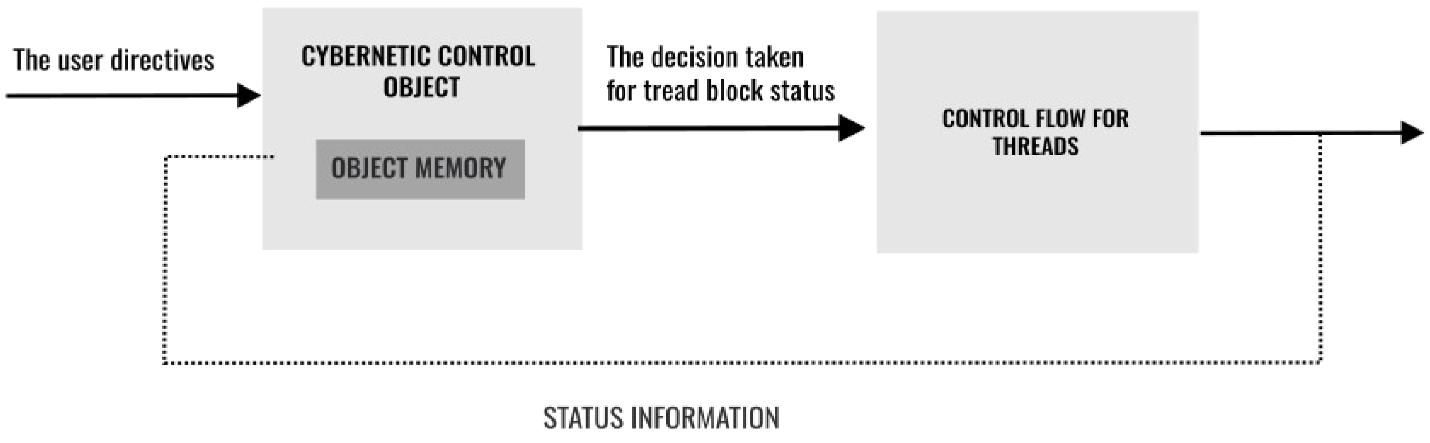
A BREIF INTRODUCTION TO THE CYBERNETIC THREAD MANAGEMENT

In order to simplify the explanation of the cybernetic thread management, the instance of the most upper level wrapper class (pcynlitx::threads) is named as cybernetic control object this section. This wrapper class includes the other classes which exist on pcynlitx multi-threading library as its member objects.

On the classical multi-threading, the program flows are changed when a particular condition is met. However, on the contrary, in cybernetic the cybernetic thread management system, the condition is received by the member function of the cybernetic control object and the cybernetic control object gives its decision based on current call to a control function and its previous knowledge about the process. More specifically, the cybernetic control object has a memory and it receives each thread's block status and the cybernetic control objects blocks or notifies the threads according to the other threads situation. ***In fact, the cybernetic object has a kind of situation awareness.*** In the pcynlitx multi-threading, the control functions (*or the control points*) of the library are used in order to collect information about the process status. A very simplified version of a cybernetic thread control function is shown below.

```
void stop(int a, int b){

      int caller_number = this→number();

      if(caller_number == a){

         TDS[caller_number].block_status = true;

         this→stop(caller_number);

         TDS[caller_number].block_status = false;
      }
}
TDS : The data sutructre holding thread related data.
```

In this control function, when the thread in which its id number is **'a'** performs a call to the **"stop(a,b)"**, the thread **"a"** is blocked and its status is set as blocked on the data structure holding thread status information. Beside this, when the corresponding control function ***"run(a,b)"*** is called by the thread **"b"**, this information has been taken into account. If the thread which is numbered as **"a"** is already blocked condition when the ***"run(a,b)"*** is called, then the thread **"a"** is notified and starts its execution again. On the other case, the thread numbered as **"b"** wait inside the function call ***"run(a,b)"*** until the thread **"b"** performs a call to the ***"stop(a,b)"***; In other words, in the cybernetic execution of the software, the cybernetic objects may not give permission for the execution of the procedures even though the related condition is met. An illustration of the cybernetic thread management technology has been given below.


The user directives
CYBERNETIC CONTROL OBJECT
OBJECT MEMORY
The decision taken for tread block status
CONTROL FLOW FOR THREADS
STATUS INFORMATION