### Санкт-Петербургский государственный политехнический университет Институт Информационных Технологий и Управления

### Кафедра компьютерных систем и программных технологий

Отчёт по практической работе

по предмету «Системное программное обеспечение»

**Утилита сбора системной информации в ОС Windows**

### Работу выполнил студент гр. 53501/3 Мартынов С. А. Работу принял преподаватель Душутина Е. В.

### Санкт-Петербург 2015

**Оглавление**

[Постановка задачи 3](#_TOC_250024)

[Введение 4](#_TOC_250023)

[Класс системной информации 8](#_TOC_250022)

[Метод GetUserTime() 8](#_TOC_250021)

[Метод GetUTCTime() 8](#_TOC_250020)

[Метод GetFUserName() 8](#_TOC_250019)

[Метод GetHostname() 9](#_TOC_250018)

[Метод GetCPUVendor() 9](#_TOC_250017)

[Метод GetCPUNumber() 9](#_TOC_250016)

[Метод GetVolumesInformation() 9](#_TOC_250015)

[Метод GetTotalMemory() 10](#_TOC_250014)

[Метод GetFreeMemory() 10](#_TOC_250013)

[Метод GetPagefileMemory() 10](#_TOC_250012)

[Метод GetVideoInformation() 10](#_TOC_250011)

[Метод GetWindowsVersion() 10](#_TOC_250010)

[Метод GetWindowsBuild() 11](#_TOC_250009)

[Метод GetWindowsRole() 11](#_TOC_250008)

[Метод GetConnectionInformation() 11](#_TOC_250007)

[Метод GetUptimeInformation() 11](#_TOC_250006)

[Метод GetConnectedHardwareList() 12](#_TOC_250005)

[Листинг 12](#_TOC_250004)

[Демонстрация работы программы 25](#_TOC_250003)

[Эксперимент 1. Виртуальная Win7 25](#_TOC_250002)

[Эксперимент 2. Реальная Win7 25](#_TOC_250001)

[Заключение 28](#_TOC_250000)

# Постановка задачи

### В рамках данной работы необходимо ознакомиться с основными механизмами сбора си- стемной информации в ОС Windows.

### Необходимо рассмотреть имеющихся штатных и нештатных механизмов извлечения си- стемной информации Windows.

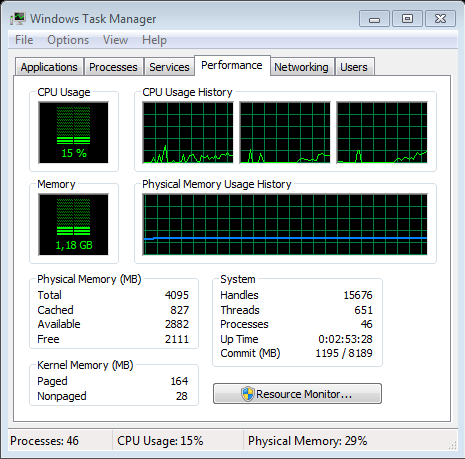
### В процессе работы предполагается изучить источники получения системной информации в Windows и разработать консольную утилиту, отображающую на экран (или в лог-файл) всю собранную информацию. Данная информация оказывается полезной, когда продукт уже передан на эксплуатацию конечному пользователю, и у него возникают проблемы, а разработчик не может получить физического доступа к машине, на которой исполняется код.

### В конце сравнить результаты полученные разработанной утилитой с результатами других средств. Провести эксперимент на нескольких устройствах.

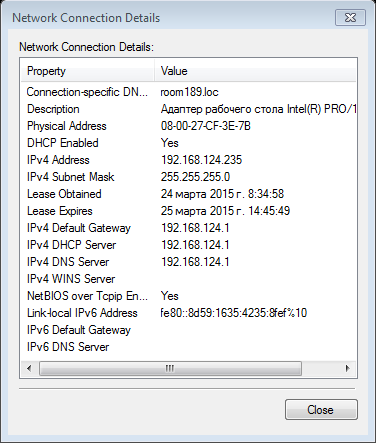
# Введение

### Основные характеристики работы системы можно получить в диспетчере задач (рис. 1). Там находится информация о количестве доступных системе ядер и их загрузка. Есть информация по доступной и занятой памяти, а также о файле подкачки. Показано общее время работы системы.

### Другие полезные данные можно почерпнуть из настроек сетевого подключения (рис. 2) и в оснастке управления дисками (рис. 3), где помимо самих томов можно увидеть файловую систему, объём и букву, под которой том смонтирован в систему.

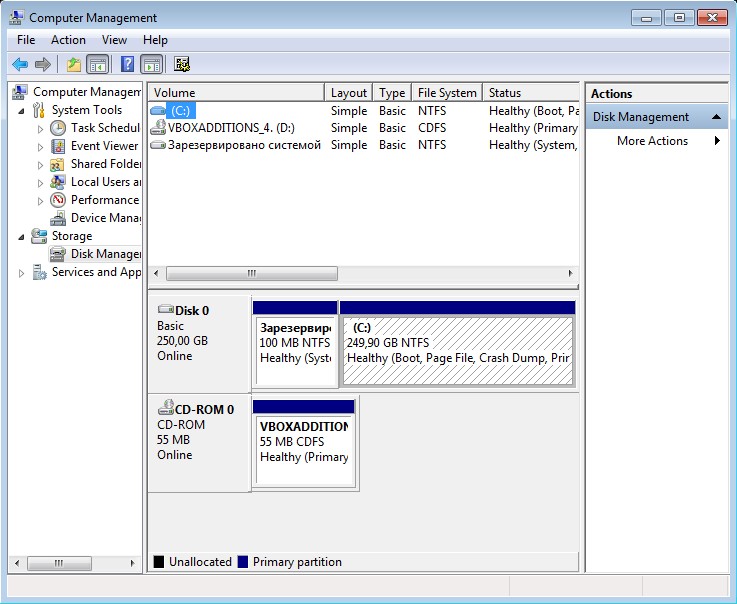


### Рис. 1: Диспетчер задач



### Рис. 2: Информация о сетевом подключении

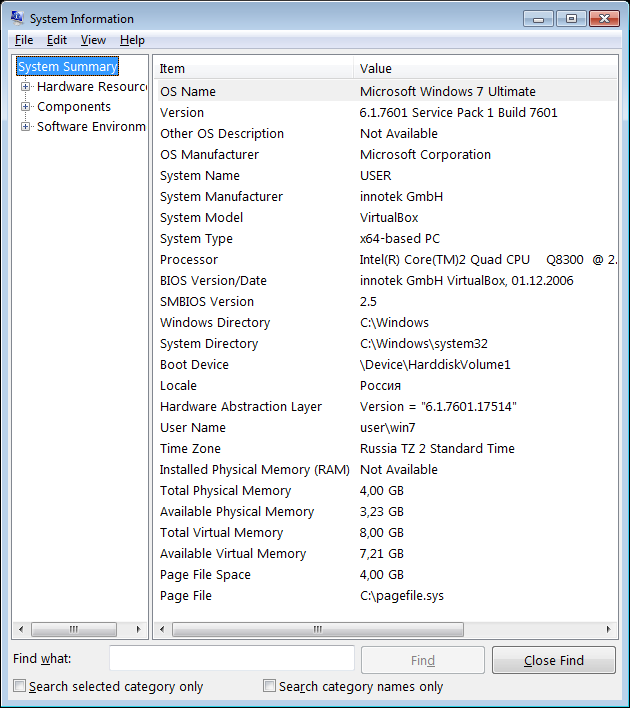
### Конфигуратор сетевых подключений даёт всю возможную статичную информацию о подключении.



### Рис. 3: Оснастка управления дисками

### Более комплексным решением является системная утилита msinfo32 (рис 4). Она собирает и отображает данные о конфигурации системы как для локальных, так и для удаленных компьютеров. Сюда входит информация о конфигурации оборудования, компонентах ком- пьютера, а также программном обеспечении, в том числе о подписанных и неподписанных драйверах. При устранении неполадок, связанных с конфигурацией системы, сотрудникам службы технической поддержки необходимы определенные данные о компьютере.

### Для хранения данных о системе предназначены файлы с расширением .nfo. Кроме того, программа работает с файлами форматов .cab и .xml. Содержимое открытого файла .cab можно просматривать средствами системы.



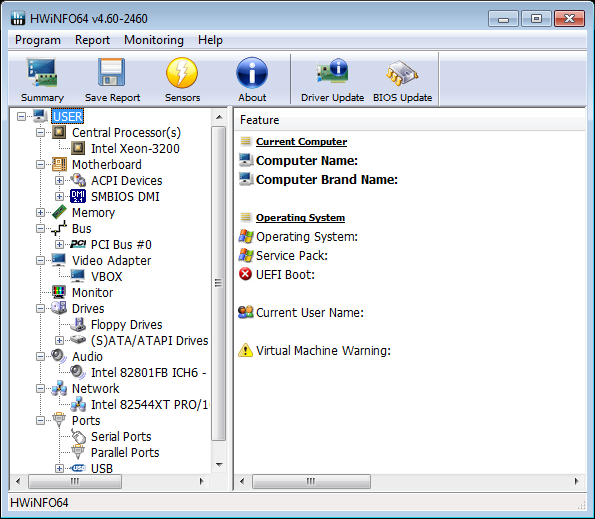
### Рис. 4: штатная утилита msinfo32

### Среди нештатных и бесплатных стредств сборки системной информации наибольшей популярностью пользуются AIDA32, Everest Home и HWiNFO (рис. 5).

### HWiNFO предоставляет детальную информацию об оборудовании в ОС Windows. Так же существует портативная версия. При запуске HWiNFO открывает несколько окон: окно состояния процессоров, окно сводной информации о системе и самое большое окно с доступом к оборудованию. Сводная информация дает возможность быстро ознакомиться с конфигурацией основного оборудования компьютера (процессор, память, диски, видеокарта, оперативная память). Всё установленное оборудование поделено по типу принадлежности (память, процессоры и т.д.) и показано в виде дерева.

### HWiNFO включает в себя монитор системы в режиме реального времени, что позволит увидеть реальные показания датчиков оборудования на текущий момент времени. Про- грамма включает в себя специальные тесты производительности системы. Вся информация, которую представляет программа, может быть сохранена в различных форматах: текст, Html, Xml, Mhtml.

### Стоит заметить, что программа сходу не правильно определила имеющийся в наличии процессор.



### Рис. 5: Запуск утилиты HWiNFO

# Класс системной информации

### Для сбора системной информации был разработан класс MySystem, методы которого отвечают за сбор различной системной информации. Ниже рассмотрены эти методы и предоставлен листинг их реализации.

## Метод GetUserTime()

### Возвращает пользовательское время, т.е. время, локальное для пользователя (с учётом часового пояса).

### Источник информации – системная структура \_SYSTEMTIME.

## Метод GetUTCTime()

### Возвращает мировое (UTC) время. Не зависит от локальных настроек пользователя. Источник информации – системная структура \_SYSTEMTIME.

## Метод GetFUserName()

### Возвращает полное имя пользователя, с учётом имени домена.

### Источник информации – системный вызов GetComputerNameEx() из sysinfoapi.h.

## Метод GetHostname()

### Возвращает имя хоста. Это не полное доменное имя, но это имя может использоваться для доступа по сети в рамках одного широковещательного домена.

### Источник информации – системный вызов GetComputerNameEx() из sysinfoapi.h.

## Метод GetCPUVendor()

### Возвращает название производителя процессора (если это возможно; если нет вернёт пустую строку). Для работы используется ассемблерный код т.к. информация получается непосредственно из регистров процессора.

### Источник информации – регистры центрального процессора.

## Метод GetCPUNumber()

### Возвращает количество доступных ядер. Если на машине включена поддержка техноло- гии Intel hyper-threading technology (или аналогичная технология виртуализации ядер), возвращаемое значение будет соответствовать количеству ядер, которое доступно ядре операционной системы.

### Источник информации – системная структура \_SYSTEM\_INFO.

## Метод GetVolumesInformation()

### Возвращает информацию о логических разделах, используемых в системе. По каждому разделу выводится его путь (как правило, заглавная буква латинского алфавита), метка (если она установлена), серийный номер и используемая файловая система (если она известна ядру операционной системы).

### Источник информации – системный вызов GetVolumeInformationW() из FileApi.h.

## Метод GetTotalMemory()

### Возвращает (в гибибайтах) общий объём физической оперативной памяти без файла подкачки.

### Источник информации – системный класс \_MEMORYSTATUSEX.

## Метод GetFreeMemory()

### Возвращает (в гибибайтах) общий объём свободной физической оперативной памяти без файла подкачки.

### Источник информации – системный класс \_MEMORYSTATUSEX.

## Метод GetPagefileMemory()

### Возвращает (в гибибайтах) общий объём системного файла подкачки. Источник информации – системный класс \_MEMORYSTATUSEX.

## Метод GetVideoInformation()

### Возвращает подробную информацию по видиосистеме. В начале формируется список всех видеоадаптеров (видеокарт), а потом список мониторов, подключённых к каждому из них.

### По видеоадаптерам выводится имя производителя (если эта информация есть в системном реестре) и системный путь. По мониторам выводится имя производителя (если эта инфор- мация есть в системном реестре), системный путь, разрешение (количество пикселей по горизонтали и по вертикали) и частота обновления.

### Источник информации – системный реестр.

## Метод GetWindowsVersion()

### Возвращает предполагаемую версию операционной системы (с точностью до номера сер- виспака) и её внутренний номер. Этот функционал системой поддерживаться довольно

### странным образом и не гарантирует точности результата, однако в рамках наших экспери- ментов ошибок не наблюдалось.

### Источник информации – множество функций из VersionHelpers.h.

## Метод GetWindowsBuild()

### Возвращает номер сборки операционной системы. Бывает полезен для выявления различий в рамках одной версии операционной системы.

### Источник информации – системная структура \_OSVERSIONINFOEXW.

## Метод GetWindowsRole()

### Возвращает роль машины. Это может быть Workstation (рабочая станция), Server (сервер) и Domain Controller (контроллер домена).

### Источник информации – системная структура \_OSVERSIONINFOEXW.

## Метод GetConnectionInformation()

### Выводит подробную информацию по сетевым соединениям. Как и в случае с видеосистемой, вначале формируется список сетевых адаптеров, а потом по каждому из них список подключений.

### По сетевому адаптеру выводится его системный путь, имя (если оно задано) и уникальный идентификатор (MAC-адрес). По сетевому подключению выводится IP-адрес, маска сети, шлюз (если указан) и источник получения адреса. Если адрес был получен по DHCP, этот факт будет указан, как и IP-адрес DHCP-сервера, выдавшего клиенту его IP-адрес.

### Источник информации – системный вызов GetAdaptersInfo() из iphlpapi.h.

## Метод GetUptimeInformation()

### Возвращает время работы системы с момента включения в часах, минутах и секундах. Источник информации – функция GetTickCount64() из sysinfoapi.h.

## Метод GetConnectedHardwareList()

### Возвращает список устройст, записи (драйвера) которых были обнаружены в реестре (для своей работы эта функция использует приватную функцию GetConnectedHardwareList(), но это сделано исключительно для упращения работы с кодом). Изначально устройства группируются по классу, но могут быть перегруппированы и отфильтрованы в пользова- тельском коде.

### Источник информации – системный реестр.

## Листинг

### Листинг 1 содержит код реализации представленных выше функций. Заголовочный файл интереса не представляет и может быть найден по ссылке из постановки задачи.

### Листинг 1: Файл реализации методов класса MySystem

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

# include

" My System . h"

# include

# include

# include

< sstream >

< iostream >

< Lmcons . h> *// UNLEN*

# include

< Version Helpers .h>

# include

< iphlpapi . h>

My System :: My System () : OneGB (1024 \* 1024 \* 1024) {

*// Fix localtime*

Get Local Time (& st Local );

*// Fix systime*

Get System Time (& st System );

*// System information ( CPU number)*

Get System Info (& sys Info );

*// Everything about memory*

memory Status . dw Length = sizeof ( MEMORYSTATUSEX ); Global Memory Status Ex (& memory Status );

*// Windows version*

osv Info . dw OSVersion Info Size = sizeof ( osv Info ); Get Version Ex (( OSVERSIONINFO \*)& osv Info );

}

My System ::~ My System () {

27 }

#### 28

1. std :: wstring My System :: Get User Time (){
2. *// format: YYYY - MM - DD , HH: MM: SS. ms*
3. std :: wstringstream ss;
4. ss << st Local . wYear << "-" << st Local . w Month << "-" << st Local . wDay << " "

<< st Local . wHour << ":" << st Local . w Minute << ":" << st Local . w Second

<< "." << st Local . w Milliseconds ;

#### 33

34 return ss. str ();

35 }

#### 36

1. std :: wstring My System :: Get UTCTime (){
2. *// format: YYYY - MM - DD , HH: MM: SS. ms*
3. std :: wstringstream ss;
4. ss << st System . wYear << "-" << st System . w Month << "-" << st System . wDay << " " << st System . wHour << ":" << st System . w Minute << ":" << st System . w Second << "." << st System . w Milliseconds ;
5. return ss. str ();

42 }

#### 43

1. std :: wstring My System :: Get FUser Name (){
2. *// Full user ’ s name*
3. TCHAR user Name [ UNLEN + 1];
4. DWORD nULen = UNLEN ;
5. Get User Name Ex ( Name Sam Compatible , userName , & nULen );

#### 49

50 return std :: wstring ( user Name );

51 }

1. std :: wstring My System :: Get Hostname (){
2. *// Computer name can be long*
3. TCHAR sc Computer Name [ MAX\_ COMPUTERNAME\_ LENGTH \* 2 + 1];
4. DWORD ln Name Length = MAX\_ COMPUTERNAME\_ LENGTH \* 2;
5. Get Computer Name Ex ( Computer Name Net BIOS , sc Computer Name , & ln Name Length );

#### 57

58 return std :: wstring ( sc Computer Name );

59 }

#### 60

61 std :: wstring My System :: Get CPUVendor (){

62 int regs [4] = { 0 };

1. char vendor [13];
2. \_\_ cpuid ( regs , 0); *// mov eax ,0; cpuid*
3. memcpy ( vendor , & regs [1] , 4); *// copy EBX*
4. memcpy ( vendor + 4 , & regs [2] , 4); *// copy ECX*
5. memcpy ( vendor + 8 , & regs [3] , 4); *// copy EDX*

68 vendor [12] = ’\0 ’;

#### 69

* 1. std :: string tmp ( vendor );
  2. return std :: wstring ( tmp . begin () , tmp . end ());

72 }

#### 73

1. int My System :: Get CPUNumber (){
2. return sys Info . dw Number Of Processors ;

76 }

#### 77

1. std :: wstring My System :: Get Volumes Information (){
2. *// see http :// www. codeproject . com/ Articles / 115061 / Determine - Information - about - System - User - Processes*
3. std :: wstringstream ss;
4. TCHAR sz Volume [ MAX\_ PATH + 1];
5. TCHAR sz File System [ MAX\_ PATH + 1];

#### 83

1. DWORD dw Serial Number ;
2. DWORD dw Max Len ;
3. DWORD dw System Flags ;

#### 87

1. TCHAR sz Drives [ MAX\_ PATH + 1];
2. DWORD dw Len = Get Logical Drive Strings ( MAX\_PATH , sz Drives );
3. TCHAR \* p Letter = sz Drives ;

#### 91

92 BOOL b Success ;

#### 93

1. while (\* p Letter ) {
2. b Success = Get Volume Information ( pLetter , *// The source*
3. szVolume , MAX\_PATH , *// Volume Label ( LABEL)*
4. & dw Serial Number , & dwMaxLen , *// Serial Number ( VOL)*
5. & dw System Flags ,
6. sz File System , MAX\_ PATH ); *// File System ( NTFS , FAT ...)*

#### 100

1. if ( b Success ) {
2. ss << \* p Letter << ":" << std :: endl ;

#### 103

1. *// LABEL command*
2. ss << "\ t Label :\ t" << sz Volume << std :: endl ;

#### 106

1. *// Standard formal to display serial number ( VOL command)*
2. ss << "\ t Numbr :\ t" << HIWORD ( dw Serial Number ) << "-" << LOWORD ( dw Serial Number ) << std :: endl ;

#### 109

1. *// File - System*
2. ss << "\ t FSysm :\ t" << sz File System << std :: endl << std :: endl << std :: endl ;

#### 112 }

1. else {
2. ss << " No data for " << p Letter << std :: endl << std :: endl << std :: endl

;

#### 115 }

#### 116

1. while (\*++ p Letter ); *// Notice Semi - colon!*
2. p Letter ++;

#### 119 }

120 return ss. str ();

#### 121 }

#### 122

1. double My System :: Get Total Memory (){
2. return memory Status . ull Total Phys / OneGB ;

#### 125 }

#### 126

1. double My System :: Get Free Memory (){
2. return memory Status . ull Avail Phys / OneGB ;

#### 129 }

#### 130

1. double My System :: Get Pagefile Memory (){
2. return memory Status . ull Total Page File / One GB ;

#### 133 }

#### 134

1. std :: wstring My System :: Get Video Information (){
2. std :: wstringstream ss;
3. int device Index = 0;
4. int result ;

#### 139

1. do {
2. PDISPLAY\_ DEVICE display Device = new DISPLAY\_ DEVICE ();
3. display Device -> cb = sizeof ( DISPLAY\_ DEVICE );

#### 143

144 result = Enum Display Devices ( NULL , device Index ++ , display Device , 0);

#### 145

1. if ( display Device -> State Flags & DISPLAY\_ DEVICE\_ ACTIVE ) {
2. PDISPLAY\_ DEVICE monitor = new DISPLAY\_ DEVICE ();
3. monitor -> cb = sizeof ( DISPLAY\_ DEVICE );

#### 149

150 Enum Display Devices ( display Device -> DeviceName , 0 , monitor , 0);

#### 151

1. ss << " Display Device :\ t" << display Device -> Device Name << std :: endl ;
2. ss << " Display String :\ t" << display Device -> Device String << std :: endl ;
3. ss << " Display ID :\ t" << display Device -> Device ID << std :: endl << std :: endl ;;

#### 155

1. ss << "\ t Monitor Device :\ t" << monitor -> Device Name << std :: endl ;
2. ss << "\ t Monitor String :\ t" << monitor -> Device String << std :: endl ;
3. ss << "\ t Monitor ID :\ t" << monitor -> Device ID << std :: endl ;

#### 159

1. PDEVMODE dm = new DEVMODE ();
2. if ( Enum Display Settings ( display Device -> DeviceName , ENUM\_ CURRENT\_ SETTINGS , dm)) {
3. ss << std :: endl ;
4. ss << "\ tFreq .: \ t" << dm -> dm Display Frequency << std :: endl ;
5. ss << "\ tBPP : \ t" << dm -> dm Bits Per Pel << std :: endl ;
6. ss << "\ t Width : \ t" << dm -> dm Pels Width << std :: endl ;
7. ss << "\ tHeig .: \ t" << dm -> dm Pels Height << std :: endl ;

#### 167 }

#### 168 }

#### 169

170 } while ( result );

#### 171

1. *// ss << n Width << " x" << n Height;*
2. return ss. str ();

#### 174 }

#### 175

1. std :: wstring My System :: Get Windows Version (){
2. *// See https :// msdn. microsoft . com/ en - us/ library/ ms 724429 %28 VS . 85 % 29 . aspx*
3. DWORD dw Win Ver = Get Version ();
4. std :: wstringstream ss;

#### 180

1. if ( Is Windows 8 Point 1 Or Greater ()) {
2. ss << " Windows 8.1 ";

#### 183 }

1. else if ( Is Windows 8 Or Greater ()) {
2. ss << " Windows 8 ";

#### 186 }

1. else if ( Is Windows 7 SP 1 Or Greater ()) {
2. ss << " Windows 7 SP1 ";

#### 189 }

1. else if ( Is Windows 7 Or Greater ()) {
2. ss << " Windows 7 ";

#### 192 }

1. else if ( Is Windows Vista SP 2 Or Greater ()) {
2. ss << " Vista SP2 ";

#### 195 }

1. else if ( Is Windows Vista SP 1 Or Greater ()) {
2. ss << " Vista SP1 ";

#### 198 }

1. else if ( Is Windows Vista Or Greater ()) {
2. ss << " Vista ";

#### 201 }

1. else if ( Is Windows XPSP 3 Or Greater ()) {
2. ss << " XP SP3 ";

#### 204 }

1. else if ( Is Windows XPSP 2 Or Greater ()) {
2. ss << " XP SP2 ";

#### 207 }

1. else if ( Is Windows XPSP 1 Or Greater ()) {
2. ss << " XP SP1 ";

#### 210 }

211 else if ( Is Windows XPOr Greater ()) {

212 ss << " XP";

#### 213 }

214 ss << ", " << LOBYTE ( LOWORD ( dw Win Ver )) << "." << HIBYTE ( LOWORD ( dw Win Ver ));

#### 215

216 return ss. str ();

#### 217 }

#### 218

1. double My System :: Get Windows Build (){
2. return osv Info . dw Build Number ;

#### 221 }

#### 222

1. std :: wstring My System :: Get Windows Role (){
2. std :: wstringstream ss;

#### 225

1. switch ( osv Info . w Product Type ) {
2. case VER\_ NT\_ WORKSTATION :
3. ss << " Workstation ";
4. break ;
5. case VER\_ NT\_ SERVER :
6. ss << " Server ";
7. break ;
8. case VER\_ NT\_ DOMAIN\_ CONTROLLER :
9. ss << " Domain Controller ";
10. break ;
11. default :
12. ss << " Unknown ";

#### 238 }

#### 239

240 return ss. str ();

#### 241 }

#### 242

1. std :: wstring My System :: Get Connection Information (){
2. *// See https :// msdn. microsoft . com/ en - us/ library/ ms 724429 %28 VS . 85 % 29 . aspx*
3. std :: wstringstream ss;
4. PIP\_ ADAPTER\_ INFO p Adapter Info ;
5. ULONG ul Out Buf Len = sizeof ( IP\_ ADAPTER\_ INFO );

#### 248

1. p Adapter Info = ( IP\_ ADAPTER\_ INFO \*) MALLOC ( sizeof ( IP\_ ADAPTER\_ INFO ));
2. if ( Get Adapters Info ( p Adapter Info , & ul Out Buf Len ) == NO\_ ERROR ) {
3. PIP\_ ADAPTER\_ INFO p Adapter = p Adapter Info ;
4. while ( p Adapter ) {
5. ss << " Adapter " << pAdapter -> Adapter Name << " /" << pAdapter -> Description << "/" << std :: endl ;
6. ss << "\ tMAC addr :\ t";
7. for ( UINT i = 0; i < pAdapter -> Address Length ; i ++) {
8. if ( i == ( pAdapter -> Address Length - 1))
9. ss << ( int ) pAdapter -> Address [ i] << std :: endl ;
10. else
11. ss << ( int ) pAdapter -> Address [ i] << "-";

#### 260 }

1. ss << "\ tIP Address :\ t " << pAdapter -> Ip Address List . Ip Address . String

<< std :: endl ;

1. ss << "\ tIP Mask :\ t " << pAdapter -> Ip Address List . Ip Mask . String << std

:: endl ;

1. ss << "\ t Gateway :\ t " << pAdapter -> Gateway List . Ip Address . String << std

:: endl ;

1. if ( pAdapter -> Dhcp Enabled ) {
2. ss << "\ tDHCP Enabled :\ t Yes " << std :: endl ;
3. ss << "\ tDHCP Server :\ t " << pAdapter -> Dhcp Server . Ip Address . String

<< std :: endl ;

#### 267 }

1. else
2. ss << "\ tDHCP Enabled : No" << std :: endl ;

#### 270

271 p Adapter = pAdapter -> Next ;

#### 272 }

#### 273 }

1. FREE ( p Adapter Info );
2. return ss. str ();

#### 276 }

1. std :: wstring My System :: Get Uptime Information (){
2. std :: wstringstream ss;

#### 279

1. unsigned long uptime = ( unsigned long ) Get Tick Count 64 ();
2. unsigned int days = uptime / (24 \* 60 \* 60 \* 1000 ) ;

282 uptime %= (24 \* 60 \* 60 \* 1000) ;

283 unsigned int hours = uptime / (60 \* 60 \* 1000) ;

284 uptime %= (60 \* 60 \* 1000 ) ;

285 unsigned int minutes = uptime / (60 \* 1000) ;

286 uptime %= (60 \* 1000) ;

287 unsigned int seconds = uptime / ( 1000 ) ;

#### 288

289 ss << days << " days , " << hours << ":" << minutes << ":" << seconds ;

#### 290

291 return ss. str ();

#### 292 }

#### 293

1. void My System :: Fill Hardware Info ( HDEVINFO & di , SP\_ DEVINFO\_ DATA & did , Hardware Info & hd) {
2. std :: locale loc ;
3. BYTE \* pbuf = NULL ;
4. DWORD req Size = 0;
5. if (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_ DEVICEDESC , NULL , NULL , 0 , & req Size ))

#### 299 {

300 *// error , but loop might continue ?*

#### 301 }

#### 302

1. pbuf = new BYTE [ req Size > 1 ? req Size : 1];
2. if (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_ DEVICEDESC , NULL , pbuf , reqSize , NULL ))

#### 305 {

1. *// device does not have this property set*
2. memset ( pbuf , 0 , req Size > 1 ? req Size : 1);

#### 308 }

1. hd. dev Descrition = ( wchar\_ t \*) pbuf ;
2. delete [] pbuf ;

#### 311

1. TCHAR dev Instance Id [ MAX\_ DEVICE\_ ID\_ LEN ];
2. memset ( dev Instance Id , 0 , MAX\_ DEVICE\_ ID\_ LEN );
3. *// pbuf = new BYTE[ req Size > 1 ? req Size : 1];*
4. if ( Setup Di Get Device Instance Id ( di , & did , dev Instance Id , MAX\_ DEVICE\_ ID\_ LEN , NULL ) == FALSE ) {
5. *// error , but loop might continue ?*

#### 317 }

1. hd. dev Instance ID . assign ( dev Instance Id );
2. *// delete [] pbuf;*

#### 320

1. req Size = 0;
2. if (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_MFG , NULL , NULL , 0 ,

|  |  |  |  |
| --- | --- | --- | --- |
| 323 | { | & req Size )) | |
| 324 |  | *// error , but loop might continue ?* | |
| 325 | } |  | |
| 326 |  |  | |
| 327 | pb | uf = new BYTE [ req Size > 1 ? req Size : 1]; | |
| 328 | if | (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_MFG , NULL , pbuf , reqSize , NULL )) | |
| 329 | { | |  |
| 330 | *// device does not have this property set* | |  |
| 331 | memset ( pbuf , 0 , req Size > 1 ? req Size : 1); | |  |
| 332 | } | |  |
| 333 | hd. hardware MFG = ( wchar\_ t \*) pbuf ; | |  |
| 334 |  | |  |
| 335 | *// Small hack for VM* | |  |
| 336 | if ( hd. hardware MFG . length () > 1 && ! std :: isalpha ( hd. hardware MFG [1] , | | loc )) |
| 337 | hd. hardware MFG . assign ( L"( none )"); | |  |
| 338 |  | |  |

339 delete [] pbuf ;

#### 340 } 341 342

1. void My System :: Get Connected Hardware List ( std :: multimap < std :: wstring , Hardware Info >& result ) {
2. result . clear ();

#### 345

1. std :: vector < GUID > all Class Guids ;
2. *// Setup Di Get Class Devs () does not work if the class GUID is NULL and the*
3. *// 4 th parameter is different from DIGCF\_ ALLCLASSES . Therefore we collect all*
4. *// the class GUIDs in the first step , and then collect all the connected devices*
5. *// by feeding the collected GUIDs to Setup Di Get Class Devs () in the 2 nd step*

#### 351

352 *// Step 1*

#### 353 {

1. HDEVINFO di = Setup Di Get Class Devs ( NULL ,
2. NULL ,
3. NULL ,
4. DIGCF\_ ALLCLASSES );

#### 358

1. if ( di == INVALID\_ HANDLE\_ VALUE ) {
2. DWORD ret = :: Get Last Error ();
3. throw - 1;

#### 362 }

|  |  |  |
| --- | --- | --- |
| 363 |  |  |
| 364 | int iIdx = 0; |
| 365 | while ( true ) { |
| 366 | SP\_ DEVINFO\_ DATA did ; |
| 367 | did . cb Size = sizeof ( SP\_ DEVINFO\_ DATA ); |
| 368 | if ( Setup Di Enum Device Info ( di , iIdx , & did ) == FALSE ) { |
| 369 | if (:: Get Last Error () == ERROR\_ NO\_ MORE\_ ITEMS ) |
| 370 | { |
| 371 | break ; |
| 372 | } |
| 373 | else { |
| 374 | *// error , but loop might continue ?* |
| 375 | } |
| 376 | } |
| 377 | if ( std :: find ( all Class Guids . begin () , all Class Guids . end () , | did . |
| 378 | Class Guid ) == all Class Guids . end ()) { all Class Guids . push\_ back ( did . Class Guid ); |  |
| 379 | } |  |
| 380 | iIdx ++; |  |
| 381 | } |  |
| 382 | if ( Setup Di Destroy Device Info List ( di) == FALSE ) { |  |
| 383 | *// error , but should be ignored ?* |  |
| 384 | } |  |
| 385 | } |  |
| 386 | *// Step 2* |  |
| 387 | for ( unsigned int i = 0; i < all Class Guids . size (); i ++) { |  |
| 388 | HDEVINFO di = Setup Di Get Class Devs (& all Class Guids [ i], |  |
| 389 | NULL , |  |
| 390 | NULL , |  |
| 391 | DIGCF\_ PRESENT ); |  |
| 392 | if ( di == INVALID\_ HANDLE\_ VALUE ) { |  |
| 393 | throw :: Get Last Error (); |  |
| 394 | } |  |
| 395 |  |  |
| 396 |  |  |
| 397 | int iIdx = 0; |  |
| 398 | Hardware Info hd; |  |
| 399 | while ( true ) |  |
| 400 | { |  |
| 401 | SP\_ DEVINFO\_ DATA did ; |  |
| 402 | did . cb Size = sizeof ( SP\_ DEVINFO\_ DATA ); |  |
| 403 | if ( Setup Di Enum Device Info ( di , iIdx , & did ) == FALSE ) { |  |
| 404 | if (:: Get Last Error () == ERROR\_ NO\_ MORE\_ ITEMS ) { |  |
| 405 | break ; |  |
| 406 | } |  |

407

408

409

410

411

412

413

414

else {

*// error , but loop might continue ?*

}

}

415

416

417

418

419

420

BYTE \* pbuf = NULL ;

DWORD req Size = 0;

if (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_ CLASS , NULL , NULL , 0 , & req Size ))

{

*// error , but loop might continue ?*

}

421

422

423

424

425

426

427

428

pbuf = new BYTE [ req Size > 1 ? req Size : 1];

if (! Setup Di Get Device Registry Property ( di , & did , SPDRP\_ CLASS , NULL , pbuf , reqSize , NULL ))

{

*// device does not have*

memset ( pbuf , 0 , req Size

*this property set*

> 1 ? req Size : 1);

}

Fill Hardware Info ( di , did ,

hd);

429

430

431

432

433

434

435

436

result . insert ( std :: multimap < std :: wstring , Hardware Info >:: value\_ type ((

wchar\_ t \*) pbuf , hd)); delete [] pbuf ;

iIdx ++;

}

if

( Setup Di Destroy Device Info List ( di) == FALSE ) {

*// error , but should be ignored ?*

}

}

}

### Листинг 2 содержит исходный код программы SystemInformation, которая была написана для деонсрации работы в классом MySystem. Как можно видеть, программа занимается только выводом информации, полученной от класса MySystem.

### Листинг 2: Исходный код программы SystemInformation

1. # include < tchar . h>
2. # include < cstdlib >
3. # include < string >
4. # include < iostream >
5. # include < fstream >

6 # include " My System . h"

#### 7

1. int \_ tmain ( int argc , \_ TCHAR \* argv []) {
2. std :: locale :: global ( std :: locale (""));
3. My System systm ;
4. std :: wcout << " >> General Information " << std :: endl ;
5. std :: wcout << " User time : \ t" << systm . Get User Time () << std :: endl ;
6. std :: wcout << " UTC time : \ t" << systm . Get UTCTime () << std :: endl ;
7. std :: wcout << " User name : \ t" << systm . Get FUser Name () << std :: endl ;
8. std :: wcout << " Hostname : \ t" << systm . Get Hostname () << std :: endl ;
9. std :: wcout << std :: endl ;

#### 17

1. std :: wcout << " Windows : \ t" << systm . Get Windows Version () << " ( build " << systm . Get Windows Build () << ")" << std :: endl ;
2. std :: wcout << " Role : \ t\ t" << systm . Get Windows Role () << std :: endl ;
3. std :: wcout << " Uptime : \ t" << systm . Get Uptime Information () << std :: endl ;
4. std :: wcout << std :: endl ;

#### 22

1. std :: wcout << " CPU : \ t\ t" << systm . Get CPUVendor () << " (" << systm . Get CPUNumber () << " cores )" << std :: endl ;
2. std :: wcout << " Physical RAM : \ t" << systm . Get Total Memory () << " ( GB)" << std :: endl ;
3. std :: wcout << " Available RAM : \ t" << systm . Get Free Memory () << " ( GB)" << std :: endl ;
4. std :: wcout << " Pagefile : \ t" << systm . Get Pagefile Memory () << " ( GB)" << std :: endl ;
5. std :: wcout << std :: endl ;

#### 28

1. std :: wcout << " >> Video System Information " << std :: endl << systm . Get Video Information () << std :: endl ;
2. std :: wcout << " >> Hard Disk Drive Information " << std :: endl << systm . Get Volumes Information ();
3. std :: wcout << " >> Network Interface Information " << std :: endl << systm . Get Connection Information ();
4. std :: wcout << std :: endl ;

#### 33

1. std :: multimap < std :: wstring , Hardware Info > results ;
2. systm . Get Connected Hardware List ( results );
3. std :: wcout << " >> Devices " << std :: endl ;

#### 37

1. std :: wstring devclass = \_T(" none ");
2. for ( std :: multimap < std :: wstring , Hardware Info >:: const\_ iterator it = results . begin (); it != results . end (); ++ it) {
3. if ( devclass . compare ( it -> first )) {
4. std :: wcout << it -> first << L":" << std :: endl ;

42

43

44

devclass . assign ( it -> first );

46

}

std :: wcout endl ;

std :: wcout endl ;

std :: wcout

<<

L"\ t Description : " << it -> second . dev Descrition << std ::

45

<<

L"\ t Instance ID: " << it -> second . dev Instance ID << std ::

<<

L"\ t Manufacturer : " << it -> second . hardware MFG << std :: endl

<< std :: endl ;

47

48

49

50

51

52

}

*// std :: wcout << L" Done !" << std :: endl;*

getwchar (); exit (0) ;

}

# Демонстрация работы программы

### Для демонстрации практической части была разработана маленькая программа SystemInformation.

### Пользуясь методами класса MySystem, она собирает системную информацию и выводит её на экран. Вывод можно перенаправить в файл.

## Эксперимент 1. Виртуальная Win7

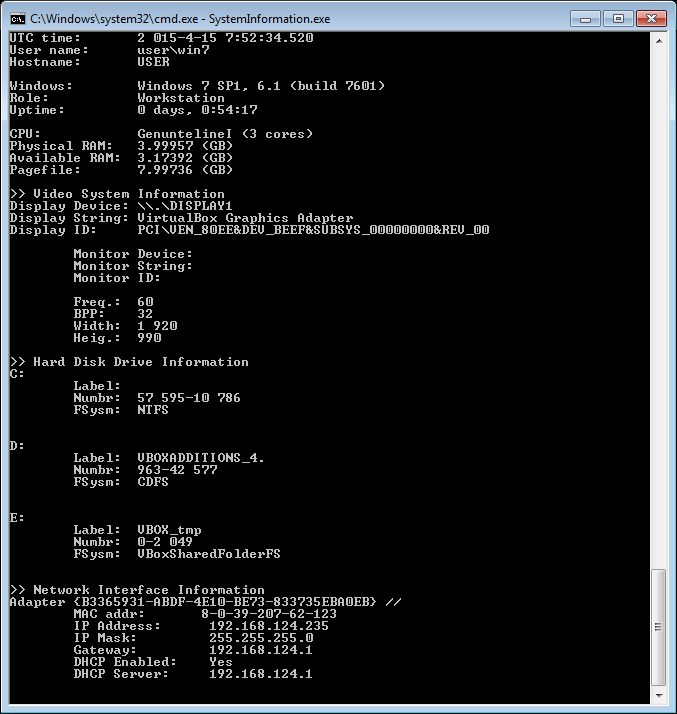
### Программа была запущена на виртуальной машине. Из особенностей следует отметить три ядра, доступные системе (это сделано специально, для комфортной работы гиперви- зора), имя видеоадаптера (виртуальная машина использует собственный драйвер) и имя производителя центрального процессора (виртуальная машина эмулирует X64 процессор).

### На рисунке 6 представлены результаты работы. Они разбиты на 5 секции: общая инфор- мация, информация о видеосистеме, информация о накопителях и информация о сетевой системе. Последней секцией идёт информация обо всех устройствах, имеющихся в систе- ме. Эта информация получена из реестра, без реального опроса оборудования. Полный текстовый лог лежит в папке со всеми прочими логами.

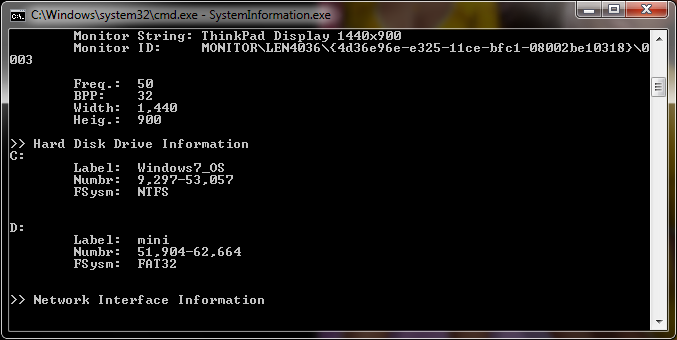
### Разница между локальным временем и UTC соответствует Московскому часовому поясу. В корректности полученной информации можно убедиться по рисункам 1, 2 и 3.

## Эксперимент 2. Реальная Win7

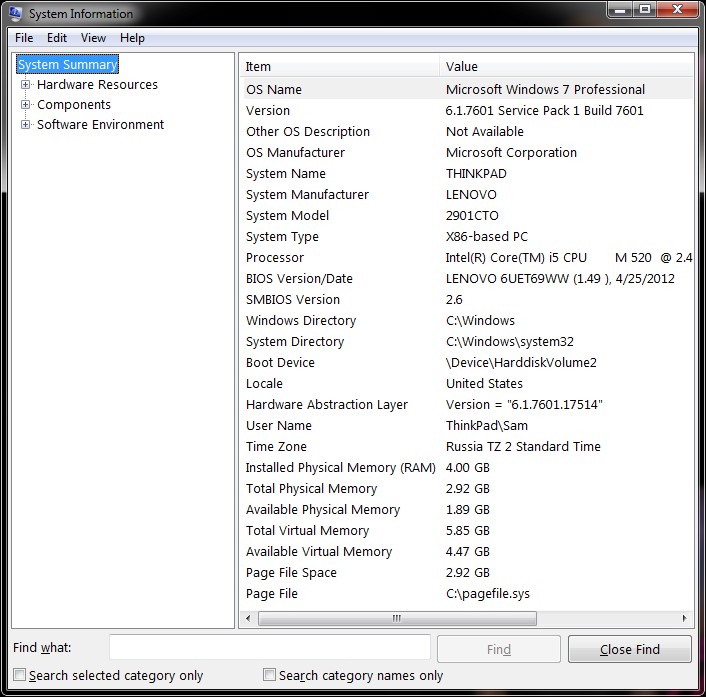
### Эксперимент проводился на реальном ноутбуке Lenovo Thinkpad T410s. Программа пока- зала значительно более высокое быстродействие, по сравнению с предыдущими запусками на виртуальных машинах. Это создало сложности при попытке сделать скриншот работы (рисунок 7). Подробный лог находится в папке с логами. Как показывает рисунок 8, данные корректы.



### Рис. 6: Результаты работы программы SystemInformation на виртуальной Win7



### Рис. 7: Результаты работы программы SystemInformation на реальной Win7.



### Рис. 8: Результаты работы программы SystemInformation

# Заключение

### В данной работе были рассмотрены основные механизмы сбора системной информации в ОС Windows.

### В отличии от мира linux, сбор системной информации осложнён разнообразностью форм её представления и разбросанностью по всей ОС.

### Разработанная программа позволяет получить следующую информацию:

### время пользователя и UTC время;

### имя пользователя и имя хоста;

### версию операционной системы, с точностью до номера сервиспака и сборки;

### производителя центрального процессора и доступных ядрах;

### различные параметры оперативной памяти (как физической так и файла подкачки);

### параметры работы видеосистемы;

### локальные файловые накопители и используемые файловые системы;

### параметры работы сетевой системы.

### Корректность работы программы была проверена путём сверки данных с другими систем- ными источниками (лог-файлы находятся в общем архиве, в папке с логами). Данный код можно использовать в качестве динамической библиотеки в других, более масштабных проектах.