Credit Hours System

CMPN303

Cairo University

Faculty of Engineering

Operating Systems Project

Presented by:

Mahmoud Ezzat ELMaghraby – 1170532

Khaled Ali Shaalan – 1170395

Ahmed Mohamed ELKhatib – 1170100

Abdullah Ayman ELSharnouby – 1170126

Student Names	ID	Phase 1	Phase 2	Phase 3
Mahmoud Ezzat ELMaghraby	1170532	25%	50%	
Khaled Ali Shaalan	1170395	25%		50%
Ahmed Mohamed ELKhatib	1170100	25%	50%	
Abdullah Ayman ELSharnouby	1170126	25%		50%

Process generator

reading the processes file

```
while(fgets(line, sizeof line, file)!=NULL) {
    reads data from file and add it to array of struct processData
}
```

Ask the user for the desired scheduling algorithm then create a message queue to send processes to scheduler

Create scheduler and clock with the parameters

```
for(int i = 0; i<2; i++) {
    pid = fork();
    if(pid == 0 && i==0) { // child one
            char* const par[]={(char*)0};
            execv("./clk.out",par);
    }
    else if(pid == 0 && i==1) { // child two
        char algo[2];
        char quan[4];
        char numberProcesses[4];
        sprintf(algo, "%d", choice);
        sprintf(quan, "%d", quantum);
        sprintf(numberProcesses, "%d", numProcesses);
        char *argv[]={"./scheduler.out", algo , quan, numberProcesses, NULL};
        execv(argv[0],argv);
    }
}
```

sends a process when the clock reaches its arrival time

```
int i=numProcesses;
while(i > 0){
    x = getClk();
    if(data[numProcesses - i].arrivaltime <= x){
        send the process using the queue
        i--;
    }
}
```

Headers

Defining struct processData that contains all information of a process

```
struct processData
{
    int id;
    int memsize;
    int pid;
    int arrivaltime;
    int priority;
    int runningtime;
    int waitingtime;
    int finishtime;
    int starttime;
    int turnaroundtime;
    int responsetime;
    float weightedturnaroundtime;
    int sleeptime;
    struct memUnit* memAssigned;
};
```

Defining struct memUnit that contains all information of a memory unit

Each block in the memory is called a memory unit, at the beginning of the project there is one unit called mainMemory with size 1024 bytes, if the unit is divided it will have two children with the same size each one having half the size of the parent

```
struct memUnit
{
    bool isAllocated;
    int processId;
    int size;
    int smallestUnit;
    int beginning;
    struct memUnit* childOne;
    struct memUnit* childTwo;
};
```

Memory Allocation

```
struct memUnit* memoryAllocation(int processId, int processSize, struct memUnit*
mem)
{
if isAllocated
    return NULL

else if it does not have children
    if size of process is smaller than unit and bigger than half the size
        allocate and return the unit
    else if size of process smaller than or equal half the unit size
        generate two units as childs each with half the size
        return the recursion on first child
    else
        return NUll
else if it has children
    it checks which child has the smallest unit under it and returns recursion on
it if     NULL is returned it returns recursion on the second child
}
```

Memory Deallocation

It returns 2 to tell the parent that the process was found and to try to merge the 2 children

```
int memoryDeallocation(int processId, int processSize, struct memUnit* mem)
{
if isAllocated
    checks if the id of the allocated process is the one we are searching for
    returns 2 if found else returns 0
else if no children or process size is bigger than unit
    return 0
do  recursion on first child
    if 1 is returned do not merge and return
    if 2 check other child if it can merge
    if 0 do recursion on second child
}
```

Priority Queue

The structure used is priority queue using linked list

```
struct node
{
    struct processData data;
    int p;
    struct node* next;
};
```

3 queues are created

- readyQueue: contains processes that are allocated in memory and ready to run
- waitingQueue: contains processes that are received but did not find space in memory
- receivedQueue: processes that have same arrival time are sorted here before adding in ready queue to allocate the processes with highest priority first

runningProcess: contains the running process to be added to ready queue if preempted or deleted if finished

Contains enqueue and dequeue functions to add or remove processes from queues

Scheduler

Main

Creates shared memory to share with the process the time waited so the process can calculate its remaining time

```
while(1)
{
receive from message queue all processes sent at the current time and sorts them
in received queue according to priority and algorithm chosen

Then dequeue every process in received queue, for each process if space was found
in memory allocate it in memory and enqueue in ready queue else enqueue in
waiting queue

Then check algorithm chosen to fork, stop, continue a process

A process is not forked until it is going to run for the first time, struct
processData contains int pid which is initially set to -1, when the process is
forked the pid is assigned to this integer, so we check it to see if we forked
before or not
}
```

handler

This function is called when SIGUSR1 is sent from process when it is terminating, it calculates the turnaround, weighted turnaround and waiting time of the process that terminated

Then it deallocates the process from memory and searches if waiting queue contains any processes to allocate the most suitable one according to arrival time and size

```
int val = memoryDeallocation(runningProcess->data.id, runningProcess-
>data.memsize, mainMemory);
if(waitingQueue != NULL){
        get the largest unit in the memory
        then dequeue the first process found that is equal or smaller than this
unit
        Allocate the process in the memory and enqueue in ready queue
        }
}
```

if all processes terminated CPU utilization, average weighted turnaround and average waiting is calculated

Process

Keeps calculating the remaining time

```
while(remainingtime > 0)
{
    remainingtime = runningtime - (getClk() - starttime - atoi((char
*)sharedWaitedTime));
}
```

when remaining time reaches zero SIGUSR1 is sent to scheduler

```
kill(getppid(), SIGUSR1);
```

Test Cases

The file with name "processes.txt" in Code folder is the one that is considered as input

Phase 1

Testcase 1

```
#id arrival runtime priority
1   1   14  6
2   10  4   3
3   11  14  1
4   21  5   2
```

Highest priority first

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 started arr 1 total 14 remain 14 wait 0
At time 15 process 1 finished arr 1 total 14 remain 0 wait 0 TA 14 WTA 1.00
At time 15 process 3 started arr 11 total 14 remain 14 wait 4
At time 29 process 3 finished arr 11 total 14 remain 0 wait 4 TA 18 WTA 1.29
At time 29 process 4 started arr 21 total 5 remain 5 wait 8
At time 34 process 4 finished arr 21 total 5 remain 0 wait 8 TA 13 WTA 2.60
At time 34 process 2 started arr 10 total 4 remain 4 wait 24
At time 38 process 2 finished arr 10 total 4 remain 0 wait 24 TA 28 WTA 7.00
```

```
CPU utilization = 100%
Avg WTA = 2.97
Avg Waiting = 9.00
Std WTA = 2.40
```

Shortest remaining time next

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 started arr 1 total 14 remain 14 wait 0
At time 10 process 1 stopped arr 1 total 14 remain 5 wait 0
At time 10 process 2 started arr 10 total 4 remain 4 wait 0
At time 14 process 2 finished arr 10 total 4 remain 0 wait 0 TA 4 WTA 1.00
At time 14 process 1 resumed arr 1 total 14 remain 5 wait 4
At time 19 process 1 finished arr 1 total 14 remain 0 wait 4 TA 18 WTA 1.29
At time 19 process 3 started arr 11 total 14 remain 14 wait 8
At time 21 process 3 stopped arr 11 total 14 remain 12 wait 8
At time 21 process 4 started arr 21 total 5 remain 5 wait 0
At time 26 process 4 finished arr 21 total 5 remain 0 wait 0 TA 5 WTA 1.00
At time 26 process 3 resumed arr 11 total 14 remain 12 wait 13
At time 38 process 3 finished arr 11 total 14 remain 0 wait 13 TA 27 WTA 1.93
```

```
CPU utilization = 100%
Avg WTA = 1.30
Avg Waiting = 4.25
Std WTA = 0.38
```

Round robin

quantum = 5

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 started arr 1 total 14 remain 14 wait 0
At time 11 process 1 stopped arr 1 total 14 remain 4 wait 0
At time 11 process 2 started arr 10 total 4 remain 4 wait 1
At time 15 process 2 finished arr 10 total 4 remain 0 wait 1 TA 5 WTA 1.25
At time 15 process 1 resumed arr 1 total 14 remain 4 wait 4
At time 19 process 1 finished arr 1 total 14 remain 0 wait 4 TA 18 WTA 1.29
At time 19 process 3 started arr 11 total 14 remain 14 wait 8
At time 24 process 3 stopped arr 11 total 14 remain 9 wait 8
At time 24 process 4 started arr 21 total 5 remain 5 wait 3
At time 29 process 4 finished arr 21 total 5 remain 0 wait 3 TA 8 WTA 1.60
At time 29 process 3 resumed arr 11 total 14 remain 9 wait 13
At time 38 process 3 finished arr 11 total 14 remain 0 wait 13 TA 27 WTA 1.93
```

```
CPU utilization = 100%
Avg WTA = 1.52
Avg Waiting = 5.25
Std WTA = 0.27
```

Testcase 2

```
#id arrival runtime priority
1   11  18  3
2   14  27  5
3   16  29  10
4   24  8   10
5   31  24  6
6   34  27  3
7   38  13  1
```

Highest priority first

```
#At time x process y state arr w total z remain y wait k
At time 11 process 1 started arr 11 total 18 remain 18 wait 0
At time 29 process 1 finished arr 11 total 18 remain 0 wait 0 TA 18 WTA 1.00
At time 29 process 2 started arr 14 total 27 remain 27 wait 15
At time 56 process 2 finished arr 14 total 27 remain 0 wait 15 TA 42 WTA 1.56
At time 56 process 7 started arr 38 total 13 remain 13 wait 18
At time 69 process 7 finished arr 38 total 13 remain 0 wait 18 TA 31 WTA 2.38
At time 69 process 6 started arr 34 total 27 remain 27 wait 35
At time 96 process 6 finished arr 34 total 27 remain 0 wait 35 TA 62 WTA 2.30
At time 96 process 5 started arr 31 total 24 remain 24 wait 65
At time 120 process 5 finished arr 31 total 24 remain 0 wait 65 TA 89 WTA 3.71
At time 120 process 3 started arr 16 total 29 remain 29 wait 104
At time 149 process 3 finished arr 16 total 29 remain 0 wait 104 TA 133 WTA 4.59
At time 149 process 4 started arr 24 total 8 remain 8 wait 125
At time 157 process 4 finished arr 24 total 8 remain 0 wait 125 TA 133 WTA 16.62
```

utilization is not 100% because first arrival time is 11

```
CPU utilization = 93%
Avg WTA = 4.59
Avg Waiting = 51.71
Std WTA = 5.04
```

Shortest remaining time next

```
#At time x process y state arr w total z remain y wait k
At time 11 process 1 started arr 11 total 18 remain 18 wait 0
At time 29 process 1 finished arr 11 total 18 remain 0 wait 0 TA 18 WTA 1.00
At time 29 process 4 started arr 24 total 8 remain 8 wait 5
At time 37 process 4 finished arr 24 total 8 remain 0 wait 5 TA 13 WTA 1.62
At time 37 process 5 started arr 31 total 24 remain 24 wait 6
At time 38 process 5 stopped arr 31 total 24 remain 23 wait 6
At time 38 process 7 started arr 38 total 13 remain 13 wait 0
At time 51 process 7 finished arr 38 total 13 remain 0 wait 0 TA 13 WTA 1.00
At time 51 process 5 resumed arr 31 total 24 remain 23 wait 19
At time 74 process 5 finished arr 31 total 24 remain 0 wait 19 TA 43 WTA 1.79
At time 74 process 2 started arr 14 total 27 remain 27 wait 60
At time 101 process 2 finished arr 14 total 27 remain 0 wait 60 TA 87 WTA 3.22
At time 101 process 6 started arr 34 total 27 remain 27 wait 67
At time 128 process 6 finished arr 34 total 27 remain 0 wait 67 TA 94 WTA 3.48
At time 128 process 3 started arr 16 total 29 remain 29 wait 112
At time 157 process 3 finished arr 16 total 29 remain 0 wait 112 TA 141 WTA 4.86
```

utilization is not 100% because first arrival time is 11

```
CPU utilization = 92%
Avg WTA = 2.43
Avg Waiting = 37.57
Std WTA = 1.35
```

Round robin

quantum = 15

```
#At time x process y state arr w total z remain y wait k
At time 11 process 1 started arr 11 total 18 remain 18 wait 0
At time 26 process 1 stopped arr 11 total 18 remain 3 wait 0
At time 26 process 2 started arr 14 total 27 remain 27 wait 12
At time 41 process 2 stopped arr 14 total 27 remain 12 wait 12
At time 41 process 3 started arr 16 total 29 remain 29 wait 25
At time 56 process 3 stopped arr 16 total 29 remain 14 wait 25
At time 56 process 4 started arr 24 total 8 remain 8 wait 32
At time 64 process 4 finished arr 24 total 8 remain 0 wait 32 TA 40 WTA 5.00
At time 64 process 1 resumed arr 11 total 18 remain 3 wait 38
At time 67 process 1 finished arr 11 total 18 remain 0 wait 38 TA 56 WTA 3.11
At time 67 process 5 started arr 31 total 24 remain 24 wait 36
At time 82 process 5 stopped arr 31 total 24 remain 9 wait 36
At time 82 process 6 started arr 34 total 27 remain 27 wait 48
At time 97 process 6 stopped arr 34 total 27 remain 12 wait 48
At time 97 process 7 started arr 38 total 13 remain 13 wait 59
At time 110 process 7 finished arr 38 total 13 remain 0 wait 59 TA 72 WTA 5.54
At time 110 process 2 resumed arr 14 total 27 remain 12 wait 81
At time 122 process 2 finished arr 14 total 27 remain 0 wait 81 TA 108 WTA 4.00
At time 122 process 3 resumed arr 16 total 29 remain 14 wait 91
At time 136 process 3 finished arr 16 total 29 remain 0 wait 91 TA 120 WTA 4.14
At time 136 process 5 resumed arr 31 total 24 remain 9 wait 90
At time 145 process 5 finished arr 31 total 24 remain 0 wait 90 TA 114 WTA 4.75
At time 145 process 6 resumed arr 34 total 27 remain 12 wait 96
At time 157 process 6 finished arr 34 total 27 remain 0 wait 96 TA 123 WTA 4.56
```

utilization is not 100% because first arrival time is 11

```
CPU utilization = 92%
Avg WTA = 4.44
Avg Waiting = 69.57
Std WTA = 0.73
```

Phase 2

Testcase 1

```
#id arrival runtime priority memsize
1   11  8   9   256
2   18  20  5   170
3   24  13  3   128
4   29  26  4   60
5   30  22  4   248
6   32  18  10  64
7   40  2   1   200
```

Highest priority first

```
#At time x allocated y bytes for process z from i to j
At time 11 allocated 256 bytes for process 1 from 0 to 255
At time 18 allocated 170 bytes for process 2 from 256 to 511
At time 19 freed 256 bytes from process 1 from 0 to 255
At time 24 allocated 128 bytes for process 3 from 0 to 127
At time 29 allocated 60 bytes for process 4 from 128 to 191
At time 30 allocated 248 bytes for process 5 from 512 to 767
At time 32 allocated 64 bytes for process 6 from 192 to 255
At time 39 freed 170 bytes from process 2 from 256 to 511
At time 40 allocated 200 bytes for process 7 from 256 to 511
At time 52 freed 128 bytes from process 3 from 0 to 127
At time 54 freed 200 bytes from process 7 from 256 to 511
At time 80 freed 60 bytes from process 4 from 128 to 191
At time 102 freed 248 bytes from process 5 from 512 to 767
At time 120 freed 64 bytes from process 6 from 192 to 255
```

Shortest remaining time next

```
#At time x allocated y bytes for process z from i to j
At time 11 allocated 256 bytes for process 1 from 0 to 255
At time 18 allocated 170 bytes for process 2 from 256 to 511
At time 19 freed 256 bytes from process 1 from 0 to 255
At time 24 allocated 128 bytes for process 3 from 0 to 127
At time 29 allocated 60 bytes for process 4 from 128 to 191
At time 30 allocated 248 bytes for process 5 from 512 to 767
At time 32 allocated 64 bytes for process 6 from 192 to 255
At time 37 freed 128 bytes from process 3 from 0 to 127
At time 40 allocated 200 bytes for process 7 from 768 to 1023
At time 42 freed 200 bytes from process 7 from 768 to 1023
At time 54 freed 170 bytes from process 2 from 256 to 511
At time 72 freed 64 bytes from process 6 from 192 to 255
At time 94 freed 248 bytes from process 5 from 512 to 767
At time 120 freed 60 bytes from process 4 from 128 to 191
```

Round robin

quantum = 10

```
#At time x allocated y bytes for process z from i to j
At time 11 allocated 256 bytes for process 1 from 0 to 255
At time 18 allocated 170 bytes for process 2 from 256 to 511
At time 19 freed 256 bytes from process 1 from 0 to 255
At time 24 allocated 128 bytes for process 3 from 0 to 127
At time 29 allocated 60 bytes for process 4 from 128 to 191
At time 30 allocated 248 bytes for process 5 from 512 to 767
At time 32 allocated 64 bytes for process 6 from 192 to 255
At time 40 allocated 200 bytes for process 7 from 768 to 1023
At time 49 freed 170 bytes from process 2 from 256 to 511
At time 82 freed 128 bytes from process 3 from 0 to 127
At time 84 freed 200 bytes from process 7 from 768 to 1023
At time 112 freed 64 bytes from process 6 from 192 to 255
At time 118 freed 60 bytes from process 4 from 128 to 191
At time 120 freed 248 bytes from process 5 from 512 to 767
```

Testcase 2

```
#id arrival runtime priority memsize
1   2   18  2   128
2   6   16  5   64
3   12  2   5   256
4   22  21  1   120
5   23  4   2   200
6   29  21  6   200
7   33  15  4   128
8   39  1   9   256
```

Highest priority first

```
#At time x allocated y bytes for process z from i to j
At time 2 allocated 128 bytes for process 1 from 0 to 127
At time 6 allocated 64 bytes for process 2 from 128 to 191
At time 12 allocated 256 bytes for process 3 from 256 to 511
At time 20 freed 128 bytes from process 1 from 0 to 127
At time 22 allocated 120 bytes for process 4 from 0 to 127
At time 23 allocated 200 bytes for process 5 from 512 to 767
At time 29 allocated 200 bytes for process 6 from 768 to 1023
At time 36 freed 64 bytes from process 2 from 128 to 191
At time 36 allocated 128 bytes for process 7 from 128 to 255
At time 57 freed 120 bytes from process 4 from 0 to 127
At time 61 freed 200 bytes from process 5 from 512 to 767
At time 61 allocated 256 bytes for process 8 from 512 to 767
At time 76 freed 128 bytes from process 7 from 128 to 255
At time 78 freed 256 bytes from process 3 from 256 to 511
At time 99 freed 200 bytes from process 6 from 768 to 1023
At time 100 freed 256 bytes from process 8 from 512 to 767
```

Shortest remaining time next

```
#At time x allocated y bytes for process z from i to j
At time 2 allocated 128 bytes for process 1 from 0 to 127
At time 6 allocated 64 bytes for process 2 from 128 to 191
At time 12 allocated 256 bytes for process 3 from 256 to 511
At time 14 freed 256 bytes from process 3 from 256 to 511
At time 22 freed 128 bytes from process 1 from 0 to 127
At time 22 allocated 120 bytes for process 4 from 0 to 127
At time 23 allocated 200 bytes for process 5 from 256 to 511
At time 27 freed 200 bytes from process 5 from 256 to 511
At time 29 allocated 200 bytes for process 6 from 256 to 511
At time 33 allocated 128 bytes for process 7 from 512 to 639
At time 39 allocated 256 bytes for process 8 from 768 to 1023
At time 40 freed 256 bytes from process 8 from 768 to 1023
At time 43 freed 64 bytes from process 2 from 128 to 191
At time 58 freed 128 bytes from process 7 from 512 to 639
At time 79 freed 120 bytes from process 4 from 0 to 127
At time 100 freed 200 bytes from process 6 from 256 to 511
```

Round robin

quantum = 10

```
#At time x allocated y bytes for process z from i to j
At time 2 allocated 128 bytes for process 1 from 0 to 127
At time 6 allocated 64 bytes for process 2 from 128 to 191
At time 12 allocated 256 bytes for process 3 from 256 to 511
At time 22 allocated 120 bytes for process 4 from 512 to 639
At time 23 allocated 200 bytes for process 5 from 768 to 1023
At time 30 freed 128 bytes from process 1 from 0 to 127
At time 32 freed 256 bytes from process 3 from 256 to 511
At time 32 allocated 200 bytes for process 6 from 256 to 511
At time 33 allocated 128 bytes for process 7 from 0 to 127
At time 38 freed 64 bytes from process 2 from 128 to 191
At time 52 freed 200 bytes from process 5 from 768 to 1023
At time 52 allocated 256 bytes for process 8 from 768 to 1023
At time 83 freed 256 bytes from process 8 from 768 to 1023
At time 98 freed 128 bytes from process 7 from 0 to 127
At time 99 freed 120 bytes from process 4 from 512 to 639
At time 100 freed 200 bytes from process 6 from 256 to 511
```