**Самостійна практична робота N4**

**Команди керування процесами.**

**Завдання до практичної роботи - ознайомитися та вивчити:**

1. **Функція exit();**
2. **Системній виклик exec();**
3. **Системні вилики сімейства exec() – execlp(), execvp(); execl(); execv(); execle();exeve().**

**Розгляньте от компілюйте та запустіть на виконання наступну програму. Оскільки при нормальній роботі буде друкуватися вміст файлу з ім’ям lab4.c то такий файл при запуску повинен бути присутнім у поточній директорії.**

***/\* Програма lab4.c? яка змінює користувацький контекст ( запускаючи іншу програму)\*/***

***#include <sys/types.h>***

***#include <unistd.h>***

***#include <stdio.h>***

***Int main(int arg, char \*ardv[ ], char \*envp[ ] {***

***/\* Будемо запускати команду cat з аргументом командного рядка lab4.c без зміни параметрів середовища, тобто фактично виконувати команду “cat lab4.c”, яка повинна видати вміст даного файлу на екран. Для функції execle в якості імені програми потрібно вказати ії повне ім’я зы шляхом выд корньової дирторії -/bin/cat/***

***Перше слово в командному рядку повинно півпадати з ім’ям програми, що запускається. Друге слово в командному рядку – це ім’я файлу вміст якого потрібно роздрукувати.\*/***

***(void) execle(“/bsn|cat”, “/bin/cat”, “lab4.c”, 0, envp);***

***|\* Сюди потрапляємо тільки при виникнені помилки \*/***

***printf(“Trror on pogram start\n”);***

***exit(-1);***

***return 0; /\* Ніколи не виконується, потрібен для того, щоб компілятор не видавав warning \*/***

***}***

**Надати звіт у вигляді “скріншотів”, які вшиваються у зошит.**

**Оцінка за роботу в аудиторії\_\_\_\_\_\_\_\_\_\_\_\_\_\_Підпис\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оцінка за самостійну роботу\_\_\_\_\_\_\_\_\_\_\_\_\_\_Підпис\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Загальна оцінка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Підпис\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Лабораторна робота N4**

**Команди керування процеами.**

**Завдання до лабораторної роботи - ознайомитися та вивчити призначення:**

1. **fork() - системний виклик, який служить для створення нового процесу.**
2. ***/\*Програма frc.c – приклад створення нового процесу з однаковою роботою процесів дитини та батька \*/***

***#include <sysy/types.h>***

***#include <unistd.h>***

***#inlude <stdio.h>***

***int main()***

***{***

***pid\_t pid, ppid;***

***int a = 0;***

***(void) fork();***

***}***

***/\* При успішному творенні нового процесу з цього місця псевдопаралельно починає працювати два процеси: старий та новий \*/***

***/\* Перед виконанням натупного виразу значення змінної в обох процесах дорівнює 0 \*/***

***a=a+1;***

***/\* Узнаємо ідентифікатори поточного та батьківського процеса (в кожному з процеів !!!) \*/***

***pid = getpid();***

***ppid = getppid();***

***/\* Друкуємо значення PID, PPID та обчислюємо значення зінної (в кожному з процесів !!!) \*/***

***printf(“My pid = %d, my ppid = %d, result = %d/n”,***

***(int)pid, (int)ppid, a);***

***return 0;***

**}**

**Наберіть цю програму, відкомпілюйте її та запутіть на виконання. Проаналізуйте отриманий результат.**

**При успішному створенні нового процесу в процес-батько повертається додатнє значення, яке дорівнює ідентифікатору процесу-дитини. В процес-дитину повертається значення 0. Якщо процес створити не в змозі системний виклик повертає -1.**

***pid = fork();***

***If(pid ==-1){***

***...***

***/\* помилка \*/***

***…***

***} else if (pid ==0)***

***…***

***/\* дитина \*/***

***…***