SMARTICS Journal, Vol.7 No. 1 2021. p1-6
ISSN online: 2476-9754, ISSN print: 2623-0429
DOI : https://doi.org/10.21067/smartics.v7i1.5229

Application of Simulation of Dining Philosophers Problem in Concurrent Process

Parasian DP Silitonga[a*,c], Irene Sri Morina[b], Romanus Damanik[a,c]
[a,c]Informatics Engineering Department, Saint Thomas Catholic University, Setia Budi N0. 479-F, Medan, Indonesia
[b]Data and Information Department, Adam Malik Hospital, Bunga Lau No. 17, Medan, Indonesia
[*]correspondence email : parasianirene@gmail.com

***Abstract*— A classic simulation that can describe concurrent processes on an operating system is the dining philosopher problem. In the dining philosophers problem, there is the possibility of a deadlock, a condition in which two or more processes cannot continue their execution. This research produces a software application that can simulate the concurrent process and prevent deadlock problems that occur. The benefit of the simulation in this study is as a visualization of the completion of dining philosophers problems and as an additional facility in the teaching and learning process, especially in operating system courses.**

***Index Terms*—Concurrent; Deadlock; Operating System; Simulation**

I. INTRODUCTION

The operating system is an interface that bridges communication between the user and the computer system. As the main component in a computer system, the operating system must control the use of all available resources [1]. In the process of designing an operating system, there is a common foundation called concurrency. Processes are declared concurrent if several processes exist at the same time [2]. Concurrent processes can be completely independent of each other but can also interact with each other.

The interacting processes require synchronization to adequately controlled. However, several problems must resolve in the interacting concurrent processes, such as deadlocks, synchronization, etc. A classic simulation that can describe concurrent processes on an operating system is the dining philosopher problem. The dining philosophers problem is illustrated as several philosophers at the dining table with some chopsticks [3]. If there is a very hungry philosopher, he will take chopsticks. If philosophers take chopsticks, then some philosophers have to wait until the chopsticks put back. In this illustration, there is a possibility of a deadlock, a condition in which two or more processes cannot continue their execution [4].

Simulation is the process of designing a model of an existing system, then conducting experiments on the model and evaluating the experimental results [5]. Simulation is a technique or method of solving problems through imitation system data processing to obtain experimental output data as material for problem solutions or input for actual system development and improvement. By doing a simulation, it can see the results that will obtain if applied to the real world and can minimize the risk.

In the world of education, simulation can use as a teaching method, assuming that not all learning processes can be carried out directly on the actual object. The simulation learning method is a learning method that imitates something real to its surroundings or process [6]. Simulation can improve understanding by providing a visualization of an object [7]. Based on these reasons, in this study, a dining philosopher problem simulation was built, which is used to visualize the concurrent process in the operating system.

II. MATERIAL AND METHOD

A. Operating System

The operating system is an interface that bridges communication between the user and the computer system. The operating system hides all the hardware complexities by providing functions at the machine level to avoid programming hassles, making it easier for application program designers [8].

Received January 12, 2021; Revised January 18, 2021; Accepted January 18, 2021

As the main component in a computer system, the operating system control the use of all available resources [1]. Some of the functions provided by the operating system include :

- Software that controls hardware is just an ordinary program.
- Programs that make hardware easier to use.
- Collection of programs that regulate hardware work.
- Resource manager or resource allocator.
- As a control program.
- As the kernel, a program that continues to run as long as the computer is turned on.
- As a guardian, which is to regulate or protect computers from various computer crimes.

As a resource manager, the operating system provides routines for handling computer resources. These routines can be grouped into five broad categories, namely [9]:

- Process management.
- Main memory management.
- I / O Management.
- File management.
- Secondary memory management concurrent

In the process of designing an operating system, there is a common foundation called concurrency. Processes are declared concurrent if several processes exist at the same time [2]. Concurrent systems allow processes to access one resource simultaneously. Causes a race condition, which is a condition in which more than one process tries to access a resource simultaneously [10]. This race condition has an impact on the consistency and validity of the conditions of the resources accessed.

Concurrency in the operating system is very important. Currently, the operating system is multiprogramming or multithreading and leading to distributed processing, which requires concurrent processes. In concurrent processes that interact with each other, several fundamental problems must be resolved :

- Mutual Exclusion
- Deadlock
- Starvation
- Synchronization

The solution to these problems is fundamental because computer system technology leads to a multiprocessing, distributed and parallel system that requires congruent processes. The scope of congruences includes the following [11] :

- Allocation of processing services to processes.
- Common use and competition for resources.
- Communication between processes.
- Multi-process activity synchronization deadlock.

One of the fundamental problems with concurrent operating system processes is deadlocks. A process is a deadlock if the process is waiting for a specific event that will never happen [12]. A set of processes is a deadlock when every process in the collection is waiting for an event that other processes can only do. The process of waiting for events that will never happen.

Deadlock occurs when processes access exclusive resources. All deadlocks that occur involve competition for exclusive resources by two or more processes. A process is said to experience starvation when the processes wait for an infinite resource allocation, while other processes can obtain resource allocations [8]. Starvation is due to bias in resource allocation policies or strategies. This condition must avoid because it is unfair, but it is a desire that avoidance was done as efficiently as possible.

Waiting for each other in an operating system process can be due to the prerequisite of each process being another process or waiting for resources used by other processes. There are four conditions for the occurrence of deadlocks, namely [10] :

- Mutual exclusion condition, where each resource at that time is given to exactly one process.
- Hold and wait condition, the processes that are holding the resource, waiting for a new resource.
- Non-preemption condition, previously given resources cannot be forcibly taken from the process. Resources must be explicitly released from the processes that hold them.

There is a circular chain of two or more processes in circular wait condition, each waiting for the resource held by the next member in the chain. The first three conditions are the necessary conditions for the occurrence of deadlocks. The existence of a deadlock always means that these conditions are met. However, the existence of these three conditions does not guarantee a deadlock. Deadlock occurs when the fourth condition is met. The fourth condition is a must for deadlocks to occur.

B. Simulation

Simulation is the process of designing a model of an existing system, then conducting experiments on the model and evaluating the experimental results [5]. Simulation is a technique or method of solving problems through imitation system data processing to obtain experimental output data as material for problem solutions or input for actual system development and improvement. Doing a simulation can see the results obtained if applied to the real world and minimize the risk.

Simulation is a technique or method of solving problems through data processing imitation system operating systems to obtain output data from investigations or research experiments as material for problem solutions or as input to develop and improve actual system structures and operations. The benefits of simulation in the real world, among others [13]:

- Describe system behavior.
- Simulate the operation of a system through a model.
- Solve a mathematical problem with numerical analysis.
- Studying the dynamics of a system.
- Provides a description of system behavior in development over time.
- Build theories and hypotheses that account for the behavior of the system that is enjoyed.
- Predicting future system behavior, namely the effect of system changes or changes in operation.

C. Dining Philosophers Problem

The Dining Philosophers Problem is one of the classic problems in synchronization. The dining philosophers problem illustrates several philosophers at the dining table with some chopsticks [3]. There are five philosophers with five bowls of noodles in front of each philosopher and one chopstick between each philosopher. Philosophers spend time thinking (when complete) and eating (when hungry). When hungry, the philosopher will take two chopsticks (in the left and right hands) and eat. However, at times, only one chopstick was taken. If a philosopher takes two chopsticks, then two philosophers beside the philosopher who is eating must wait until the chopsticks put back. The deadlock condition can occur if every philosopher is hungry and takes the left chopstick, then all the values of the chopsticks will be zero, and then each philosopher will take the right chopstick, then a deadlock will occur. There are several ways to avoid deadlocks, including :

- Allow a maximum of 4 philosophers to sit together at one table.
- Allows a philosopher to pick up chopsticks only if they are present.
- Using an asymmetric solution, the philosopher on an odd number takes the left chopstick first, then the right chopstick. Meanwhile, the philosopher who sits on an even chair takes the right chopstick first, then the left chopstick.

III. RESULT

Table 1. Process Illustration

	Time A	Time B	Initial Condition
First Philosopher	7	10	15
Second Philosopher	5	4	4
Third Philosopher	6	11	14
Fourth Philosopher	5	13	11
Fifth Philosopher	6	12	18

From Table 1, it know that the condition of each philosopher, namely philosopher-1, is in a complete condition because the initial condition of 15 seconds is above the B-time, which is only 10 seconds (philosopher-1 will feel hungry and look for chopsticks 5 seconds later). Philosopher-2 is in a hungry state because the initial condition 4 seconds = time-B, philosopher-3 is in a state of fullness, philosopher-4 is in a state of hunger, and philosopher-5 is in a state of fullness. The initial conditions of the dining philosophers problem can describe as in Figure 1.

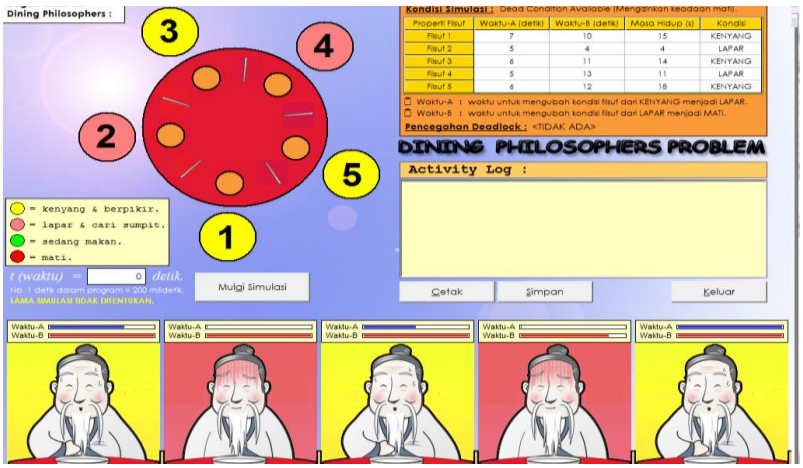

Fig. 1. Simulation Initial Condition

When t = 1 second, philosophers-1, philosophers-3 and philosophers-5 are full and thinking, the simulation process is complete and thinking, while philosophers-2 and philosophers-4 are hungry and get chopsticks in their left hand. At t = 2 seconds, philosophers-2 and philosophers-4 get two chopsticks and start eating, while philosophers-1, philosophers-3 and philosophers-5 are still full and thinking. At time t = 3 seconds, the philosopher-3 feels hungry (because the current philosopher-3 life span = time-B is 11 seconds) and starts looking for chopsticks but does not get chopsticks because the chopstick on the left use by philosopher-4 and chopsticks on the right used by philosophers-2.

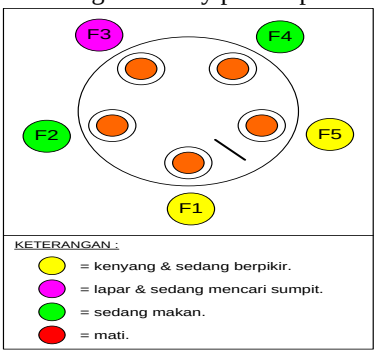

Fig. 2. Illustration of the Simulation Condition when t = 3 Seconds

At time t = 5 seconds, philosopher-1 feels hungry (because philosopher-1's current lifetime = time-B is 10 seconds) and looks for chopsticks. Philosopher-1 got chopsticks in his right hand. At time t = 6 seconds, the philosopher-5 feels hungry (because the current philosopher-5's lifetime = time-B is 12 seconds) and looks for chopsticks. The philosopher-5 did not get the chopsticks. At t = 9 seconds, the philosopher-2 is full (because his life has reached the maximum value, which is 9 seconds = time-A + time-B) and begins to think. Philosopher-3 gets chopsticks in his right hand.

At t = 10 seconds, the philosopher-1 gets two chopsticks and starts eating. At t = 11 seconds, the philosopher-4 is complete and begins to think. Philosopher-5 gets chopsticks in his right hand. At t = 12 seconds, the philosopher-3 gets two chopsticks and starts eating.

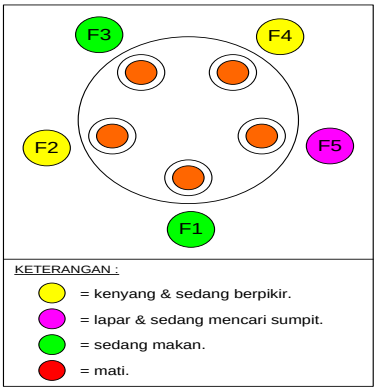

Fig. 3. Illustration of the Simulation Condition when t = 12 Seconds

The simulation process will continue according to the procedure. The simulation will only stop if a deadlock condition occurs. Table 2 is an illustration of the simulation of a deadlock.

Table 2. Process Deadlock Illustration

	Time A	Time B	Initial Condition
First Philosopher	7	12	27
Second Philosopher	5	3	2
Third Philosopher	15	10	25
Fourth Philosopher	6	5	5
Fifth Philosopher	20	5	20

At t = 1 second, philosophers-2 and philosophers-4 are hungry and get chopsticks in their left hand, while philosophers-1, philosophers-3 and philosophers-5 are full and thinking. At t = 2 seconds, philosopher-2 and philosopher-4 get two chopsticks and start eating. At t = 10 seconds, philosophers-2 and philosophers-4 are full and start thinking. At t = 15 seconds, all philosophers are simultaneously hungry and take the chopstick in their left hand. There was a deadlock condition because all the philosophers holding chopsticks in their left hands were waiting for chopsticks on the right. All philosophers will wait for each other.

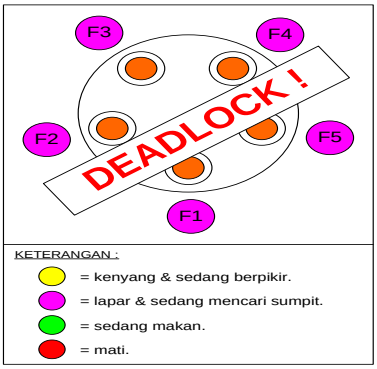

Fig. 4. Deadock Condition Illustration

In this simulation, deadlock conditions that occur can avoid by choosing one of the following three solutions :

- Solution-A allows a maximum of 4 philosophers to sit together at one table. There will be no deadlock condition if there are less than four philosophers seated together around a table with five seats (one philosopher is considered dead).
- Solution-B, which is to allow a philosopher to take chopsticks only if the two chopsticks are present. If all philosophers are hungry simultaneously, only two philosophers can eat because the philosopher takes two chopsticks simultaneously.

- Solution-C is an asymmetric solution. The philosopher on the odd number takes the left chopstick then the right chopstick, while the philosopher on the even number takes the right chopstick first, then the left chopstick. Suppose all philosophers are hungry at the same time. In that case, the asymmetric method of taking will prevent all philosophers from taking the left chopstick simultaneously to avoid the deadlock condition.

IV. CONCLUSION

Based on the results obtained in this study, several conclusions draw, namely, the dining philosopher problems simulation built in this study can visualize concurrent processes well. The simulation application built in this study can visualize deadlock problem handlers through 3 deadlock handling solutions. It recommends that the simulation built in this study develop by illustrating a more complex process to visualize the concurrent process better.

ACKNOWLEDGMENT

Thanks very much to the Catholic University of Saint Thomas Medan for providing support so that this research can carry out well. Thanks also to Kanjuruhan Malang University for being willing to publish the results of this research. The author is open to all suggestions and criticisms for this research's improvement and perfection in the future.

REFERENCES

[1] M. Marufuzzaman, S. Al Karim, M. S. Rahman, N. M. Zahid, and L. M. Sidek, "A review on reliability, security and memory management of numerous operating systems," *Indones. J. Electr. Eng. Informatics*, vol. 7, no. 3, pp. 577–585, 2019, doi: 10.11591/ijeei.v7i3.987.

[2] W. Salem and M. Hefnawi, "Management of the Processes for Evaluating External Human Induced Events Using Operating Systems Concept," no. January 2019, pp. 23–29, 2018, doi: 10.13140/RG.2.2.21473.35687.

[3] V. Choppella, A. Sanjeev, K. Viswanath, and B. Jayaraman, "Generalised Dining Philosophers as Feedback Control," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11969 LNCS, pp. 144–164, 2020, doi: 10.1007/978-3-030-36987-3_9.

[4] A. W. Salim *et al.*, "Implementation Resource Request Alghoritm in Simulation of Deadlock Avoidance," *J. Phys. Conf. Ser.*, vol. 1230, no. 1, 2019, doi: 10.1088/1742-6596/1230/1/012096.

[5] D. Mourtzis, "Simulation in the design and operation of manufacturing systems: state of the art and new trends," *Int. J. Prod. Res.*, vol. 58, no. 7, pp. 1927–1949, 2020, doi: 10.1080/00207543.2019.1636321.

[6] S. Zulfiqar, R. Zhou, F. Asmi, and A. Yasin, "Using simulation system for collaborative learning to enhance learner's performance," *Cogent Educ.*, vol. 5, no. 1, p. 1424678, 2018, doi: 10.1080/2331186X.2018.1424678.

[7] P. D. P. Silitonga, H. Himawan, and R. Damanik, "Forecasting acceptance of new students using double exponential smoothing method," *J. Crit. Rev.*, vol. 7, no. 1, pp. 300–305, 2020, doi: 10.31838/jcr.07.01.57.

[8] S. Husnjak, I. Jović, I. Cvitić, and J. Štefanac, "Overview: Operating Systems of Modern Terminal Devices," *Proc. 5th Int. Virtual Res. Conf. Tech. Discip.*, vol. 6, no. December, pp. 8–13, 2018, doi: 10.18638/rcitd.2018.6.1.124.

[9] G. P. S. Brijender Kahanwal, Tejinder Pal Singh, Ruchira Bhargava1, "File System – A Component Of Operating System," *Asian J. Comput. Sci. Inf. Technol.*, 2011.

[10] T. Farah, R. Shelim, M. Zaman, M. M. Hassan, and D. Alam, "Study of race condition: A privilege escalation vulnerability," *WMSCI 2017 - 21st World Multi-Conference Syst. Cybern. Informatics, Proc.*, vol. 2, no. December 2020, pp. 100–105, 2017.

[11] H. Maghsoudloo, "A Survey of Concurrency Control Algorithms in the Operating Systems," vol. 3, no. 2, pp. 302–307, 2014.

[12] P. Chahar and S. Dalal, "Deadlock Resolution Techniques: An Overview," *Int. J. Sci. Res. Publ.*, vol. 3, no. 7, pp. 2250–3153, 2013, [Online]. Available: www.ijsrp.org.

[13] T. Feliciani *et al.*, *A scoping review of simulation models of peer review*, vol. 121, no. 1. Springer International Publishing, 2019.

Parasin DP Silitonga is a lecturer assisting the Higher Education Service Institute Region I, seconded at the Faculty of Computer Science, Catholic University of Saint Thomas, Medan. He completed his Masters in Computer Science from Gadjah Mada University Yogyakarta in 2010. Currently, he is the Head of the Computer Laboratory at the Catholic University of Saint Thomas, Medan.

Irene Sri Morina completed her Bachelor of Computer education in 2002 from the Faculty of Computer Science, Saint Thomas Catholic University, Medan. Then completed her Masters in Computer education in 2015 from the University of North Sumatra. She was currently serving in the Data and Information Section of the Adam Malik Hospital Medan.

Romanus Damanik completed his Masters in Computer from the University of North Sumatra. He is a permanent lecturer at the Catholic University of Saint Thomas. Currently given the trust to occupy the position of Head of Quality Assurance Unit at the Faculty of Computer Science.