International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 9, pp: 264-269 www.ijaem.net ISSN: 2395-5252

Context Switching On High Priority Process

Srinithi B, SnehaM, Harshith S, Dr.M. Sujithra M.C.A, M.Phil., PhD,Dr.A.D. Chitra M.C.A, M.Phil., PhD,

2nd Year, M.Sc. Software Systems (Integrated), Coimbatore Institute of Technology,Coimbatore .
Assistant Professor, AssistantProfessor, Department of Data Science,Department of Software Systems, Coimbatore Institute of Technology,Coimbatore Institute of Technology,Coimbatore Coimbatore

Date of Submission: 10-11-2020 Date of Acceptance: 24-11-2020

ABSTRACT: Time is needed to spare the setting of one process that is in the running state and afterward getting the setting of another process that is going to come in the running state . during that time there is no valuable work done by the CPU from the client point of view . thus , setting exchanging is unadulterated overhead in this condition . time taken for changing from one process to other is unadulterated over head . since the framework accomplishes no helpful work while exchanging . so this paper illustrates to go for stringing(threads) at whatever point conceived .

keywords:Process Control Block (PCB),Control Processing Unit (CPU)

I. INTRODUCTION

What is process in operating system ?Process is a variables program in execution including the current values the program counter, registers and variables . the difference between a process the programmers that the program is a group of instructions where is the process is the activity When a program is loaded into the memory and it becomes a process , it can be divided into 4 section-stack ,text and data .

- Stack- the process that contains the temporary data
- Heap – this is dynamically allocated memory to a process
- Text-this includes the current activity represented by the value of program counter and contents of the processor register
- Data-this segment contains the all worldwide and static factors

PROCESS MODEL TECHNIQUES :
PROCESSES :

The framework it is lot simple to consider an assortment of cycles running in (pseudo) equal at that point to attempt to monitor how the CPU switch is from program to program

Types of process:
Predominately , there are three types of process model

Multi programming

We have single mutual processor by all program there is just one program counter all projects in memory .in this program, counter is introduced to execute the piece of a program A then with the assistance of processwhich it move

Multiprocessing

Each with its own progression of control and every when running freely of different ones there is just a single actual program counter .So when each process runs its legitimate program counter is stacked into the genuine program counter

One program at one time :

At some random moment just one processpursue and different processsome time period in other perspective all cycles progress yet just process really runs at given moment of time

PROCESS STATES :
When all is said done , a process can have each of the accompanying five states in turn

- Start – this is underlying state when a processis first begun / made
- Ready – the processis waiting to be allotted to a processor .prepared process are holding back to have processor designated to them by the workers framework with goal that they can run process
- Running – after ready express, the processstate is set to running and the processor executes its guidance
- Waiting – process moves into the holding up the state in the event that it needs to sit tight for an assets for example : sitting tight for client info , or trusting that a record will open up

- Terminated or exit : once the processcompletes its execution , or it is ended by the working framework it is moved to the ended state where it is holds back to be taken out from primary memory

PROCESS CONTROL BLOCK:

What is process control block ? It is the information structure kept up by the working framework for each processSome operating systems keep up just PCB in that PCB has all the passages put away by a process table The process control block keeps all the data expected to monitor a process

IMPLEMENATION OF PROCESSES:

Process model is actualized by process table and process control block which keep track all data of process.at the hour of formation of another process working framework distributes a memory for it stack a process code in the allotted memory and arrangement information space for it The condition of processis put away as new in its PCB and when this process move to prepared express its states is additionally changes PCB .at the point when a running processneeds to hang tight for an info yield gadgets its state is changed to hinderd the different lines utilized as connected rundown

EVOLUTION OF PROCESS STATE MODEL:

In the early working frameworks there were no multi tasks, when a process execution was begun, it utilized the processor until it was done thus those frameworks didn't require any process running state. In the recent working frameworks that performs various tasks and cycles can execute interleaved, any process in the course of its life could have unique running states

Two-State Model:

In this model, that is the essential model in working frameworks, a process is in the RUNNING state (Process currently executing) or NOT RUNNING state (Process waiting for execution). At the point when a process is first made by the OS, it instates the program control block for the process and the new process enters the framework in Not-running state. After some time, the right now running process will be hindered by certain functions, and the OS will move the presently running process from Running state to Not-running state. The dispatcher at that point chooses one process from Not-running cycles and moves the process to the Running state for execution

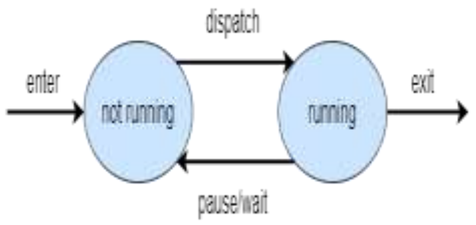

Fig.1 : State Transition

Three-State Model:

There is one significant disadvantage of two state measure model. At the point when dispatcher brings another process from not-running state to running state, the process may in any case be hanging tight for some function or I/O demand. Along these lines, the dispatcher must cross the line and discover a not-running process that is prepared for execution. It can corrupt execution . To defeat this issue, we split the not-running state into two states: Ready State and Waiting (Blocked) State.

Ready State: The process in the main memory that is prepared for execution.

Waiting or Blocked State: The process in the main memory that is hanging tight for some function.

The OS keeps up a different queue for both Ready State and Waiting State. A process moves from Waiting State to Ready State once the function it's been sitting tight for finishes.

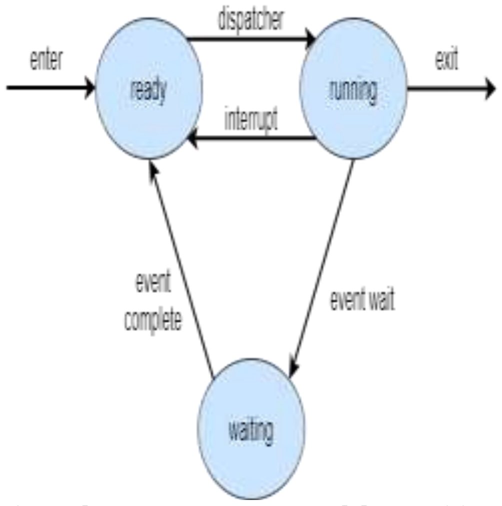

Fig.2:Three State Process Model Transition

Five-State Model:

- **New** : A process has been created but has not yet been admitted to the pool of executable processes.
- **Ready :** Processes that are prepared to run if given an opportunity. That is, they are not waiting on anything except the CPU availability.

- **Running:** The process that is currently being executed. (Assume single processor for simplicity.)
  Logically, the 'Running' and 'Ready' states are similar. In both cases the process is willing to run, only in the case of 'Ready' state, there is temporarily no CPU available for it.
- **Blocked :** A process that cannot execute until a specified event such as an IO completion occurs.
- **Exit:** A process that has been released by OS either after normal termination or after abnormal termination (error).

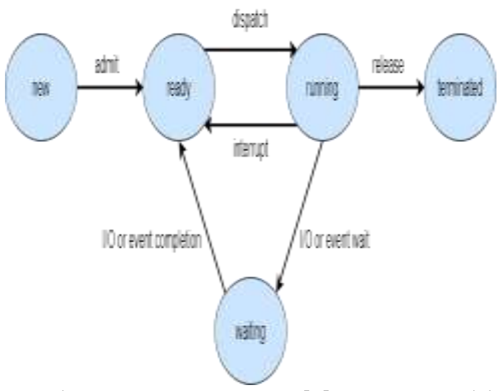

Fig.3:Five-State Process Model State Transition

Possible State Transitions are:
Null→ New: A new process is created to execute a program.
New→ Ready: The OS will move a process from the New state to the Ready state when it is prepared to take on an additional process.
Ready → Running: When it is time to select a process to run, the OS chooses one of the processes in the Ready state. This is the job of the scheduler or dispatcher.
Running → Exit: The currently running process is terminated by the OS if the process indicates that it has completed or if it aborts.
Running → Ready: The most common reason for this transition is that the running process has reached the maximum allowable time for uninterrupted Execution
Running → Blocked: A process is put in the Blocked state if it requests something for which it must wait.
Blocked → Ready: A process in the Blocked state is moved to the Ready state when the event for which it has been waiting occurs.
Ready → Exit: For clarity, this transition is not shown on the state diagram. In some systems, a parent may terminate a child process at any time. Also, if a parent terminates, all child processes associated with that parent may be terminated.
Blocked → Exit: The comments under the preceding item apply.

Six State Process Model:
There is one significant defect in the five-state model. As we probably are aware, that the processor is lot quicker than I/O gadgets. Accordingly, a circumstance may happen where the processor executes so quick that the entirety of the cycles moves to waiting state and no process is in ready state. The CPU sits inert until, at least one process completes the I/O activity. This prompts low CPU utilizationCPU would now be able to acquire some different process in the primary memory. There are two choices. First is to bring a fresh new process and the subsequent choice is to take another process from the suspended queue back to the main memory. Bringing a process from the suspended queue is mostly preferred.

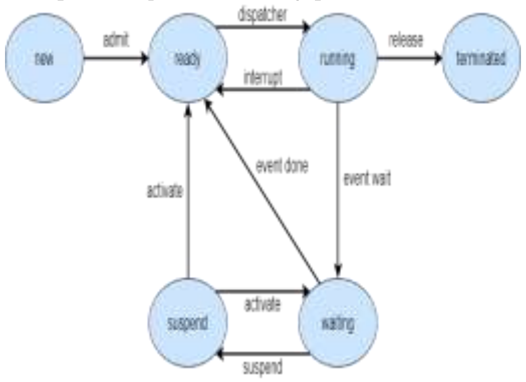

Fig.4: Six State Process Model Transition

State Transitions:
Waiting -> Suspend: In this transition, if all the processes are in waiting state the process is moved from waiting state to suspended state by OS .
Suspend -> Ready: In this transition, where the process is in the suspended state moves back to the main memory for execution when there is sufficient memory available.
Suspend -> Waiting: Here, the os which moves the process from secondary memory to main memory will be waiting for some event.
Seven State Process Model:
It is known as Five state process model with two suspended states:
Blocked/Suspend : The process is in secondary memory and awaiting an event.
Ready/Suspend : The process is in secondary memory but is available for execution as soon as it is loaded into main memory.

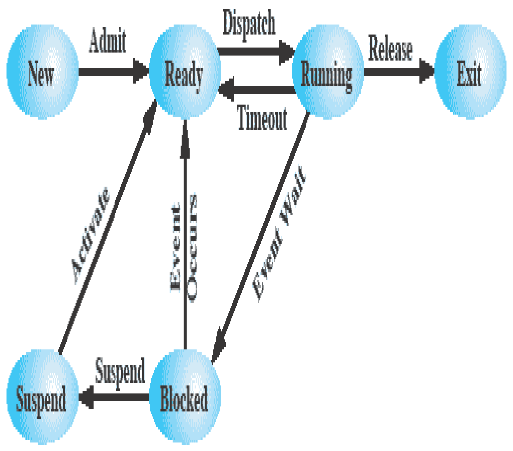

Fig.5 a.) With one suspended state

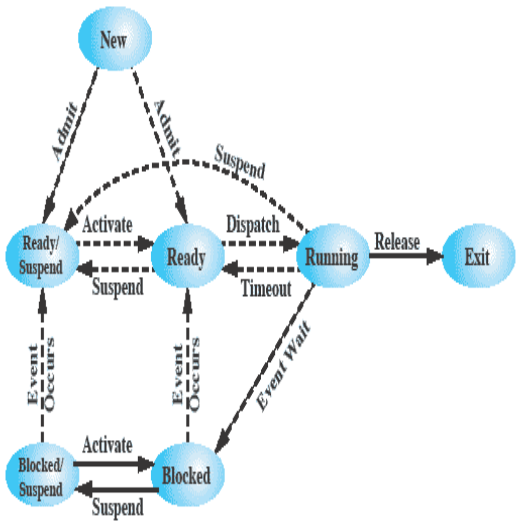

Fig.5b.) With two suspended state

State Transitions
Waiting -> Blocked/Suspend
Blocked/Suspend -> Waiting
Blocked/Suspend -> Ready/Suspend
Ready/Suspend -> Ready
Ready-> Ready/Suspend
New -> Ready/Suspend

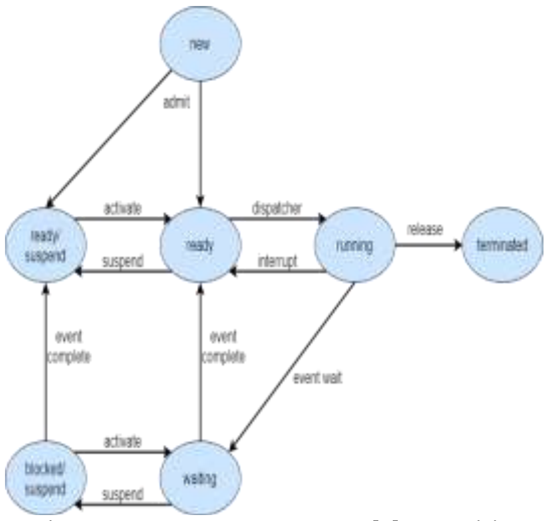

Fig.6:Seven State Process Model Transition

II. RESULT OBTAINED :

ISSUE: The disservice of context switching
The disadvantages of setting exchanging is that it requires some an ideal opportunity for setting exchanging for example the setting exchanging time .
ARRANGEMENT : Threads

PROPOSED SOLUTION :
Following are some reasons why we use threads in designing operating systems.

- A process with multiple threads make a great server for example printer server.
- Because threads can share common data, they do not need to use inter -process communication.
- Because of the very nature, threads can take advantage of multiprocessors.
- Threads are cheap in the sense that They only need a stack and storage for registers therefore, threads are cheap to create.
- Threads use very little resources of an operating system in which they are working. That is, threads do not need new address space, global data, program code or operating system resources.
- Context switching are fast when working with threads. The reason is that we only have to save and/or restore PC, SP and registers.

What are threads? the program is isolated into various understandings utilizing strings . these assignments give the hallucination of running equal however are completed consistently . string has a program counter , registers and stack . the program counter has the data of the apparent multitude of errands and the circumstance , registers , spare the current working factors and stack stores the historical back drop of execution .From the multiple points of view string work similarly as that of process . a portion of the likeness and contrasts are :

Similitudes :

Like process threads share CPU and just one string dynamic (running) at a time .threads inside a process execute successively . Similar to measure , in the event that one string is impeded another string can run .

Contrasts

In contrasts, threads are not independent of one another . all threads can access every address in the task . threads are designed to assist one other . note that process might or might not assist one another because process may originate from different users

Following are some reasons why we use threads in designing operating systems.

- A process with multiple threads make a great server for example printer server.
- Because threads can share common data, they do not need to use inter -process communication.
- Because of the very nature, threads can take advantage of multiprocessors.
- Threads are cheap in the sense thatThey only need a stack and storage for registers therefore, threads are cheap to create.
- Threads use very little resources of an operating system in which they are working. That is, threads do not need new address space, global data, program code or operating system resources.
- Context switching are fast when working with threads. The reason is that we only have to save and/or restore PC, SP and registers.

III. CONCLUSION:

A question you might ask is whether processes or threads can run at the same time. The answer is: it depends. On a system with multiple processors or CPU cores (as is commonwith modern processors), multiple processes or threads can be executed in parallel. On asingle processor, though, it is not possible to have processes or threads truly executing atthe same time. In this case, the CPU is shared among running processes or threads using aprocess scheduling algorithm that divides the CPU's time and yields the illusion of parallelexecution. The time given to each task is called a "time slice." The switching back and forthbetween tasks happens so fast it is usually not perceptible. The terms parallelism (genuines i multaneousexecution) and concurrency (interleaving of processes in time to give the appearance of simultaneous execution), distinguish between the two types of real orapproximate simultaneous operation.

IV. FUTURE SCOPE :

Today's operating systems are conceptually upside-down. They developed the hard way, gradually struggling upwards from the machinery (processors, memory, disks and displays) toward the user. In the future, operating systems and information management tools will grow top-down.

When we think of the future, we think of the IoT, big data, AI, machine learning, additive manufacturing, and robotics. We're pushing forward with these new technologies, but there's going to be a need for convergence.

If we want a future with autonomous vehicles moving in harmony, drone deliveries, and refrigerators that purchase milk when we run out, then the systems we put in place need to communicate and operate together efficiently.

Smart trash cans are also a perfect example of one of the fears people have about this type of automated future—the potential for lost jobs and careers. Once trucks are automated, will we need human drivers for trash removal? For taxis?

Do we really want systems that are perfectly automated, that don't allow for any sort of flexibility?

I'm not advocating for people to cheat the system all the time, but we like being able to bend rules and make exceptions. Is there a local store that you visit every week where they know your face? Well, imagine you go there next week but forget your wallet. They'll probably tell you, "Don't worry about it. Come back with the money later."

REFERENCES:

[1]. Kartik Pandya, "Network Structure or Topology", International Journal of Advance Research in Computer Science and Management Studies, Volume 1, Issue

[2]. July 2013 2. Dhananjay M.DhamDhere, "Operating Systems A Concept – Based Approach", 3rdEdition, McGraw Hill Education (India) Private Limited, New Delhi, 2003

[3]. David P. Reed and Rajendra K. Kanodia. Synchronization with eventcounts and sequencers. Communications of the ACM,

22(2), February 1979. Proposes different synchronization mechanism.
[4]. K.A.Sumitradevi, N. P. Banashree, “Operating Systems”, second edition, SPD publications.
[5]. Brian P. Crow, Jeong Geun Kim, Prescott T. Sakai,” IEEE 802.11 Wireless Local Area Networks”, IEEE Communications Magazine, September 1997.
[6]. William Stallings, “Operating Systems: Internals and Design Principles”, seventh Edition, Pearson Publications.