НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 4**

**з дисципліни «Операційні системи»**

**Лектор:**

Ст. викладач Грицай О.Д.

**Виконав:**

студ. групи ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

Ст. викладач Грицай О.Д.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

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

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

**Теоретичні відомості**

Процес - визначають як екземпляр виконуваної програми. Кожному процесу при завантаженні виділяються ресурси операційної системи. Складові елементи процесу:

-об'єкт ядра, через який операційна система керує процесом і де зберігається статична інформація про процес

- адресний простір де зберігається код програми та дані.

При створенні процесу система створює таблицю дескрипторів, які використовуються для об’єктів ядра. Фактично це масив структур даних. Кожна структура містить вказівник на якийсь об'єкт ядра, маску доступу і деякі прапорці. Коли процес ініціалізується перший раз, таблиця пуста. Коли викликається функція що створює об'єкт ядра , ядро виділяє для цього об'єкта блок пам'яті і ініціалізує його.

Адресний простір процесу складається з набору адрес віртуальної пам'яті, які він може використати. Адресний простір процесу недоступний іншим процесам. Процес володіє системними ресурсами, такими як файли, мережеві з'єднання, пристрої введення-виведення, об'єкти синхронізації тощо. Процес містить деяку стартову інформацію для потоків, які в ньому створюватимуться. Процес має містити хоча б один потік, який система скеровує на виконання

**Постановка завдання**

1.Виконати в окремому процесі табулювання функцій ( Можна замінити алгоритмом заданим у лабораторній роботі №3).

2. Реалізувати табулювання функцій у 2-ох, 4-ох, 8-ох процесах. Виміряти час роботи процесів. Порівняти результати роботи в одному і в багатьох процесах.

3. Реалізувати можливість зміни пріоритету виконання процесу.

4. Реалізувати можливість зупинки і відновлення роботи процесу

5. Реалізувати можливість вбиття процесу.

6. Порівняти результати виконання програми під ОС Windows та Linux.

7. Результати роботи відобразити у звіті.

**Хід роботи**

Код програми:

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QMessageBox"

#include <sys/resource.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#include <vector>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

for(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

for(int j = 0; j < ui->tableWidget->columnCount(); j++)

{

QTableWidgetItem\* item = new QTableWidgetItem;

item->setText("");

item->setTextAlignment(Qt::AlignCenter);

ui->tableWidget->setItem(i,j,item);

}

}

connect(ui->pushButton\_start,SIGNAL(clicked()),this,SLOT(MySlot()));

connect(ui>pushButton\_suspend,SIGNAL(clicked()),this,SLOT(MySlot()));

connect(ui->pushButton\_resume,SIGNAL(clicked()),this,SLOT(MySlot()));

connect(ui->pushButton\_kill,SIGNAL(clicked()),this,SLOT(MySlot()));

connect(ui>pushButton\_killall,SIGNAL(clicked()),this,SLOT(MySlot()));

timer = new QTimer();

connect(timer,SIGNAL(timeout()),this,SLOT(MyTimer()));

timer->start(1000);

}

MainWindow::~MainWindow()

{

for(int i = 0; pid\_array.size();i++){

int status;

waitpid(pid\_array[i],&status,0);

}

delete ui;

}

void MainWindow::MyTimer(){

for(int i = 0; i<ui->tableWidget->rowCount();i++){

if(ui->tableWidget->item(i,0)->text()!=""){

struct rusage all\_time;

getrusage(ui->tableWidget->item(i,0)->text().toInt(),&all\_time);

char buff[40];

snprintf(buff,sizeof(buff),"%ld.%06ld",all\_time.ru\_utime.tv\_sec,all\_time.ru\_utime.tv\_usec);

ui->tableWidget->item(i,3)->setText(buff);

}

}

}

void MainWindow::MySlot()

{

QPushButton\* btn = (QPushButton\*) sender();

if(btn->text() == "Start"){

int ProcessCount = ui->comboBox\_process\_count->currentText().toInt();

int rowCount = ui->spinBox\_row\_count->value();

int columnCount = ui->spinBox\_column\_count->value();

for(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

for(int j = 0; j < ui->tableWidget->columnCount(); j++)

{

QTableWidgetItem\* item = new QTableWidgetItem;

item->setText("");

item->setTextAlignment(Qt::AlignCenter);

ui->tableWidget->setItem(i,j,item);

}

}

std::string cmd = "/home/pavlo/source1\_lab\_4/lab\_4";

std::string cmd\_1 = std::to\_string(rowCount/ProcessCount);

std::string cmd\_2 = std::to\_string(columnCount/ProcessCount);

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

{

pid\_t pid = fork();

switch(pid){

case -1:

perror("fork");

exit(1);

case 0:

execl(cmd.c\_str(),cmd.c\_str(),cmd\_1.c\_str(),cmd\_2.c\_str(),0);

perror("failed");

exit(1);

default: // parent process, pid now contains the child pid

ui->tableWidget->item(i,0)->setText(QString::number(pid));

pid\_array.push\_back(pid);

ui->tableWidget->item(i,1)->setText("Running");

ui->tableWidget->item(i,2)->setText(QString::number(20));

ui->tableWidget->item(i,3)->setText(QString::number(0));

break;

}

}

}else if(btn->text() == "Suspend"){

kill(ui->tableWidget->item(ui->tableWidget->currentRow(),0)->text().toInt(),SIGSTOP);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Suspended");

}else if(btn->text() == "Resume"){

kill(ui->tableWidget->item(ui->tableWidget->currentRow(),0)->text().toInt(),SIGCONT);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Running");

}else if(btn->text() == "Kill"){

kill(ui->tableWidget->item(ui->tableWidget->currentRow(),0)->text().toInt(),SIGKILL);

ui->tableWidget->item(ui->tableWidget->currentRow(),1)->setText("Terminated");

}else if(btn->text() == "Kill all"){

for(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

kill(ui->tableWidget->item(i,0)->text().toInt(),SIGKILL);

ui->tableWidget->item(i,1)->setText("Terminated");

}

}

}

void MainWindow::on\_comboBox\_process\_count\_currentTextChanged(const QString &arg1)

{

ui->tableWidget->setRowCount(arg1.toInt());

for(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

for(int j = 0; j < ui->tableWidget->columnCount(); j++)

{

QTableWidgetItem\* item = new QTableWidgetItem;

item->setText("");

item->setTextAlignment(Qt::AlignCenter);

ui->tableWidget->setItem(i,j,item);

}

}

}

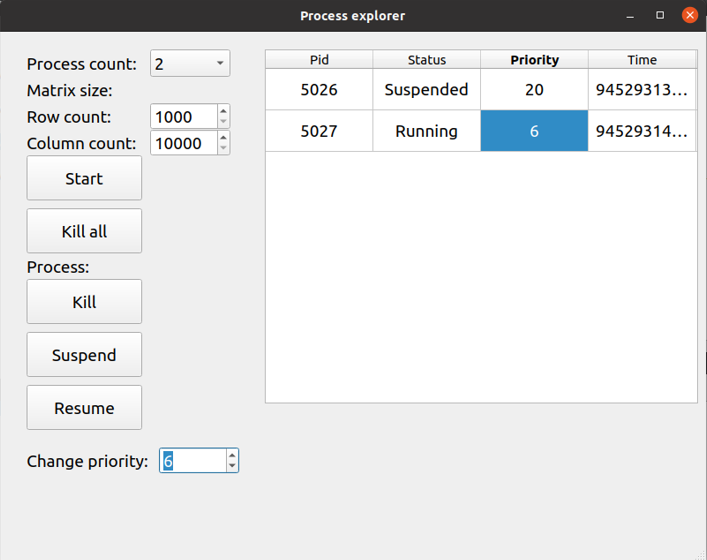
void MainWindow::on\_spinBox\_priority\_valueChanged(int arg1)

{

setpriority(PRIO\_PROCESS,ui->tableWidget->item(ui->tableWidget->currentRow(),0)->text().toInt(),arg1);

ui->tableWidget->item(ui->tableWidget->currentRow(),2)->setText(QString::number(arg1));

}



##### Висновок

На даній лабораторній роботі я навчився працювати з процесами в ОС Linux з використанням функцій API.

За допомогою функцій fork і execl я реалізував індивідуальне завдання у 1, 2, 4 і 8 процесах. Також навчився використовувати такі функції, як Setpriority, kill.