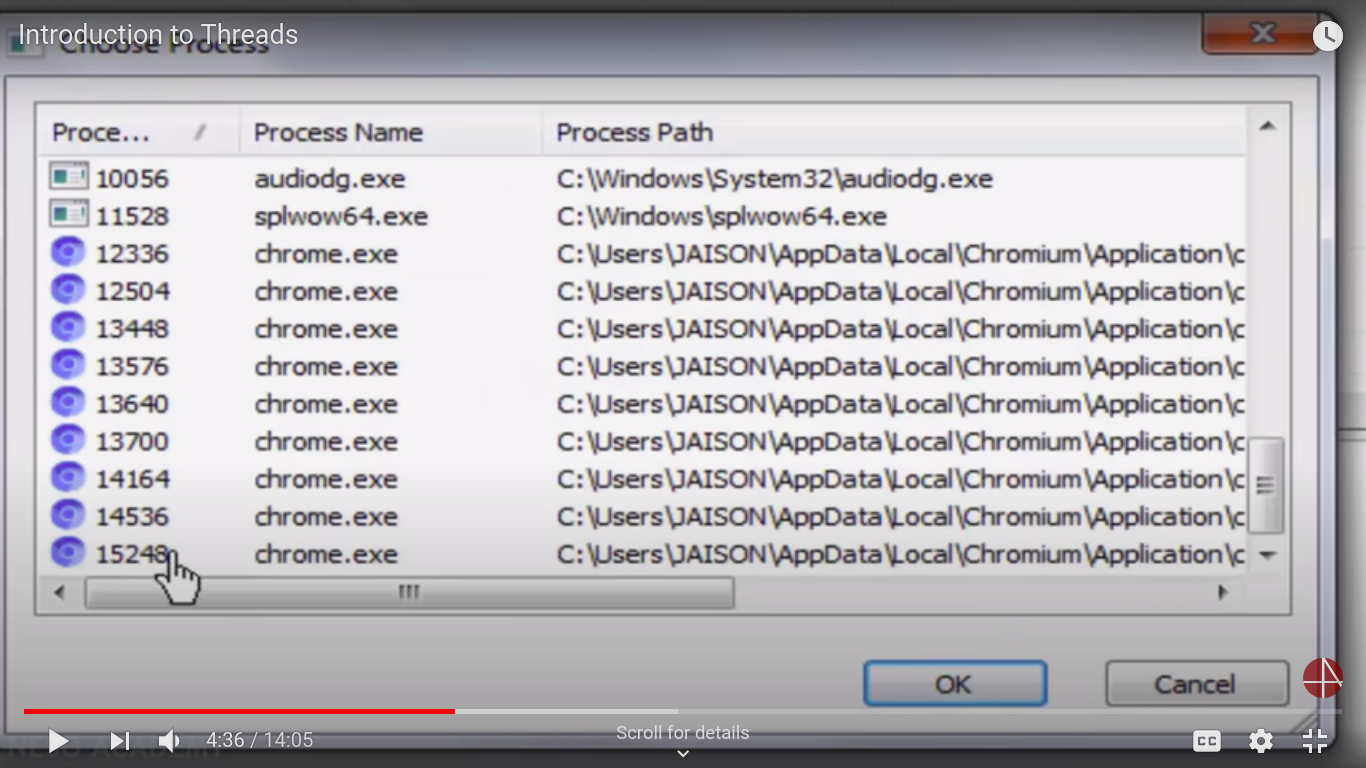
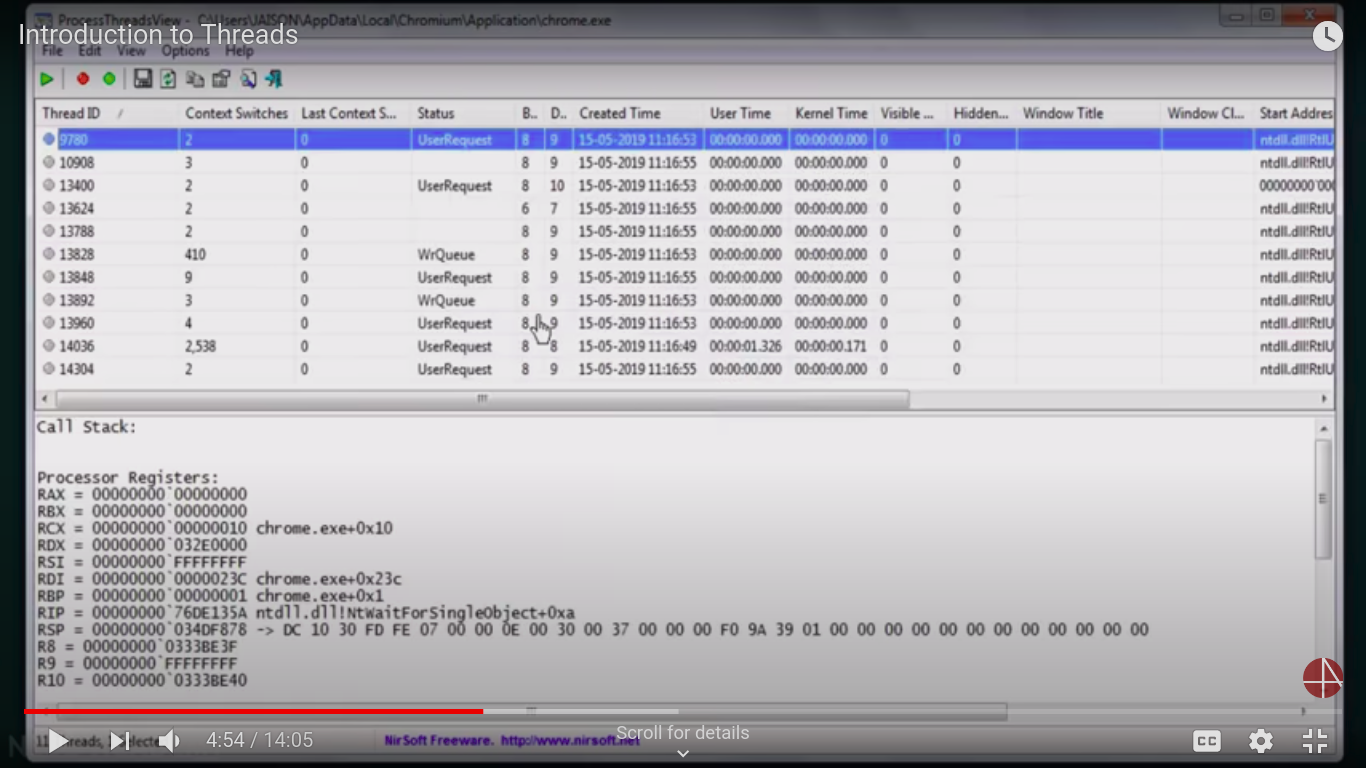
**Treading:**

Program under execution is known as **process** and thread is a basic unit of execution.

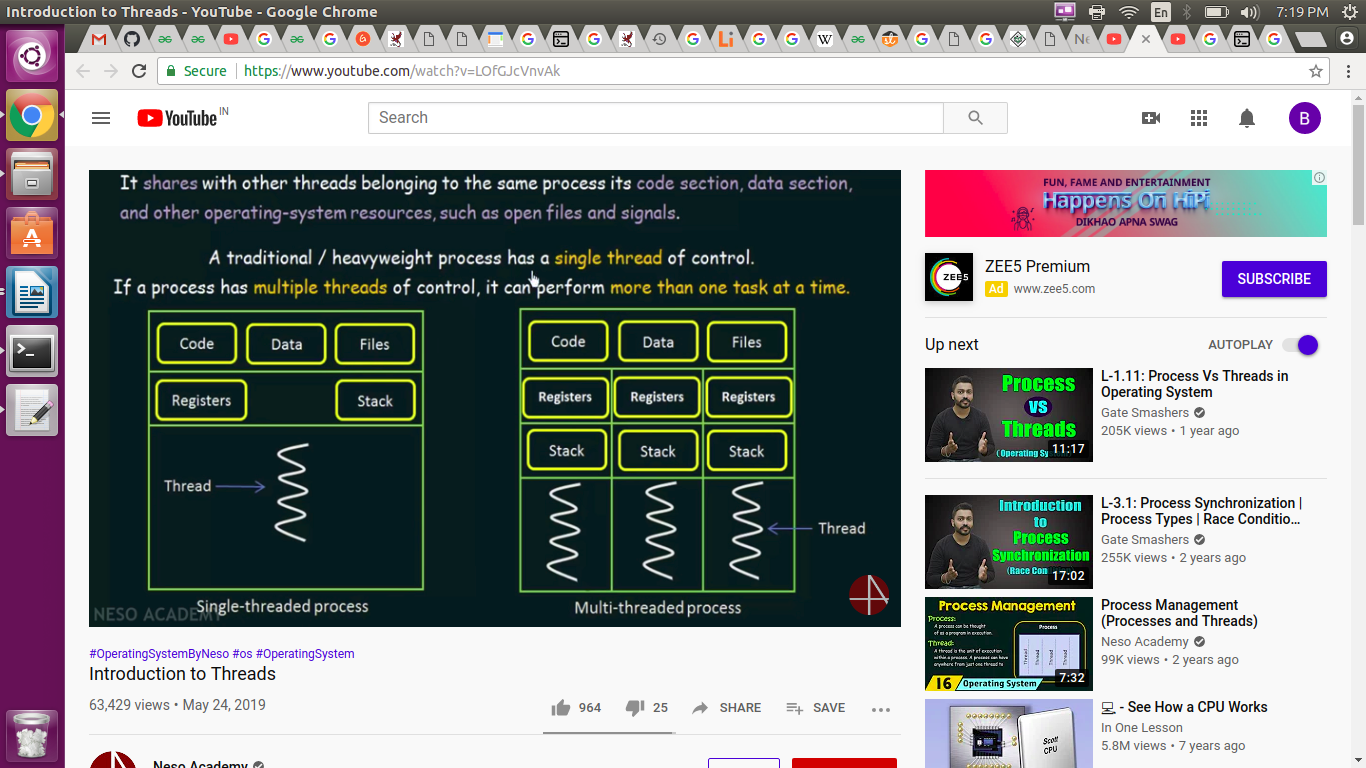




Ex: One thread using for display window and another thread download from internet.

Each program may have number of processes associated with it and each process can have number of threads executing in it.

Threads shares with other threads belonging to the same process its code section, data section and other operating system resources. If a process has single thread it will perform single task, if a process has multiple threads of control, it can perform more than one task at a time.



**Single threaded proess:**

The block considered as process. The code, data, files, registers and stack belonging to the single thread process. This perform only one task at a time.

**Multi threaded proess:**

It a single process contain multiple threads and each of thread has own stack and register. Code, data section and files belonging to these process are shared by these threads.

Multi thread process much more efficient than single thread process it make computation faster and efficient.

Stack and registers can’t be shared among the threads. Each thread has its own stack and registers.

A register may hold an instruction, a storage address, or other kind of data needed by the process.

The threads has share resources like code, data, and files among all threads within a process.

## **Thread Scheduling:**

## When a thread is created, it inherits its priority from the thread that created it. You also can modify a thread's priority at any time after its creation by using the setPrioritymethod.

## Thread priorities are integers ranging between MIN\_PRIORITY and MAX\_PRIORITY. The higher the integer, the higher the priority.

## If two threads of the same priority are waiting for the CPU, the scheduler arbitrarily chooses one of them to run.

**Process control block (PCB):**

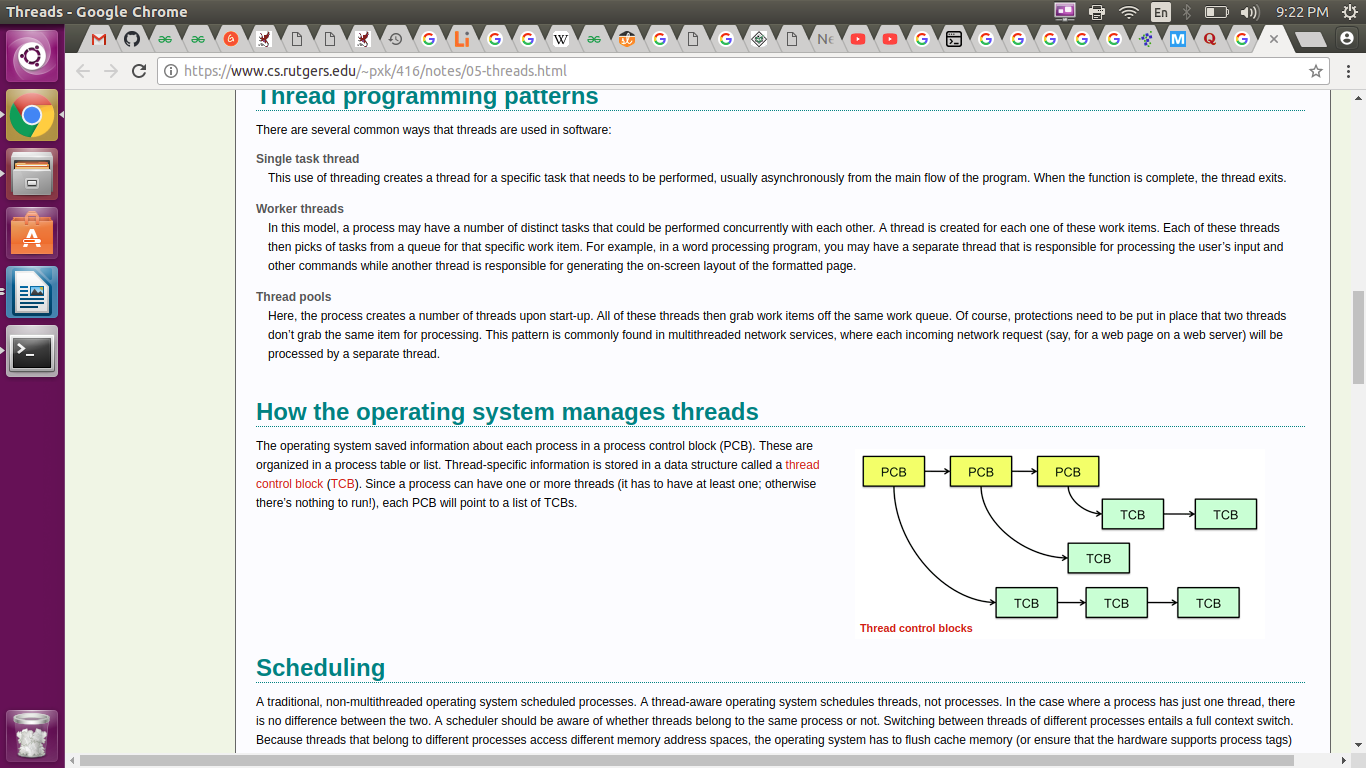
An operating system had to keep track of processes, and stored its per-process information in a data structure called a process control block (PCB).

The items that the operating system must store that are unique to each thread are:

* Thread ID
* Saved registers, stack pointer, instruction pointer
* Stack (local variables, temporary variables, return addresses)
* Signal mask
* Priority (scheduling information)

# **How the operating system manages threads:**

The operating system saved information about each process in a process control block (PCB). These are organized in a process table or list. Thread-specific information is stored in a data structure called a thread control block(TCB), each PCB will point to a list of TCBs.



The items that are shared among threads within a process are:

* Text segment (instructions)
* Data segment (static and global data)
* BSS segment (uninitialized data)
* Open file descriptors
* Signals
* Current working directory
* User and group IDs

### Multi-threading Models:

### Types of relation between user thread and kerne thread.

### There are two types of threads to be managed in a system.

### User threads : Supported above the kernel and are managed without kernel

### support. These are threads operated in user level which are

### created by users.

### 

### Kernel threads : Supported and managed directly by the operating system.

### 1. Many-To-One Model:

### Many user threads are accessing one kernel thread.

### Thread management is done by the thread library in user space.

### 

**Limitation:**

The entire process will block if a thread makes a blocking system call.

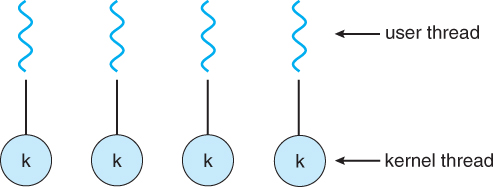
Because only one thread can access the kernel at a time, multiple threads are unable to run parallel on multi processors.

**2. One-To-One Model:**

One user thread map to one kernel thread.

Provides more concurrent than Many-To-One Model by allowing another thread to run when a thread makes a blocking system call.

Also multiple threads run parallel on multi processors.



**Limitations:**

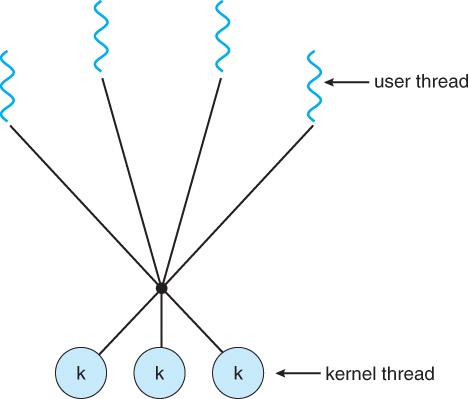
Because of creating creating number of kernel threads can burden the performance of application.

The implementation of this model restrict the number of threads supported by the system.

**3. Many-To-Many Model:**

Many user threads map to many kernel threads.

In this model number of user threads map to smaller or equal number of kernel threads.



The number of kernel threads may be specific to either a particular application or a particular machine.

Developers can create as many user threads as necessary and the corresponding kernel threads can run in parallel on a multi-processors.

Also when a thread performs a blocking system call, the kernel can schedule another thread for execution.

**Benefits of multi thread :**

**Resource Sharing:**

All the threads of a process share its resources such as memory, data, files etc. A single application can have different threads within the same address space using resource sharing.

**Responsiveness:**

Program responsiveness allows a program to run even if part of it is blocked using multithreading. This can also be done if the process is performing a lengthy operation. For example - A web browser with multithreading can use one thread for user contact and another for image loading at the same time.

**Utilization of Multiprocessor Architecture:**

In a multiprocessor architecture, each thread can run on a different processor in parallel using multithreading. This increases concurrency of the system. This is in direct contrast to a single processor system, where only one process or thread can run on a processor at a time.

**Economy:**

It is more economical to use threads as they share the process resources. Comparatively, it is more expensive and time-consuming to create processes as they require more memory and resources. The overhead for process creation and management is much higher than thread creation and management.