The paper illustrate the concept of process states through the figure(1) that show the life time of processes\(^3\). Finally, we show the important of process control block through the figure(2) that contains many pieces of information associated with a specific process, (i.e.;) the PCB represents a record or a data structure that holds information about each process\(^4\).

![Diagram of Process State](image1)

**Fig.(1):** Diagram of Process State \(^4\)

![Process Control Block (PCB)](image2)

**Fig.(2):** Process Control Block (PCB)\(^4\)

---

**2- DESIGN SYSTEM FOR CPU SCHEDULING**

Building system that will implement the concepts of process schedules in visual way and the system must able to show to the user how the process will be execute and what is the highest priority and what is the first process to execute. The system allow user to choose the type of schedule (FCFS, Round Robin, priority, short job first, shortest remaining time). On most multitasking systems, only one process can truly be active at a time - the system must therefore share its time between the execution of many processes. This sharing is called scheduling\(^5\). Different methods of scheduling are appropriate for different kinds of execution. There are many type of scheduling.
- Queuing. This is appropriate for serial or batch jobs like print spooling and requests from a server. There are two main ways of giving priority to the jobs in a queue. One is a first-come first-served (FCFS) basis, also referred to as first-in first-out (FIFO); the other is Sorted queue, in which the elements are regularly ordered according to some rule. The most prevalent example of this is the shortest job first (SJF) rule.

- Round-robin. This is the time-sharing approach in which several tasks can coexist. The scheduler gives a short time-slice to each job, before moving on to the next job, polling each task round and round. This way, all the tasks advance, little by little, on a controlled basis \(^{(6)(7)}\).

The system should be able to allow user to add process, delete process and show how will execute in visual way. Using C# language, where the C# is a modern object-oriented language.

**4.1. The system form and entering form**

The graphical user interface will include a main window figure(3) that holds scheduling algorithm selection from an option list and a command for launching the sub window which holds specific data about the selected algorithm where the user can manipulate with the scheduling execution.

![Fig.(3):- The Main Window](image)

When the user presses the select button a new window should be viewed to him containing data fields for the selected scheduling algorithm and some commands for managing the execution of processes and threads.
Each scheduling algorithms will have its unique representation, in the FIFO algorithm for example as shown in the above figure(4), the running process is the process that is sliced out from the pie chart.

The figure(4) represents all processes and their information and it allows the filling of an initial data, this data can be change while run time with the help of the commands in the command panel.

2.1.1. Interaction scenario

This scenario represents the interaction between the user and the system.

- The user enters the system.
- The system views the main window.

In the main window the user selects the type of scheduling algorithm he wants to
visualize.

Fig.(6):- The Selection of Algorithm

- The system views a sub window which is the scheduling window and contains initial fields and specific data for each algorithm.

Fig.(7):- After Selecting the Algorithm

- The user sets the initial data and presses run.

Fig.(8):- Before Starting the Visualize

The system starts the scheduling and views it in a visual manner.
The user can then do some commands like creating processes or threads and killing them.

When the user calls a command the system will affect the scheduling with the command and the visualization will change according to it.

When the user wants to restart the scheduling he has only to press the restart command then the system will end the current scheduling and set it to the beginning of a new scheduling.

When a user wants to change the scheduling he may press the change scheduling and the current window will release resources and let him start a new window with a new scheduling process.

When a user presses the exit command all the processes and threads will terminate and the system will release all resources.

When the user wants to create a new process he fills down a new row in the table with the process values then press's create process button.
Fig.(10):- Creating Process

The result for the new process creation will affect the visualization by adding a new slice for that process.

Fig.(11):- Creating Process Slice

- As the system allows the creation of a process the system will allow the killing of them by simply selecting the process to kill from the table and pressing the kill process button.
The result for process killing is shown in figure(13).

The user may even create a thread for a certain process by selecting the process and pressing create thread.
Fig. (14): Creating Thread

Fig. (15): Cutting the Slice to two Pieces that Represents the Threads

- To dispatch process the user selects a process that is not running from the table and make it run by pressing the dispatch process button.
Fig. (16):- Dispatching A Process

- To block a process the user selects the process from the table and presses the block button the system will turn its visual slice to black.

Fig. (17):- Blocking A Process

- To suspend a process the user just selects the process and presses suspend and the system turns it to gray.
To resume a process the user selects a process that is already suspended and the system resume it to execution.

To change the priority of a process in any other scheduling algorithm other than FIFO the user just set the new priority of the process in the table and updates it.

**TESTING**

System testing involves integration components that implement system functions or features and then testing this integrated system.

3.1.**Black Box Testing:**

Black Box Testing is testing technique having no knowledge of the internal functionality/structure of the system. This testing technique treats the system as black box or closed box. Tester will only know the formal inputs and projected results. Tester does not know how the program actually arrives at those results. Hence tester tests the system based on the functional specifications given to him. That is the reason black box testing is also considered as functional testing. This testing technique is also called as behavioral testing or opaque box testing or simply closed box testing. Although black box testing is a behavioral testing, Behavioral test design is slightly different from black-box test design because the use of internal knowledge is not illegal in behavioral testing \(^8\).
3.1.1 Run

<table>
<thead>
<tr>
<th>Function Being Tested</th>
<th>Initial System State</th>
<th>Input</th>
<th>Expected Output</th>
<th>Post Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>User will choose to start running algorithm.</td>
<td>Click run Button</td>
<td>Data for algorithm</td>
<td>Display diagram</td>
<td>Starting execution of the algorithm with the selected data</td>
</tr>
<tr>
<td>User will choose to start running algorithm.</td>
<td>Click run Button</td>
<td>Wrong data</td>
<td>Error Message</td>
<td>Error Message</td>
</tr>
</tbody>
</table>

3.1.2. Fill Scheduling data

<table>
<thead>
<tr>
<th>Function Being Tested</th>
<th>Initial System State</th>
<th>Input</th>
<th>Expected Output</th>
<th>Post Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>User will choose to fill the initial algorithm data</td>
<td>Click fill Button</td>
<td>Data for algorithm</td>
<td>Visualization form is visible</td>
<td>Storing data and enabling run button</td>
</tr>
<tr>
<td>User will choose to start running algorithm.</td>
<td>Click fill Button</td>
<td>Wrong data</td>
<td>Error Message</td>
<td>Error Message</td>
</tr>
</tbody>
</table>

4- CONCLUSIONS

1. This system is built to show how scheduling algorithms are work in visual way and show how process are executing.
2. This system show process acts in different process states and show the difference between schedule algorithm.
3. From this system we obtain easy tools for the user to easy manages process and easy tools to the user to add or remove process and show the effect.
4. Easy tools to show how different process scheduling are working and the user have better understand the process schedule.
5. The system more realistic process model that can be configured easily by the user.

5- REFERENCES

7. Haldar and Subramanian, 1991, "Discuss fairness in processor scheduling in time sharing system".
محاكاة لجدولة وحدة المعالجة المركزية

يسرى احمد فاضل
مدرس مساعد
كلية الهندسة، جامعة ديالى

هدى محمد صالح
مدرس مساعد
كلية الهندسة، جامعة ديالى

الخلاصة

جدولة وحدة المعالجة المركزية هي أساس عمل نظام التشغيل متعدد البرامج. إذ عن طريق قيام وحدة المعالجة المركزية بالتحكم في التنفيذ، يصبح من الممكن أن نجعل نظام تشغيل الكمبيوتر يجعل من الممكن أن نجد نظراً لدورة التدوين الذي استخدمه في جميع الأوقات، من أجل تعظيم الاستفادة القصوى من عمل وحدة المعالجة المركزية.

هذا البحث يقدم محاكاة لعمل وحدة المعالجة المركزية باستخدام التمثيل الرسومي لتمثيل مفاهيم خوارزميات جدولة وحدة المعالجة المركزية، وهو يتيح للمستخدم اختبار وزيادة فهمه للمفاهيم المدرسية عن طريق جدولة الدراسات. من خلال واجهة مستخدم رسومية سهلة، إذ أنه رسومياً يصور كل برنامج من حيث ما يقوم به حاليا في علمنه ضمن زمن محدد. وبالاستخدام هذا التمثيل يصبح أسهل بكثير، فهم ما يجري داخل النظام، وما هو سبب ترشيح مجموعه مختلفة من البرامج وتخصيصها لوحدة المعالجة المركزية في أوقات مختلفة.

في هذا البحث، تم تقديم مفاهيم أساسية للجدولة، وعرض العديد من خوارزميات جدولة وحدة المعالجة المركزية المختلفة بطريقة مرئية.

الكلمة المفتاحية: جدولة وحدة المعالجة المركزية، ونظام متعدد البرامج، البرنامج بصيغته التنفيذية.

Diyala Journal of Engineering Sciences, Vol. 02, No. 02, December 2009
52