------------------------")

def print\_table():

print("Number | appear time | execute time | start time | finish time | wait time | full time |")

middle\_wait\_time = 0

middle\_executive\_time = 0

for element in FINISHED\_PROCESS\_LIST:

middle\_wait\_time += element.wait\_time

middle\_executive\_time += element.wait\_time + element.executive\_time

print(" %5d | %5d | %5d | %5d | %5d | %5d | %5d |" % (element.name, element.appear\_time, element.executive\_time, element.start\_time, element.finish\_time, element.wait\_time, element.wait\_time+element.executive\_time))

middle\_wait\_time = middle\_wait\_time/len(FINISHED\_PROCESS\_LIST)

middle\_executive\_time = middle\_executive\_time/len(FINISHED\_PROCESS\_LIST)

print("Middle wait time for processes = ", middle\_wait\_time)

print("Middle executive time with waiting for processes = ", middle\_executive\_time)

if \_\_name\_\_ == '\_\_main\_\_':

PROCESS\_APPEAR\_PROBABILITY = 0.3

PROCESS\_LIST = []

CURRENT\_ITERATION = 0

FREE\_PROCESSOR = True

TASK\_NUMBER = 10

FINISHED\_TASK\_NUMBER = 0

CURRENT\_PROCESS = []

CURRENT\_TASK\_NUMBER = 0

FINISHED\_PROCESS\_LIST = []

planning();

print\_table()

**SJF.py**

import process as pr

import random as rnd

global PROCESS\_APPEAR\_PROBABILITY

global PROCESS\_LIST

global FINISHED\_PROCESS\_LIST

global CURRENT\_ITERATION

global CURRENT\_PROCESS

global FREE\_PROCESSOR

global TASK\_NUMBER

global FINISHED\_TASK\_NUMBER

global CURRENT\_TASK\_NUMBER

global BEST\_ELEMENT

global BEST\_ELEMENT\_INDEX

global INDEX

def planning():

global CURRENT\_ITERATION

global TASK\_NUMBER

global FINISHED\_TASK\_NUMBER

global FREE\_PROCESSOR

global CURRENT\_PROCESS

global CURRENT\_TASK\_NUMBER

global FINISHED\_PROCESS\_LIST

global BEST\_ELEMENT

global BEST\_ELEMENT\_INDEX

global INDEX

while True:

if TASK\_NUMBER == FINISHED\_TASK\_NUMBER:

break

print("-------------------------------------------------------------------------------")

print("Iteration ", CURRENT\_ITERATION)

print()

if CURRENT\_TASK\_NUMBER < TASK\_NUMBER:

if rnd.random() < PROCESS\_APPEAR\_PROBABILITY:

CURRENT\_TASK\_NUMBER += 1

PROCESS\_LIST.append(pr.Process(rnd.randint(3, 10), CURRENT\_ITERATION, CURRENT\_TASK\_NUMBER))

print("Task ", CURRENT\_TASK\_NUMBER, " appeared")

else:

print("Task didn't appear")

else:

print("All tasks have already appeared. There will not be other tasks")

if FREE\_PROCESSOR and PROCESS\_LIST == []:

print ("Processor is free. There are no task to compute")

if (FREE\_PROCESSOR and PROCESS\_LIST):

FREE\_PROCESSOR = False

BEST\_ELEMENT = PROCESS\_LIST[0]

BEST\_ELEMENT\_INDEX = 0

INDEX = -1

for element in PROCESS\_LIST:

INDEX +=1

if element.executive\_time < BEST\_ELEMENT.executive\_time:

BEST\_ELEMENT = element

BEST\_ELEMENT\_INDEX = INDEX

CURRENT\_PROCESS = PROCESS\_LIST.pop(BEST\_ELEMENT\_INDEX)

BEST\_ELEMENT = []

CURRENT\_PROCESS.start\_time = CURRENT\_ITERATION

if CURRENT\_PROCESS != []:

CURRENT\_PROCESS.execute();

print("Task ", CURRENT\_PROCESS.name, " is computing. Time to finish ", CURRENT\_PROCESS.time\_to\_finish)

for element in PROCESS\_LIST:

element.wait();

if CURRENT\_PROCESS.time\_to\_finish == 0:

FREE\_PROCESSOR = True

FINISHED\_TASK\_NUMBER +=1

CURRENT\_PROCESS.finish\_time = CURRENT\_ITERATION

FINISHED\_PROCESS\_LIST.append(CURRENT\_PROCESS)

CURRENT\_PROCESS = []

CURRENT\_ITERATION += 1

print("-------------------------------------------------------------------------------")

def print\_table():

print("Number | appear time | execute time | start time | finish time | wait time | full time |")

middle\_wait\_time = 0

middle\_executive\_time = 0

for element in FINISHED\_PROCESS\_LIST:

middle\_wait\_time += element.wait\_time

middle\_executive\_time += element.wait\_time + element.executive\_time

print(" %5d | %5d | %5d | %5d | %5d | %5d | %5d |" % (element.name, element.appear\_time, element.executive\_time, element.start\_time, element.finish\_time, element.wait\_time, element.wait\_time+element.executive\_time))

middle\_wait\_time = middle\_wait\_time/len(FINISHED\_PROCESS\_LIST)

middle\_executive\_time = middle\_executive\_time/len(FINISHED\_PROCESS\_LIST)

print("Middle wait time for processes = ", middle\_wait\_time)

print("Middle executive time with waiting for processes = ", middle\_executive\_time)

if \_\_name\_\_ == '\_\_main\_\_':

PROCESS\_APPEAR\_PROBABILITY = 0.3

PROCESS\_LIST = []

CURRENT\_ITERATION = 0

FREE\_PROCESSOR = True

TASK\_NUMBER = 10

FINISHED\_TASK\_NUMBER = 0

CURRENT\_PROCESS = []

CURRENT\_TASK\_NUMBER = 0

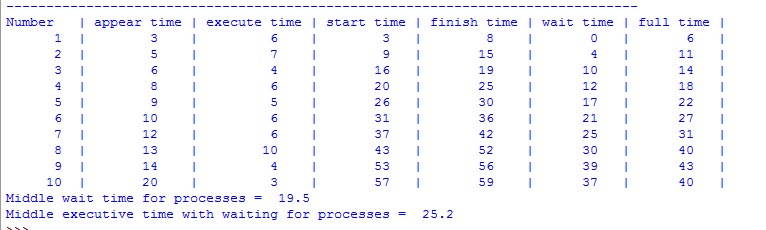
FINISHED\_PROCESS\_LIST = []

planning();

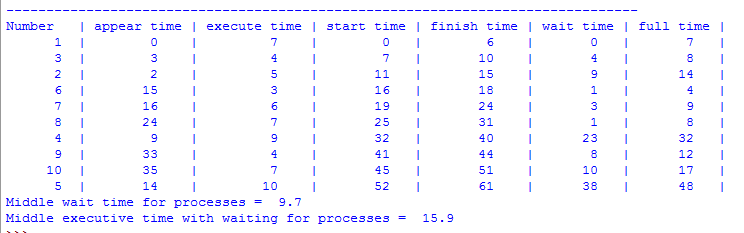
print\_table()

**Тести**

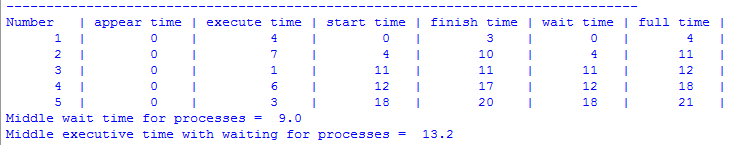
Тест алгоритму FCFS на випадковому наборі даних:



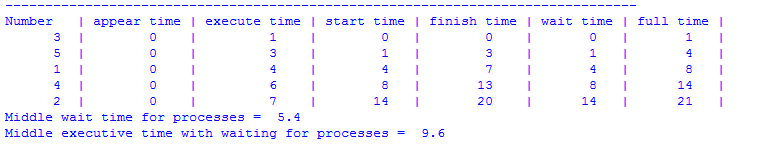
Тест алгоритму SJF на випадковому наборі даних:



Тест алгоритму FCFS на однаковому визначеному наборі даних:



Тест алгоритму SJF на однаковому визначеному наборі даних:



**Висновок**: Як видно на останніх результатах, алгоритми FCFS та SJF мають значну різницю в часі затримки виконання задач, а отже і в часі повного виконання задачі. Алгоритм SJF розподіляє виконання задач більш раціонально, зменшуючи час затримки за рахунок послідовного виконання задач від найкоротшої до найдовшої.