Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра программной инженерии](https://www.belstu.by/fakultety/fit/vm)

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №3**

Дисциплина: Операционные системы

Тема: Процессы

Выполнила:

студентка 3 курса 5 группы

Городилина Анастасия Сергеевна

Минск 2024

**Задание 01**

int main()

{

for (int i = 0; i < 1000; i++)

{

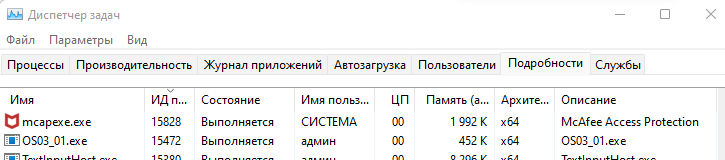
cout << "PID: " << GetCurrentProcessId() << endl;

Sleep(2000);

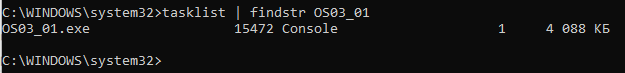
}

}

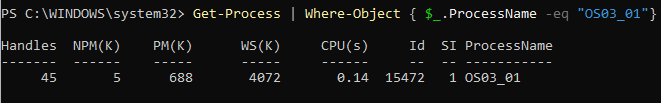
Task manager



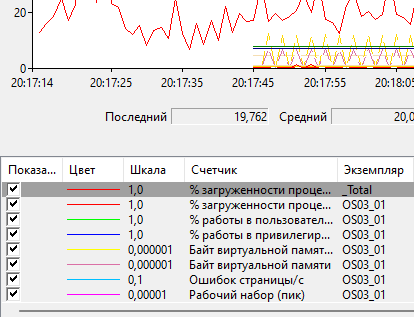
Tasklist



PowerShell



Performance monitor



**Задание 02**

int main()

{

for (int i = 0; i < 50; i++)

{

cout << "PID: " << GetCurrentProcessId() << endl;

Sleep(1000);

}

}

int main()

{

LPCWSTR path1 = L"C:\\AllMine\\Labs\\ОС\\Lab3\\OS03\_01\\Debug\\OS03\_02\_1.exe";

LPCWSTR path2 = L"C:\\AllMine\\Labs\\ОС\\Lab3\\OS03\_01\\Debug\\OS03\_02\_2.exe";

STARTUPINFO si;

PROCESS\_INFORMATION pi1;

PROCESS\_INFORMATION pi2;

ZeroMemory(&si, sizeof(STARTUPINFO));

si.cb = sizeof(STARTUPINFO);

if (CreateProcess(path1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi1))

{

std::cout << "Child process OS\_02\_1 has been created" << endl;

}

else std::cout << "Child process OS\_02\_1 hasn't been created" << endl;

if (CreateProcess(path2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi2))

{

std::cout << "Child process OS\_02\_2 has been created" << endl;

}

else std::cout << "Child process OS\_02\_2 hasn't been created" << endl;

for (int i = 0; i < 100; i++)

{

cout << "PID: " << GetCurrentProcessId() << endl;

Sleep(1000);

}

WaitForSingleObject(pi1.hProcess, INFINITE);

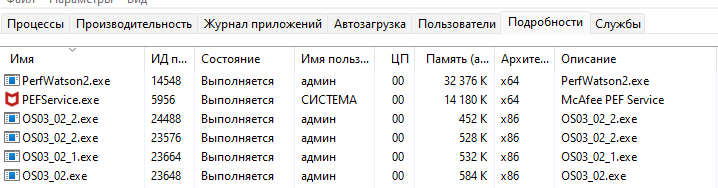
WaitForSingleObject(pi2.hProcess, INFINITE);

CloseHandle(pi1.hProcess);

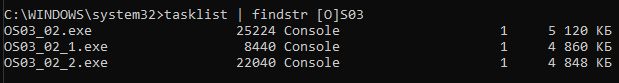
CloseHandle(pi2.hProcess);

}

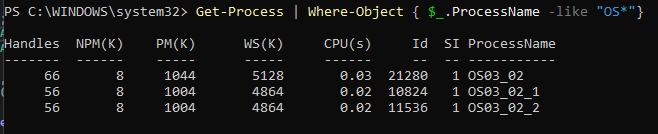
Task manager



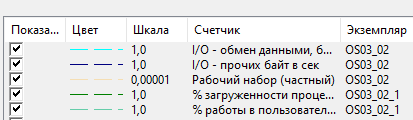
Tasklist

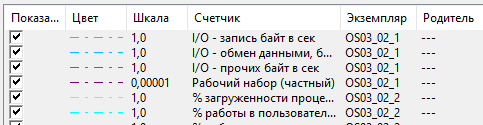


Powershell



Performance monitor





**Задание 03**

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <TlHelp32.h>

using namespace std;

int main()

{

DWORD pid = GetCurrentProcessId();

HANDLE snap = CreateToolhelp32Snapshot(TH32CS\_SNAPALL, 0);

PROCESSENTRY32 peProcessEntry;

peProcessEntry.dwSize = sizeof(PROCESSENTRY32);

wcout << L"Current PID = " << pid << endl << L"-----------------------------------" << endl;

try

{

if (!Process32First(snap, &peProcessEntry))

throw L"Process32First";

do

{

wcout << L"Name\t\t" << peProcessEntry.szExeFile << endl

<< L"PID\t\t" << peProcessEntry.th32ProcessID;

if (peProcessEntry.th32ProcessID == pid)

wcout << " --> current process";

wcout << L"\nParent ID\t" << peProcessEntry.th32ParentProcessID;

wcout << endl << L"-----------------------------------" << endl;

} while (Process32Next(snap, &peProcessEntry));

}

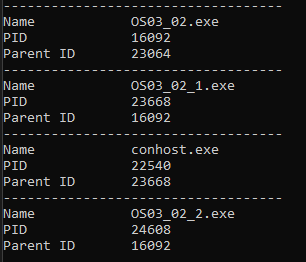
catch (char\* msg)

{

wcout << L"ERROR: " << msg << endl;

}

}



**Задание 04**

int main(void) {

for (int i = 1; i <= 1000; i++) {

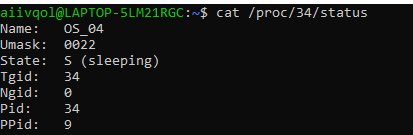
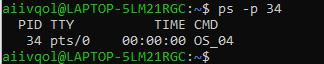
printf("%d-%d\n", \_getpid(), i);

Sleep(1000);

}

return 0;

}

**Задание 05**

void cycle(int counter, char\* message)

{

for (int i = 1; i <= counter; ++i)

{

printf("%d [%s]: %d\n", getpid(), message, i);

sleep(2);

}

}

int main()

{

pid\_t pid;

switch (pid = fork())

{

case -1: perror("Error");

exit(-1);

case 0: cycle(50, "OS03\_05\_1");

exit(0);

default: cycle(100, "OS03\_05");

wait(0);

}

exit(0);

}

int main()

{

for (int i = 1; i <= 50; ++i)

{

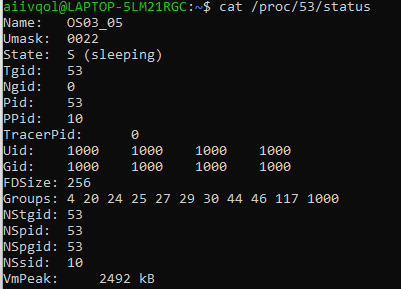
printf("%d [OS03\_05\_1]: %d\n", getpid(), i);

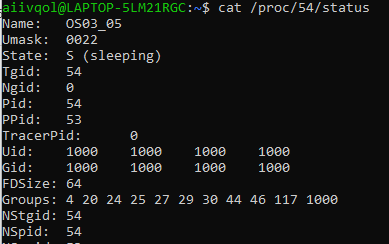
sleep(1);

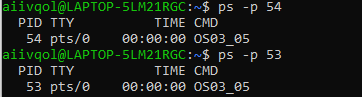
}

exit(0);

}







**Задание 06**

int main()

{

system("./OS03\_05\_1");

for (int i = 1; i <= 100; ++i)

{

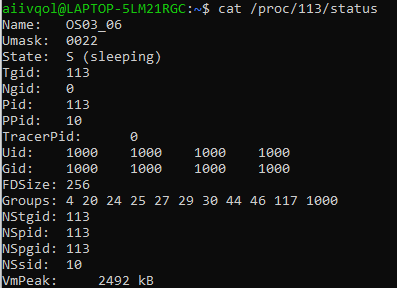
printf("%d [OS03\_06]: %d\n", getpid(), i);

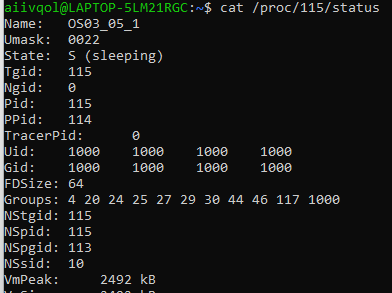
sleep(1);

}

exit(0);

}









**Задание 07**

int main()

{

char\* const argv[] = { "ps", "-ef", 0 };

execv("./OS03\_05\_1", argv);

for (int i = 1; i <= 100; ++i)

{

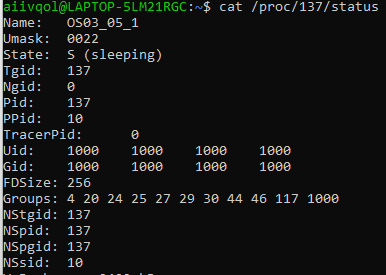
printf("%d [OS03\_07]: %d\n", getpid(), i);

sleep(1);

}

exit(0);

}





system — создаёт новый процесс, exec — заменяет текущий на новый