# Київський національний університет імені Тараса Шевченка факультет радіофізики, електроніки та комп’ютерних систем

Лабораторна робота №4

Роботу виконав

Студент 3-го курсу

ФРЕКС КІ-МА

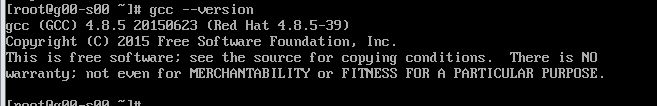
Паньшин Дмитро

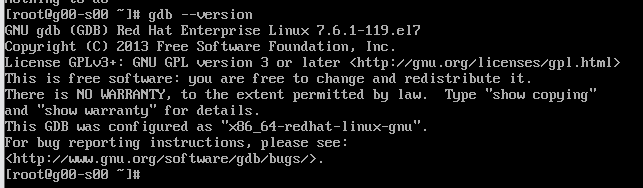
Київ 2020

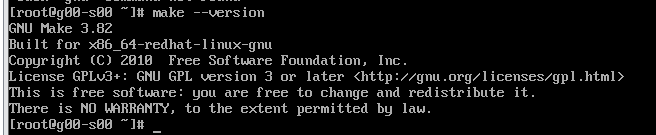
1.

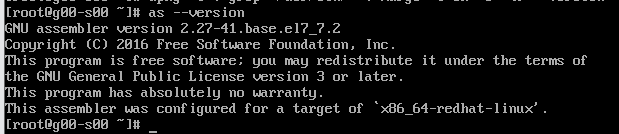
Підготовка середовища

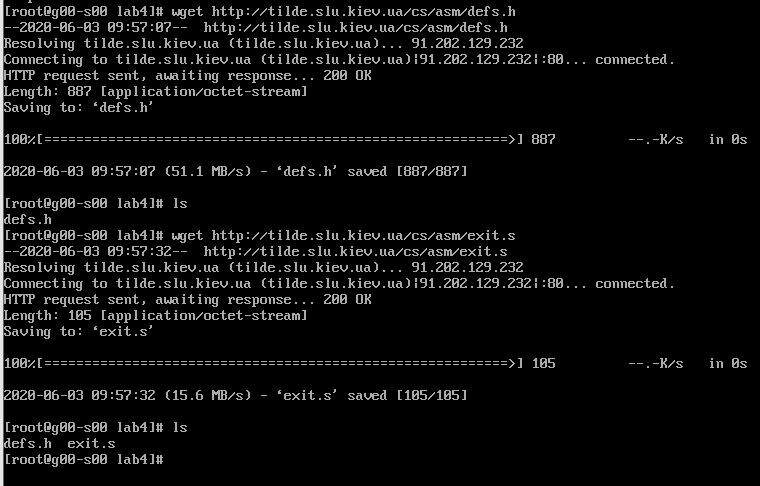
sudo yum group install "Development Tools"



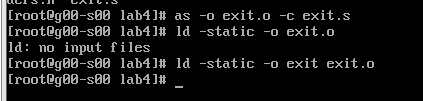






Завантажуємо файли за допомогою команди:wget  






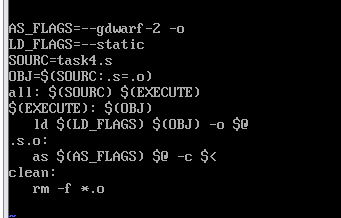
# 2. Автоматизація збірки

Створіть Makefile, який за командою make exit та make all виконає збірку, а за командою make clean очистить об'єктні та виконувані файли.

Модифікуйте Makefile так, щоб опції асемблера та лінкера задавалися змінними ASFLAGS та LDFLAGS.

Додайте опцію асемблера для генерації відлагоджувальних символів DWARF.

Використайте шаблонні правила так, щоб можна було збирати декілька асемблерних файлів в окремі виконувані файли. Це знадобиться при виконанні індивідуального завдання.



# 3. Навички відлагоджування

Завантажте одержаний виконуваний файл у відлагоджувач за допомогою команди:

gdb ./exit

Встановіть точку зупинки на початок програми (мітка \_start):

b \_start

Запустіть програму

run

Після зупинки виконання програми перегляньте вміст регістрів:

info registers або i r

Переходьте до виконання наступної команди:

next або n

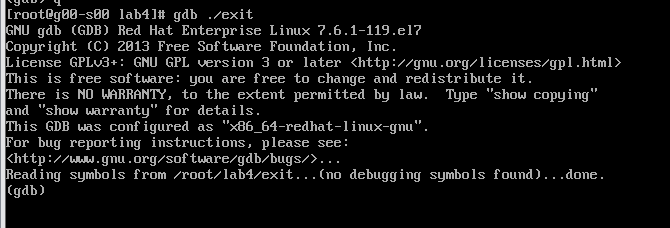
Для виходу із режиму покрокового виконання використовуйте команду

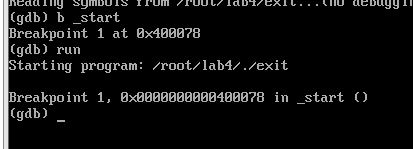
continue або с

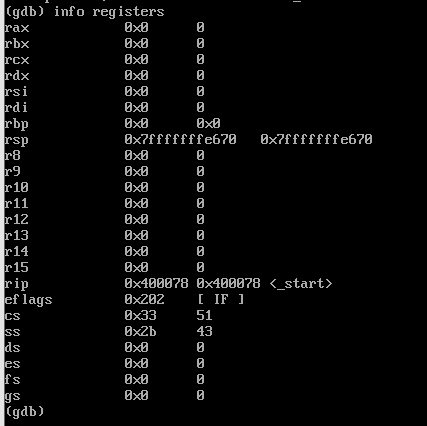
Програма працюватиме до наступної точки зупинки або до повного чи аварійного завершення.

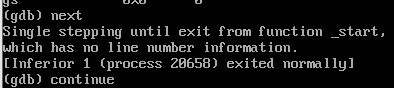
Для перегляду адресного простору процесу скористайтесь командою x, наприклад:

x/16gx 0x12345678









4. Контрольований fork-вибух

Створіть програму, яка "розмножується" за допомогою fork() 5 разів.

Кожна нова копія виводить свою генерацію (число від 1 до 5) на стандартний вивід та завершується.

Батьківський процес очікує завершення дочірнього та також завершується.

### **Псевдокод**

byte nforks = 5;

main() {

    quad pid;

    byte n;

fork:

    pid = fork();

    if(pid != 0) goto parent;

child:

    n = nforks + 0x30;      /\* 1 + 0x30 = '1' \*/

    write(stdout, 1, &n);

    nforks--;

    if(nforks == 0) goto end;

    goto fork;

parent:

    wait4(pid, 0, 0, 0);

end:

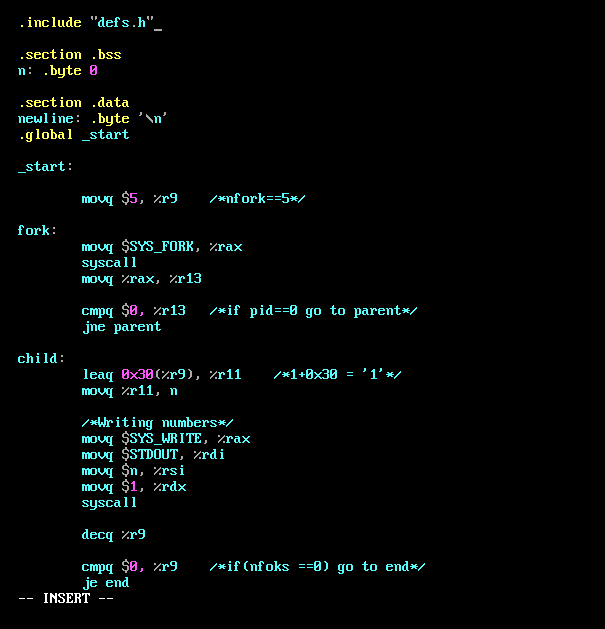
    exit(0);

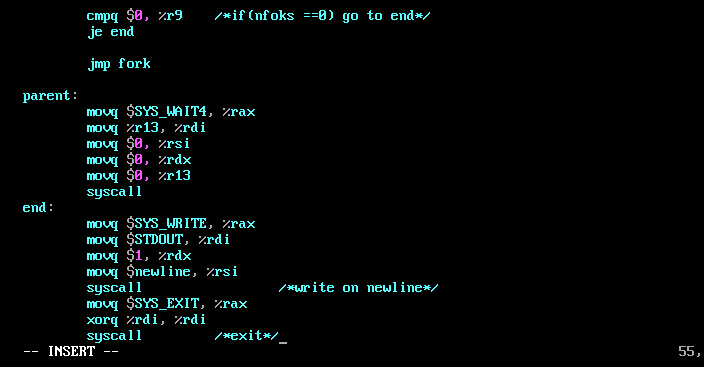
}

### **Перевірка**

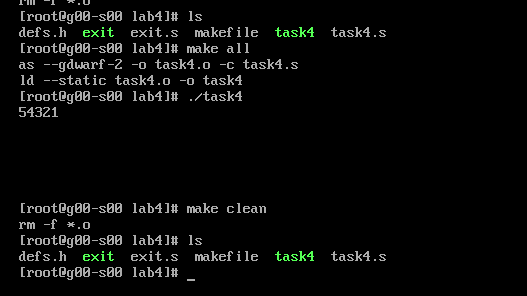
В результаті ви маєте отримати вивід: 54321

Код программи:

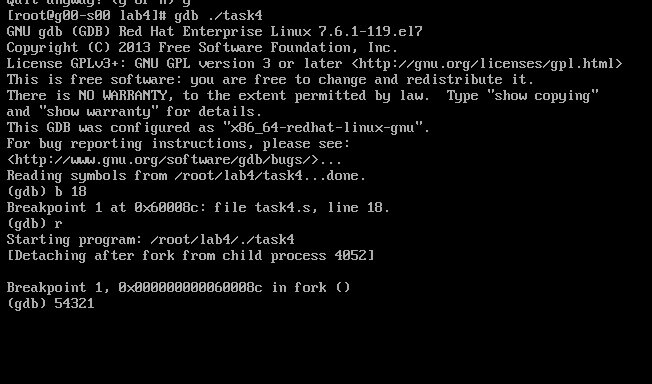


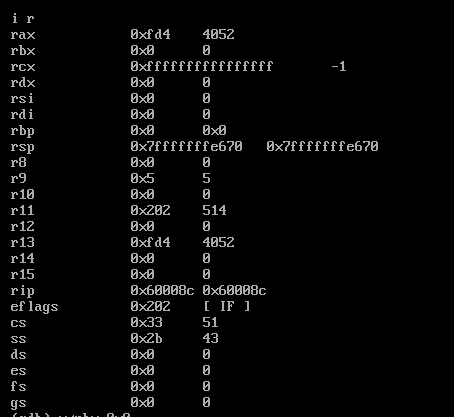


Робота make file та самої программи:



Debugging with gdb:





**Висновки:** під час виконання лабораторної роботи були здобуті навички з основ Assembler для архітектури x86\_64, також було розроблено програму для створення системного виклику fork процесів, відлагоджено за допомогою утиліти для налагодження GDB. Був створений Makefile для автоматизації компіляції та зв’язування програми.

Репозитрорій: https://github.com/N0thingLikeTheSun/CS\_LABS