Белорусский государственный технологический университет
Факультет информационных технологий
Кафедра программной инженерии

Лабораторная работа 3

По дисциплине «Операционные системы»

На тему «Управление процессами»

Выполнил:
Студент 3 курса 9 группы
Павлович Ян Андреевич
Преподаватель: Савельева М.Г.

Минск, 2025

Введение

Цель работы:

Получение практических навыков системного программирования, связанных с созданием, управлением и анализом процессов в операционных системах Windows и Linux. В ходе работы требуется научиться использовать системные вызовы и функции API для организации многопоточных приложений, освоить передачу данных между процессами с помощью аргументов командной строки и переменных окружения, а также ознакомиться с инструментами диагностики и мониторинга процессов.

Задачи работы:

1. Разработать несколько приложений для ОС Windows.
2. Разработать несколько приложений для ОС Linux.

Используемые инструменты:

- Parallels Desktop
- Командная оболочка cmd
- Process Explorer
- Компилятор g++

1 Windows

1.1 Приложение Lab-03x

Программа выполняет простой цикл с заданным количеством итераций, где каждая итерация сопровождается задержкой в 500 мс.
Перед началом работы программа выводит количество итераций, а затем — PID процесса на каждой итерации.
Количество итераций передается через аргументы командной строки. Если аргумент отсутствует, программа пытается получить значение из переменной окружения ITER_NUM. При отсутствии и аргумента, и переменной окружения программа завершает работу с ошибкой с помощью ExitProcess.

Создаём файл *Lab-03x.cpp* и записываем в него код (листинг 1.1).

После этого компилируем исходный файл в исполняемый (*.exe*) и запускаем программу через командную строку, передавая параметры либо задавая переменную окружения.

```
C:\lab3\x64\Debug>lab3x.exe
Error: no number of iterations is specified

C:\lab3\x64\Debug>lab3x.exe 4
Number of iterations: 4
Iteration 1, PID: 6592
Iteration 2, PID: 6592
Iteration 3, PID: 6592
Iteration 4, PID: 6592

C:\lab3\x64\Debug>set ITER_NUM=4

C:\lab3\x64\Debug>lab3x.exe
Number of iterations: 4
Iteration 1, PID: 8880
Iteration 2, PID: 8880
Iteration 3, PID: 8880
Iteration 4, PID: 8880
```

Рисунок 1.1 – Запуск приложения Lab-03x

При корректной передаче параметров программа отрабатывает без ошибок. Если не задать ни аргумент, ни переменную окружения, приложение завершает выполнение с сообщением об ошибке.

1.2 Приложение Lab-03a:

Программа создает три дочерних процесса с помощью функции CreateProcess, каждый из которых запускает приложение *Lab-03x*.

Условия запуска:

- Первый процесс**:** используется только первый параметр функции CreateProcess. В этот же параметр передается и имя программы, и количество итераций.
- Второй процесс**:** используется только второй параметр функции CreateProcess, куда передается строка.
- Третий процесс**:** первый параметр содержит имя программы, второй — количество итераций (перед ним может потребоваться пробел).

Родительский процесс должен дождаться завершения всех дочерних процессов, выполняющихся одновременно. Необходимо предусмотреть обработку ошибок и освобождение ресурсов.

Для анализа созданных процессов используются Process Explorer, cmd и Task Manager. В Process Explorer следует убедиться в наличии дескрипторов дочерних процессов у родительского. Создаём файл *Lab-03a.cpp* (листинг 1.2), компилируем и запускаем программу.

```
C:\lab3\x64\Debug>lab3a.exe
Error when creating the process 1: 2
Press child processes are completed.
Number of iterations: 5
Iteration 1, PID: 11692
Number of iterations: 5
Iteration 1, PID: 10436
Iteration Iteration 2, PID: 2, PID: 10436
11692
Iteration Iteration 33, PID: , PID: 1169210436

Iteration Iteration 44, PID: , PID: 1043116962

Iteration 5, PID: 11692
Iteration 5, PID: 10436
```

Рисунок 1.2 – Приложение Lab-03a

```
C:\>tasklist | findstr lab3x
lab3x.exe                    11692 Console                    1      4,284 K
lab3x.exe                    10436 Console                    1      4,272 K
```

Рисунок 1.3 – Проверка процессов через cmd

Затем проверим процессы через Task Manager

Name	PID	Status	User name	CPU	Worki...	Platform	UAC virtualizati...
lab3a.exe	12088	Running	yanpaulovich	00	0 K	64 bit	Disabled
lab3x.exe	11604	Running	yanpaulovich	00	0 K	64 bit	Disabled
lab3x.exe	7544	Running	yanpaulovich	00	0 K	64 bit	Disabled

Рисунок 1.4 – Проверка процессов через Task Manager

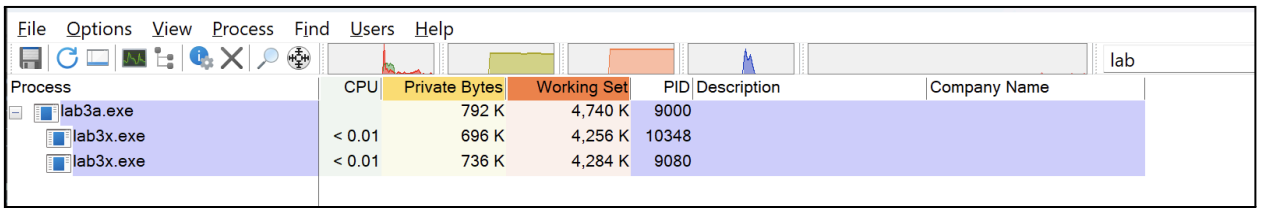

Рисунок 1.5 – Проверка процессов через Process Explorer

Первый процесс не создается из-за ошибки — система не находит файл, так как аргумент был передан в параметре lpApplicationName, который должен содержать только путь к исполняемому файлу. Второй и третий процессы создаются корректно, так как параметры были переданы через lpCommandLine, что соответствует требованиям функции CreateProcess.

После запуска выполняется проверка созданных процессов через cmd, Task Manager и Process Explorer.

В дереве процессов видно, что Lab-03a является родительским, а Lab-03x — дочерними. Всего успешно запущено два дочерних процесса.

1.3 Приложение Lab-03b

Данное приложение аналогично *Lab-03a*, но с измененными параметрами вызова процессов:

- В первом вызове CreateProcess передается только имя файла (аргумент с количеством итераций отсутствует).
- В третьем вызове вторым параметром передается NULL, при этом в коде заранее устанавливается локальная переменная окружения ITER_NUM, значение которой отличается от глобальной.

Создаём файл *Lab-03b.cpp* (листинг 1.3), компилируем и запускаем.

```
C:\lab3\x64\Debug>set ITER_NUM=3

C:\lab3\x64\Debug>lab3b.exe
Number of iterations: 3
Iteration 1, PID: 11404
Number of iterations: 4
Iteration 1, PID: 7740
Number of iterations: 5
Iteration 1, PID: 2156
Iteration 2, PID: 11404
Iteration Iteration 2, PID: 7740
2, PID: 2156
Iteration 3, PID: 11404
Iteration Iteration 3, PID: 73740, PID:
2156
Iteration Iteration 4, PID: 74, PID: 740
2156
Iteration 5, PID: 2156
All child processes are completed.
```

Рисунок 1.6 – Приложение Lab-03b

Все процессы были успешно созданы.

- Первый процесс выполняется 3 раза (значение взято из глобальной переменной окружения).
- Второй — 4 раз (значение передано через аргумент командной строки).
- Третий — 5 раз (значение локальной переменной окружения).

1.4 Приложение Lab-03c

Программа выводит список всех запущенных в системе процессов, включая их имена, идентификаторы (PID) и идентификаторы родительских процессов (PPID). Для реализации используется Tool Help Library.

Создаём файл *Lab-03c.cpp* (листинг 1.4), компилируем и запускаем.

```
C:\lab3\x64\Debug>lab3c.exe
List of running processes:
-----------------------------------------
Process Name: [System Process], PID: 0, PPID: 0
Process Name: System, PID: 4, PPID: 0
Process Name: Registry, PID: 124, PPID: 4
Process Name: smss.exe, PID: 520, PPID: 4
Process Name: csrss.exe, PID: 704, PPID: 688
Process Name: wininit.exe, PID: 784, PPID: 688
Process Name: csrss.exe, PID: 792, PPID: 776
Process Name: winlogon.exe, PID: 856, PPID: 776
Process Name: services.exe, PID: 932, PPID: 784
Process Name: lsass.exe, PID: 952, PPID: 784
Process Name: svchost.exe, PID: 656, PPID: 932
Process Name: fontdrvhost.exe, PID: 780, PPID: 856
Process Name: fontdrvhost.exe, PID: 776, PPID: 784
Process Name: svchost.exe, PID: 1052, PPID: 932
Process Name: svchost.exe, PID: 1112, PPID: 932
Process Name: dwm.exe, PID: 1176, PPID: 856
Process Name: svchost.exe, PID: 1284, PPID: 932
Process Name: svchost.exe, PID: 1324, PPID: 932
Process Name: svchost.exe, PID: 1416, PPID: 932
Process Name: svchost.exe, PID: 1428, PPID: 932
Process Name: svchost.exe, PID: 1524, PPID: 932
Process Name: svchost.exe, PID: 1532, PPID: 932
Process Name: svchost.exe, PID: 1580, PPID: 932
Process Name: svchost.exe, PID: 1616, PPID: 932
Process Name: svchost.exe, PID: 1640, PPID: 932
Process Name: svchost.exe, PID: 1680, PPID: 932
```

Рисунок 1.7 – Приложение Lab-03c

На экране отображается полный список активных процессов с их именами, PID и PPID.

2 Linux

2.1 Приложение Lab-03x

Программа полностью аналогична Windows-варианту. Создаём файл *Lab-03x.cpp* (листинг 2.1), компилируем и запускаем.

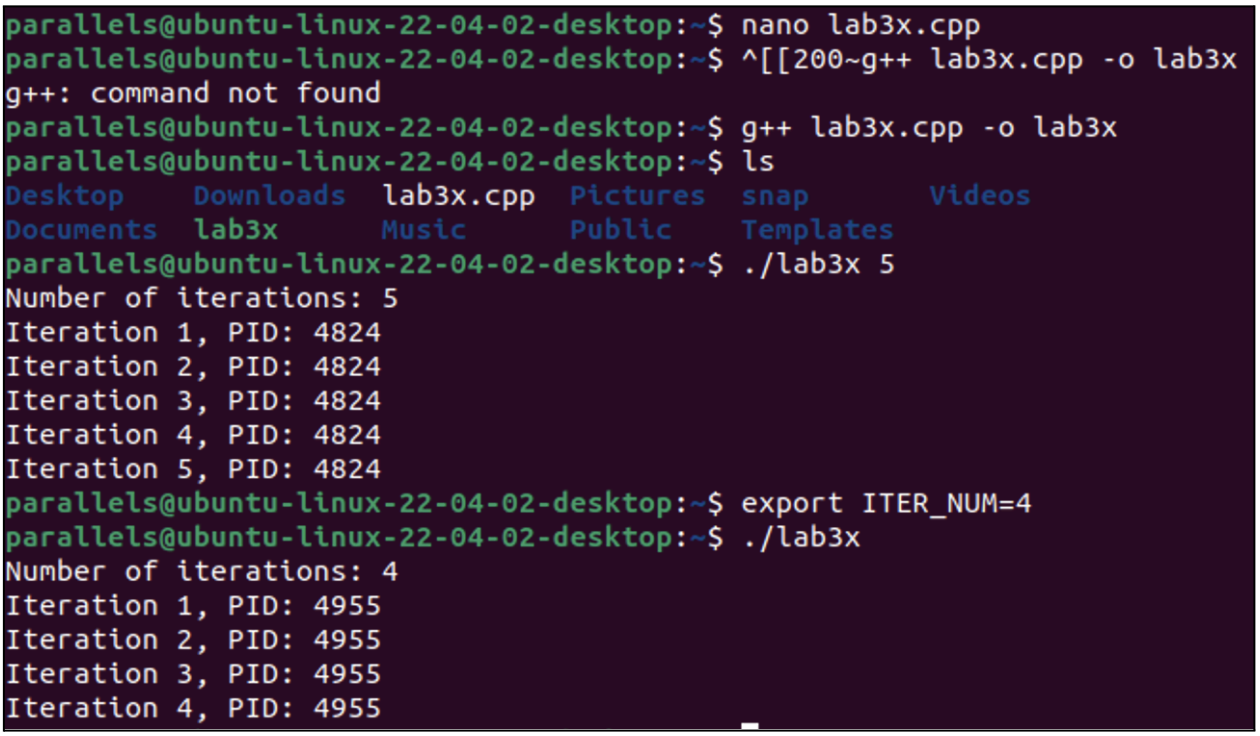

Рисунок 2.1 – Приложение Lab-03x

Приложение корректно выполняет цикл с заданным числом итераций, как при передаче параметра через командную строку, так и через переменную окружения.

2.2 Приложение Lab-03a

Программа создает два дочерних процесса, загружая в них исполняемый файл *Lab-03x*.

Первый дочерний процесс получает количество итераций через аргумент командной строки. Второй — через переменную окружения ITER_NUM, которая создаётся в родительском процессе перед его запуском.

Родительский процесс ожидает завершения обоих дочерних процессов. Для анализа используются каталог **/proc** и утилита **ps**. Необходимо также предусмотреть обработку ошибок и очистку ресурсов.

Создаём файл *Lab-03a.cpp* (листинг 2.2), компилируем и запускаем.

```
parallels@ubuntu-linux-22-04-02-desktop:~$ ./lab3a 5
Number of iterations: 5
Iteration 1, PID: 6668
Number of iterations: 7
Iteration 1, PID: 6669
Iteration 2, PID: 6668
Iteration 2, PID: 6669
Iteration 3, PID: 6668
Iteration 3, PID: 6669
Iteration 4, PID: 6669
Iteration 4, PID: 6668
Iteration 5, PID: 6668
Iteration 5, PID: 6669
Iteration 6, PID: 6669
Iteration 7, PID: 6669
Both child processes are completed.
```

Рисунок 2.2 – Приложение Lab-03a

```
parallels@ubuntu-linux-22-04-02-desktop:~$ ps -ef | grep lab3x
paralle+    6358    6357  0 22:00 pts/0    00:00:00 ./lab3x 5
paralle+    6359    6357  0 22:00 pts/0    00:00:00 ./lab3x
paralle+    6367    6195  0 22:00 pts/1    00:00:00 grep --color=auto lab3x
parallels@ubuntu-linux-22-04-02-desktop:~$
```

Рисунок 2.3 – Использование утилиты ps

```
parallels@ubuntu-linux-22-04-02-desktop:~$ ls /proc | grep -E '^[0-9]+$' | xargs
 -I{} sh -c 'cat /proc/{}/comm 2>/dev/null | grep lab3x && echo PID: {}'
lab3x
PID: 8027
lab3x
PID: 8028
```

Рисунок 2.4 – Использование /proc

Оба процесса были успешно созданы. При помощи утилиты **ps** получены PID дочерних процессов. Через файловую систему **/proc** были просмотрены параметры: командная строка, переменные окружения, имя процесса, состояние, PID и использование памяти.

Заключение

В ходе выполнения лабораторной работы были получены практические навыки работы с процессами в операционных системах Windows и Linux.
В среде Windows были реализованы приложения Lab-03x, Lab-03a, Lab-03b и Lab-03c, демонстрирующие создание дочерних процессов с помощью CreateProcess, передачу параметров через аргументы командной строки и переменные окружения, а также обработку ошибок и корректное завершение процессов. Отдельное внимание уделено изучению дескрипторов процессов и мониторингу через Process Explorer, Task Manager и командную строку.
В Linux были разработаны аналоги этих программ с использованием системных вызовов fork, exec, waitpid, а также анализа процессов через файловую систему /proc и утилиту ps. Реализованы сценарии передачи параметров через аргументы и переменные окружения, предусмотрена обработка ошибок и корректное завершение дочерних процессов.
Сравнение поведения процессов в Windows и Linux позволило выявить архитектурные различия между платформами и углубить понимание принципов системного программирования. Полученные знания создают основу для дальнейшего изучения многозадачности, межпроцессного взаимодействия и анализа процессов в современных операционных системах.

Приложение А – Листинг А.1

```
#define _CRT_SECURE_NO_WARNINGS
#include <windows.h>
#include <iostream>
#include <cstdlib>
int main(int argc, char* argv[]) {
    int iterations = 0;
    if (argc > 1) {
        iterations = std::atoi(argv[1]);
    }
    else {
        char* envVar = std::getenv("ITER_NUM");
        if (envVar != nullptr) {
            iterations = std::atoi(envVar);
        }
        else {
            std::cerr << "Error: no number of iterations is specified" <<
std::endl;
            ExitProcess(1);
        }
    }
    std::cout << "Number of iterations: " << iterations << std::endl;
    DWORD pid = GetCurrentProcessId();
    for (int i = 0; i < iterations; ++i) {
        std::cout << "Iteration " << (i + 1) << ", PID: " << pid << std::endl;
        Sleep(500);
    }
    return 0;
}
```

Приложение Б – Листинг Б.1

```
#undef UNICODE
#undef _UNICODE
#include <windows.h>
#include <iostream>
#include <string>
int main() {
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi[3];
    BOOL success;
    std::string exePath = "C:\\lab3\\x64\\Debug\\lab3x.exe";
    std::string arg = "5";

    // 1
    std::string full1 = exePath + " " + arg;
    success = CreateProcess(
        full1.c_str(), // ← всё в первом параметре
        NULL,
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[0]);
    if (!success) {
        std::cerr << "Error when creating the process 1: " << GetLastError()
<< std::endl;
    }

    // 2
    std::string full2 = exePath + " " + arg;
    success = CreateProcess(
        NULL,
        (LPSTR)full2.c_str(),
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[1]);
    if (!success) {
        std::cerr << "Error when creating the process 2: " << GetLastError()
<< std::endl;
    }
    // 3
    std::string arg3 = " " + arg;
    success = CreateProcess(
        exePath.c_str(),
        (LPSTR)arg3.c_str(),
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[2]);
    if (!success) {
        std::cerr << "Error when creating the process 3: " << GetLastError()
<< std::endl;
    }

    HANDLE handles[3] = { pi[0].hProcess, pi[1].hProcess, pi[2].hProcess };
    WaitForMultipleObjects(3, handles, TRUE, INFINITE);
    for (int i = 0; i < 3; ++i) {
        CloseHandle(pi[i].hProcess);
        CloseHandle(pi[i].hThread);
    }
```

```
    std::cout << "Press child processes are completed." << std::endl;
    std::cin.get();
    std::cout << "All child processes are completed." << std::endl;
    return 0;
}
```

Приложение В – Листинг В.1

```
#undef UNICODE
#undef _UNICODE
#include <windows.h>
#include <iostream>
#include <string>
int main() {
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi[3];
    BOOL success;
    std::string exePath = "C:\\lab3\\x64\\Debug\\lab3x.exe";
    std::string arg = "4";

    // 1
    success = CreateProcess(
        exePath.c_str(),
        NULL,
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[0]);
    if (!success) {
        std::cerr << "Error when creating the process 1: " << GetLastError()
<< std::endl;
    }

    // 2
    std::string full2 = "\"" + exePath + "\" " + arg;
    success = CreateProcess(
        NULL,
        (LPSTR)full2.c_str(),
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[1]);
    if (!success) {
        std::cerr << "Error when creating the process 2: " << GetLastError()
<< std::endl;
    }

    // 3
    SetEnvironmentVariableA("ITER_NUM", "5");
    success = CreateProcess(
        exePath.c_str(),
        NULL,
        NULL, NULL, FALSE, 0, NULL, NULL,
        &si, &pi[2]);
    if (!success) {
        std::cerr << "Error when creating the process 3: " << GetLastError()
<< std::endl;
    }

    HANDLE handles[3] = { pi[0].hProcess, pi[1].hProcess, pi[2].hProcess };
    WaitForMultipleObjects(3, handles, TRUE, INFINITE);
    for (int i = 0; i < 3; ++i) {
        CloseHandle(pi[i].hProcess);
        CloseHandle(pi[i].hThread);
    }
```

```
    std::cout << "All child processes are completed." << std::endl;
    return 0;}
```

Приложение Г – Листинг Г.1

```
#include <windows.h>
#include <tlhelp32.h>
#include <iostream>
int main() {
    HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (hSnapshot == INVALID_HANDLE_VALUE) {
        std::cerr << "Failed to create a snapshot of processes. Error code: "
<< GetLastError() << std::endl;
        return 1;
    }
    PROCESSENTRY32 pe32;
    pe32.dwSize = sizeof(PROCESSENTRY32);
    if (!Process32First(hSnapshot, &pe32)) {
        std::cerr << "Failed to create a snapshot of processes. Error code: "
<< GetLastError() << std::endl;
        CloseHandle(hSnapshot);
        return 1;
    }
    std::cout << "List of running processes:\n";
    std::cout << "-----------------------------------------\n";
    do {
        std::wcout << L"Process Name: " << pe32.szExeFile
            << L", PID: " << pe32.th32ProcessID
            << L", PPID: " << pe32.th32ParentProcessID << std::endl;
    } while (Process32Next(hSnapshot, &pe32));
    CloseHandle(hSnapshot);
    return 0;
}
```

Приложение Д – Листинг Д.1

```
#include <iostream>
#include <cstdlib>
#include <unistd.h> // для getpid() и usleep()

int main(int argc, char* argv[]) {
    int iterations = 0;

    if (argc > 1) {
        iterations = std::atoi(argv[1]);
    } else {
        const char* envVar = std::getenv("ITER_NUM");
        if (envVar != nullptr) {
            iterations = std::atoi(envVar);
        } else {
            std::cerr << "Error: no number of iterations is specified" <<
std::endl;
            return 1;
        }
    }

    std::cout << "Number of iterations: " << iterations << std::endl;
    pid_t pid = getpid();

    for (int i = 0; i < iterations; ++i) {
        std::cout << "Iteration " << (i + 1) << ", PID: " << pid << std::endl;
        usleep(500000); // 500 ms
    }

    return 0;
}
```

Приложение Е – Листинг Е.1

```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
#include <cstdlib>

int main(int argc, char* argv[]){
    const char* path = "./lab3x";
    const char* arg1 = argv[1];

    pid_t pid1 = fork();
    if (pid1 == 0) {
        execl(path, path, arg1, NULL);
        std::cerr << "Error starting the first process" << std::endl;
        exit(1);
    }

    pid_t pid2 = fork();
    if (pid2 == 0) {
        setenv("ITER_NUM", "7", 1);
        execl(path, path, NULL);
        std::cerr << "Error starting the second process" << std::endl;
        exit(1);
    }

    waitpid(pid1, NULL, 0);
    waitpid(pid2, NULL, 0);

    std::cout << "Both child processes are completed." << std::endl;
    return 0;
}
```