МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 4
На тему: "*Програмне створення та керування процесами в операційній системі LINUX*"
З дисципліни: "Операційні системи"

Лектор:
старший викладач кафедри ПЗ
Грицай О.Д.

Виконав:
студент групи ПЗ-22
Коваленко Д.М.

Прийняла:
старший викладач кафедри ПЗ
Грицай О.Д.

«_____» _____ 2022 р.
$\sum$ = _____.....

Львів — 2022

Тема. Програмне створення та керування процесами в операційній системі LINUX.

Мета. Ознайомитися з багатопоточністю в ОС Linux. Навчитися працювати з процесами, в ОС Linux.

Лабораторне завдання

1. Створити окремий процес, і здійснити в ньому розв'язок задачі згідно варіанту у відповідності до порядкового номера у журнальному списку (підгрупи).
2. Реалізувати розв'язок задачі у 2-ох, 4-ох, 8-ох процесах. Виміряти час роботи процесів за допомогою функцій WinAPI. Порівняти результати роботи в одному і в багатьох процесах
3. Для кожного процесу реалізувати можливість його запуску, зупинення, завершення та примусове завершення («вбиття»).
4. Реалізувати можливість зміни пріоритету виконання процесу
5. Продемонструвати результати виконання роботи, а також кількість створених процесів у "Диспетчері задач", або подібних утилітах (н-д, ProcessExplorer)

2. Вивести посортовані по зростанню методом «бульбашки» рядки матриці матриці $N \cdot N$ ($N > 1000$ задається користувачем, матриця визначається випадково).

Хід роботи

```
#include <stdio.h>
#include "stdlib.h"
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <iostream>
#include <string>
#include <fstream>
#include <sys/times.h>

using namespace std;

void gettimes(int pid);
void suspend(int pid);
void resume(int pid);
void kill(int pid);
void renice(int pid);

time_t time_begin;
int main(void) {
    int* pids;
    int pid;
    int status, PC = 0, N = 0;
    char op;

    while (N < 1) {
        cout << "Enter value of N (> 1000): ";
        cin >> N;
    }

    while (PC < 1) {
        cout << "Enter number of processes: ";
        cin >> PC;
    }

    pids = new int[PC];
```

```
cout << "Generating random array " << N << "x" << N << " ... ";

int** array = new int* [N];
for (int i = 0; i < N; i++) array[i] = new int[N];

srand(static_cast<unsigned int>(time(nullptr)));
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < N; j++)
    {
        array[i][j] = rand();
    }
}
cout << "Done!" << endl;

cout << "Writing array to file ... ";

ofstream file;
file.open("array.txt");
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < N; j++)
    {
        if (j == N-1)
        file << array[i][j];
        else
        file << array[i][j] << ",";
    }
    file << "\n";
}
file.close();

cout << "Done!" << endl;

time_begin = clock();

for (int i = 0; i < PC; i++) {
    pid = fork();
    switch(pid) {
        case -1:
        // Error
        break;
        case 0:
        printf("Started child process (pid: %d)\n", getpid());
        char s1[10];
        char s2[10];
        char s3[10];
        sprintf(s1, "%d", N);
        sprintf(s2, "%d",(N/PC)*i);
        sprintf(s3, "%d", (N/PC)*(i+1));
        execl("/bin/footclient", "--", "./process", s1, s2, s3, NULL);
        return 0;
        default:
        usleep(100000);
        string line;
        ifstream file("pid");
        getline(file, line);
        file.close();

        pids[i] = atoi(line.c_str());
        break;
```

```
        }
    }

    int i;
    while (true) {
        cout << "Suspend [s], Resume [r], Exit [e], Kill [k], Times [t], Priority [p]: ";
        cin >> op;
        if (op == 'e') break;
        cout << "Process index: ";
        cin >> i;
        if (op == 's') suspend(pids[i]);
        if (op == 'r') resume(pids[i]);
        if (op == 'k') kill(pids[i]);
        if (op == 't') gettimes(pids[i]);
        if (op == 'p') renice(pids[i]);
    }

    for (int i = 0; i < PC; i++) {
        kill(pids[i]);
    }

    return 0;
}

void gettimes(int pid) {
    char s[100];
    sprintf(s, "ps -o time %d | grep 0", pid);
    system(s);
}

void kill(int pid) {
    kill(pid, SIGKILL);
}

void resume(int pid) {
    kill(pid, SIGCONT);
}

void suspend(int pid) {
    cout << pid << endl;
    kill(pid, SIGSTOP);
}

void renice(int pid) {
    int nice;
    char cmd[100];
    cout << "New priority: ";
    cin >> nice;
    sprintf(cmd, "/bin/doas /bin/renice -n %d -p %d", nice, pid);
    system(cmd);
}
```

```
#include <iostream>
#include <chrono>
#include <fstream>
#include <string>
#include <vector>
#include <unistd.h>
```

```
using namespace std;
using namespace std::chrono;

void bubble_sort(int* array, int N);
vector<string> split(string s, string delimiter);

int main(int argc, char** argv)
{
    ofstream f;
    f.open("pid");
    f << getpid();
    f.close();

    if (argc < 4) return -1;

    int Ncol = atoi(argv[1]);
    int startRow = atoi(argv[2]);
    int endRow = atoi(argv[3]);
    int Nrow = endRow - startRow;

    cout << "Reading array from file " << Ncol << "x" << Nrow << " ... ";

    string line;
    ifstream file("/home/dmytro/lpnu/OS/lab4/array.txt");
    int** array = new int* [Nrow];
    for (int i = 0; i < startRow; i++)
    {
        getline(file, line);
    }
    for (int i = 0; i < Nrow; i++)
    {
        getline(file, line);
        array[i] = new int[Ncol];

        auto vec = split(line, ",");
        for (int j = 0; j < Ncol; j++)
        {
            array[i][j] = atoi(vec[j].c_str());
        }
    }
    file.close();

    cout << "Done!" << endl;

    cout << "Sorting array ... " << endl;

    auto start = high_resolution_clock::now();
    for (int i = 0; i < Nrow; i++)
    {
        cout << "Sorting row " << i+startRow << endl;
        bubble_sort(array[i], Ncol);
    }
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);
    cout << "Sorting took: " << duration.count() << "ms" << endl;

    cout << "Sorted array: " << endl;

    for (int i = 0; i < Nrow; i++)
    {
```

```
        for (int j = 0; j < Ncol; j++)
        {
            cout << array[i][j] << ", ";
        }
        cout << endl;
    }

    return 0;}

void bubble_sort(int* array, int N)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N - i - 1; j++)
        {
            if (array[j] > array[j + 1])
            {
                swap(array[j], array[j + 1]);
            }
        }
    }
}

vector<string> split(string s, string delimiter) {
    size_t pos_start = 0, pos_end, delim_len = delimiter.length();
    string token;
    vector<string> res;

    while ((pos_end = s.find(delimiter, pos_start)) != string::npos) {
        token = s.substr(pos_start, pos_end - pos_start);
        pos_start = pos_end + delim_len;
        res.push_back(token);
    }

    res.push_back(s.substr(pos_start));
    return res;
}
```

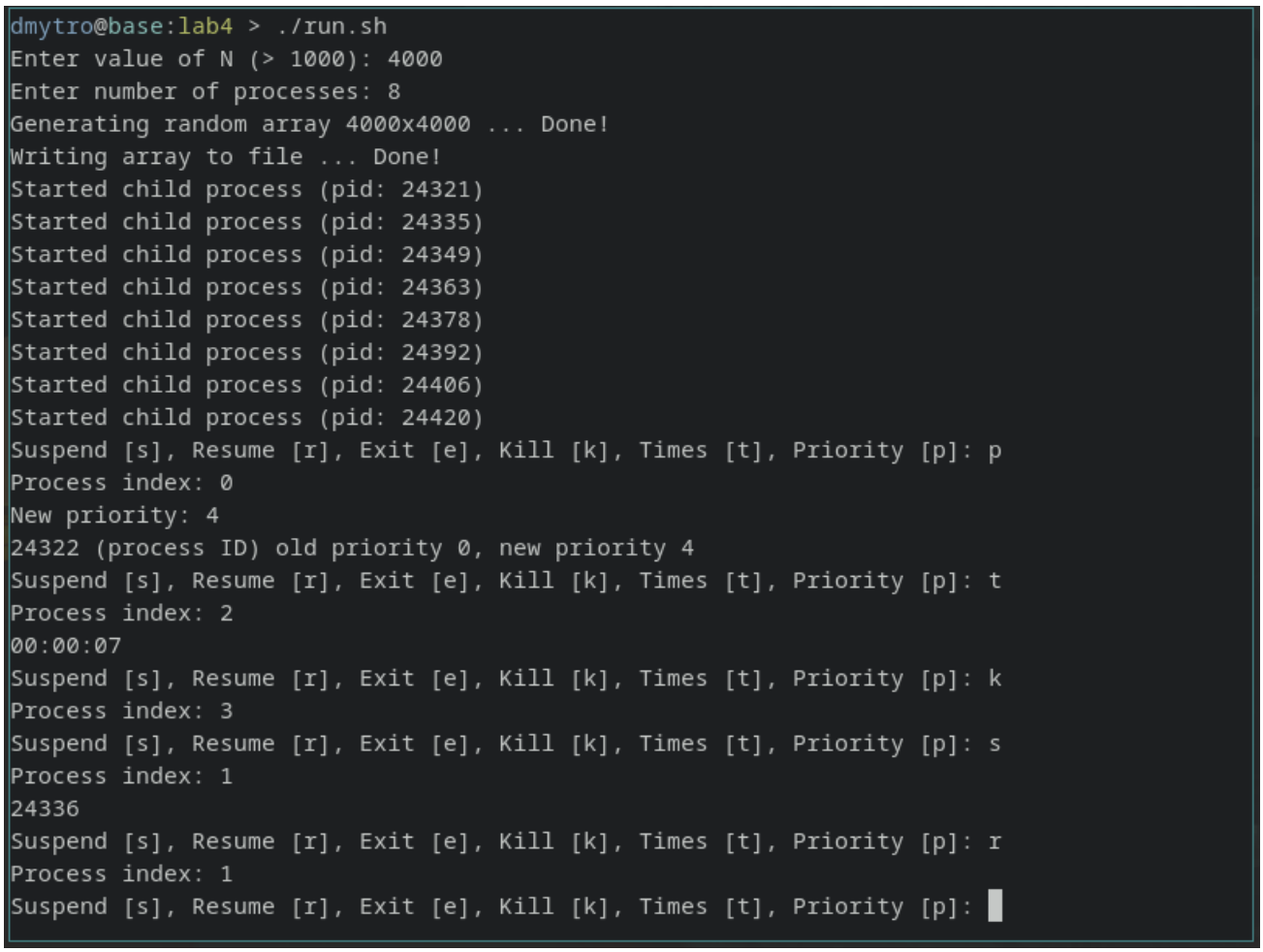

Рис. 1: Виконання програми

Висновок

Під час виконання лабораторної роботи я Ознайомився з багатопоточністю в ОС Linux. Навчився працювати з процесами, в ОС Linux. Навчився створювати нові процеси, призупиняти, завершувати та продовжувати їх роботу. Навчився отримувати та встановлювати пріоритет процесу. Навчився отримувати час виконання процесу.