THREADS

R. Abhinav
CB.EN.U4CSE19453

- **What is a thread?**
  - A thread is a path of execution within a process. A process can contain multiple threads.
  - The thread consists of a program counter, stack and a set of registers
- **Why threads:-**
  - The main and primary difference between a thread and a process is that thread run in a shared memory space and process run in seperate memory space.
  - Threads share lot of things among themselves
    Eg:- Code section, data section, OS resources etc.,
- **How do we use threads in our program?**

The API's for using threads &/ managing threads are provided by a library called as "thread library".

- These 'thread libaries can be implemented by user space &/ by kernel space.
- Three most popular thread libraries that are used are as follows:-
  (i) Posix thread
  (ii) Java threads
  (iii) Win32 threads

Windows 7 threads:-

- The windows 7 uses Win-32 thread library to create and manage threads.
- This library can be used by including the following header file in the program:

```
#include <windows.h>
```

Some system calls:-

① createThread ( ) :-

Usage:- This system call is to create a thread to execute within the virtual address space of the calling process.

Parameters that this system call has :-

① Security attributes (default - NULL)

② Thread stack size (default - 0)

③ Thread start routine (function)

④ variable (pointer to variable) that needs to be passed to the thread.

⑤ Creation Flags (default - 0)

⑥ This is a pointer to a variable that receives the thread-id.

- Security attributes:-

This pointer to the structure determines whether the returned handle can be inherited by the child process or not. If 'NULL' the handle cannot be inherited.

2. Thread stack size :-

If this parameter is zero, the thread uses the size of executable. This is basically the initial size of the stack.

3. Thread start routine :-

This is basically the function that has to be executed by the thread.

4. Parameters :-

A pointer to variable that is passed to the thread.

5. Creation Flags :-

This parameter controls the creation of threads. If '0' is passed gets executed immediately after creation.

6. Thread-id :-

A pointer to a variable that recieves the thread-id.

Return-value :-
If thread execution succeed return value is a handle to a new thread &, fails returns "NULL"

• Close Handle ( ) :

Usage: This function closes the open handle

parameter :-

Valid handle : to an object that is open.

This system call returns a non-zero value if the function succeeds

- **Wait for Single Object ( ) :**

* **Usage:-**

Waits until the specified object is in the State (8) time out interval elapses

* **Parameters :-**

Handle: The return value that thread creation gives i.e, Thread Handle in this case.

Time out in milliseconds. If a non-zero value is specified. the function waits until the object is signaled (8) interval elapses.