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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**Лабораторна робота №3  
з дисципліни «Операційні системи»**

**Варіант 9**

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

Київ – 2017

**Постановка задачі**

Динамічні розділи без використання зовнішньої пам’яті. Кількість розділів менша, ніж кількість процесів. Процеси утворюють загальну чергу до розділів пам’яті. Використовується лінійний адресний простір. Розміри процесів задаються випадково.

**Приклад роботи програми**

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

1

0000 | aaaa aaaa aaaa aaaa

0010 | aaaa abbb bbbb bbbc

0020 | cccc cccc cccc cccc

0030 | cccc cccc cccc cccc

0040 | cccc cc

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

2

Start Length Process

0 21 P1

15 10 P2

1F 39 P3

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

4

Process id: P2

New process data: jjjj

Process P2 edited, 6 free space in the section

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

1

0000 | aaaa aaaa aaaa aaaa

0010 | aaaa ajjj j~~~ ~~~c

0020 | cccc cccc cccc cccc

0030 | cccc cccc cccc cccc

0040 | cccc cc

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

4

Process id: P2

New process data: kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk

Not enough space in the section: 44 more required

1. Show memory map

2. Show memory table

3. Launch new process

4. Edit process

5. Read from process

6. Write to process

0. Exit

0

**Лістинг**

processes = [  
 {"id": "P1", "data": "aaaaaaaaaaaaaaaaaaaaa"},  
 {"id": "P2", "data": "bbbbbbbbbb"},  
 {"id": "P3", "data": "ccccccccccccccccccccccccccccccccccccccc"}  
]  
memory = ""  
max\_mem\_size = 160  
memory\_table = [{"start": 0, "length": 0, "process": ""}]  
  
  
def init\_memory():  
 global memory\_table  
 global memory  
 memory\_table = []  
 p\_address = 0  
 for p in processes:  
 memory += p["data"]  
 memory\_table.append({"start": p\_address, "length": len(p["data"]), "process": p["id"]})  
 p\_address += len(p["data"])

def show\_memory\_map():  
 for i in range(0, len(memory)):  
 if i % 16 == 0:  
 print("\n%04X\t|" % i, end="")  
 if i % 4 == 0:  
 print(" ", end="")  
 if not memory[i]:  
 print("0", end="")  
 else:  
 print("%s" % memory[i], end="")  
 print("\n")  
  
  
def show\_memory\_table():  
 print("Start\tLength\tProcess")  
 for d in memory\_table:  
 print("%X\t%d\t%s" % (d["start"], d["length"], d["process"]))  
 print()  
  
  
def launch\_process(process\_id, process\_data):  
 global memory  
 global memory\_table  
 if process\_data == "" or process\_id == "" or len(memory) + len(process\_data) > max\_mem\_size:  
 return 1  
 p\_address = len(memory)  
 memory += process\_data  
 memory\_table.append({"start": p\_address, "length": len(process\_data), "process": process\_id})  
 return 0  
  
  
def launch\_new\_process\_interface():  
 print("Process id: ", end="")  
 process\_id = input()  
 print("Process data: ", end="")  
 process\_data = input()  
 res = launch\_process(process\_id, process\_data)  
 if res != 0:  
 print("Failed to launch process")  
 else:  
 print("Process %s launched" % process\_id)  
 print()  
  
  
def edit\_process(process\_id, new\_data):  
 global memory  
 global memory\_table  
 for p in memory\_table:  
 if p["process"] == process\_id:  
 p\_length = p["length"]  
 p\_start = p["start"]  
 if len(new\_data) > p\_length:  
 return 1, len(new\_data) - p\_length  
 memory = memory[0:p\_start] + new\_data + "~"\*(p\_length-len(new\_data)) + memory[p\_start + p\_length:]  
 return 0, p\_length - len(new\_data)  
 return 2, None

def edit\_process\_interface():  
 print("Process id: ", end="")  
 process\_id = input()  
 print("New process data: ", end="")  
 new\_data = input()  
 res, space = edit\_process(process\_id, new\_data)  
 if res == 0:  
 print("Process %s edited, %d free space in the section" % (process\_id, space))  
 elif res == 1:  
 print("Not enough space in the section: %d more required" % space)  
 else:  
 print("Failed to edit process %s" % process\_id)  
 print()  
  
  
def read\_from\_process(process\_id, address):  
 try:  
 for p in memory\_table:  
 if p["process"] == process\_id and 0 <= int(address) <= p["length"] - 1:  
 return memory[p["start"] + int(address)]  
 return ""  
 except ValueError:  
 return ""  
  
  
def read\_from\_process\_interface():  
 print("Process id: ", end="")  
 process\_id = input()  
 print("Address in process: ", end="")  
 address = input()  
 res = read\_from\_process(process\_id, address)  
 if res == "":  
 print("Failed to read from process %s" % process\_id)  
 else:  
 print(res)  
 print()  
  
  
def write\_to\_process(process\_id, address, value):  
 global memory  
 try:  
 for p in memory\_table:  
 if p["process"] == process\_id and 0 <= (int(address)) <= p["length"] - 1:  
 abs\_address = p["start"] + int(address)  
 memory = memory[0:abs\_address] + value[0] + memory[abs\_address+1:]  
 return 0  
 return 1  
 except ValueError:  
 return 1

def write\_to\_process\_interface():  
 print("Process id: ", end="")  
 process\_id = input()  
 print("Address in process: ", end="")  
 address = input()  
 print("Value: ", end="")  
 value = input()  
 res = write\_to\_process(process\_id, address, value)  
 if res != 0:  
 print("Failed to write to process %s" % process\_id)  
 else:  
 print("Wrote successfully")  
 print()  
  
  
def show\_menu():  
 print("1. Show memory map")  
 print("2. Show memory table")  
 print("3. Launch new process")  
 print("4. Edit process")  
 print("5. Read from process")  
 print("6. Write to process")  
 print("0. Exit")  
  
  
init\_memory()  
while True:  
 show\_menu()  
 ch = input()  
 while ch not in ["0", "1", "2", "3", "4", "5", "6"]:  
 ch = input()  
 if ch == "0":  
 break  
 elif ch == "1":  
 show\_memory\_map()  
 elif ch == "2":  
 show\_memory\_table()  
 elif ch == "3":  
 launch\_new\_process\_interface()  
 elif ch == "4":  
 edit\_process\_interface()  
 elif ch == "5":  
 read\_from\_process\_interface()  
 elif ch == "6":  
 write\_to\_process\_interface()