King Abdulaziz University

Faculty of Computing and Information Technology

Department of Computer Science

CPCS 361 Operation System

Project Report

Memory Management

Main and Virtual Memory

Group Number: 16

Table of Contents

Team Members:

Name	ID
Shaima Abdullah Bashammakh	1914892
Roaa Abdullah Alzhrani	2005863
Hanin Suleiman Alhaj	2010269
Alaa Emad Alhamzi	2010304

Introduction

In this project of CPCS 361, we will use java programming language to satisfy project's requirements. This project is about one of the most important operating system services which is memory management, and it contains two parts main memory and virtual memory.

General Information

Compiler Name and Version

We used NetBeans, and it uses the internal API of javac for compiling.

Hardware Configuration

Hardware configuration is the settings of the system resource that assigned to a specific device. Software and hardware input/output configurations can be defined from a single interactive interface through hardware configuration.

Hardware configuration has benefits, for example, easy maintenance, increase reliability and help to cost predict.

Operation System and Version

During the implementation of the project we used 2 versions of Window operation system, version 10 and 11.

Part 1: Main Memory

The Idea Description

Designing a Main Memory with size 1048576 then allocate processes based on the chosen algorithm such as First Fit, Best Fist, and Worst Fit. In addition, Release process from the memory and compact all free blocks as a single block then print a report illustrates the free and allocated blocks.

The Instructions for Running the Program

Open our project in NetBeans.

Run the code.

The Description of Experiments

In part 1 of the project, we create a main memory of size 1048576, which can allocate the processes using three algorithms. At the beginning the memory will be empty, then the user will enter the command RQ to allocate a process in the memory and determine the size of the process and the desired algorithm for example, (RQ p1 40000 F). If the user selects" First Fit" algorithm the memory will allocate the process to the first possible region. However, if the selected algorithm is the "Best Fit" the memory will allocate the process to the smallest possible region. Finally, if the selected algorithm is the "worst Fit" the memory will allocate the process in the largest possible region.

The second command is the release command, which removes a process from the memory and compact its region with the next free region. For example, (RL p4) this command will remove the process 4 from the memory.

Third command is Compact, which finds all free blocks in the memory and compact them as a single free block.

The Last command is STAT, which will report the blocks of memory that are allocated and unused.

Finally, if the memory is allocated completely it will display an error message “The Memory is Full!”.

The Output

First Run

```
./allocator 1048576
allocator> RQ p1 40000 F
allocator> RQ p2 50000 F
allocator> RQ p3 240000 B
allocator> RQ p4 8000 W
allocator> RQ p5 20000 W
allocator> RQ p6 500 B
allocator> RQ p7 800 F
allocator> RQ p8 9000 W
allocator> RQ p9 1000 F
allocator> RQ p10 700 B
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:337999] Process  P4
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Process  P8
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:1048575] Unused
```

Second Run

```
allocator> RL p4
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:337999] Unused
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Process  P8
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:1048575] Unused
```

Third Run

```
allocator> RL p8
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:337999] Unused
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Unused
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:1048575] Unused
```

Fourth Run

```
allocator> RQ p11 800 B
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:330799] Process  P11
Addresses [330800:337999] Unused
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Unused
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:1048575] Unused
allocator>
```

Fifth Run

```
allocator> RQ p12 1000 F
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:330799] Process  P11
Addresses [330800:331799] Process  P12
Addresses [331800:337999] Unused
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Unused
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:1048575] Unused
allocator>
```

Sixth Run

```
allocator> RQ p13 50000 W
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:330799] Process  P11
Addresses [330800:331799] Process  P12
Addresses [331800:337999] Unused
Addresses [338000:357999] Process  P5
Addresses [358000:358499] Process  P6
Addresses [358500:359299] Process  P7
Addresses [359300:368299] Unused
Addresses [368300:369299] Process  P9
Addresses [369300:369999] Process  P10
Addresses [370000:419999] Process  P13
Addresses [420000:1048575] Unused
allocator>
```

Seventh Run

```
allocator> c
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:330799] Process  P11
Addresses [330800:331799] Process  P12
Addresses [331800:351799] Process  P5
Addresses [351800:352299] Process  P6
Addresses [352300:353099] Process  P7
Addresses [353100:354099] Process  P9
Addresses [354100:354799] Process  P10
Addresses [354800:404799] Process  P13
Addresses [404800:1048575] Unused
```

Eighth Run

```
allocator> RQ p14 1000000 F
allocator> stat
Addresses [0:39999] Process  P1
Addresses [40000:89999] Process  P2
Addresses [90000:329999] Process  P3
Addresses [330000:330799] Process  P11
Addresses [330800:331799] Process  P12
Addresses [331800:351799] Process  P5
Addresses [351800:352299] Process  P6
Addresses [352300:353099] Process  P7
Addresses [353100:354099] Process  P9
Addresses [354100:354799] Process  P10
Addresses [354800:404799] Process  P13
Addresses [404800:1048575] Process  P14
allocator> RQ p15 1000 F
The Memory is Full!
allocator> x
BUILD SUCCESSFUL (total time: 7 minutes 26 seconds)
```

Part 2: Virtual Memory

The Idea Description

The idea of part two is depend on the translation from logical address to physical address using page table. Take a logical address, extract offset and page number and then calculate a physical address.

The Instructions for Running the Program

Open our project in NetBeans.

Make sure that the files we read from exist.

Run the code.

The Description of Experiments

There are three parts:

First part:

Read 100 address from addresses.txt file, extract page number and offset and then generate the frame number for each page number using page table. NOTE: each page number has a unique frame number

After that find and calculate the physical address, and then store the signed byte that cross ponds a specific logical address into a calculated physical address.

Second part:

Read 5 address from 100 addresses that read in part one, extract page number and offset and then find the frame number for each page number using an exists page table that generated in part one.

After that find and calculate the physical address, and then read the signed byte from a calculated physical address.

Three part:

Firstly, update 30 random page number in page table and change its value to -1.

Secondly, read 80 address from 100 addresses that read before. NOTE: the 30 page numbers do not have a frame number.

Thirdly, extract page number and offset and then find the frame using an exists page table that generated in part one. If the page number has a frame number increase the pageHit variable, and if the page number has no a frame number (frame number = -1) increase the pageFault variable.

The Output

First run

```
run:
----------------------------------PART 1-----------------------------------------
                            NOTHING to PRINT

----------------------------------PART 2-----------------------------------------
Logical Address         Page Number         Offset        Frame Number         Signed Byte Value         Same as Model Answer
32865                    128                 97             21                   0                          Yes
62493                    244                 29             120                  0                          Yes
64815                    253                 47             184                  75                         Yes
64747                    252                 235            59                   58                         Yes
61006                    238                 78             175                  59                         Yes

----------------------------------PART 3-----------------------------------------
969  40185  28781  48128  38929  6091  12218  19358
58882  42932  7591  6727  3067  17665  50552  32315
22760  61006  49213  22501  54894  28964  64357  38929
54388  44770  48128  49294  12218  22760  49213  59162
24462  49294  17866  36529  38336  8620  48399  51476
49294  36529  34561  38929  58982  692  2315  58882
14557  59162  51476  62493  41118  5129  49213  30705
50552  34561  41118  6727  5003  40178  59162  40185
14557  44954  19358  44954  58882  46919  32315  64747
32541  42632  8620  19358  64815  54894  10392  46919

Page Fault : 32
Page Hit : 48
BUILD SUCCESSFUL (total time: 0 seconds)
```

Second run

```
run:
----------------------------------PART 1-----------------------------------------
                            NOTHING to PRINT

----------------------------------PART 2-----------------------------------------
Logical Address         Page Number         Offset        Frame Number         Signed Byte Value         Same as Model Answer
64454                    251                 198            150                  62                         Yes
6727                     26                  71             75                   -111                       Yes
12218                    47                  186            1                    11                         Yes
22760                    88                  232            211                  0                          Yes
64747                    252                 235            133                  58                         Yes

----------------------------------PART 3-----------------------------------------
44954  42252  27444  59162  36922  38336  43765  34561
42632  17665  22514  54894  32865  56657  62615  60645
57982  30198  18145  36374  30198  969  42252  6308
41118  18633  27966  61006  3784  5129  6308  64815
3884  20259  54894  38336  22760  3784  60086  46919
6727  34561  15913  3884  33405  7591  64243  40178
36529  50227  43121  27966  5129  53683  28718  22514
58982  58982  58584  43121  46919  62615  28718  55041
48128  32315  49294  22760  58982  30705  41118  59162
28781  21395  21311  33405  58982  32315  40891  62493

Page Fault : 25
Page Hit : 55
BUILD SUCCESSFUL (total time: 0 seconds)
```

Third run

```
run:
-----------------------------------PART 1-----------------------------------------
                            NOTHING to PRINT

-----------------------------------PART 2-----------------------------------------
Logical Address        Page Number        Offset        Frame Number        Signed Byte Value        Same as Model Answer
30198                   117                246            51                   29                       Yes
38929                   152                17             73                   0                        Yes
14557                   56                 221            182                  0                        Yes
22760                   88                 232            6                    0                        Yes
27966                   109                62             228                  27                       Yes

-----------------------------------PART 3-----------------------------------------
58892  49294  24462  42632  2315  63258  5129  21311
32865  42632  51476  57857  28781  58882  46919  22760
54894  64454  36529  22501  28781  8620  27444  55041
28964  38336  62615  31649  28718  6091  36922  27444
969  59162  64815  57857  42632  59240  7591  40178
58892  12218  10842  5003  40891  40178  62493  62615
58882  42252  21395  64357  61006  28781  64747  27444
42632  57857  58982  42252  27966  59240  57982  57857
6727  41118  15757  56657  33405  42252  57857  41118
21395  40178  15679  42632  30705  44770  27444  17665

Page Fault : 34
Page Hit : 46
BUILD SUCCESSFUL (total time: 0 seconds)
```