

*Department of Software Engineering*

***Bahria University Karachi Campus***

***Course Title: System Programming***

***Course code:******CEN 449***

***Course instructor:******Engr Rizwan Fazal***

***Lab instructor:******Engr Rehan Baig***

***< Process Monitoring System >***

***BSE. (5(B))***

***SYSTEM PROGRAMMING SEMESTER PROJECT REPORT***

***<MUSTUFA> <02-131202-012>***

***<Talha Zafar> <02-131202-080>***

***<Muaz Shahzad> <02-131202-081>***

***<HasSaN AKHTAR> <02-131202-019>***

***Table of Contents***

* 1. *Acknowledgement*
  2. *Abstract*
  3. *Introduction*
  4. *Project Scope*
  5. *Technologies*
  6. *Project code*
  7. *Project Snaps*

***acknowledgement***

*I take this occasion to thank God, almighty for blessing us withhis grace and taking our endeavor to a successful culmination*

***Abstract***

Process monitoring systems are used to monitor the real-time status of all processes. They are used to track production and performance of a process, as well as to identify any abnormalities or inefficiencies. The system can also be used to provide feedback to operators, alert them of problems and suggest corrective action. The system can also be used to analyze performance trends. The system usually consists of handles, pointers, controllers, displays, and data collection and analysis software.

***introduction***

The fast growth rate of the online grocery market presents a challenge for

supermarket chains competing for share, in terms of balancing their online and ofﬂine

investments. A better understanding of the triggers which inﬂuence the adoption (and

the discontinuation) of online grocery shopping is vital for the strategic management of

this sector,

The fast growth rate of the online grocery market presents a challenge for

supermarket chains competing for share, in terms of balancing their online and ofﬂine

investments. A better understanding of the triggers which inﬂuence the adoption (and

the discontinuation) of online grocery shopping is vital for the strategic management of

this sector,

The fast growth rate of the online grocery market presents a challenge for

supermarket chains competing for share, in terms of balancing their online and ofﬂine

investments. A better understanding of the triggers which inﬂuence the adoption (and

the discontinuation) of online grocery shopping is vital for the strategic management of

this sector,

The fast growth rate of the online grocery market presents a challenge for

supermarket chains competing for share, in terms of balancing their online and ofﬂine

investments. A better understanding of the triggers which inﬂuence the adoption (and

the discontinuation) of online grocery shopping is vital for the strategic management of

this sector,

Process monitoring systems are used to monitor the execution of processes in an operating system. It provides a way to monitor the performance, resource utilization, and other aspects of running processes. Process monitoring systems collect information about running processes, such as their CPU and memory utilization, threads information, and report the data to a central server. Process monitoring systems can also be used to detect malicious activity on a system, such as malware infection or data theft. They can also be used to monitor user activity, such as which applications and websites are being used.

***PROJECT SCOPE***

The scope of process monitoring system in operating system includes the following:

1. Monitoring of system resources such as CPU, memory, disk, and network usage.

2. Analysis of process execution, including identification of processes that are consuming system resources, identification of processes that are not responding, and identification of abnormal system behavior.

3. Implementation of policies for resource utilization, including setting limits on resource usage and process execution.

4. Automated alerting of system administrators in case of any abnormal system behavior.

5. Automated or manual process suspension or termination in case of any abnormal system behavior.

***TECHNOLOGIES***

*We work on a visual studio in which we use C++ and C language.*

***PROJECT CODE***

#include <windows.h>

#include <tlhelp32.h>

#include <tchar.h>

#include <iostream>

// Forward declarations:

BOOL GetProcessList();

BOOL ListProcessModules(DWORD dwPID);

BOOL ListProcessThreads(DWORD dwOwnerPID);

void printError(const TCHAR\* msg);

int main(void)

{

//printf("");

printf("");

printf("\n\t\t\t\t|=============================|");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t| \*PROCESS MONITORING SYSTEM\* | ");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t|=============================|");

printf("\n");

printf("\n");

printf("");

printf("\n\t\t\t\t|=============================|");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t| \*Prepared By: Muaz & Talha \*| ");

printf("\n\t\t\t\t| Mustufa & Hassan \*| ");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t|=============================|");

printf("\n");

printf("\n");

GetProcessList();

printf("\n");

printf("Task Completed Succesfully!!!!!");

printf("\n");

system("pause");

return 0;

}

BOOL GetProcessList()

{

HANDLE hProcessSnap;

HANDLE hProcess;

PROCESSENTRY32 pe32;

DWORD dwPriorityClass;

// Take a snapshot of all processes in the system.

hProcessSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hProcessSnap == INVALID\_HANDLE\_VALUE)

{

printError(TEXT("CreateToolhelp32Snapshot (of processes)"));

return(FALSE);

}

// Set the size of the structure before using it.

pe32.dwSize = sizeof(PROCESSENTRY32);

// Retrieve information about the first process,

// and exit if unsuccessful

if (!Process32First(hProcessSnap, &pe32))

{

printError(TEXT("Process32First")); // show cause of failure

CloseHandle(hProcessSnap); // clean the snapshot object

return(FALSE);

}

// Now walk the snapshot of processes, and

// display information about each process in turn

int itr=0;

do

{

itr++;

hProcess = OpenProcess(PROCESS\_ALL\_ACCESS, FALSE, pe32.th32ProcessID);

if (hProcess == NULL) {

//printError(TEXT("OpenProcess"));

//printf("cONTINUES THIS ONE1--------");

continue;

}

else

{

dwPriorityClass = GetPriorityClass(hProcess);

if (!dwPriorityClass) {

printError(TEXT("GetPriorityClass"));

printf("cONTINUES THIS ONE2--------");

}

CloseHandle(hProcess);

}

// Retrieve the priority class.

dwPriorityClass = 0;

\_tprintf(TEXT("\n%d:\tPROCESS NAME: %.15s"),itr,pe32.szExeFile);

\_tprintf(TEXT("\t Process ID = 0x%08X"), pe32.th32ProcessID);

\_tprintf(TEXT("\n Thread count = %d"), pe32.cntThreads);

\_tprintf(TEXT("\n Parent process ID = 0x%08X"), pe32.th32ParentProcessID);

\_tprintf(TEXT("\n Priority base = %d"), pe32.pcPriClassBase);

/\*if (dwPriorityClass)

\_tprintf(TEXT("\n Priority class = %d"), dwPriorityClass);

// List the modules and threads associated with this process

/\*ListProcessModules(pe32.th32ProcessID);

ListProcessThreads(pe32.th32ProcessID);\*/

} while (Process32Next(hProcessSnap, &pe32));

std::cout << "\nEnter process Id for displaying thread information: ";

int myId;

std::cin >> myId;

while (myId<100 || myId>itr) {

std::cout << "Please Enter Correct ID: ";

std::cin >> myId;

}

PROCESSENTRY32 selectedProcess;

selectedProcess.dwSize = sizeof(PROCESSENTRY32);

if (!Process32First(hProcessSnap, &selectedProcess))

{

printError(TEXT("Process32First")); // show cause of failure

CloseHandle(hProcessSnap); // clean the snapshot object

return(FALSE);

}

for (int i = 1; i < myId; i++) {

Process32Next(hProcessSnap, &selectedProcess);

}

printf("\n");

printf("Selected Process is.....");

printf("\n");

\_tprintf(TEXT("%d\nPROCESS NAME : %.15s"),myId,selectedProcess.szExeFile);

\_tprintf(TEXT("\t\nProcess ID = 0x%08X"), selectedProcess.th32ProcessID);

\_tprintf(TEXT("\nThread count = %d"), selectedProcess.cntThreads);

\_tprintf(TEXT("\nParent process ID = 0x%08X"), selectedProcess.th32ParentProcessID);

\_tprintf(TEXT("\nPriority base = %d"), selectedProcess.pcPriClassBase);

printf("\n");

system("pause");

ListProcessThreads(selectedProcess.th32ProcessID);

ListProcessModules(selectedProcess.th32ProcessID);

CloseHandle(hProcessSnap);

return(TRUE);

}

BOOL ListProcessModules(DWORD dwPID)

{

HANDLE hModuleSnap = INVALID\_HANDLE\_VALUE;

MODULEENTRY32 me32;

// Take a snapshot of all modules in the specified process.

hModuleSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPMODULE, dwPID);

if (hModuleSnap == INVALID\_HANDLE\_VALUE)

{

printError(TEXT("CreateToolhelp32Snapshot (of modules)"));

return(FALSE);

}

// Set the size of the structure before using it.

me32.dwSize = sizeof(MODULEENTRY32);

// Retrieve information about the first module,

// and exit if unsuccessful

if (!Module32First(hModuleSnap, &me32))

{

printError(TEXT("Module32First")); // show cause of failure

CloseHandle(hModuleSnap); // clean the snapshot object

return(FALSE);

}

// Now walk the module list of the process,

// and display information about each module

do

{

\_tprintf(TEXT("\n\nMODULE NAME: %s"), me32.szModule);

\_tprintf(TEXT("\nExecutable = %s"), me32.szExePath);

\_tprintf(TEXT("\nProcess ID = 0x%08X"), me32.th32ProcessID);

\_tprintf(TEXT("\nRef count (g) = 0x%04X"), me32.GlblcntUsage);

\_tprintf(TEXT("\nRef count (p) = 0x%04X"), me32.ProccntUsage);

\_tprintf(TEXT("\nBase address = 0x%08X"), (DWORD)me32.modBaseAddr);

\_tprintf(TEXT("\nBase size = %d"), me32.modBaseSize);

printf("\n");

system("pause");

} while (Module32Next(hModuleSnap, &me32));

CloseHandle(hModuleSnap);

return(TRUE);

}

BOOL ListProcessThreads(DWORD dwOwnerPID)

{

HANDLE hThreadSnap = INVALID\_HANDLE\_VALUE;

THREADENTRY32 te32;

// Take a snapshot of all running threads

hThreadSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPTHREAD, 0);

if (hThreadSnap == INVALID\_HANDLE\_VALUE)

return(FALSE);

// Fill in the size of the structure before using it.

te32.dwSize = sizeof(THREADENTRY32);

// Retrieve information about the first thread,

// and exit if unsuccessful

if (!Thread32First(hThreadSnap, &te32))

{

printError(TEXT("Thread32First")); // show cause of failure

CloseHandle(hThreadSnap); // clean the snapshot object

return(FALSE);

}

// Now walk the thread list of the system,

// and display information about each thread

// associated with the specified process

do

{

if (te32.th32OwnerProcessID == dwOwnerPID)

{

\_tprintf(TEXT("\n\nTHREAD ID = 0x%08X"), te32.th32ThreadID);

\_tprintf(TEXT("\nBase priority = %d"), te32.tpBasePri);

\_tprintf(TEXT("\nDelta priority = %d"), te32.tpDeltaPri);

\_tprintf(TEXT("\n"));

system("pause");

}

} while (Thread32Next(hThreadSnap, &te32));

CloseHandle(hThreadSnap);

return(TRUE);

}

void printError(const TCHAR\* msg)

{

DWORD eNum;

TCHAR sysMsg[256];

TCHAR\* p;

eNum = GetLastError();

FormatMessage(FORMAT\_MESSAGE\_FROM\_SYSTEM | FORMAT\_MESSAGE\_IGNORE\_INSERTS,

NULL, eNum,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT), // Default language

sysMsg, 256, NULL);

// Trim the end of the line and terminate it with a null

p = sysMsg;

while ((\*p > 31) || (\*p == 9))

++p;

do { \*p-- = 0; } while ((p >= sysMsg) &&

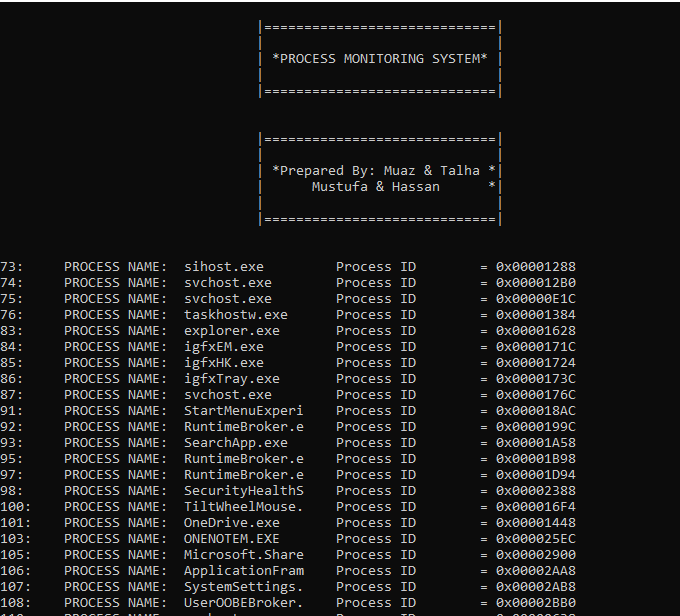
((\*p == '.') || (\*p < 33)));

// Display the message

\_tprintf(TEXT("\n WARNING: %s failed with error %d (%s)"), msg, eNum, sysMsg);

}

*Project Snaps:*



A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

***THE END***