**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине «Операционные системы»

на тему: «Командный интерфейс *ОС Windows*»

Выполнил: студент гр. ИТП-11

Лисицин Н.С.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2022

**Цель**: разработать программу, осуществляющую моделирование режима работы с разделение времени.

**Задание**:

В соответствии с вариантом (выдается преподавателем), разработать программу на языке *С#* (или другом), осуществляющую моделирование работы планировщика процессов.

**Ход работы**

В соответствии с вариантом (выдается преподавателем), разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов. Моделирование режима разделения времени выполняется в соответствии с заданием (таблица 2.1). На экран выводится следующая информация:

1. Номер текущего кванта времени процессора;

2. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления;

3. Таблица планирования процессов с отображением текущего состояния процессов.

После запуска, программа должна диалоговом режиме, ввести информацию процессах – имя, длительность, приоритет, время появления. Для алгоритмов *RR* число квантов времени. Выполнение должно производиться в пошаговом режиме (по нажатию на кнопку). По окончанию работы процесса на экране должно выводится сообщение о его завершении «Процесс такой-то завершен».

В соответствии с вариантом необходимо сделать Алгоритм *FCFS* вытесняющий.

На рисунке 1 представлен вариант задания



Рисунок 1 – Вариант задания

Сначала происходит ввод данных через диалоговые окна и выводиться таблица с полученными данными.

Вывод таблицы с исходными данными представлен на рисунке 2.

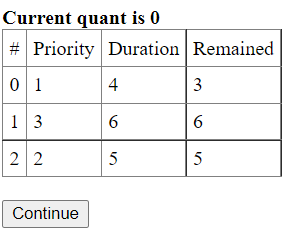


Рисунок 2 – Таблица с введенными данными

После по нажатии кнопки *continue* происходит формирование таблицы с алгоритмом *FCFS* вытесняющий*.*

Результат выполнения скрипта представлен на рисунке 2.

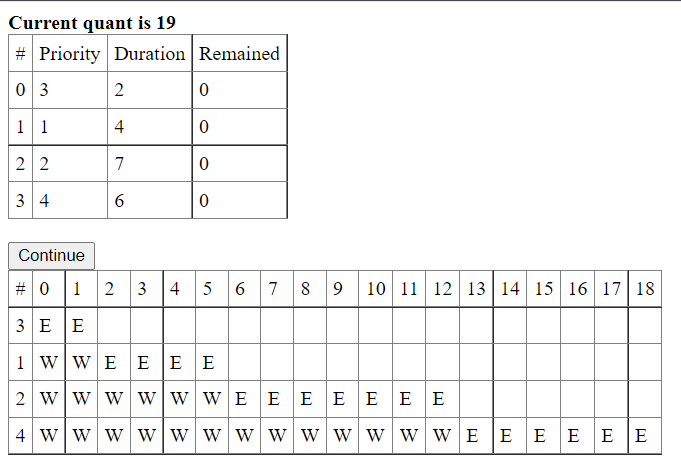


Рисунок 2 – Результат выполнения алгоритма *FCFS* вытесняющего

Текст программы, выполненной на языке *JavaScript*, представлен в листинге **А**. Выполняется процесс, который первее прибыл. Если процесс выполняется, его состояние — “*E*”; ожидает своей очереди — “*W*”, исполнен — “*C*”. Согласно программе, нам необходимо отображать, когда процесс начался, когда закончился, и когда ожидает очереди.

**Вывод:** в результате выполнения лабораторной работы, была разработана программа, которая осуществляет моделирование режима работы с разделением времени.

**ПРИЛОЖЕНИЕ А**

**Текст программы**

**App.js:**

"*use* *strict*";

*let* *div* = *document*.*getElementById*("*tableDiv*");

*let* *btn* = *document*.*getElementById*("*btn*");

*let* *process* = [];

*let* *processN* = +*prompt*("*Enter* *amount* *of* *processess*: ");

*for* (*let* *i* = 0; *i* < *processN*; *i*++)

{

*let* *priority* = +*prompt*("*Enter* *priority* *of* *process*: ");

*let* *duration* = +*prompt*("*Enter* *duration* *of* *process*: ");

*process*.*push*({*priority*, *duration*, *remained*: *duration*, *appearance*: "", *isCompleted*: *false*, *states*: []});

}

*let* *table* = [];

*let* *statesN* = [];

*let* *wAmount* = 0, *eAmount* = 0;

*let* *isPrEx* = *false*, *f* = *false*;

*let* *numOfExPr* = 0;

*let* *q* = 0;

*btn*.*addEventListener*("*click*", (*e*) =>

{

*if* (*process*.*some*(*el* => !*el*.*isCompleted*))

    {

*f* = *false*;

*if* (!*isPrEx*)

        {

*let* *processQ* = *findQprocess*(*process*);

*isPrEx* = *true*;

*process* = *putStates*(*process*, *processQ*, *numOfExPr*);

*process*[*numOfExPr*].*remained*--;

        }

*else* *if* (*isPrEx* && *process*[*numOfExPr*].*remained*)

        {

*let* *processQ* = *findQprocess*(*process*);

*process* = *putStates*(*process*, *processQ*, *numOfExPr*);

*process*[*numOfExPr*].*remained*--;

        }

*else* *if* (*isPrEx* && !*process*[*numOfExPr*].*remained*)

        {

*process*[*numOfExPr*].*isCompleted* = *true*;

*console*.*log*(`*Process* *with* *priority* ${*process*[*numOfExPr*].*priority*} *is* *over*!\*n*`);

*isPrEx* = *false*;

*numOfExPr*++;

*f* = *true*;

        }

*div*.*innerHTML* = "";

*showQuantTable*(*div*, *process*, *q*);

*if* (!*f*) {

*if* (*isPrEx*)

*console*.*log*(`*Process* *with* *priority* ${*process*[*numOfExPr*].*priority*} *is* *executing*!`);

*statesN*.*push*(*q*++);

        }

    } *else*

*showTable*();

});

*btn*.*click*();

*function* *findQprocess*(*process*) {

*let* *arr* = [];

*process*.*filter*((*pr*, *i*) => !*pr*.*isCompleted* && *arr*.*push*({*pr*, *i*}));

*return* *arr*;

}

*function* *putStates*(*process*, *processQ*, *numOfExPr*)

{

*process*.*forEach*((*el*, *i*) =>

    {

*if* (*i* != *numOfExPr* && !*processQ*.*some*(*e* => *e*.*i* == *i*))

*el*.*states*.*push*(" ");

*else* *if* (*i* != *numOfExPr* && *processQ*.*some*(*e* => *e*.*i* != *i*))

*el*.*states*.*push*("*W*");

*else* *if* (*i* == *numOfExPr* && !*el*.*isCompleted*)

*el*.*states*.*push*("*E*");

    });

*return* *process*;

}

*function* *showQuantTable*(*el*, *process*, *q*)

{

*let* *str*1 = ["#", '*Priority*', '*Duration*', '*Remained*'];

*let* *b* = *document*.*createElement*("*b*");

*b*.*innerHTML* = `*Current* *quant* *is* ${*q*}`;

*el*.*append*(*b*);

*el*.*append*(*document*.*createElement*("*br*"));

*let* *table* = *document*.*createElement*("*table*");

*let* *thead* = *document*.*createElement*("*thead*");

*let* *tbody* = *document*.*createElement*("*tbody*");

*table*.*style*.*borderCollapse* = "*collapse*";

*table*.*setAttribute*("*border*", "1");

*str*1.*forEach*((*el*) => {

*let* *td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *el*;

*thead*.*append*(*td*);

    })

*process*.*forEach*((*el*, *ind*) => {

*let* *tr* = *document*.*createElement*("*tr*");

*let* *td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *ind*;

*tr*.*append*(*td*);

*Object*.*values*(*el*).*slice*(0, 3).*forEach*((*propName*) =>

        {

*let* *td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *propName*;

*tr*.*appendChild*(*td*);

        });

*tbody*.*append*(*tr*);

    })

*table*.*append*(*tbody*);

*table*.*append*(*thead*);

*el*.*append*(*table*);

}

*function* *showTable*()

{

*table*.*push*(["#", ...*statesN*]);

*let* *t* = *document*.*createElement*("*table*");

*let* *thead* = *document*.*createElement*("*thead*");

*let* *tbody* = *document*.*createElement*("*tbody*");

*t*.*style*.*borderCollapse* = "*collapse*";

*t*.*setAttribute*("*border*", "1");

*let* *tr* = *document*.*createElement*("*tr*");

*table*[0].*forEach*((*el*) => {

*let* *td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *el*;

*tr*.*append*(*td*);

    })

*thead*.*append*(*tr*);

*t*.*append*(*thead*);

*for* (*let* *i* = 0; *i* < *process*.*length*; *i*++)

    {

*let* *tr* = *document*.*createElement*("*tr*");

*let* *td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *process*[*i*].*priority*;

*tr*.*append*(*td*);

*process*[*i*].*states*.*forEach*((*state*) => {

*td* = *document*.*createElement*("*td*");

*td*.*innerHTML* = *state*;

*tr*.*append*(*td*);

        });

*tbody*.*append*(*tr*);

    }

*t*.*append*(*tbody*);

*document*.*body*.*appendChild*(*t*);

}

**ПРИЛОЖЕНИЕ B**

**Текст программы**

**Index.html:**

<!*DOCTYPE* *HTML*>

<*html*>

<*head*>

    <*style*>

*td* {

*padding*: 5*px*;

        }

    </*style*>

</*head*>

<*body*>

    <*div* *id*="*tableDiv*">

    </*div*>

    <*br*><*input* *type*="*button*" *value*="*Continue*" *id*="*btn*">

    <*script* *src*="./*lab*.*js*"></*script*>

</*body*>

</*html*>