Implementation Resource Request Algorithm In Simulation of Deadlock Avoidance

A W Salim*, F Wiranata, C M Mahidin, R F Waruwu, Vikram, S Wardani, A Dharma

Computer Science Department, Faculty of Technology and Computer Science, Universitas Prima Indonesia, Medan, Indonesia
*abdidharma@unprimdn.ac.id

Abstract. Deadlock is a condition in the operating system, where processes wait for an event that will never occurs. If a deadlock occurs in a system, the system will hang and cannot be operated. The solution to this problem is to avoid deadlock by closing the possibility that might cause deadlock. This research uses Resource Request algorithm to avoid deadlock. The algorithm will try to lend resources to the process and analyse whether the state after lending resources is safe state or unsafe state. If the state is a safe state, then the loan is given, otherwise if not the loan will be pending until the loan does not cause unsafe state. Application can simulate the requesting of resources by processes and prevent deadlock in the simulation using Resource Request algorithm, which is to ensure that every situation after lending resource is in a safe state. Deadlock will not occur in safe state.

1. Introduction
One of the roles of the operating system is related to process management and resources [1]. When the computer starts, a number of processes will be carried out by the CPU by using existing resources according to needs [2]. A resource can be a hardware device (for example: a printer) that is connected to a computer or a piece of information stored on a storage medium. The role of the operating system is to allocate one or more resources for use by certain processes [3]. In practice, often a process requires a resource that is also needed by other processes. This is commonly known as resource sharing. If scheduling goes well, each process will get the required resources so that the processes can be completed [2]. A process is called deadlock if the process waits for a particular event that will never occur. A set of deadlock conditions if each process in the collection waits for an event that can only be done by another process that is also in the collection. Deadlock occurs when the processes of accessing resources exclusively are holding resources and requesting other resources. (which is being held by another process) [4]. One method to prevent deadlocks is the Banker Algorithm [5].

Banker’s Algorithm is a deadlock handling algorithm besides the Ostrich Algorithm. Bankers themselves are taken from the word Bank. Bankers here have such meaning (Bank people), because this algorithm refers to the banking system in dealing with dead ends. That is, this algorithm is used by banks to ensure that the bank will not lack resources (resources), which of course in the case of bank resources is the customer's money [6]. In the Banker Algorithm, there are two states, namely safe state and unsafe state. A safe situation is a situation where the bank still has enough money (resource) when a bank customer withdraws his money. Unsafe state (unsafe state) occurs when bank customers make withdrawals in excess of existing resources (money). Handling, namely by allowing customers to
withdraw money if the withdrawal will not bring the bank to unsafe conditions. If the withdrawal of money brings the bank to an unsafe condition, then the customer must wait until there is additional money from other customers in the bank. [2]

2. Method

2.1. Research Methods

The researcher completes the research using the System Development Life Cycle (SDLC) method, which consists of stages of data collection, analysis, design, coding and testing. The SDLC method can be seen in Figure 1.

![Figure 1. Research Methods](image)

2.2. Data Collection

Data collection is done using the Library Study method, which collects reference material from electronic journals, such as: scientific journals and articles, and sources from print media, such as the book Operating Systems, especially those related to process synchronization and deadlocks, and deadlock prevention.

2.3. Analysis

The analysis discusses deadlock problems that occur inside the computer and how the Resource Request algorithm works in avoiding deadlocks and simulation processes that will run in the application. Exclusive access to resources by several processes within a computer is one example of a possible case for dealing with deadlock problems. Some processes can run at once and compete to get a loan of resources from a computer. Some examples include the process of printing documents with a printer, using a scanner, read / write with a floppy disk or CD / DVD-RW and other applications, such as photo editor applications and office applications. The processes that run on a computer require resources, in the form of memory to complete the execution of the process. This limited memory is borrowed from the operating system. In borrowing and using memory, the operating system allows for use and hold and wait conditions, where the process can request resources according to their needs at that time. When the process requires additional resources, the process can request new resources while using old resources. This process runs until it is finished and the process of returning all resources (borrowed) to the operating system. The resources that are being used by the process cannot be forcibly taken from the process (non-preemption condition), because it will cause the terminate process and the process must be repeated (reset / restart) again from the beginning. The process of borrowing demand and the simultaneous use of resources by the process will cause a situation, where the operating system no longer has the available resources, while the processes, which are using and using resources, still need additional resources to complete execution. In this situation, all processes will wait for each other and hope the other process is completed, so that additional resources are available to be borrowed and used. Circularly waiting for each other this condition is also called circular wait condition. As a result, all processes cannot run and deadlocks occur. One way to avoid deadlock known in the science of Operating Systems is to use the Resource Request Algorithm.

2.4. Design
Designing the appearance of the system (prototype) is done by designing the form in general in the application. The application used to design the prototype display is Microsoft Visual Basic .NET. The form design and reports in the application consist of:

- **Input Form.** The Input Form serves to input the number of resource types, names and maximum total resources owned by the system.

- **Simulation Form.** The Simulation Form serves to conduct borrowing simulation poses and resource use by processes that are running simultaneously. Each process that occurs in the simulation will be recorded on the History form and each step of the resource request algorithm can be seen by selecting the checkbox “Show Algorithm Analysis Results”.

- **New Process Form.** The New Process Form function to start a new process if the simulation is done manually, and enter the maximum amount of resources needed by the process.

- **Borrowing Resource Form.** The Borrowing Resource Form serves to make resource borrowing when the simulation is done manually, and includes the amount of resources that the process will borrow.

- **History Process Form.** The History Process Form used to display the history of processes that have appeared in the simulation process.

- **Time Report Form.** The Time Report Form used to display the simulation process that occurs at any time.

- **Algorithm Analysis Form.** The Algorithm Analysis Form function to display the workings of the Resource Request algorithm in deciding whether to borrow resources or not to the process. This form will appear every time there is a loan request from the process, and the algorithm performs an analysis, whether the loan can be given or not.

- **About Form.** The About Form contains the identity of the researcher or compiler of the thesis, and brief information about the application.

### 2.5. Coding

Simulation software is designed using the Microsoft Visual Basic .Net 2008 programming language. The components used in making this software are standard VB .Net standard components.

### 2.6. Testing

Software testing is done to correct errors that might occur during the coding process.

### 3. Result and Discussion

When the application starts, the Input form will appear. On this form, the user can choose the number of resource types, fill in the name and the number of resources owned by the computer and choose whether the simulation will use an algorithm to avoid deadlocks. Display Form Input can be seen in Figure 2.
In Figure 2, if the deadlock avoidance algorithm is not used, the simulation might produce a deadlock condition. If the user clicks on the 'Simulation Process' button on the Input Form, the Simulation form will appear. This form will display the simulation process from the software. Simulation, borrowing resources, refusing loans and lending resource agreements will run automatically or can be set manually. Users can temporarily stop the simulation process that is running and the user can also set the speed of the simulation process as desired. On this form, users can see the results of the analysis of resource request algorithms for each request to borrow resources from the process. Display of the Simulation form can be seen in Figure 3.
If the user chooses to manually manage the simulation, the user can set the appearance of a new process by pressing the 'New Process' button under the blank process image. The New Process form display image can be seen in Figure 4.

![Figure 4. Display New Process Form](image)

To request borrowing resources manually by the process, the user can press the 'Request Resource' button under the process image. Display of Resource Borrowing forms can be seen in Figure 5.

![Figure 5. Display of Resource Borrowing Form](image)
If the user presses the 'Process History' button on the Simulation form, the Process History form will appear, as shown in Figure 6.

![Figure 6. Display of Process History Form](image)

If the user presses the 'Time Report' button on the Simulation form, the Time Report form will appear, as shown in Figure 7.

![Figure 7. Display of Time Report Form](image)
When the user checks the checkbox "Display the results of algorithm analysis" in the Form Simulation, the Analysis Algorithm form will appear every time a resource request occurs by the process. The Algorithm Analysis form display image can be seen in Figure 8.

![Algorithm Analysis Form](image)

**Figure 8. Display of Algorithm Analysis Form**

To display the About form, the user can press the 'About' button in the Input form. The About form display can be seen in Figure 9.

![About Form](image)

**Figure 9. Display of About Form**
If the resource request algorithm is not used in the simulation process, a deadlock state may occur. Examples of deadlock conditions can be seen in Figure 10.

![SIMULATION FORM](image)

**Figure 10.** Deadlock conditions

Figure 10 shows the situation where there is no process that can run because all processes are waiting for each other to return resources from other processes, while other processes have not finished using resources. This mutual waiting situation is called deadlock. If the user uses the Resource Request algorithm, then the deadlock state in the simulation can be avoided.

4. Conclusions and Recommendations

4.1 Conclusion
After completing the application design implementation of the Resource Request algorithm in a deadlock avoidance simulation, the authors draw the following conclusions:
- Applications can prevent deadlocks in simulations using the Resource Request algorithm, which is to ensure that every situation after borrowing resources must be safe (safe state). With a safe state, deadlock will not occur.
- Applications can display the results of a resource request algorithm in approving or rejecting resource requests from each process, so that it can help learning about this deadlock avoidance algorithm in the operating system course.
- The application provides a feature for users not to use deadlock avoidance algorithms in simulations. In this situation, a deadlock may occur.

4.2 Recommendations
Some suggestions that can be given that can help in developing this application further are by improving the animation quality of the simulation process by using Adobe Flash software and also developing applications so that it can simulate deadlock conditions that occur in real time on a computer device.

References

[1] K Hameed, et. al. 2016, Resource Management in Operating Systems-A Survey of Scheduling Algorithms, *Int. Conf. on Innovative Computing (ICIC)*, (Pakistan: University Of Management and Technology) vol 1
[2] B Haryanto, 2014, Sistem Operasi, Bandung: Penerbit Informatika
[3] S Kusumadewi, 2013, Sistem Operasi, Jakarta: Graha Ilmu
[4] M Lawley, 2001, Deadlock Avoidance for Sequential Resource Allocation Systems: Hard and Easy Cases, *Int. Journal of Flexible Manufacturing Systems*, (Atlanta: Purdue University)
[5] Okojie, et. al. 2012, A Software Aided Approach in Avoiding Deadlock Problems in Operating Systems Using Banker’s Algorithm, University of Port Harcourt, Nigeria
[6] B Yao, J Yin, and W Wu, 2016, Deadlock Avoidance Based on Graph Theory, Hainan University, China, Vol 9 No. 2, pp.353-382
[7] G A Ansari, 2012, A Modeling and Detection of Deadlock in Early Stage of System using UML, Qassim University, Saudi Arabia Vol 39 No 9
[8] P Kawadkar, S Prasad, and A D Dwivendi, 2014, Deadlock Avoidance Based on Banker’s Algorithm for Waiting State Processes, *Int. Journal of Innovative Science and Modern Engineering (IJISME)*, vol 2 issue 12, ISSN: 2319-6836
[9] Pangera A dan Ariyus D, 2015, Sistem Operasi, (Yogyakarta : STMIK AMIKOM)
[10] Setiawan R, 2017, Sistem Operasi, Malang: Seribu Bintang
[11] R Shakir, 2014, Comparison Between Proposed and Existing Algorithms for Deadlock Avoidance and Recovery, *Int. Journal of Information Technology and Electrical Engineering* (Islamabad: Federal Urdu University of Arts, Science and Technology), vol 3 issue 3, ISSN:--708X
[12] Sufian I, 2015, Sistem Operasi, Jakarta: Graha Ilmu
[13] Kondhalka V S, 2011, Deadlock Detection And Recovery in Linux (United States: San Diego State University)
[14] Watrianthos R dan Purnama I, 2018, Buku Ajar Sistem Operasi, Ponorogo: Uwais Inspirasi Indonesia
[15] Xu Gang and W Zhiming, 2013, Deadlock Avoidance Based on Banker’s Algorithm for FMS, Shanghai Jiaotong University, Shanghai

**Acknowledgements**

This research was supported by the Computer Science Department, Faculty Of Technology and Computer Science, Universitas Prima Indonesia (UNPRI), Medan.