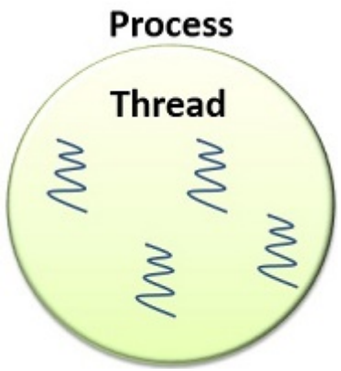
Difference Between Process and Thread in Java

July 6, 2016 — 1 Comment



Java provides the concept of multitasking, which allows more than two processes to run concurrently, and allows more than two threads to run concurrently. The main difference between process and thread is that a process is a program in execution whereas, a thread is part of that running process. Process and thread share a relationship where a process provides an environment for the execution of the thread. A process can contain multiple threads. There are some other differences between process and thread which are discussed in the comparison chart.

Search the site ...

TOP 10 DIFFERENCES

Difference Between Logical and Physical Address in Operating System

Difference Between Preemptive and Non-Preemptive Scheduling in OS

Difference between Synchronous and Asynchronous Transmission

Difference Between Paging and Segmentation in OS

Difference Between Internal and External fragmentation

Difference Between while and do-while Loop

Difference Between LAN, MAN and WAN

Content: Process Vs Thread in Java

Comparison Chart

BASIS FOR COMPARISON	PROCESS	THREAD
Basic	An executing program is called a process.	A thread is a small part of a process.
Address Space	Every process has its separate address space.	All the threads of a process share the same address space cooperatively as that of a process.
Multitasking	Process-based multitasking allows a computer to run two or more than two programs concurrently.	Thread-based multitasking allows a single program to run two or more threads concurrently.

Difference Between Pure ALOHA and Slotted ALOHA

Difference Between Recursion and Iteration

Difference Between Go-Back-N and Selective Repeat Protocol

RECENT ADDITION

Difference between CGI and Servlet

Difference between XML and HTML

Difference between Syntax and Semantics

Difference between Cellpadding and Cellspacing

Difference between GET and POST Method in HTML

CATEGORIES

DBMS

Internet

Networking

BASIS FOR COMPARISON	PROCESS	THREAD
Communication	Communication between two processes is expensive and limited.	Communication between two threads is less expensive as compared to process.
Switching	Context switching from one process to another process is expensive.	Context switching from one thread to another thread is less expensive as compared to process.
Components	A process has its own address space, global variables, signal handlers, open files, child processes, accounting information.	A thread has its own register, state, stack, program counter.
Substitute	Process are also called heavyweight task.	Thread are also called lightweight task.
Control	Process-based multitasking is not under the control of Java.	Thread-based multitasking is under the control of Java.

BASIS FOR COMPARISON	PROCESS	THREAD
Example	You are working on text editor it refers to the execution of a process.	You are printing a file from text editor while working on it that resembles the execution of a thread in the process.

Definition of Process

An executing program itself is called a process. It can be explained with an example say when you click on any application, and it starts we can say that a process has been started. Every process has its own address space hence, they do not overlap each other. In a process-based multitasking, more than two processes can run simultaneously on the computer. A process can contain multiple threads, where every thread is responding different request of the user.

Process are also called heavyweight task. Process-based multitasking leads to more overhead. Inter-process communication is very expensive is also limited. Switching from one process to also expensive. Java does not control the process based multitasking.

Definition of Thread

The threads are extensively used in Java-enabled browsers. A thread is a part of a process. A process can contain multiple threads. These multiple threads in a

process share the same address space of the process. Each thread has its own stack and register to operate on. Let us see how a thread is created. There are two methods to create a thread. **First**, you have to define a class that must extend the "**Thread**" class and further it must override the run( ) method of the Thread class.

```
class A extends Thread{
public void run( ){
for(int i = 5; i > 0; i--){
System.out.println("thread of class A is running: i=",+i);
}
}
class B extends Thread{
public void run( ){
for(int i = 5; i > 0; i--){
System.out.println("thread of class B is running: j=",+j);
}
}
public static void main(String args[ ]){
A a1=new A( );
a1.start( );
b1.start( );
}
}

//Output
thread of class A is running: i= 5
thread of class A is running: i= 4
thread of class A is running: j= 5
thread of class A is running: i= 3
thread of class A is running: i= 2
thread of class A is running: j= 4
thread of class A is running: j= 3
thread of class A is running: i= 1
thread of class A is running: j= 2
thread of class A is running: j= 1
```

Second, you can also implement the **Runnable** interface that contains only the run( ) method, that is to be defined by the implementing class.

```
class A implements Runnable{
public void run(){
System.out.println("thread A is running");
}

public static void main(String args[]){
A a1=new A( );
Thread t1 =new Thread(a1);
t1.start();
}
}
//Output
thread A is running
```

The run( ) method is defined in such a way to perform the action as the user wants. The start( ) method is used to invoke the run( ) method. Which method to apply is based upon what type of class we are creating. If a class is already extending another class, then it cannot extend the Thread class as in Java a class can not extend two classes then, no choice it has to implement the Runnable interface.

Threads are also called lightweight task. The overhead of the thread-based multitasking is very low. The cost of inter-thread communication and context switching from one thread to another thread is very low. Multithreaded multitasking is under the control of Java.

Key Differences Between Process and Thread in Java

1. A process is an executing program whereas, the thread is a small part of a process.
2. Each process has its own address space whereas, the threads of the same process share the address space as that of the process.
3. In process based multitasking, more than two processes can run at the same time whereas, in thread-based multitasking, more than two thread can run at the same time.
4. Inter-process communication between two processes is costlier than inter-thread communication.
5. Context switching between two processes is expensive and limited as compared to context switching between two threads.
6. A process is also called the heavyweight task whereas, the thread is called lightweight task.
7. Multitasking over a process is not under the control of Java whereas, the Multitasking over multithreading is under the control of Java.
8. Components contained by a process its own address space, global variables, signal handlers, open files, child processes, accounting information. On the other hand, a thread contains its own register, state, stack, program counter.

Conclusion:
If we look at the process and thread, the thread is a piece of code which requires the environment of the process to run smoothly.

Related Differences:

1. **Difference Between Multitasking and Multithreading in OS**

2. **Difference Between Circuit Switching and Packet Switching**
3. **Difference Between Multiprocessing and Multithreading**
4. **Difference between Process and Thread**
5. **Difference between Thread Class and Runnable Interface in Java**

Filed Under: Programming

Comments

saud sheikh says
May 6, 2017 at 2:13 pm

I have always got the proper differences from this website ….
Superb work…
Keep it on.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT