International Research Journal of Computer Science (IRJCS)
Issue 10, Volume 07 (November 2020)
ISSN: 2393-9842
https://www.irjcs.com/archives

APPLICATION OF THREADS

Aishika Saha
Computer Science,
Vellore Institute of Technology
aishisaha1@gmail.com

Publication History
Manuscript Reference No: IRJCS/RS/Vol.07/Issue10/NVCS10081
Received: 02, November 2020
Accepted: 17, November 2020
Published: 25, November 2020
DOI: https://doi.org/10.26562/irjcs.2020.v0710.001
Citation: Aishika saha (2020). Application of Threads. IRJCS:: International Research Journal of Computer Science, Volume VII, 267-275. https://doi.org/10.26562/irjcs.2020.v0710.001
Peer-review: Double-blind Peer-reviewed
Editor: Dr.A.Arul Lawrence Selvakumar, Chief Editor, IRJCS, AM Publications, India

Abstract: The motive behind the selection of this topic is the wide range of application and usage of threads in today's technology. It's become essential to cut back the time for execution of various processes. So, developing solutions to problems using threads helps us solve multiple tasks in a very time economical manner. To accomplish this, a deep understanding of the concept of multi-threading is crucial. Our motive is to get thorough knowledge concerning this concept to enable us to use this concept ourselves towards a real-life problem.

We will be using this concept to solve the subsequent problem:

A man needs to cross a river. He possesses a tiger, a goat and a basket containing grass. There is a ship on the river. The utmost capacity of the boat is to hold two entities at a time. The man has to cross the river with his tiger, goat and basket of grass. If he goes with the basket at the opposite side of the river the tiger can eat the goat, if he takes the tiger with him to the opposite side then the goat can eat the grass from the basket. Here processes require resources which can be made available once the other process gets completed, these condition results in deadlock. The problem is how to design a discipline of behaviour (a concurrent algorithm) to seek out the sequence of the four will cross the stream safely.

Keywords: application threads based calibration computer

I. INTRODUCTION

There is seldom any computer or program nowadays that doesn't support the concept of Multi-threading. Multi-threading helps to cut back the latent period by carrying out two unrelated process simultaneously, therefore reducing the total process time considerably.

Microsoft Word: The foremost basic example for multi-threading is word, where spell-check and input/output process is completed at the same time.

Web Browser: Responding to multiple user requests at the same time.

Operating Systems: One the most widely used Multi-threading concept in everyday life is in OS. Sorting massive set of information is often done by using multi-threading concept.

Bankers algorithm: The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by testing the allocation for predetermined maximum possible quantities of all resources, then makes an "s-state" check to test for possible activities, before deciding whether or not allocation ought to be allowed to continue.

Reader Writer:

Problem parameters:

- One set of data is shared among several processes
- Once a writer is ready, it performs its write. Just one writer can write at a time
- If a process is writing, no alternative process can read it
- If a minimum of one reader is reading, no alternate process can write
- Readers might not write and solely read

Sleeping barber: The analogy is based upon a theoretical barber shop with one barber. There's a barber shop that has one barber, one chair for the barber, and n chairs for waiting for clients if the barber is occupied.

- If there's no client, then the barber sleeps in his own chair.
- When a client arrives, he should awaken the barber.
- If there are many clients and the barber is cutting a customer's hair, then the remaining clients either wait if there are empty chairs within the waiting room or they leave if no chairs are empty.

Producer Consumer: We have a buffer of static size. A producer will produce an item and will place in the buffer. A client will pick items and will consume them. We want to confirm that once a producer is placing an item in the buffer, then no client will consume any item. In this problem, buffer is the crucial section.
Dining philosopher: The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between every pair of philosophers. There is a single chopstick between every philosopher. A philosopher will eat if he can pick up the two chopsticks adjacent to him. One chopstick can be picked up by any one of its adjacent followers however not both.

II. PROBLEM STATEMENT

We will be using this concept to solve the subsequent problem:
A man needs to cross a river. He possesses a tiger, a goat and a basket containing grass. There is a ship on the river. The utmost capacity of the boat is to hold two entities at a time. The man has to cross the river with his tiger, goat and basket of grass. If he goes with the basket at the opposite side of the river the tiger can eat the goat, if he takes the tiger with him to the opposite side then the goat can eat the grass from the basket. Here processes require resources which can be made available once the other process gets completed, these condition results in deadlock. The problem is how to design a discipline of behaviour (a concurrent algorithm) to seek out the sequence of the four will cross the stream safely.

III. METHODOLOGY: EXPERIMENTAL / SIMULATION

We have two objectives to be achieved by this project and they are as follows:
A. Our main objective is to carefully understand the varied applications of threads.
B. To study how these are used and to be able to use these methods ourselves and apply them by formulating our own problem.
We will analyze the following problems:
1) Bankers problem
2) Reader-Writer problem
3) Sleeping barber's problem
4) Producer-consumer problem
5) Dining philosopher's problem
The study of these problems and their solutions will provide us better understand about the varied applications of concepts in operating systems such as resource sharing, buffer, allocation, deadlock avoidance, etc. This will help attain our first objective. After an intensive study of these problems, we'll be able to formulate our own problem (based on real-world issues) which can be resolved using the concepts learned from these problems. This is how we will accomplish our second objective.
Existing method:
The existing solution to the problem includes a java program, which sequentially checks all the conditions and formulates output. The problem with this solution is that the output can be much faster if the checking of these conditions is done on separate threads, using the concept of threading.
Proposed solution:
Our program using the idea of splitting the task and dealing with each part simultaneously, so as to get to the output quicker. Using the concept of threading, the code runs on multiple cores simultaneously to check if the conditions for the game satisfies to be continued to play.
Conditions when the game can end:
A. If the tiger eats the goat when the man is not present on or near the shore.
B. If the goat eats the grass when the man is not present on or near the shore.
Modules:
A. Thread for boat: This thread checks if man is present on the bat or not, before the boat starts to sail the sea. If the man is not present on the boat, then other threads are stopped and a message is displayed on the screen saying "The boat must contain the man to sail through the sea."
B. Thread for left shore: This thread checks if game terminating conditions like the goat eats the grass, or the tiger eats the goat on the left shore are satisfies and leads to the termination of the game or not.
C. Thread for right shore: This thread checks if the game terminating conditions like the goat eats the grass, or the tiger eats the goat on the right shore are satisfies and leads to the termination of the game or not. When the boat reaches the right shore, a prompt requests the user for an input, as to which animal must be dropped on the right shore. Another prompt request asks the user if any entity (tiger, grass or the goat) must be sailed back to the left beach. The game terminating conditions are checked again when the boat starts to sail to the left beach.

D. Parent thread: The parent thread is supposed to run the main of the program. This thread controls the start and termination of all other threads. It also prompts the user for an input when the boat sails from right to the left shore asking if any entity needs to be dropped on the let shore or not.

IV. PSEUDOCODE

1) Start
2) Display the members on the left shore, the boat and the right shore.
3) Ask the user for the input of a member from one part to another.
4) Check if the conditions for the game fails (example: if the tiger eats the goat or the goat eats the grass).
5) Create three threads and a binary semaphore to control the threads, run them simultaneously. One of the thread checks the left shore, another one checks the right shore, and the last thread checks the conditions on the boat (if the boat can be ridden by one of the elements).
6) If the game can be continued, go to step 2.
7) If the conditions lead to the failure of the play, display the message accordingly.

V. RESULTS

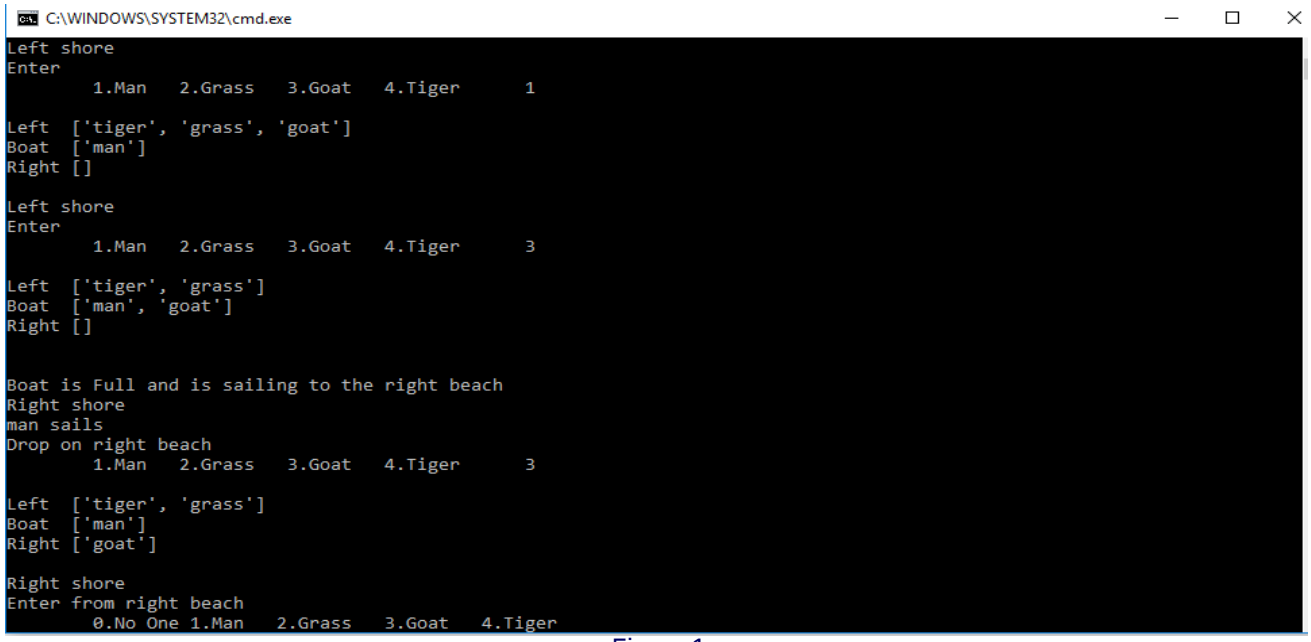

Figure 1

Initially, the boat array and the right array are empty and all the entities reside in the left array. The program asks the user for an input and provides the state of the game.

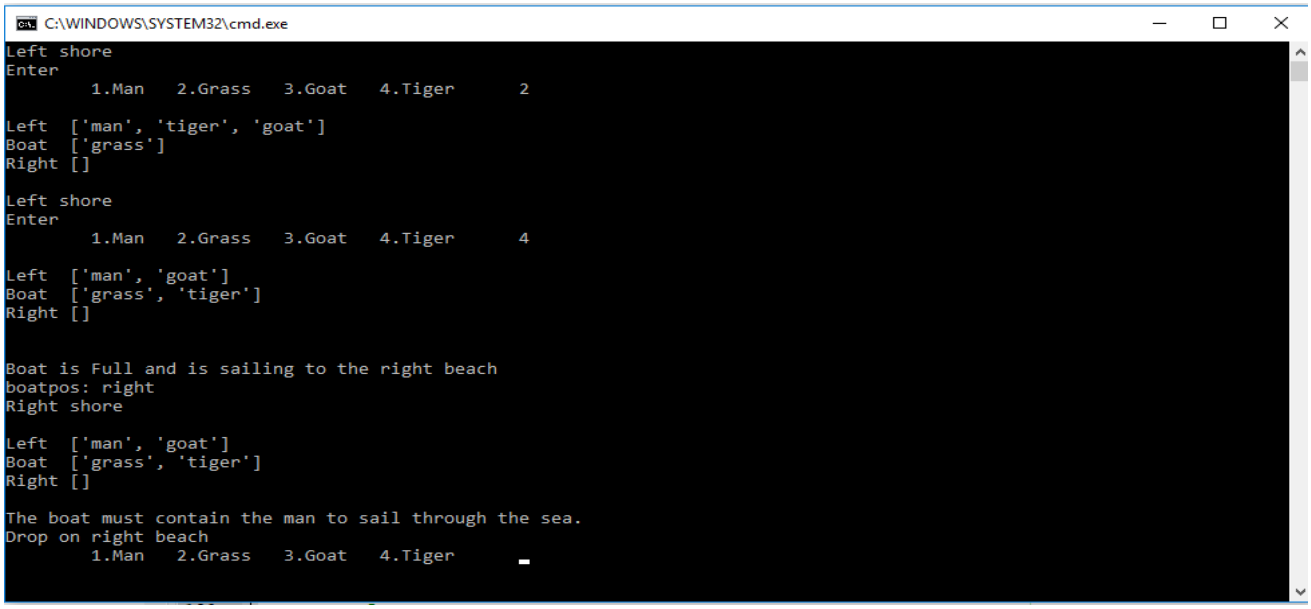

Figure 2.

Currently, the boat is on the left shore. After taking input as '1', the man is transferred from the left array to the boat array. The program requests for another input. Here the input is of '3', so the goat is transferred to the boat. Now the boat is full as the capacity of the boat is two entities. The boat sails to the right beach. Now the boat is on the right shore and the man sails the boat. The program asks input for whom to drop on right shore. The input given is '3'. So now the goat is transferred to the right array. When entities are swapped from one array to another array, each array is check for violation of the game conditions. If the conditions are satisfied by each array no error is reported. All the four modules are implemented, and since no error is encountered, the game flows smoothly. The program asks for input of two entities. We choose '2' and '4', which transfers grass and tiger to the boat. As the man is not on the boat condition for all three arrays is not satisfied and hence we obtain the error 'The boat must contain the man to sail through the sea'. The boat module returns an error message since; the man was not present on the boat when the boat was sailing through the sea.

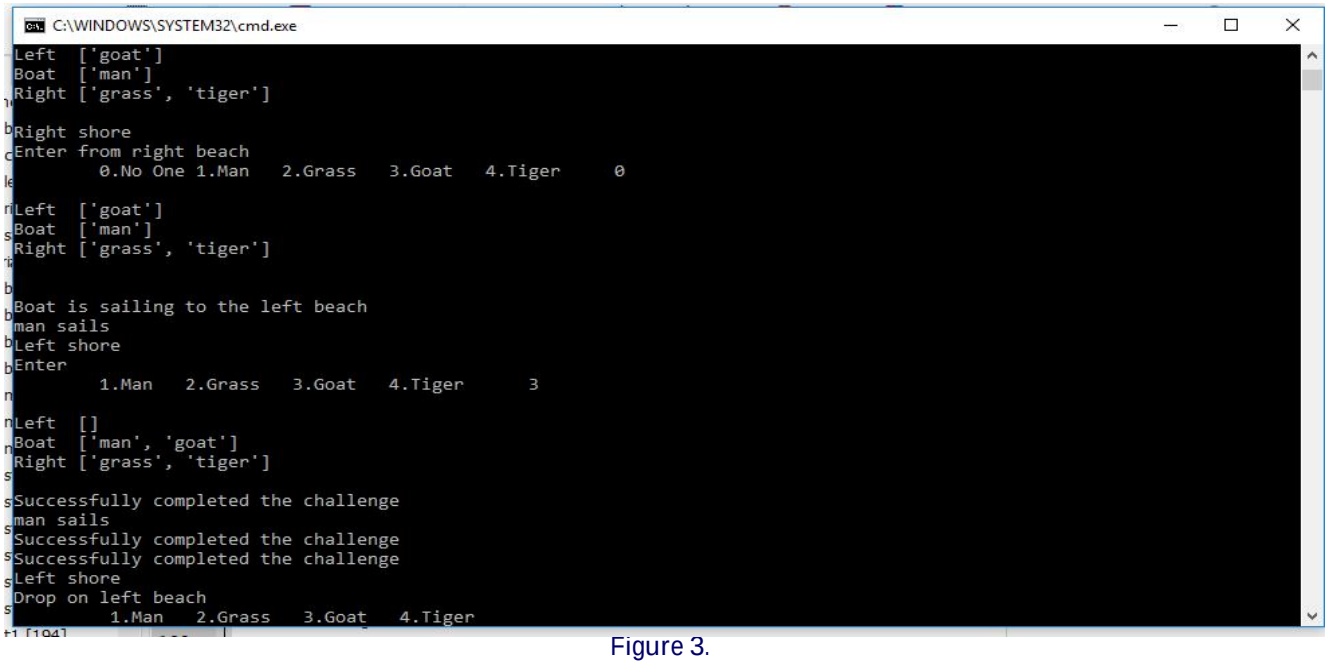

Figure 3.

The game is already in progress and the current status of the arrays is displayed. Left shore has the grass, the boat has the man and right has the grass and tiger. The boat is currently on the right shore. The program requesting input for whom to pick up from the right shore. The input given is '0', hence no transfers take place. The boat sails to the left beach and the man sails the boat. Now at the left beach, again input is taken for whom to add to the boat array. The input is '3', hence the goat is transferred to the boat array. Now the arrays satisfy game successfully completed conditions and hence the message is displayed. All the modules are implemented in this case. The games flows smoothly and a message is prompted on the used screen saying that the challenge was successfully completed.

V. CONCLUSIONS

We learned a lot about threads through this task and came across many problems which are solved by using threads and there are many more problems that are solvable using the thread's concept. We shortlisted the solved problems to five classical problems. These were the very basic problems which helped us in out in understanding the use of the thread and its functionality which made us able to understand and use the inbuilt libraries available for the thread manipulation in Python and, helped us stay close to the syllabus and understand it better. The whole point of our project was to learn how to apply the thread's concept in real-world problems. So, we have provided just one of the many examples of what we can solve using threads. And hence we have completed our task.

TECHNICAL SPECIFICATION

I have used python to execute this problem statement.

CODE:

```
meml    = ['man','tiger','grass','goat']
memr    = []
memb    = []
stop    = 0
start1  = 0
start2  = 0
start3  = 0
boatpos = 'left'
```



```
def checkarea(a,place):
        global stop
        if stop == 1:
                return
        if( 'tiger' in a  and 'goat' in a and 'man' not in a):
                stop=1
                print('\n*****OOPS the Tiger at the Goat on the',place,'*****')
        if( 'grass' in a  and 'goat' in a  and 'man' not in a):
                stop=1
                print('\n*****OOPS the Goat at the Grass on the',place,'*****')
def sail():
        global boatpos
        if boatpos=='right':
          print('\nBoat is Full and is sailing to the right beach ')
        else:
          print('\nBoat is sailing to the left beach ')
def left():
        global memb
        global memr
        global meml
        global stop
        global boatpos
        if (len(memb) == 2):
                checkarea(meml,'left shore')
        else:
                return
        if stop==0:
                return;
        if (len(meml) == 0):
                print('Successfully completed the challenge')
def right():
        global memb
        global memr
        global meml
        global stop
        global boatpos
        checkarea(memr,'right shore')
        if (len(meml) == 0):
                print('Successfully completed the challenge')
                return;
        while ( boatpos=='right' and stop==0 ):
                sail()
                x=5
                while(x>4 or x<0):
                        print('Right shore')
                        x=int(input('Drop on right beach \n\t1.Man  2.Grass  3.Goat  4.Tiger\t'));
                        if(x>4 or x<0):
                                print('Invalid input. ');
                        else:
                                if (x==1):
                                        if('man' not in memb):
                                                print('Man in not on the boat')
                                                continue
                                        memb.remove('man');
                                        memr.append('man');
                                elif (x==2):
                                        if('grass' not in memb):
                                                print('Grass in not on the boat')
                                                continue
```



```
                        memb.remove('grass');
                        memr.append('grass');
                elif (x==3):
                        if('goat' not in memb):
                                print('goat in not on the boat')
                                continue
                        memb.remove('goat');
                        memr.append('goat');
                elif (x==4):
                        if('tiger' not in memb):
                                print('tiger in not on the boat')
                                continue
                        memb.remove('tiger');
                        memr.append('tiger');
                boatpos='left'
                print('\nLeft ',meml)
                print('Boat ',memb)
                print('Right',memr,'\n')
                if (len(meml) == 0):
                        print('Successfully completed the challenge')
                        return;
        x=5
        while(x>4 or x<0):
                print('Right shore')
                x=int(input('Enter from right beach \n\t0.No One 1.Man  2.Grass  3.Goat  4.Tiger\t'));
                if(x>4 or x<0):
                        print('Invalid input. ');
                else:
                        if (x==1):
                                if('man' not in memr):
                                        print('Man in not on the right shore')
                                        continue
                                memr.remove('man');
                                memb.append('man');
                        elif (x==2):
                                if('grass' not in memr):
                                        print('Grass in not on the right shore')
                                        continue
                                memr.remove('grass');
                                memb.append('grass');
                        elif (x==3):
                                if('goat' not in memr):
                                        print('goat in not on the right shore')
                                        continue
                                memr.remove('goat');
                                memb.append('goat');
                        elif (x==4):
                                if('tiger' not in memr):
                                        print('tiger in not on the right shore')
                                        continue
                                memr.remove('tiger');
                                memb.append('tiger');
                        boatpos='left'
                        print('\nLeft ',meml)
                        print('Boat ',memb)
                        print('Right',memr,'\n')
                        sail()
checkarea(memr,'right shore')
```



```
def boat():
        global memb
        global memr
        global meml
        global stop
        global boatpos
        checkarea(memb,'boat')
        if (stop == 1):
                return;
        if('man' not in memb):
                stop=1
                print('boatpos: ' + boatpos  )
                print('\nLeft ',meml)
                print('Boat ',memb)
                print('Right',memr,'\n')
                print('The boat must contain the man to sail through the sea.')
                else:
                print('man sails')
        if (len(meml) == 0):
                print('Successfully completed the challenge')
        if __name__ == "__main__":
                while (stop == 0):
                x=5
                boatpos='left'
                        while(x>4 or x<0):
                        t1 = threading.Thread(target=left,  args=())
                        t2 = threading.Thread(target=right, args=())
                        t3 = threading.Thread(target=boat,  args=())
                        print('Left shore')
                        x=int(input('Enter \n\t1.Man  2.Grass  3.Goat  4.Tiger\t'));
                        if(x>4 or x<0):
                                print('Invalid input. ');
                        else:
                                if (x==1):
                                        if('man' not in meml):
                                                print('Man in not on the left shore')
                                                continue;
                                        meml.remove('man')
                                        memb.append('man')
                                elif (x==2):
                                        if('grass' not in meml):
                                                print('grass in not on the left shore')
                                                continue;
                                        meml.remove('grass')
                                        memb.append('grass')
                                elif (x==3):
                                        if('goat' not in meml):
                                                print('goat in not on the left shore')
                                                continue;
                                        meml.remove('goat');
                                        memb.append('goat');
                                elif (x==4):
                                        if('tiger' not in meml):
                                                print('tiger in not on the left shore')
                                                continue;
                                        meml.remove('tiger');
                                        memb.append('tiger');
                                print('\nLeft ',meml)
                                print('Boat ',memb)
```



```
print('Right',memr,'\n')
if (len(memb) == 2):
        boatpos='right'
        t1.start()
        if stop==0:
                t2.start()
                start2 = 1
        if stop==0:
                t3.start()
                start3 = 1
        t1.join()
        if start2 == 1:
                t2.join()
        if start2 == 1:
                t3.join()
        if boatpos == 'left':
                t3 = threading.Thread(target=boat, args=())
                t3.start()
        x=5;
        if (len(meml) == 0):
                print('Successfully completed the challenge')

        while((x>4 or x<0) and len(memb)==2):
                print('Left shore')
        x=int(input('Drop on left beach \n\t 1.Man  2.Grass  3.Goat  4.Tiger\t'));
                if(x>4 or x<0):
                        print('Invalid input. ');
                else:
                        if (x==1):
                                if('man' not in memb):
                                        print('Man in not on the boat')
                                        continue
                                memb.remove('man');
                                meml.append('man');
                        elif (x==2):
                                if('grass' not in memb):
                                        print('Grass in not on the boat')
                                        continue
                                memb.remove('grass');
                                meml.append('grass');
                        elif (x==3):
                                if('goat' not in memb):
                                        print('goat in not on the boat')
                                        continue
                                memb.remove('goat');
                                meml.append('goat');
                        elif (x==4):
                                if('tiger' not in memb):
                                        print('tiger in not on the boat')
                                        continue
                                memb.remove('tiger');
                                meml.append('tiger');
                        boatpos='left'
                        print('\nLeft ',meml)
                        print('Boat ',memb)
                        print('Right',memr,'\n')
                        if (len(meml) == 0):
                                print('Successfully completed the challenge')
```


REFERENCES

1. Eggers, Susan J., et al. "Simultaneous multithreading: A platform for next-generation processors." IEEE micro 17.5 (1997): 12-19
2. Akkary, Haitham, and Michael A. Driscoll. "A dynamic multithreading processor." Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture. IEEE Computer Society Press, 1998.
3. Akkary, Haitham, and Michael A. Driscoll. "A dynamic multithreading processor." Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture. IEEE Computer Society Press, 1998.
4. https://codereview.stackexchange.com/questions/118516/the-wolf-goat-and-cabbage-game
5. https://en.wikipedia.org/wiki/Thread_(computing)
6. https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)
7. Tullsen, Dean M. "Simulation and modeling of a simultaneous multithreading processor." The 1996 22 nd International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems, CMG. Part 2(of 2). 1996.
8. Tullsen, Dean M., et al. "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor." ACM SIGARCH Computer Architecture News. Vol. 24. No. 2. ACM, 1996.
9. https://codereview.stackexchange.com/questions/118516/the-wolf-goat-and-cabbage-game
10. https://en.wikipedia.org/wiki/River_crossing_puzzle
11. https://www.cs.unm.edu/~luger/ai-final/code/LISP.fwgc.html
12. Ascher, Marcia. "A river-crossing problem in cross-cultural perspective." Mathematics Magazine 63.1 (1990): 26-29.
13. Jeffries, Robin, et al. "A process model for missionaries-cannibals and other river-crossing problems." Cognitive Psychology 9.4 (1977): 412-440.
14. Lueckenhaus, Maximilian, and Wolfgang Eckstein. "Thread concept for automatic task parallelization in image analysis." Parallel and Distributed Methods for Image Processing II. Vol. 3452. International Society for Optics and Photonics, 1998. https://whatis.techtarget.com/definition/multithreading
15. F.N.Sibai,"Performance effect of localized thread schedules in heterogeneous multi-core processors," in Innovations in Information Technology, 2007. IIT '07. 4th International Conference on, nov. 2007, pp. 292 – 296.
16. Lichen Weng and Chen Liu, "On better performance from scheduling threads according to resource demands in MMMP," in Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, sept. 2010, pp. 339 – 345.
17. Cheng Lian and Yang Quansheng, "Thread priority sensitive simultaneous multi-threading fair scheduling strategy," in Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, dec. 2009, pp. 1 –4. [8] J.C. Saez, J.I. Gomez, and M
18. P.De,V.Mann, and U. Mittaly, "Handling OS jitter on multicore multithreaded systems," in Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, may 2009, pp. 1 –12.
19. G. Parmer and R. West, "Predictable interrupt management and scheduling in the composite component-based system," in Real-Time Systems Symposium, 2008, dec. 2008, pp. 232 –243.

