Jurnal Mantik, 5 (2) (2021) 1334-1340

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik/index

Multi-threading on Linux Operating System Using Scheduling Algorithm

Rahmad Syuhada

Department of Information Technology,Universitas Battuta, Medan, Indonesia
Email : rahmadsyuhada@yahoo.co.id

ARTICLE INFO

Article history:
Received: 30 July 2021
Revised: 12 August 2021
Accepted:30 August, 2021

Keywords:
Operating Systems, Round Robin Algorithm, Concurrency, Linux

ABSTRACT

The choice of operating system in terms of convenience and operation is a top priority for users, where the operating system has a systematic schedule, which includes data processing, data storage, calculation of memory usage and other resources. The operating system continues to grow today so that the operating system is also the center of attention because of its very rapid development. In research conducted on process scheduling, the Linux scheduling implementation is made to calculate input on the data, where the main menu input produces the amount of waiting time with the result of how many times the process performs an activity, then with the process scheduling algorithm, efficiency, completion time, throughput, and waiting time. The scheduling algorithm system that is of concern in this development is the Round Robin Algorithm. Therefore, an analysis of multi-threading techniques with this algorithm is carried out to see processes running simultaneously so as to produce an average total waiting time of 7.4 ms and an average turnaround time 11.6 ms. by counting the existence of threads in a process so that there is concurrency in many processes in one execution

1. Introduction

An OS provides various types of services in exploiting resources, which requires one or many processes to the user, these resources are closely related to the computer system. This is because the operating system regulates the supporting components of the computer system such as memory, I/O modules when receiving or sending signals by calling the system [1], or I/O devices and other constituent components so that process execution can be carried out and hide the complexity of hardware settings from users and application makers. This causes the need to understand how the computer system works, knowing how the operating system's ability to support threads in a process both by using resource ownership and execution/scheduling.

The Round Robin algorithm is one of the scheduling processes, where this algorithm rotates the processes in the queue [2]. The process will get a share of the time quantum, if the time quantum runs out or the process is complete, the CPU will be allocated to the next process. In this process scheduling, no process is prioritized, all processes get the same time share from the CPU. Therefore, to find out how the operating system's ability to support multi-threads in a process, both in resource ownership and scheduling/execution on the Linux operating system will be analyzed. Based on research from the utilization of the process that can be achieved [3], to set an upper limit on processor usage on system priority they define the processor utilization factor to be the fraction of processors in the time spent in performing each task set. therefore, the utilization factor is equal to one minus the fraction of idle processor time, because Ci|Ti is a fraction of the processor time spent executing task τ_i, so for *m* the task of the utilization factor is :

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

$$U \sum_{i=1}^{m} (C_i / T_i)$$

Although the processor utilization factor can be increased by increasing the C_i's values or by decreasing the T_i's values, it can be limited by the requirement that all tasks meet deadlines at critical times. Analyzed the ability of the operating system to support multiple threads in a process. Analyzed the ability of the operating system to support multiple threads in a process. The Linux architecture[4] discusses some of the architectures that support real-time characteristics and discusses what constitutes a real-time architecture. The task of the classical periodic model in Liu and Layland's research, is the most widely used in real-time system modeling[3]. This model allows multiple settings for a job. These parameters are of two types: Static parameters for the task itself and dynamic parameters on each instance. Basic static parameters of periodic τi is :

$$\tau_i = (R_i, C_i, D_i, T_i) \quad (1)$$

R_i : (time release) the date of the first activation of task i, when τ_i was able to start for the first time.
C_i : (time computing): execution time τ_i. This parameter is considered in several works in real-time scheduling as worst-time execution (WCET for time worst execution) on the processor it will run on.
D_i : relative maturity or critical time frame for each activation τ_i.
T_i : implementation period τ_i.
The parameters in this task in addition to bringing determinism to interrupt processing[8], task scheduling also supports periodic intervals required for real-time processing. can be seen in Figure 1 below:

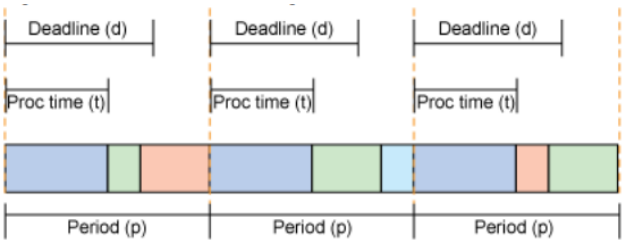

Figure 1. Periodic task scheduling

Figure 1. shows the scheduling of periodic tasks, many control systems require sampling and processing. In the specified period (p), certain tasks must be executed for system stability.

2. Literature Review

2.1 Round Robin Algorithm

The operating system assigns a fixed priority to each of its processes, arranging processes in a queue-ready state in order of priority. Processes with high priority will not be disturbed by processes with low priority [7]. The Round Robin scheduling algorithm in research [4], has the following steps:

1. The scheduler maintains a queue of ready processes and a list of swapped and blocked processes.
2. The PCB from the new process is created and added to the end of the queue. The PCB is removed from the scheduling data structure.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

3. The scheduler always picks the PCB from the start of the queue. this is a disadvantage because all processes are given essentially the same priority.
4. When the running process completes its time division, then the process is moved to the end of the queue, there is each process on the processor per iteration of the Round Robin algorithm.
5. The controller performs the following actions :
   a. When a sub-process makes an input-output request, the PCB is removed from the queue ready to be blocked/swapped list.
   b. When an input-output operation awaited by a process is completed or a process is swapped in the PCB, it is removed from the blocked/swapped list to the end of the queue.

2.2 Race Condition

Race Condition is a condition where shared memory/data can be accessed by one or more processes simultaneously and the final result is not as desired [12]. In the spooler directory there are parts with large sizes, given the order of 0, 1, 2, 3, 4, … each sequence can contain a file name.

TABLE 1
RACE CONDITION

Print spooler	Spooler directory
Process X →	Checks the input section that doesn't contain (7) and stop.
	interrupt…
Process Y →	data is put to print in section (7), and stops (this section 7+1 = 8)
	Processor performs other processes…
Process X continues →	the data that is placed will be printed in section (7), so that it can overwrite the process of data Y that is placed in section (7).
Then process Y→	not executed, because no error has been detected.

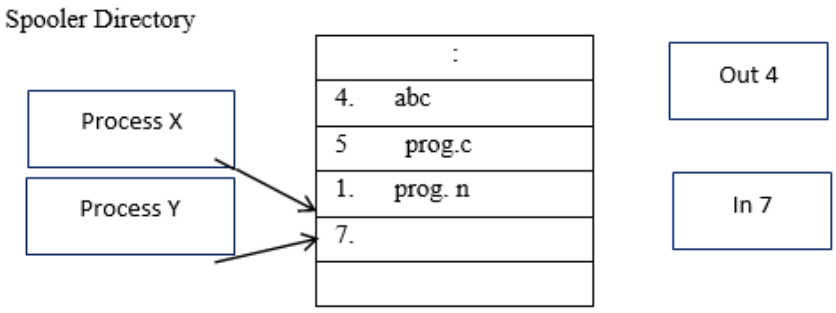

Figure 2. Race Condition in the spooler directoy

2.3 Multi-threaded Analysis

Multithreading is the operating system's process of managing its use by more than one user at a time or the ability of a program to manage multiple requests by users without having to have multiple copies of the program running on the computer..

The User-level thread model can be implemented into four models[14], including :

1. *Many-to-one*
   In Many-to-one modeling all user-level threads run on the same kernel, so processes can only run with one thread at a time..
2. *One-to-one*
   In One-to-one modeling, each user-level thread is run into a separate kernel-level thread.
   Many-to-many
   In the Many-to-many phase, the process is allocated m of kernel-level threads to execute n number of user-level threads.
3. *Two-level*

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

The Two-level model is similar to the Many-to-many model but also allows level threads to be tied to kernel-level threads. All user-level thread models in the kernel are execution units that are scheduled by the kernel to be executed on the CPU. The term "virtual processor" is often used instead of the kernel.

2.4 Operating System Linux

Linux is an operating system with resources [15], hardware (hardware) will function if there is an operating system (OS). software and hardware (hardware) are basically managed by the operating system. The Linux operating system has several parts including:

a. Bootloader : computer software in managing the boot process.
b. Kernel : The core part of a system in managing peripherals, CPU, memory.
c. Init System : its job of controlling the daemon on the sub-system, managing the boot process, after the initial boot is handled by the boot loader.
d. Daemons : Voice and scheduling services that start after logging in to the desktop or at boot.
e. Graphical Server : Displaying graphics on monitor.
f. Desktop Environment : Part to interact with users. (KDE, Gnome, Enlightenment).
g. Aplications : High quality software can be installed easily on linux.

2.5 Linux Scheduling Implementation

The CFS scheduling structure does not discuss time, but discusses and takes into account the time in which each process runs, so it is necessary to ensure that each process runs well on the processor part. CFS uses an entity structure, the struct sched_entity, like this : defined in <linux / sched.h>, to track a process:

```
struct sched_entity {
struct load_weight load;
struct rb_node run_node;
struct list_head group_node;
unsigned int on_rq;
u64 exec_start;
u64 sum_exec_runtime;
u64 vruntime;
u64 prev_sum_exec_runtime;
www.it-ebooks.info
ptg
The Linux Scheduling Implementation 51
u64 last_wakeup;
u64 avg_overlap;
u64 nr_migrations;
u64 start_runtime;
u64 avg_wakeup;
/* many stat variables elided, enabled only if CONFIG_SCHEDSTATS is set */
};
```

3. Result and Discussion

Discussion of the results of previous studies, critically analyzed [5], and associated with the existence of multithreads in the queue process according to Figure 2.1 race condition if it continues to run. The following flowchart gives a brief idea of the sequence of the scheduling process :

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

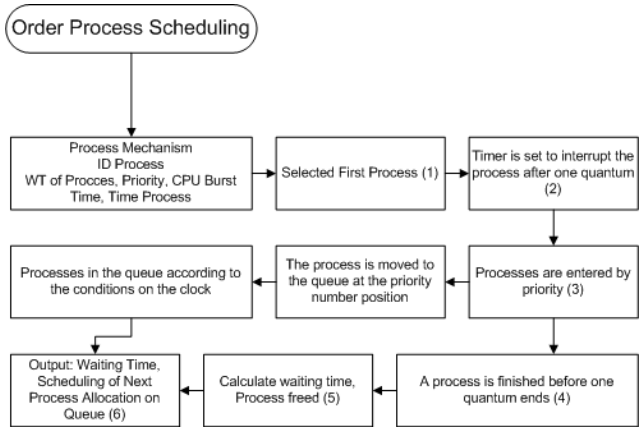

Figure 3. Sequence Analysis of Scheduling Process

The process is determined by CPU burst time and priority, then this process is scheduled to perform implementation of CPU scheduling algorithm based on incoming time priority with the experiment that will be discussed here for sharing the time quantum. The Round Robin Algorithm with the average completion time is calculated so as to get the results.

3.1 Process Mechanism:

Every incoming process must have a field:

a. ID process (integer, max. 3bit)
b. Waiting Time of Process
c. Number of priority (1 – 3, priority)
d. CPU burst time : unit of time that describes the time interval for carrying out the process
e. Processing time I/O

Queue Process :

1. Every queue that is executed first becomes pseudo parallelism
2. If the first queue is empty, the second queue will continue to run even if a new process enters.
3. I/O uses the SPOOLING system, and I/O processing runs parallel to the CPU.

3.2 Case Study

Tabel 3.1. Input Components of the Processor

ID Process	Entry Time	Process Time
R0	0	5
R1	1	7
R2	3	5
R3	6	3
R4	7	1

The result data obtained in table 3.1 shows the process with burst time and priority using the Round-Robin Algorithm with time quantum = 3, the Gantt Chart is obtained in the following picture :

R0	R1	R2	R3	R4	R0	R1	R2	R1	
0	3	6	9	12	13	15	18	20	21

figure 4. *Gantt Chart* Algorithm Round-Robin with time quantum = 3

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Tabel 3.2. Input Components of the Processor 1

Process ID	Arrivel time	Long Process	When You Start	When Done	AWT	ATT
R0	0	5	0	5	0	5
R1	1	7	5	21	13	20
R2	3	5	8	20	12	17
R3	6	3	11	14	5	8
R4	7	1	14	15	7	8

Total average waiting time $= \frac{\sum Wt}{P} = \frac{37}{5} =$ 7.4 ms

Average turnaround time=11.6

Multithread in various queuing processes, where the calculation process uses scheduling algorithms that have different variations, the ability of this OS to support concurrency in the number of processes or executions, to calculate the existence of threads from each process to be examined from the operating system is shown in Figure 5 Multithread below.

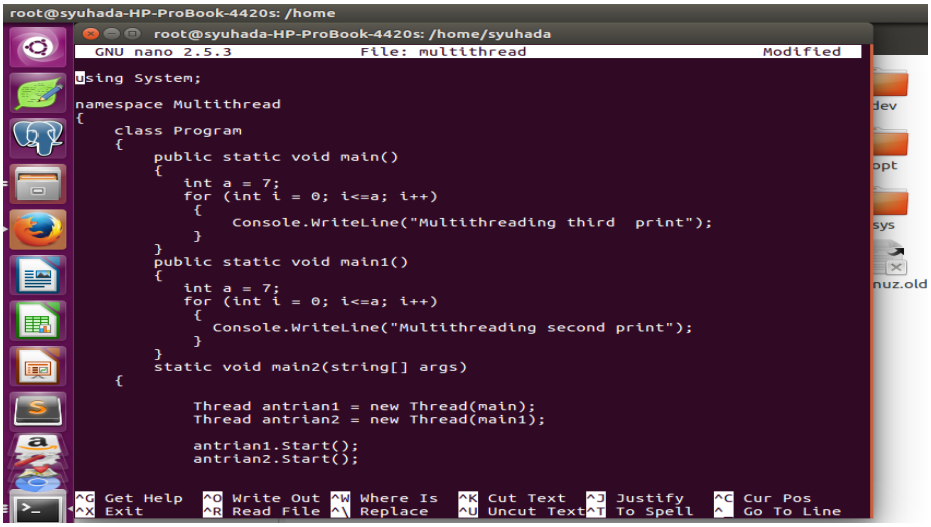

Figure 5. Multithreading in the queue process

4 Conclusion

The results in the study concluded that, from the observed data based on 5 running task priority processes, namely R0, R1, R2, R3, R4 using the Round Robin algorithm, the results obtained where Total AWT (Average Waiting Time) = 7.4 ms and ATT (Average Turnover Time)= 11.6 ms, the number of threads seen in the round robin method has its own advantages and disadvantages, each unique scheduling technique that may be suitable for different operating systems, many processes on the system are categorized in the form of interaction, namely: Process independent and the process cares about each other indirectly. In future research, it is better to look at the differences in scheduling techniques using 3 types of queues, namely using FIFO and Non Premptive SJF scheduling techniques with the maximum priority determined by each.

5 Reference

[1] Abbas, N., Ali, K., and Kadry, S., "A New Round Robin Based Sceduling Algorithm for Operating System: Dinamyc Quantum using the mean average", Faculty of Business Lebanase University.

[2] Ajit, S., Goyal, P., and Bahtra, S., "An Optimized Round Robin Sceduling algorithm for CPU sceduling", International Journal on Computer Science and Engineering Vol. 02 No 07 Tahun, 2012.
[3] C. L. Liu., J.W. Layland, "Scheduling Algorithms for Multiprogramming in a hard realtime environment", ACM 20, vol. 20, pp. 46-61, 1973.
[4] Atmadja, W., Christian, B., and Kristofel, L. "Real Time Operating System on embedded linux with ultrasonic sensor for mobile robot", International Conference on Industrial Automation, Information and Communications Technology (IAICT), Electronic ISBN:978-1-4799-4909-0:2014.
[5] Zouaoui, S., Boussaid, L., and Mtibaa, A., "Priority based round robin (PBRR) CPU scheduling algorithm", International Journal of Electrical and Computer Engineering (IJECE). 2019 Vol.9, No.1, pp.190-202
[6] Yaashuwanth, C., Ramesh, R., "A New Scheduling algorithm for real time task", International Journal of computer science and invormation security, vol 6 no 2, 2009.
[7] Yadav, R.K., Mishra, A. K., Prakash, N., Sharma, H., "An Improved Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066, 2010.
[8] Jones, M. (2008). Anatomy of real-time Linux architectures: IBM
[9] Rajput, I. S., Gupta, D. "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", International Journal of Innovations in Engineering and Technology (IJIET), 1(3), 1-11.
[10] Randal, E., Bryant, D.R., O'Hallaron,(2003) Computer System A Programmer's Perspective: Prentice Hall.
[11] Ali P.A., Ariyus, D.(2010) Sistem Operasi: ANDI, Yogyakarta.
[12] Tanenbaum, A.S.(2008). Modern Operating Systems, Pearson Education Inc, New Jersey.
[13] Zhou, H.Y., Hou K.M., and Vaulx, C., "A Supersmall Distributed REAl-time Microkernel dedicated to wireless sensors", Journal of PERVASIVE COMPUT.&COMM. 2006 issue 4 Vol(2), pp398-410.
[14] Spring (2018). Implementing threads Operating System, Uppsala university.
[15] Robert, L. (2005). Linux Kernel Development: Novell Press.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.