IJCSMS (International Journal of Computer Science & Management Studies) Vol. 19, Issue 01
Publishing Month: October 2015
An Indexed and Referred Journal with ISSN (Online): 2231–5268
www.ijcsms.com

Preemptive and Non Preemptive Priority Scheduling

Ledina Hoxha Karteri[1] and Anisa Shehu[2]

[1]PhD Student in Management of Information Systems, Informatics, Mathematics and Statistics Department, Faculty of Economics, European University of Tirana, Tirana, Albania
ledinakarteri@gmail.com

[2]MSc Student in Computer Engineering, Polytechnic University of Tirana, Faculty of Information Technology, Tirana, Albania

Abstract

Software is the main point of development, due to it become "alive" everything hard. Operating system is the revolution in computer science, based on which we do everything. There are lots of well-known job that operating system does background, such as memory management, processing, file system management, input-output requests and even more, but there will always be a "hole" in OS that can every time do something new on it to become better and better. One "hole" is process management. OS can manage in different way based on some scheduling algorithms, so I decided to compare a priority scheduling algorithms in multitasking with preemptive, non-preemptive and aging technique. We will notice the differences between them.

Keywords: *OS, Priority Scheduling algorithms, preemptive, non-preemptive and aging technique.*

1. Introduction

Operating system changed our life, since it do a lots of duty. Sometimes the OS overall is defined as an abstraction of human life actions. I want to remember how terrible is for a student the wait for his time in front of secretary door, and time by time he stay in the end, because sometimes came a teacher that entered first, another time came a department chief, and then came a rector end so on, so the day "went" and the secretary left. That's why is so important to analyze scheduling algorithms. Sharing of computer resources between multiple processes is also called scheduling [1]. So the scheduler chooses the next one to be admitted and run. And the stage over which pass a process are shown in **Figure1**[10].

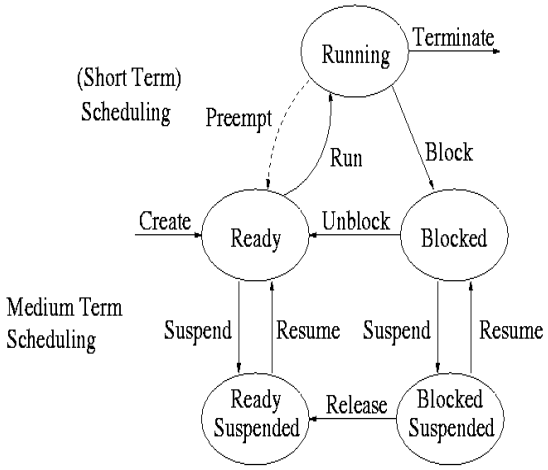


Figure 1: State of Process

A priority scheduling algorithm can leave some low priority processes in the ready queue indefinitely. If the system is heavily loaded, it is great probability that there is a higher-priority process to grab the processor. This is called starvation problem. One solution for the starvation problem might be to gradually increase the priority of processes that stay in the system for a long time [13].

The objectives of a good scheduling policy include [10]:

- Fairness.
- Efficiency.
- Low response time (important for interactive jobs).

- Low turnaround time (important for batch jobs).
- High throughput [the above are from Tanenbaum].
- Repeatability.

1.1. Priority Scheduling

This algorithm is based on execution of process over a priority value. Priorities may be *internal* or *external*. Internal priorities are determined by the system using factors such as time limits, a process' memory requirements, its anticipated I/O to CPU ratio, and any other system-related factors. External priorities are assigned by administrators.[11]
[9]This way every process has its own priority on which will depend if it is going to be run or wait. The first one to run is going to be the process with highest priority, while others will wait for their turn. This algorithm can be:

1.1.1 Preemptive when the priority of the newly arrived process is compared to the process running and if its priority is higher it will occupy the CPU. In this case unfortunately go out from secretary if the rector came in.

1.1.2 Non-Preemptive when the recently arrived process is positioned at the head of the queue. In this case, no one can forced us leave the secretary even if it the rector.

1.1.3 Aging Technique

Anyway, the drawback of this algorithm is indefinite blocking for lower priority processes which seem to never have the chance to be run.
This is solved with a technique called ***aging*** which gradually increases the priority of the processes that have been waiting for a long time. The aging technique estimates the time a process will run based on a weighted average of previous estimates and measured values.[6] Aging can be used to reduce starvation of low priority tasks[12]. So this technique ensure that students with "low priority" to become in home empty-hand. So this trend to be fair in queue.
In this paper we will figure out the differences between non-preemptive, and preemptive with and without aging.

2. Related work

The very first book that brought me on the field of operating system, and specially illumination over scheduling CPU processes was the Modern Operating System by Andrew Tanenbaum read during my studies.

In fact there are a lot of research and papers over scheduling algorithm since there are lot of scheduling algorithms such as Round Robin, Priority scheduling, Shortest Job First etc. I have read a lot of documents about scheduling, and implementation and improvement of efficiency of management of processes that need to access the same resources.

But what I noticed during my researches, there is not a published paper about the comparison of the preemptive and non-preemptive, even something about aging technique.

So I decided to figure out these differences between a preemptive and non-preemtive algorithm, and how to solve the situation of the starvation. There are some works related to this scheduling algorithm, but I am focused on work done by Sukumar Babu Bandarupalli, Neelima Priyanka Nutulapati, Prof. Dr. P.Suresh Varma "A Novel CPU Scheduling Algorithm–Pre-emptive & Non-Preemptive"[16], in wich is made a mixture of preemtive and nonpreemtive algorithm based on arrival time , wich improve the CPU efficiency in real-time uni-processor-multi-processors operating system[16]. This paper is a real ig deal about the efficiency of CPU. They proposed a new CPU algorithm called Novel CPU Scheduling in wich can implement different algorithm pairs even we can noticed that this proposed algorithm gives almost equal performance like SJF Pre-emtive and Non Pre-emtive algorithm[16].
Another interesting work was a comparison of Priority Scheduling algorithm and \Round Robin one, done by Alban Allkoci, Elona Dhima and Igli Tafa [17] where they concluded that if they use Round Robin, the processes will need more time to finish executing compared with Priority.[17] So this is the big deal that give us reason that if we have ideas to implement on CPU scheduling algorithms, it should implement on Priority

Scheduling to become better and better, to improve all good scheduling characteristics.

Another mathematical well-done work about the response time analysis with preemption thresholds where is calculated blocking time and based on concept of level-I period, they drive the equation for computing the worst case response time of periodic or sporadic tasks in this general model.[17]

3. Theory of experiment

In this part of research we will figure out all the concepts of our work over priority scheduling in situation, preemption and non-preemption scheduling.

3.1 The Environment of Working

As part of working process was clarifying some necessary aspect such as operating system where we do experiment, which is Windows 8 OS. Another thing should noticed is programming language, where I can say that it's more reasonable and easy to me to work with C language. It's more understandable to me, and it is easier to modify.

Something else important to mention is the random and not meaningful name and value of algorithm executing.

4. Priority Scheduling algorithm

As now is so often mention during this research, priority scheduling algorithm arranges processes in queue based on its priority. These processes go to the running state based on this algorithm. Something else that gives priority on running state is preemption. So, below we will give a flowchart of these kind of scheduling.

4.1 Preemptive algorithm

In this kind of algorithm processes that has higher priority occupy the CPU even it is running another on. The flowchart of this part of code be in figure2.

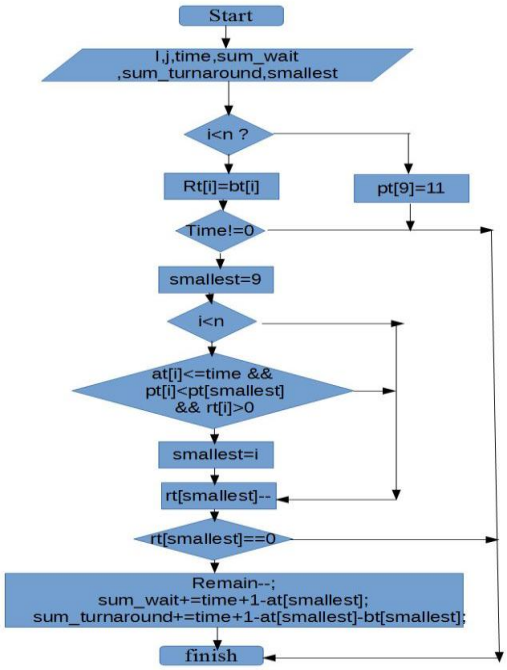


Figure 2: Flowchart of Preemptive scheduling

In this case a "rector" can force us to go out, even if he is a teacher

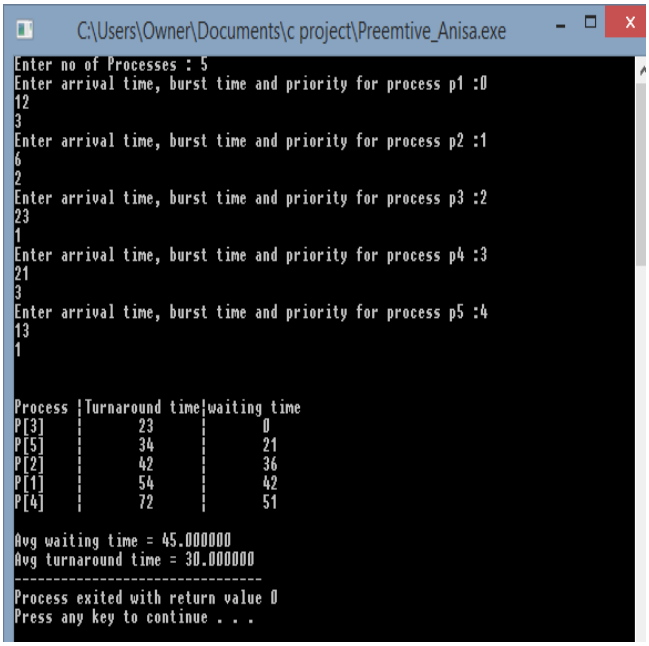


Figure 3: Execution of Preemptive Scheduling

4.2 Non-Preemptive algorithm without aging

In this kind of algorithm processes that has higher priority *can't* occupy the CPU till the process running hasn't finish. The flowchart of this part of code be in figure4.

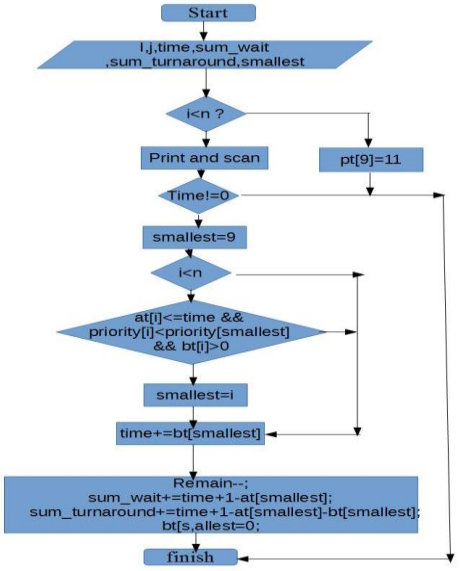


Figure 4: Flowchart of Non-Preemptive Scheduling without Aging

In this case a "rector" *can't* force us to go out, even if he is a teacher, but this situation can drive to starvation. This situation is solved by aging technique. This technique implemented increase of the lowest priority process.

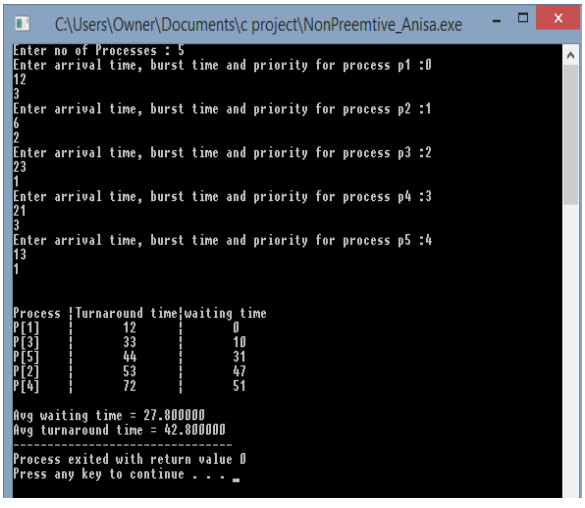


Figure 5: Execution of Non-Preemptive Scheduling

4.3 Non-Preemptive algorithm *with* aging

In this kind of algorithm processes that has higher priority *can't* occupy the CPU till the process running hasn't finish, but the priority of the lowest increase time by time.
In this case I prefer to let for the next research, where I can figure out some interesting about the non-preemptive priority scheduling

For a very first idea about the algorithm with aging technique implemented I suggest to look on a code written on this site [19].

4.4 Comparison of Algorithms

As we can noticed from the test above, it's obvious the average waiting and turnaround time of both of preemptive and non-preemtive algorithms. It's clear that the average waiting time of preemptive algorithm is lower that the nonpreemtive one, and the opposite for the average turnaround time.

5. Conclusion and Future work

As my purpose was to comparison the both preemptive and non-preemtive priority scheduling for the same pairs of processes, I realized it and suggest the best one, and leave the "hole" for a better CPU performance scheduling algorithm. During our project we verify the time waiting and turnaround one, seen in both of algorithms, conclude that waiting time is in favor in non-preemtive algorithm, and turnaround time is in favor in preemptive one.

As the future work I suggest, as I mention above, to take a special care about aging technique and improve it.

Another great topic would be analyzing and suggesting a scheduling algorithm for a bigger system, such as GRID and Cluster, or even bigger. It would be really meaningful to suggest a scheduling algorithm for these. Hope to be part of working group of them.

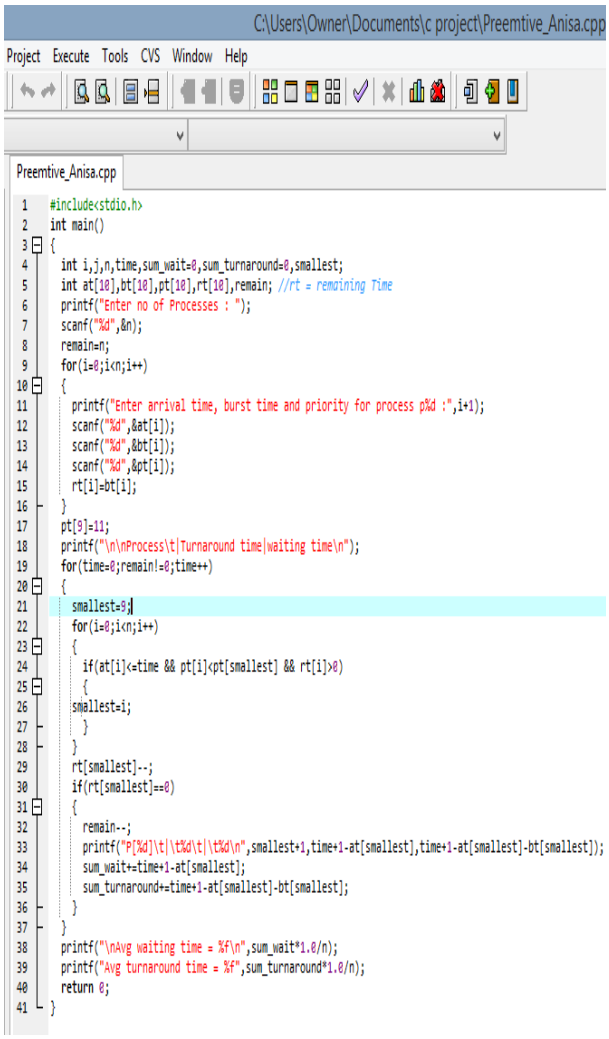
References

[1] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", Sixth Edition.

[2] Milan Milenkovic, "Operating Systems Concepts and Design", McGRAM-HILL, Computer Science Series, second edition.
[4] A. Dhore "Opeating Systems", Technical Publications.
[5] A Novel CPU Scheduling Algorithm– Preemptive & Non-Preemptive- Sukumar Babu Bandarupalli, Neelima Priyanka Nutulapati, Prof. Dr. P.Suresh Varma- ISSN: 2249-6645
[6] Modern Operating Systems – Andrew Tanenbaum
[7] Scheduling Algorithms – Peter Brucker
[8] Scheduling Theory, Algorithms and Systems – Michael Pinedo
[9]http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node124.html
[10]http://cs.nyu.edu/~gottlieb/courses/2000-01-spring/os/chapters/chapter-2.html
[11]http://www.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html
[12] Processor Scheduling - Notes - Operating Systems - Computer Science Now
[13]http://www.eee.metu.edu.tr/~halici/courses/442/Ch2%20Process%20Scheduling.pdf
[14]http://programersparadise.tk/c-program-preemptive-priority-scheduling/
[15] Theory of Scheduling – R. Conway
[16] "A Novel CPU Scheduling Algorithm– Preemptive & Non-Preemptive"- Sukumar Babu Bandarupalli, Neelima Priyanka Nutulapati, Prof. Dr. P.Suresh Varma
[17]Comparing Priority and Round Robin Scheduling Algorithms by Alban Allkoci, Elona Dhima, Igli Tafa.
[18] http://programersparadise.tk/c-program-for-non-preemptive-priority-scheduling-program-in-c/
[19]http://electrofriends.com/source-codes/software-programs/c/c-advanced-programs/priority-schedueling-preemtion-aging-arrival-times/

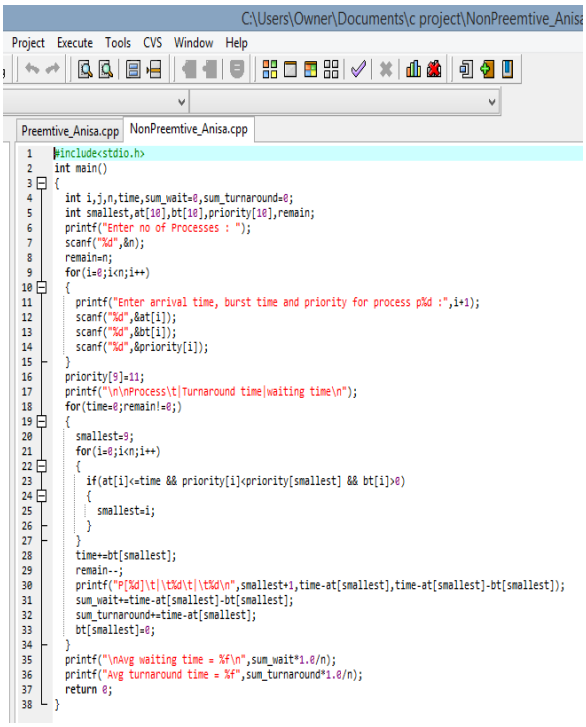
Appendix

Source code of Preemptive Priority Scheduling [14]




```
#include<stdio.h>
int main()
{
  int i,j,n,time,sum_wait=0,sum_turnaround=0,smallest;
  int at[10],bt[10],pt[10],rt[10],remain; //rt = remaining Time
  printf("Enter no of Processes : ");
  scanf("%d",&n);
  remain=n;
  for(i=0;i<n;i++)
  {
    printf("Enter arrival time, burst time and priority for process p%d :",i+1);
    scanf("%d",&at[i]);
    scanf("%d",&bt[i]);
    scanf("%d",&pt[i]);
    rt[i]=bt[i];
  }
  pt[9]=11;
  printf("\n\nProcess\t|Turnaround time|waiting time\n");
  for(time=0;remain!=0;time++)
  {
    smallest=9;
    for(i=0;i<n;i++)
    {
      if(at[i]<=time && pt[i]<pt[smallest] && rt[i]>0)
      {
    smallest=i;
      }
    }
    rt[smallest]--;
    if(rt[smallest]==0)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",smallest+1,time+1-at[smallest],time+1-at[smallest]-bt[smallest]);
      sum_wait+=time+1-at[smallest];
      sum_turnaround+=time+1-at[smallest]-bt[smallest];
    }
  }
  printf("\nAvg waiting time = %f\n",sum_wait*1.0/n);
  printf("Avg turnaround time = %f",sum_turnaround*1.0/n);
  return 0;
}
```

Source code of Non-Preemptive Priority Scheduling [18]




```
#include<stdio.h>
int main()
{
  int i,j,n,time,sum_wait=0,sum_turnaround=0;
  int smallest,at[10],bt[10],priority[10],remain;
  printf("Enter no of Processes : ");
  scanf("%d",&n);
  remain=n;
  for(i=0;i<n;i++)
  {
    printf("Enter arrival time, burst time and priority for process p%d :",i+1);
    scanf("%d",&at[i]);
    scanf("%d",&bt[i]);
    scanf("%d",&priority[i]);
  }
  priority[9]=11;
  printf("\n\nProcess\t|Turnaround time|waiting time\n");
  for(time=0;remain!=0;)
  {
    smallest=9;
    for(i=0;i<n;i++)
    {
      if(at[i]<=time && priority[i]<priority[smallest] && bt[i]>0)
      {
        smallest=i;
      }
    }
    time++bt[smallest];
    remain--;
    printf("P[%d]\t|\t%d\t|\t%d\n",smallest+1,time-at[smallest],time-at[smallest]-bt[smallest]);
    sum_wait+=time-at[smallest]-bt[smallest];
    sum_turnaround+=time-at[smallest];
    bt[smallest]=0;
  }
  printf("\nAvg waiting time = %f\n",sum_wait*1.0/n);
  printf("Avg turnaround time = %f",sum_turnaround*1.0/n);
  return 0;
}
```