#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include "typesStructs.h"

#include "oneSnapShot.h"

#pragma warning(disable:4996)

int FirstListProcess = 0;

// get the memory usage for process, ID, name of process, names of is dll

// Calls two functions, that create a linked list of DLLs and processes

// get ID process

void getMemoryInfo(DWORD processID) {

//create a new variable

PROCESS\* ret = (PROCESS\*)malloc(sizeof(PROCESS));

HANDLE hProcess;

PROCESS\_MEMORY\_COUNTERS pmc;

ret->prev = NULL;

ret->next = NULL;

//add the process Id into the new variable

ret->processId = processID;

// Open process in order to receive information

hProcess = OpenProcess(PROCESS\_QUERY\_INFORMATION |

PROCESS\_VM\_READ,

FALSE, processID);

if (NULL == hProcess) {

// Write to log a warning

return;

}

HMODULE hMods[1024];

DWORD cbNeeded;

TCHAR FoundProcessName[MAX\_PATH];

TCHAR wDllName[MAX\_PATH];

char regularCharArr[MAX\_PATH];

// Get Process Name

if (GetModuleFileNameEx(hProcess, 0, FoundProcessName, MAX\_PATH)) {

// At this point, buffer contains the full path to the executable

size\_t numConverted;

//Changing the string from TCHAR to char

//add the name Of Process into the new variable

wcstombs\_s(&numConverted, ret->nameOfProcess, MAX\_PATH, FoundProcessName, MAX\_PATH);

//Checking that there are no nameless processes

if (strlen(ret->nameOfProcess) < 1) {

//not have name to the process

//not added to the process linked list

return 1;

}

}

else {

// couldn't get the name

// You better call GetLastError() here

// Write To log

return 1;

}

// Gets the memory data and puts it in a new variable

if (GetProcessMemoryInfo(hProcess, &pmc, sizeof(pmc))) {

ret->PageFaultCount = pmc.PageFaultCount;

ret->WorkingSetSize = pmc.WorkingSetSize;

ret->QuotaPeakPagedPoolUsage = pmc.QuotaPeakPagedPoolUsage;

ret->QuotaPagedPoolUsage = pmc.QuotaPagedPoolUsage;

ret->PagefileUsage = pmc.PagefileUsage;

}

// Get Dlls List

// Receive all dll and put in hMods Array

if (EnumProcessModules(hProcess, hMods, sizeof(hMods), &cbNeeded)) {

// Calculate how many dll identifiers were returned.

// and loop of all dlls

for (int i = 0; i < (cbNeeded / sizeof(HMODULE)); i++) {

// Get the full path to the module's file.

// "name of the dll"

if (GetModuleFileNameEx(hProcess, hMods[i], wDllName, MAX\_PATH)) {

char dllName[MAX\_PATH];

size\_t numConverted;

wcstombs\_s(&numConverted, dllName, MAX\_PATH, wDllName, MAX\_PATH);

// get a name and links it to a dll linked list

addDLL(dllName);

}

}

}

//add the DLLName\_Tail, DLLName\_Head, countDLL into the new variable

ret->dllTail = DLLName\_Tail;

ret->dll = DLLName\_Head;

ret->numbersOfDLL = DLLName\_Tail->countDLL;

//Close the process

CloseHandle(hProcess);

//get the new variable and links it to a process linked list

addProcess(ret);

}

//function that include 3 Options

//receive all processes once, 20 times and until the user is pressed 4.

//the choice is made by the user

void getProcessesInfo() {

//Get Processes

DWORD aProcesses[1024] = { 0 };

DWORD cbNeeded, cProcesses;

//Receive all process ID and put in aProcesses Array

if (!EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded)) {

// Error. Write to log

return 1;

}

// Calculate how many process identifiers were returned.

cProcesses = cbNeeded / sizeof(DWORD);

// put in struct process the memory usage for each process, ID, name of process, linked list of is dll

// and connecting the struct to process linked list

if (userResponse == 1) {

// Loop of all processes

for (int y = 0; y < cProcesses; y++) {

//function that makes a frocess linked list

getMemoryInfo(aProcesses[y]);

}

}

}