Jegyzőkönyv Operációs rendszerek

1. szorgalmi feladat

Timkó András
HZS05V
ge-BGI

Szorgalmi feladat

Írjon egy programot, ami 3 fajta ütemezési feladatot megold(FCFS, SJF, RR).
A processzek számát a felhasználó adja meg, futási időben.
Valamint a hozzá szükséges adatokat.
Írja ki az ütemezési táblákat.
Rajzoljon Gantt diagrammot.

(*Megjegyzés: bekér prioritási adattagot is, de azt ki lehet kapcsolni/kommentelni)*

```
root@elegemvan:/home/elegemvan/Desktop/HZS05V0sGyak2/HZS05V_0321# g++ hzs05v7.cpp -o hzs05v7
root@elegemvan:/home/elegemvan/Desktop/HZS05V0sGyak2/HZS05V_0321# ./hzs05v7
 Megadni kivant processek szamossaga: 4

 Adja meg az Erkezesi idot, a CPU idejet valamint a prioritast a process:[1] -nek
 Erkezesi ido:  0

 Cpu ido:       15

 Process prioritasa:
1

 Adja meg az Erkezesi idot, a CPU idejet valamint a prioritast a process:[2] -nek
 Erkezesi ido:  8

 Cpu ido:       7

 Process prioritasa:    1

 Adja meg az Erkezesi idot, a CPU idejet valamint a prioritast a process:[3] -nek
 Erkezesi ido:  12

 Cpu ido:       26

 Process prioritasa:    1

 Adja meg az Erkezesi idot, a CPU idejet valamint a prioritast a process:[4] -nek
 Erkezesi ido:  20

 Cpu ido:       10

 Process prioritasa:    1
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
| Process sz. | Erkezesi ido | CPU ido    |  Befejezesi ido | Atfordulasi ido | Varakoz ido |   Valasz ido  |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
|     1       |      0      |     15     |       15       |       15       |      0      |      0      |
|     2       |      8      |     7      |       22       |       14       |      7      |      7      |
|     3       |      12     |     26     |       48       |       36       |      10     |      10     |
|     4       |      20     |     10     |       58       |       38       |      28     |      28     |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+

Total Befejezesi ido :- 143
Atlagos Befejezesi ido :- 35.75

Total Atfordulasi ido :- 103
Atlagos Atfordulasi ido :- 25.75

Total Varakozasi ido :- 45
Atlagos Varakozasi ido :- 11.25

Total Valasz ido :- 45
Atlagos Valasz ido :- 11.25


Gantt abra(IS = idle statusz) :-

+-----------------------------+-------------+-----------------------------------------------------+-------------------+
|             P1              |      P2     |                         P3                          |         P4        |
+-----------------------------+-------------+-----------------------------------------------------+-------------------+
0                             15            22                                                    48                  58
```

First Come First Served

```
 -SJF-
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
| Process sz. | Erkezesi ido | CPU ido    |  Befejezesi ido | Atfordulasi ido | Varakoz ido |   Valasz ido  |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
|     1       |      0      |     15     |       15       |       15       |      0      |      0      |
|     2       |      8      |     7      |       22       |       14       |      7      |      7      |
|     3       |      12     |     26     |       58       |       46       |      20     |      20     |
|     4       |      20     |     10     |       32       |       12       |      2      |      2      |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+

Total Befejezesi ido :- 127
Atlagos Befejezesi ido :- 31.75

Total Atfordulasi ido :- 87
Atlagos Atfordulasi ido :- 21.75

Total Varakozasi ido :- 29
Atlagos Varakozasi ido :- 7.25

Total Valasz ido :- 29
Atlagos Valasz ido :- 7.25


Gantt abra(IS = idle statusz) :-

+-----------------------------+-------------+-------------+-----------------------------------------------------+
|             P1              |      P2     |      P4     |                         P3                          |
+-----------------------------+-------------+-------------+-----------------------------------------------------+
0                             15            22            32                                                    58
```

Sortest Job First

```
 -RR-
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
| Process sz. | Erkezesi ido | CPU ido    |  Befejezesi ido | Atfordulasi ido | Varakoz ido |   Valasz ido  |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+
|     1       |      0      |     15     |       27       |       27       |      12     |      0      |
|     2       |      8      |     7      |       25       |       17       |      10     |      0      |
|     3       |      12     |     26     |       58       |       46       |      20     |      6      |
|     4       |      20     |     10     |       42       |       22       |      12     |      7      |
+-------------+-------------+------------+----------------+----------------+-------------+-------------+

Total Befejezesi ido :- 152
Atlagos Befejezesi ido :- 38

Total Atfordulasi ido :- 112
Atlagos Atfordulasi ido :- 28

Total Varakozasi ido :- 54
Atlagos Varakozasi ido :- 13.5

Total Valasz ido :- 13
Atlagos Valasz ido :- 3.25

 MS:- 5


Gantt abra(IS = idle statusz) :-

+---------------+-------------+-------------+-------+-------+-------------+-------------+-------------+-----------------------------+
|       P1      |      P2     |      P1     |      P3     | P2  |  P1   |      P4     |      P3     |      P4     |             P3              |
+---------------+-------------+-------------+-------------+-----+-------+-------------+-------------+-------------+-----------------------------+
0               8             13            18            23    25      27            32            37            42                            58

root@elegemvan:/home/elegemvan/Desktop/HZS05V0sGyak2/HZS05V_0321#
```

Round Robin

(Githhubon hzs05v7.cpp néven megtalálható)

Nyers program kód:

// C++ kulonbozo utemezesi algoritmusokhoz

```
#include <cstdlib>
#include <iostream>
#include <queue>
using namespace std;

class process {
public:
   pid_t p_no = 0;
   time_t start_AT = 0, AT = 0,
         BT_left = 0, BT = 0, temp_BT = 0,
         CT = 0, TAT = 0, WT = 0, RT = 0;
   int priority = 0;

   // Befejezesi ido
   void set_CT(time_t time)
   {
      CT = time;
      set_TAT();
      set_WT();
   }

   // Atfordulasi ido (turn around time)
   void set_TAT()
   {
      TAT = CT - start_AT;
   }

   // Varakozasi ido
   void set_WT()
   {
      WT = TAT - BT;
   }
   // Mivel push()-nal frissul az Erkezesi ido, ezert azt kezelni kell
   void P_set()
   {
      start_AT = AT;
      BT_left = BT;
   }
   // Valasz ido
   void set_RT(time_t time)
   {
      RT = time - start_AT;
   }

   // '<' Operator tulterhelese
   // mivel az erkezesi idonek nagyobb a prioritasa
   // priority_queue elsonek poppolja a nagyobb erteket
   // ezert ki kell csereni a '<' -t '>' hogy a legkisebbet poppolja
   friend bool operator<(const process& a, const process& b)
   {
      return a.AT > b.AT;
   }
};


process pop_index(priority_queue<process>* main_queue, int index)
{
   priority_queue<process> rm_index;
   int i;
   process p;
```

```
   switch (index) {
   case 0:
      p = (*main_queue).top();
      (*main_queue).pop();
      break;
   default:
      for (i = 0; i < index; i++) {
         rm_index.push((*main_queue).top());
         (*main_queue).pop();
      }
      p = (*main_queue).top();
      (*main_queue).pop();
      while (!(*main_queue).empty()) {
         rm_index.push((*main_queue).top());
         (*main_queue).pop();
      }
      (*main_queue) = rm_index;
      break;
   }
   return p;
}

// Legkisebb CPU ido
time_t min_BT(priority_queue<process> main_queue, time_t clock)
{
   time_t min = 0;
   while (!main_queue.empty() && main_queue.top().AT <= clock) {
      if (min == 0 || min > main_queue.top().BT_left)
         min = main_queue.top().BT_left;
      main_queue.pop();
   }
   return min;
}
int min_BT_index(priority_queue<process> main_queue, time_t limit)
{
   int index, i = 0;
   time_t min = 0;
   while (!main_queue.empty() && main_queue.top().AT <= limit) {
      if (min == 0 || main_queue.top().BT_left < min) {
         min = main_queue.top().BT_left;
         index = i;
      }
      main_queue.pop();
      i++;
   }
   return index;
}


// RR algoritmus
priority_queue<process> RR_run(priority_queue<process> ready_queue,
                        time_t Time_Slice,
                        queue<process>* gantt)
{
   priority_queue<process> completion_queue;
   process p;
   time_t clock = 0;

  // Amig a bekert processek nem fogynak el ( a ready_queue-ban )
   while (!ready_queue.empty()) {
      while (clock < ready_queue.top().AT) {
         p.temp_BT++;
         clock++;
      }
      if (p.temp_BT > 0) {
         p.p_no = -1;
```

```
      p.CT = clock;
      (*gantt).push(p);
    }
    p = ready_queue.top();
    ready_queue.pop();

    if (p.AT == p.start_AT)
      p.set_RT(clock);

                                                    //Time_Slice az ms, ahol darabolja a hosszu processeket

    while (p.BT_left > 0 && (p.temp_BT < Time_Slice
                || ready_queue.empty()
                || clock < ready_queue.top().AT)) {
      p.temp_BT++;
      p.BT_left--;
      clock++;
    }

    if (p.BT_left == 0) {
      p.AT = p.start_AT;
      p.set_CT(clock);
      (*gantt).push(p);
      p.temp_BT = 0;
      completion_queue.push(p);
    }
    else {
      p.AT = clock;
      p.CT = clock;
      (*gantt).push(p);
      p.temp_BT = 0;
      ready_queue.push(p);
    }
  }

  return completion_queue;
}

// FCFS algoritmus
priority_queue<process> FCFS_run(priority_queue<process> ready_queue,
                queue<process>* gantt)
{
  priority_queue<process> completion_queue;
  process p;
  time_t clock = 0;

  // Amig a bekert processek nem fogynak el ( a ready_queue-ban )
  while (!ready_queue.empty()) {

    // Amig az eltelt ido kevesebb az erkezesi idonel
    while (clock < ready_queue.top().AT) {
      p.temp_BT++;
      clock++;
    }
    if (p.temp_BT > 0) {
      p.p_no = -1;
      p.CT = clock;
      (*gantt).push(p);
    }
    p = ready_queue.top();
    ready_queue.pop();
    p.set_RT(clock);
    while (p.BT_left > 0) {
      p.temp_BT++;
      p.BT_left--;
      clock++;
```

```
      }
      p.set_CT(clock);

      // Gantt diagram frissitese
      (*gantt).push(p);
      p.temp_BT = 0;

      // Befejezesi ido frissitese
      completion_queue.push(p);
    }
    return completion_queue;
}

// SJF algoritmus
priority_queue<process> SJF_P_run(priority_queue<process> ready_queue,
                          queue<process>* gantt)
{
    priority_queue<process> completion_queue;
    process p;
    time_t clock = 0;

                                                                  // Amig a bekert processek nem fogynak el ( a ready_queue-ban )
    while (!ready_queue.empty()) {
      while (clock < ready_queue.top().AT) {
        p.temp_BT++;
        clock++;
      }
      if (p.temp_BT > 0) {
        p.p_no = -1;
        p.CT = clock;
        (*gantt).push(p);
      }
      p = pop_index(&ready_queue, min_BT_index(ready_queue, clock));
      if (p.AT == p.start_AT)
        p.set_RT(clock);
      while (p.BT_left > 0 && (ready_queue.empty()
                          || clock < ready_queue.top().AT
                          || p.BT_left <= min_BT(ready_queue, clock))) {
        p.BT_left--;
        p.temp_BT++;
        clock++;
      }
      if (p.BT_left == 0) {
        p.AT = p.start_AT;
        p.set_CT(clock);
        (*gantt).push(p);
        p.temp_BT = 0;
        completion_queue.push(p);
      }
      else {
        p.AT = clock;
        p.CT = clock;
        (*gantt).push(p);
        p.temp_BT = 0;
        ready_queue.push(p);
      }
    }

    return completion_queue;
}


// Processek bekerese
priority_queue<process> set_process_data()
{
    priority_queue<process> ready_queue;
```

```
    process temp;

    int NOP, i;
    printf(" Megadni kivant proccessek szamossaga: ");
    scanf("%d", &NOP);
    for(i=0; i<NOP; i++)
    {
    printf("\n Adja meg az Erkezesi idot, a CPU idejet valamint a prioritast a process:[%d] -nek \n", i+1);
    printf(" Erkezesi ido: \t");
    scanf("%d", &temp.AT);
    printf(" \n Cpu ido: \t");
    scanf("%d", &temp.BT);
    printf(" \n Process prioritasa: \t");
    scanf("%d", &temp.priority);
    temp.p_no = i + 1;
    temp.P_set();
    ready_queue.push(temp);

    }

    return ready_queue;
}
// Atlagok szamitasa:
// Osszes varakozasi ido
double get_total_WT(priority_queue<process> processes)
{
    double total = 0;
    while (!processes.empty()) {
        total += processes.top().WT;
        processes.pop();
    }
    return total;
}

// Osszes atfordulasi ido
double get_total_TAT(priority_queue<process> processes)
{
    double total = 0;
    while (!processes.empty()) {
        total += processes.top().TAT;
        processes.pop();
    }
    return total;
}

// Osszes befejezesi ido
double get_total_CT(priority_queue<process> processes)
{
    double total = 0;
    while (!processes.empty()) {
        total += processes.top().CT;
        processes.pop();
    }
    return total;
}

// Osszes valasz ido
double get_total_RT(priority_queue<process> processes)
{
    double total = 0;
    while (!processes.empty()) {
        total += processes.top().RT;
        processes.pop();
    }
    return total;
}
```

```
//FCFS tabla kirajzolasa

void disp(priority_queue<process> main_queue, bool high)
{
   int i = 0, temp, size = main_queue.size();
   priority_queue<process> tempq = main_queue;
   double temp1;
   cout << "+-------------+--------------";
   cout << "+------------+-----------------";
   cout << "+-----------------+--------------+---------------+";
   if (high == true)
      cout << "----------+" << endl;
   else
      cout << endl;
   cout << "| Process sz. | Erkezesi ido ";
   cout << "| CPU ido     | Befejezesi ido ";
   cout << "| Atfordulasi ido | Varakozasi ido | Valasz ido |";
   if (high == true)
      cout << " Priority |" << endl;
   else
      cout << endl;
   cout << "+-------------+--------------";
   cout << "+------------+-----------------";
   cout << "+-----------------+--------------+---------------+";
   if (high == true)
      cout << "----------+" << endl;
   else
      cout << endl;
   while (!main_queue.empty()) {
      temp = to_string(main_queue.top().p_no).length();
      cout << '|' << string(6 - temp / 2 - temp % 2, ' ')
         << main_queue.top().p_no << string(7 - temp / 2, ' ');
      temp = to_string(main_queue.top().start_AT).length();
      cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
         << main_queue.top().start_AT << string(7 - temp / 2, ' ');
      temp = to_string(main_queue.top().BT).length();
      cout << '|' << string(6 - temp / 2 - temp % 2, ' ')
         << main_queue.top().BT << string(6 - temp / 2, ' ');
      temp = to_string(main_queue.top().CT).length();
      cout << '|' << string(8 - temp / 2 - temp % 2, ' ')
         << main_queue.top().CT << string(9 - temp / 2, ' ');
      temp = to_string(main_queue.top().TAT).length();
      cout << '|' << string(8 - temp / 2 - temp % 2, ' ')
         << main_queue.top().TAT << string(9 - temp / 2, ' ');
      temp = to_string(main_queue.top().WT).length();
      cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
         << main_queue.top().WT << string(7 - temp / 2, ' ');
      temp = to_string(main_queue.top().RT).length();
      cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
         << main_queue.top().RT << string(8 - temp / 2, ' ');
      if (high == true) {
         temp = to_string(main_queue.top().priority).length();
         cout << '|' << string(5 - temp / 2 - temp % 2, ' ')
            << main_queue.top().priority << string(5 - temp / 2, ' ');
      }
      cout << "|\n";
      main_queue.pop();
   }
   cout << "+-------------+--------------";
   cout << "+------------+-----------------";
   cout << "+-----------------+--------------+---------------+";
   if (high == true)
      cout << "----------+";
   cout << endl;
   temp1 = get_total_CT(tempq);
```

```
    cout << "\nTotal Befejezesi ido :- " << temp1
         << endl;
    cout << "Atlagos Befejezesi ido :- " << temp1 / size
         << endl;
    temp1 = get_total_TAT(tempq);
    cout << "\nTotal Atfordulasi ido :- " << temp1
         << endl;
    cout << "Atlagos Atfordulasi ido :- " << temp1 / size
         << endl;
    temp1 = get_total_WT(tempq);
    cout << "\nTotal Varakozasi ido :- " << temp1
         << endl;
    cout << "Atlagos Varakozasi ido :- " << temp1 / size
         << endl;
    temp1 = get_total_RT(tempq);
    cout << "\nTotal Valasz ido :- " << temp1
         << endl;
    cout << "Atlagos Valasz ido :- " << temp1 / size
         << endl;
}

//Gantt rajzolasa
void disp_gantt_chart(queue<process> gantt)
{
    int temp, prev = 0;
    queue<process> spaces = gantt;
    cout << "\n\nGantt abra(IS = idle statusz) :- \n\n+";

    // 1. sor
    while (!spaces.empty()) {
        cout << string(to_string(spaces.front().p_no).length()
                       + (spaces.front().p_no != -1)
                       + 2 * spaces.front().temp_BT,
                   '-')
             << "+";
        spaces.pop();
    }
    cout << "\n|";
    spaces = gantt;

    // 2. sor
    while (!spaces.empty()) {
        cout << string(spaces.front().temp_BT, ' ');
        if (spaces.front().p_no == -1)
            cout << "IS" << string(spaces.front().temp_BT, ' ') << '|';
        else
            cout << "P" << spaces.front().p_no
                 << string(spaces.front().temp_BT, ' ') << '|';
        spaces.pop();
    }
    spaces = gantt;
    cout << "\n+";

    while (!spaces.empty()) {
        cout << (string(to_string(spaces.front().p_no).length()
                        + (spaces.front().p_no != -1)
                        + 2 * spaces.front().temp_BT,
                    '-'))
             << "+";
        spaces.pop();
    }
    spaces = gantt;
    cout << "\n0";
    //3. sor
    while (!spaces.empty()) {
        temp = to_string(spaces.front().CT).length();
```

```
        cout << (string(to_string(spaces.front().p_no).length()
                        + (spaces.front().p_no != -1)
                        + 2 * spaces.front().temp_BT - temp / 2 - prev,
                    ' '))
             << spaces.front().CT;
        prev = temp / 2 - temp % 2 == 0;
        spaces.pop();
    }
    cout << "\n\n";
}


int main()
{
    // Tablak inicializasa
    priority_queue<process> ready_queue;
    priority_queue<process> completion_queue, completion_queue2, completion_queue3;
    queue<process> gantt, gantt2, gantt3;




    // Adatok bekerdezese
    ready_queue = set_process_data();


    completion_queue = FCFS_run(ready_queue, &gantt);

    completion_queue2 = SJF_P_run(ready_queue, &gantt2);


    int ms = 5;

    completion_queue3 = RR_run(ready_queue,ms ,&gantt3);

    // Tabla rajzolas fcfs
    disp(completion_queue, false);

    // Gantt rajzolas fcfs
    disp_gantt_chart(gantt);

     cout << "\n -SJF-  "
          << endl;

    // Tabla rajzolas sjf
    disp(completion_queue2, false);

    // Gantt rajzolas sjf
    disp_gantt_chart(gantt2);

     cout << "\n -RR-  "
          << endl;

    // Tabla rajzolas RR
    disp(completion_queue3, false);

    cout << "\n MS:- " << ms << endl;

    // Gantt rajzolas RR
    disp_gantt_chart(gantt3);

    return 0;
}
```