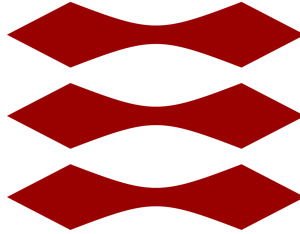

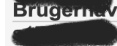


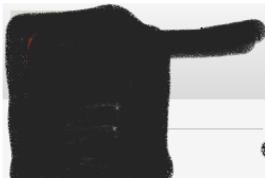
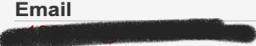




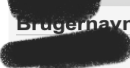
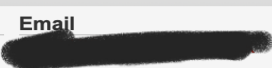

# DTU



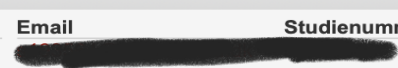
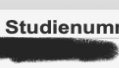






62588- Operating System  
Opgave 2 -Windows Operating system  
Gruppe 16

				^
<b>Brugernavn</b>	<b>Email</b>	<b>Studienummer</b>	<b>Uddannelse</b>	
			diploming. Softwaretek.	

				^
	<b>Email</b>	<b>Studienummer</b>	<b>Uddannelse</b>	
			diploming. Softwaretek.	

				^
<b>Brugernavn</b>	<b>Email</b>	<b>Studienummer</b>	<b>Uddannelse</b>	
			diploming. Softwaretek.	

				^
<b>Brugernavn</b>	<b>Email</b>	<b>Studienummer</b>	<b>Uddannelse</b>	
			diploming. Softwaretek.	

				^
<b>Brugernavn</b>	<b>Email</b>	<b>Studienummer</b>	<b>Uddannelse</b>	
			diploming. Softwaretek.	

The pictures are blacked due to the law of the statement of consent. I and the other authors are the contributors to this report. Enjoy the report.

# Indholdsfortegnelse

<b>How does threads work in Windows</b>	<b>3</b>
Different types of Thread	4
User - Level Thread	4
Kernel - Level Thread	4
<b>Object in Windows</b>	<b>5</b>
<b>Windows thread States</b>	<b>5</b>
<b>Multithreading</b>	<b>6</b>
<b>Relationship between threads and process</b>	<b>7</b>
<b>Process vs Thread</b>	<b>7</b>
<b>How does IPC work in Windows</b>	<b>9</b>
Clipboard	9
COM	9
Data Copy	9
DDE	10
File Mapping	10
Mailslots	10
Pipes	10
RPC	11
Windows sockets	11
<b>Sources</b>	<b>12</b>

# How does threads work in Windows

## Introduction

Thread is an execution unit that consists of its own program counter, a stack, and a set of registers where the program counter mainly keeps track of which instruction to execute next, a set of registers mainly hold its current working variables, and a stack mainly contains the history of execution

Threads are also known as **Lightweight processes**. Threads are a popular way to improve the performance of an application through parallelism. Threads are mainly used to represent a software approach in order to improve the performance of an operating system just by reducing the overhead thread that is mainly equivalent to a classical process.

The CPU switches rapidly back and forth among the threads giving the illusion that the threads are running in parallel. As each thread has its own independent resource for process execution; thus Multiple processes can be executed parallelly by increasing the number of threads.

It is important to note here that each thread belongs to exactly one process and outside a process no threads exist. Each thread basically represents the flow of control separately. In the implementation of network servers and web servers threads have been successfully used. Threads provide a suitable foundation for the parallel execution of applications on shared-memory multiprocessors.

A *process* is a software program that is currently running in Windows. Every process has an ID, a number that identifies it. A *thread* is an object that identifies which part of the program is running. Each thread has an ID, a number that identifies it.

A process may have more than one thread. The purpose of a thread is to allocate processor time. On a machine with one processor, more than one thread can be allocated, but only one thread can run at a time. Each thread only runs a short time and then the execution is passed on to the next thread, giving the user the illusion that more than one thing is happening at once. On a machine with more than one processor, true multithreading can take place. If an application has multiple threads, the threads can run simultaneously on different processors.

The Windows kernel-mode process and thread manager handles the execution of all threads in a process. Whether you have one processor or more, great care must be taken in driver programming to make sure that all threads of your process are designed so that no matter what order the threads are handled, your driver will operate properly.

If threads from different processes attempt to use the same resource at the same time, problems can occur. Windows provides several techniques to avoid this problem. The technique of making sure that threads from different processes do not touch the same resource is called *synchronization*.

## Different types of Thread

- **User - Level Thread**

These threads are implemented and used in the user library. They cannot be created using the system. While doing thread switching, if the OS is called there will be distractions. The user thread avoids all distractions and does not interrupt the kernel system. These threads are considered as single-threaded processes by the kernel. These are implemented on the systems that do not support multithreading as it is a single thread process. These are simply represented with a single, register, counter and stack. The user threads do not create any separate tasks for creation. The switching is also fast as there is no OS intervention. There is no coordination between threads and kernel and hence if one thread breaks, the entire process is blocked.

- **Kernel - Level Thread**

Kernel manages the threads and knows each and every thread. This is a multithreading type. The kernel manages a table to track the threads in each process. Also, there is a separate table to track the processes and update whenever the changes are made. OS makes the changes in the thread while creating and managing them. Knowledge of threads is shared with the kernel and hence the time for each process is scheduled according to the execution. Kernel threads are used for applications that break in between the process. Kernel threads are slow when compared with user threads. The thread control block is needed to control the tasks.

# Object in Windows

An object in the Windows operating system is a structure used to store data. An object consists of system-defined data types called attributes and functions that treat the attributes. Objects are widely used in Windows operating system, e.g. files, windows, pictures, semaphores, processes and threads. Nonetheless, not all data in Windows operating system are objects, as only data that needs sharing, protection or to be visible to user programs is located in objects. Objects in the Windows operating system are similar to objects used in object-oriented programming. However, objects in Windows lack some of the characteristics in object-oriented programming systems, such as inheritance and polymorphism.

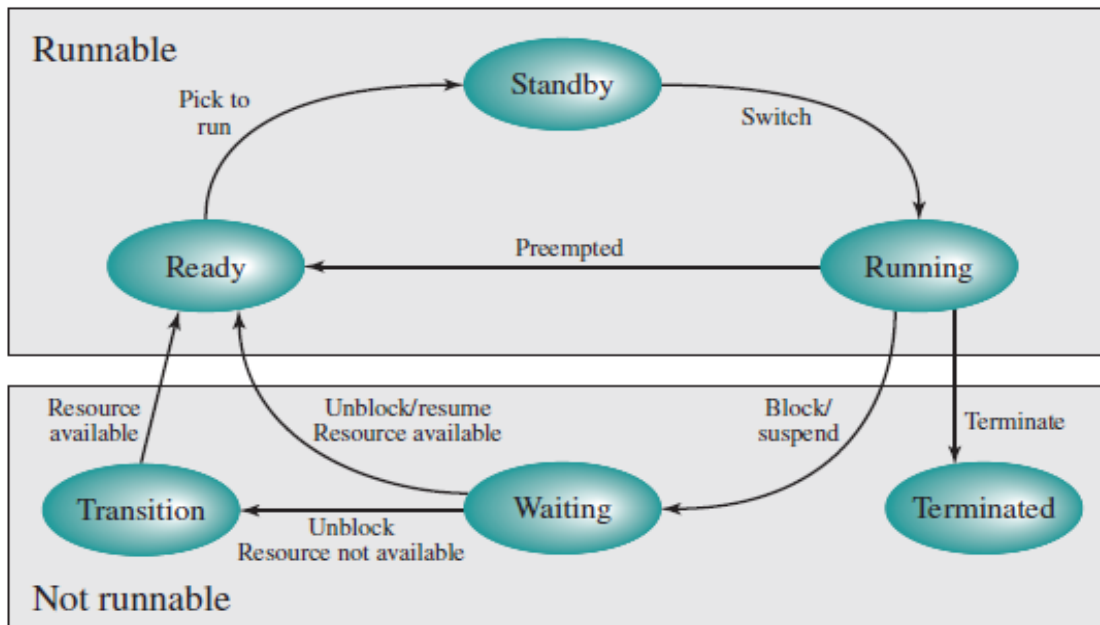
In Windows, a thread is a process object to which the operating system is allocating processor time. The process itself is a primary thread, which can launch other threads. The process has a list with the threads that belong to the process.

## Windows thread States

While a process must have one thread of execution, the process can create other threads to execute tasks in parallel. Threads share the process environment, thus multiple threads under the same process use less memory (resource) than the same number of processes. From the Win32 documentation (unmanaged), a thread can be in the following states:

Initialized	A state that indicates the thread has been initialized, but has not yet started.
Ready	A state that indicates the thread is waiting to use a processor because no processor is free. The thread is prepared to run on the next available processor.
Running	A state that indicates the thread is currently using a processor.
Standby	A state that indicates the thread is about to use a processor. Only one thread can be in this state at a time.
Transition	A state that indicates the thread is waiting for a resource, other than the processor, before it can execute. For example, it might be waiting for its execution stack to be paged in from disk.

Unknown	The state of the thread is unknown.
Wait	A state that indicates the thread is not ready to use the processor because it is waiting for a peripheral operation to complete or a resource to become free. When the thread is ready, it will be rescheduled.
Terminated	A state that indicates the thread has finished executing and has exited.



## Multithreading

Windows supports concurrency among processes because threads in different processes may execute concurrently (appear to run at the same time). Moreover, multiple threads within the same process may be allocated to separate processors and execute simultaneously (actually run at the same time). A multithreaded process achieves concurrency without the overhead of using multiple processes. Threads within the same process can exchange information through their common address space and have access to the shared resources of the process. Threads in different processes can exchange information through shared memory that has been set up between the two processes. Multithreading is the ability of a CPU to supply multiple threads of execution synchronously. Multithreading has many benefits in windows applications, such as:

- Each child window in an MDI application can be assigned to a different thread.

- If the drawing part (OnDraw code) of the program takes a long time to execute, the GUI will be blocked until the redraw is completed. However, we can assign a separate thread to the OnDraw function, thus causing the application to be responsive when long redraws occur.
- Multiple threads can be concurrently executed if there are multiple CPUs in the system thus speeding up the execution of the program.
- Complex simulations can be carried out efficiently by assigning a separate thread to each simulation entity.
- Important events can be handled efficiently by assigning those to a high priority thread.

## Relationship between threads and process

Threads: Processes	Description	Example Systems
<b>1:1</b>	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
<b>M:1</b>	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
<b>1:M</b>	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
<b>M:N</b>	It combines attributes of M:1 and 1:M cases.	TRIX

## Process vs Thread

Process	Thread
Process means any program is in execution.	Thread means segment of a process.
Process takes more time to terminate.	Thread takes less time to terminate.
It takes more time for creation.	It takes less time for creation.
It also takes more time for context	It takes less time for context switching.

switching.	
Process is less efficient in term of communication.	Thread is more efficient in term of communication.
Processes consume more resources.	Thread consumes less resources.
Process is isolated.	Threads share memory.
Process is called heavy weight process.	Thread is called light weight process.
Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.
If one process is blocked then it will not effect the execution of other process	Second thread in the same task could not run, while one server thread is blocked.
Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.



# How does IPC work in Windows

Inter Process Communication also known as IPC is a process that is used to share data between two different processes or applications on the same computer and / or network. Most commonly this type of mechanism is used in client and server environments, where a client or user will request some sort of data and the server then provides the data to them. Below is the different types of IPC Windows contains.

- Clipboard

Almost everyone who has ever touched a computer will know this simple but yet extremely useful function. The clipboard is a central depository to share data between applications. The clipboard either copy's or clips data in a standard or application-specific format, then a different application can then retrieve that data set as long as the data is available in a format it understands. So in short a clipboard is the middle man between two applications as long as both applications can agree on a data format.

- COM

Component object model (COM) is what allows us to create a document with data from different applications, an example would be when we create a word document where we put a spreadsheet from Excel or slides from powerpoint or anything else from other applications, so that even when we update something in Excel on the component, it will also update in the word document. A requirement for this function is that the software component is compatible or using the (COM) in order to communicate with the many different components.

- Data Copy

Data copy is what enables one application to another using the WM\_COPYDATA message, but this requires the two applications to form some sort of handshake or cooperation, because the application who is receiving data must know the format of the data it's receiving and also who the sender is. A key point is that the sending application can't modify the memory by any pointers.

- DDE

Dynamic Data exchange ( DDE ) can be thought of as an extension to the clipboard mechanism. When using the clipboard it's a one time use, where as DDE can also be a one time use but it can also be used for ongoing exchanges like the example i used in (COM) where the Excel component is updated in Excel but then the same spreadsheet will update in the word document as well. DDE will also continue to run the background and not require any user interference, again using the example with the word document and the Excel spreadsheet, i do not have to push any button in order for the spreadsheet to update, it does so automatically. Using the Clipboard on the contrary always requires user interaction, since we have to push buttons in order for it to do anything.

- File Mapping

File mapping is a way for at least two processes to share data as long as synchronization is provided between them.

File mapping cannot be used over a network, but only between processes on a local computer.

- Mailslots

Mailslot is a message sending process that provides you with a one way-communication capability, any process that is creating a mailslot process is called a mailslot server. Incoming messages are always added to the mailslot, then the mailslot saves the messages until the mailslot server is able to read it. It is possible for a one process to be both a mailslot server as well as a mailslot client, so therefore it is possible to have multiple instances of mailslot running simultaneously. A message can be sent to multiple mailslot instances on the network as long as it named the same, but if you broadcast a message to multiple recipients it can be no longer then 400 bytes, where as a message's length to a single mailslot instance is defined by mailslot server that the mailslot.

- Pipes

There are two different types of pipes in the Windows system, one is called *anonymous pipes*, this one enables related processes to communicate with each other.

One such process is when a child's process input or output ( I/O ) is redirected to the parent's process. In order to exchange data in both input and output two *anonymous pipe* instances need to be established; this is called ( duplex operation ). In this situation the parent process will use one pipe to write data for the child to read, and in the other pipe the child will write data for the parent to read. *Anonymous pipes* can only be used locally and between related processes.

*Named pipe* has the exact opposite capabilities from *anonymous pipes* in the sense that it can transfer data between unrelated process on a network, and a single pipe is able to take both input and output ( I/O )

- **RPC**

RPC is when an application makes a function call remotely, this can either be on the same computer or on another computer on the same network, the key difference is that both computers do not have to be using the same operating system. But in order for the two different applications on different operating systems to communicate together, they both have to support the Open Software Foundation ( OSF ) Distributed Computing Environment ( DCE ).

- **Windows sockets**

Takes advantage of the underlying protocols communication capabilities through the protocol-independent interface through the networking capabilities, it is through this socket type all ( I/O ) from the above is being processed.

# Sources

An Introduction to Windows Operating System - Einar Krogh

[Windows thread manager](#)

[Windows Thread](#)

[Multithreading](#)

[Activity](#)

William Stallings - Operating Systems- Ninth Global edition

[Service](#)

[Content Service](#)

[Broadcast Service](#)

[GCD](#)

[Block](#)

[Benefits of Interrupts as threads](#)

Windows own documentation on IPC.

<https://docs.microsoft.com/en-us/windows/win32/ipc/interprocess-communications>

