НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**Лабораторна робота №1  
з дисципліни «Операційні системи»**

**Варіант 9**

Виконав  
студент 4-го курсу  
групи КВ-41  
Курач Віктор

Київ – 2017

**Постановка задачі**

Написати програму, що моделює процедуру обслуговування процесів, які знаходяться у двох рівнях черг готових до виконання процесів:

1) черга інтерактивних процесів (алгоритм **RR)**,

2)черга фонових процесів (алгоритм **SRTF**).

Час розподіляється між чергою фонових та інтерактивних процесів: для інтерактивних процесів – 80 % і для фонових – 20 % часу. Якщо при досягненні ліміту часу виконується фоновий процес, він не переривається. Перехід на інтерактивний процес відбувається тільки після завершення поточного фонового процесу.

Зробити візуалізацію роботи програми у часі, а також представити кінцеві результати роботи на екрані у вигляді таблиці, діаграми, графіку, малюнку тощо.

**Приклад карти процесів**

Time IP1 IP2 IP3 IP4 IP5 BP1 BP2 BP3 BP4

0 \*

1 \*

2 \*

3 \*

4 \*

5 \*

6 \*

7 \*

8 \*

9 \*

10 \*

11 \*

12 \*

13 \*

14 \*

15 \*

16 \*

17 \*

18 \*

19 \*

20 \*

21 \*

22 \*

23 \*

24 \*

25 \*

26 \*

27 \*

28 \*

29 \*

30 \*

31 \*

32 \*

33 \*

34 \*

35 \*

36 \*

37 \*

38 \*

39 \*

40 \*

41 \*

42 \*

43 \*

44 \*

45 \*

46 \*

47 \*

48 \*

49 \*

50 \*

51 \*

52 \*

53 \*

54 \*

55 \*

56 \*

57 \*

58 \*

59 \*

60 \*

61 \*

62 \*

63 \*

64 \*

65 \*

66 \*

67 \*

68 \*

69 \*

70 \*

71 \*

72 \*

73 \*

74 \*

75 \*

76 \*

77 \*

78 \*

79 \*

80 \*

81 \*

82 \*

83 \*

84 \*

85 \*

86 \*

87 \*

88 \*

**Результати**

Process Income Length Start Estimate Delay Full time

BP1 0 8 8 36 20 28

BP2 2 5 8 13 0 5

BP3 4 10 8 48 30 40

BP4 5 7 8 25 10 17

IP1 0 12 0 77 65 77

IP2 1 15 1 86 70 85

IP3 3 8 3 68 57 65

IP4 6 14 6 89 69 83

IP5 7 10 7 83 66 76

Background processes average delay: 15

Background processes average full time: 22

Interactive processes average delay: 65

Interactive processes average full time: 77

**Лістинг програми**

BP1 = {"name": "BP1", "income": 0, "length": 8, "start": -1, "remain": 0, "estimate": -1}  
BP2 = {"name": "BP2", "income": 2, "length": 5, "start": -1, "remain": 0, "estimate": -1}  
BP3 = {"name": "BP3", "income": 4, "length": 10, "start": -1, "remain": 0, "estimate": -1}  
BP4 = {"name": "BP4", "income": 5, "length": 7, "start": -1, "remain": 0, "estimate": -1}  
IP1 = {"name": "IP1", "income": 0, "length": 12, "start": -1, "remain": 0, "estimate": -1}  
IP2 = {"name": "IP2", "income": 1, "length": 15, "start": -1, "remain": 0, "estimate": -1}  
IP3 = {"name": "IP3", "income": 3, "length": 8, "start": -1, "remain": 0, "estimate": -1}  
IP4 = {"name": "IP4", "income": 6, "length": 14, "start": -1, "remain": 0, "estimate": -1}  
IP5 = {"name": "IP5", "income": 7, "length": 10, "start": -1, "remain": 0, "estimate": -1}  
background\_processes = [BP1, BP2, BP3, BP4]  
interactive\_processes = [IP1, IP2, IP3, IP4, IP5]  
  
  
def print\_map\_header(output=None):  
 print("Time\t", end="", file=output)  
 for x in interactive\_processes:  
 print("%s\t" % x["name"], end="", file=output)  
 for x in background\_processes:  
 print("%s\t" % x["name"], end="", file=output)  
 print(file=output)  
  
  
def print\_results(output=None):  
 print("Process\tIncome\tLength\tStart\tEstimate\tDelay\tFull time\n", file=output)  
 average\_delay\_background = 0  
 average\_full\_time\_background = 0  
 for x in background\_processes:  
 print("%s\t%d\t%d\t%d\t%d\t\t%d\t%d" %  
 (x["name"], x["income"], x["length"], x["start"], x["estimate"],  
 x["estimate"] - x["start"] - x["length"], x["estimate"] - x["start"]),  
 file=output)  
 average\_delay\_background += x["estimate"] - x["start"] - x["length"]  
 average\_full\_time\_background += x["estimate"] - x["start"]  
 average\_delay\_background /= len(background\_processes)  
 average\_full\_time\_background /= len(background\_processes)  
 average\_delay\_interactive = 0  
 average\_full\_time\_interactive = 0  
 for x in interactive\_processes:  
 print("%s\t%d\t%d\t%d\t%d\t\t%d\t%d" %  
 (x["name"], x["income"], x["length"], x["start"], x["estimate"],  
 x["estimate"] - x["start"] - x["length"], x["estimate"] - x["start"]),  
 file=output)  
 average\_delay\_interactive += x["estimate"] - x["start"] - x["length"]  
 average\_full\_time\_interactive += x["estimate"] - x["start"]  
 average\_delay\_interactive /= len(interactive\_processes)  
 average\_full\_time\_interactive /= len(interactive\_processes)  
 print("\nBackground processes average delay: %d" % average\_delay\_background, file=output)  
 print("Background processes average full time: %d\n" % average\_full\_time\_background, file=output)  
 print("Interactive processes average delay: %d" % average\_delay\_interactive, file=output)  
 print("Interactive processes average full time: %d\n" % average\_full\_time\_interactive, file=output)  
  
  
def logging(process\_name, time, output=None):  
 if process\_name[0] == "I":  
 print("%s\t%s\*" % (str(time), "\t" \* (int(process\_name[2]) - 1)), file=output)  
 elif process\_name[0] == "B":  
 print("%s\t%s\*" % (str(time), "\t" \* (int(process\_name[2]) + 4)), file=output)  
  
  
def check\_for\_new\_process(processes, queue, time):  
 for x in processes:  
 if x["income"] <= time and x["start"] == -1:  
 queue.append(x)  
 x["start"] = time  
 return queue  
  
  
def remove\_estimated(queue, active\_process):  
 if active\_process["remain"] <= 0:  
 queue.remove(active\_process)  
 return None  
 return active\_process  
  
  
def choose\_active\_for\_srtf(queue, active\_process):  
 for x in queue:  
 if not active\_process or x["remain"] <= active\_process["remain"]:  
 active\_process = x  
 return active\_process  
  
  
def choose\_active\_for\_rr(queue, active\_process):  
 if not active\_process or active\_process == queue[len(queue) - 1]:  
 return queue[0]  
 return queue[queue.index(active\_process) + 1]  
  
  
def change\_current\_time(active\_process, time, n):  
 time += n  
 active\_process["remain"] -= n  
 if active\_process["remain"] == 0:  
 active\_process["estimate"] = time  
 return time, active\_process["remain"]  
  
  
def change\_time\_bounds(current\_time, time\_bound):  
 start = current\_time  
 while time\_bound % 10 != 8:  
 time\_bound += 1  
 if start == time\_bound:  
 time\_bound += 10  
 return start, time\_bound  
  
  
def srtf\_planner(processes, queue, start, time\_bound, output=None):  
 if not (start > 8 and not queue):  
 time = start  
 active\_process, old\_active\_process = None, None  
 while time < time\_bound or old\_active\_process == active\_process:  
 queue = check\_for\_new\_process(processes, queue, time)  
 active = choose\_active\_for\_srtf(queue, active\_process)  
 if time == time\_bound and old\_active\_process != active or not queue:  
 break  
 logging(active["name"], time, output=output)  
 time, active["remain"] = change\_current\_time(active, time, 1)  
 old\_active\_process = active  
 active\_process = remove\_estimated(queue, active)  
 start, time\_bound = change\_time\_bounds(time, time\_bound)  
 return queue, start, time\_bound  
  
  
def rr\_planner(processes, queue, start, time\_bound, active\_rr\_process, time\_quantum=1, output=None):  
 if not (start > 0 and not queue):  
 time = start  
 while time < time\_bound:  
 queue = check\_for\_new\_process(processes, queue, time)  
 active\_process = choose\_active\_for\_rr(queue, active\_rr\_process)  
 logging(active\_process["name"], time, output=output)  
 time, active\_process["remain"] = change\_current\_time(active\_process, time, time\_quantum)  
 active\_rr\_process = remove\_estimated(queue, active\_process)  
 if not queue:  
 break  
 start, bound = time\_bound, time\_bound + 2  
 return queue, start, bound, active\_rr\_process

def general\_planner(output=None):  
 time, start, bound = 0, 0, 8  
 back\_queue, inter\_queue, active\_rr = [], [], None  
 for x in background\_processes + interactive\_processes:  
 x["remain"] = x["length"]  
 while True:  
 inter\_queue, start, bound, active\_rr = \  
 rr\_planner(interactive\_processes, inter\_queue, start, bound, active\_rr, output=output)  
 back\_queue, start, bound = \  
 srtf\_planner(background\_processes, back\_queue, start, bound, output=output)  
 if not back\_queue and not inter\_queue:  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 f = open("map.txt", "w")  
 print\_map\_header(output=f)  
 general\_planner(output=f)  
 f.close()  
 g = open("result.txt", "w")  
 print\_results(g)  
 g.close()