Hi,

I found these 2 codes for windows and linux. I think they are reasonable to have a base. On windows I did the test and the values don't match with the manager since they use different structure (as the note in the example). On linux the values are already close to the test application!

windows:

```
void defP(){
    // Task Manager windows 10 uses VM_COUNTERS_EX2 similar structure is defined (in
ntddk.h)
  MEMORYSTATUSEX memInfo;
  memInfo.dwLength = sizeof(MEMORYSTATUSEX);
  GlobalMemoryStatusEx(&memInfo);
  DWORDLONG totalVirtualMem = memInfo.ullTotalPageFile;

  DWORDLONG virtualMemUsed = memInfo.ullTotalPageFile - memInfo.ullAvailPageFile;

  PROCESS_MEMORY_COUNTERS_EX pmc;
  GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS*)&pmc,
sizeof(pmc));
  SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

  DWORDLONG totalPhysMem = memInfo.ullTotalPhys;

  DWORDLONG physMemUsed = memInfo.ullTotalPhys - memInfo.ullAvailPhys;

  SIZE_T physMemUsedByMe = pmc.WorkingSetSize;

  SIZE_T  x=pmc.PagefileUsage;
  SIZE_T y=pmc.PrivateUsage;
  Cout() <<"m virtual usada-----> " << virtualMemUsedByMe/1024 << EOL;
  Cout() <<"m fisica usada-----> " << physMemUsed/1024/1024 << EOL;
  Cout() <<"total m  usada-----> " << totalPhysMem << EOL;
  Cout() <<"m privada usada-----> " << y/1024 << EOL;

}

CONSOLE_APP_MAIN
{
  SetLanguage("pt-br");
```

```
  SetDateFormat("%3:02d/%2:02d/%1:4d"); // format code in editor
  SetDateScan("dmy");

  defP();

}


linux:
#include <iostream>
#include <Core/Core.h>
#include <SysInfo/SysInfo.h>

using namespace Upp;

int parseLine(char* line){
   // This assumes that a digit will be found and the line ends in " Kb".
   int i = strlen(line);
   const char* p = line;
   while (*p <'0' || *p > '9') p++;
   line[i-3] = '\0';
   i = atoi(p);
   return i;
}

int getValue(){ //Note: this value is in KB!
   FILE* file = fopen("/proc/self/status", "r");
   int result = -1;
   char line[128];

   while (fgets(line, 128, file) != NULL){
      if (strncmp(line, "VmRSS:", 6) == 0){
         result = parseLine(line);
         break;
      }
   }
   fclose(file);
   return result;
}

CONSOLE_APP_MAIN
{
 Cout() << "--> " << getValue() << EOL;
}
```