**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по операционным системам

«Исследование процессов и памяти»

Студент АС-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Руководитель

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Останков А. И.

(подпись, дата)

Липецк 2023

Содержание:

2. Задание

3. Текст программы

4. Результат работы программы

5. Что же такое в proc/<pid>/maps ?

**2. Задание**

1. Разработать небольшую программу, которая выполняет следующую последовательность действий:

а) после запуска ожидает от оператора ввода числа N

б) запускает три дочерних процесса, выводит на stdout их PID, дожидается их завершения и выводит на stdout полученный код возврата

в) дочерние процессы в цикле N раз выводят на stdout сообщения через псевдослучайный интервал времени 0.1-8 сек. Сообщения должны содержать отметку времени, PID дочернего процесса, и номер цикла (можно сделать что-то поинтереснее - на выбор студента, главное, чтобы был вывод и длилось это некоторое время)

г) после завершения итераций дочерний процесс заканчивается, код возврата устанавливается равным количеству исполненных итераций

2. Скомпилировать и отладить программу, добиться её устойчивого исполнения. Пока программа работает:

2.1 Наблюдать процесс выполнения с помощью интерактивной утилиты top

2.2 Наблюдать процесс выполнения с помощью команды ps (изучить основные ключи форматов -l, -f, -F, -v, -u ...)

2.3 Наблюдать содержимое псевдофайлов в /proc/<pid> (где <pid> - это идентификатор запущенных процессов):

- /proc/<pid>/cmdline

- /proc/<pid>/environ

- /proc/<pid>/maps

2.4 Наблюдать содержимое псевдокаталогов в /proc/<pid>

- /proc/<pid>/fd

- /proc/<pid>/fdinfo

Описание вышеприведённого (и не только) можно узнать по man proc

3. Объяснить содержимое /proc/<pid>/maps с помощью objdump/readelf

**3. Текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <time.h>

#define MAX\_CHILDREN 3

void child\_process(int id, int N) {

srand(time(NULL) + id);

int i;

for (i = 1; i <= N; ++i) {

int sleep\_time = rand() % 8000 + 100;

sleep\_time /= 1000;

sleep(sleep\_time);

time\_t t;

time(&t);

printf("[TIME]%s[CHILD-INFO]Child %d (PID: %d) - Iteration %d\n\n", ctime(&t), id, getpid(), i);

}

exit(i - 1);

}

int main() {

int n\_iters;

printf("Enter the value of N: ");

scanf("%d", &n\_iters);

pid\_t children[MAX\_CHILDREN];

for (int i = 0; i < MAX\_CHILDREN; ++i) {

children[i] = fork();

if (children[i] == 0) {

child\_process(i + 1, n\_iters);

}

}

int status;

for (int i = 0; i < MAX\_CHILDREN; ++i) {

waitpid(children[i], &status, 0);

printf("===Child %d (PID: %d) exited with status: %d\n\n===", i + 1, children[i], WEXITSTATUS(status));

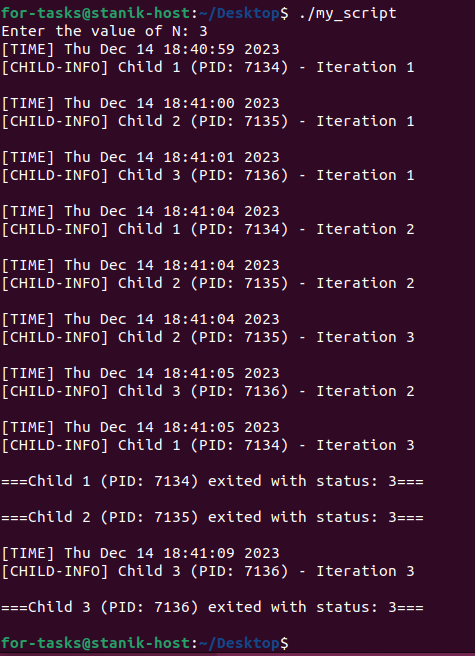
}

return 0;

}

**4. Результат работы программы**

На входе задано число циклов = 3.



-------------------------------------------------------------------

top -c -o PID

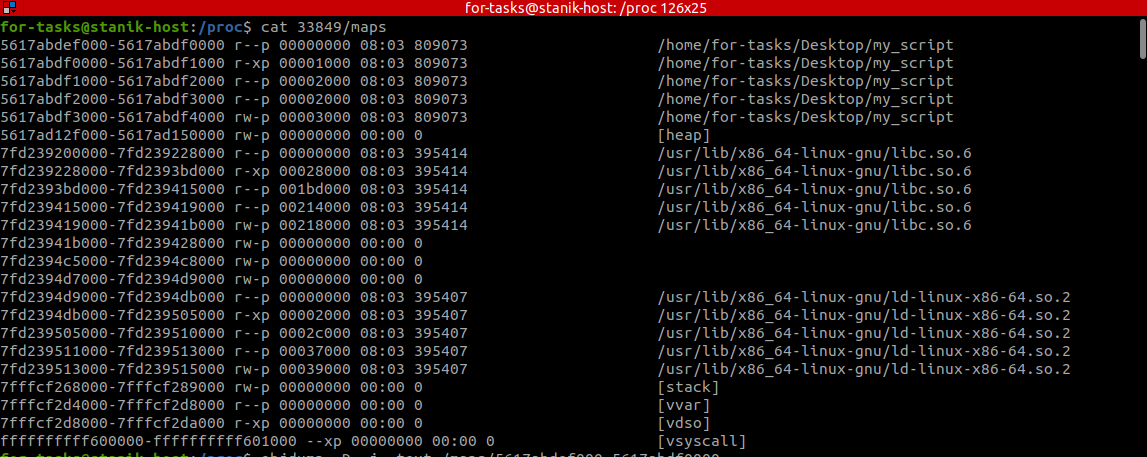
watch -n 1 'ps aux | { head -n 1; grep my\_program; }'

ps -v

ps -f -F -u for-tasks

-------------------------------------------------------------------

3. Направляемся в proc/<PID>/maps, чтобы узнать, какие команды используют виртуальную память процесса. Конкатенируем файл maps интересующего нас процесса (CHILD 2).



Здесь видны диапазоны адресов // разрешения // занимаемое виртуальное место // мажорные:минорные номера устройства // id файла // команда

Именно здесь хранятся сведения о, занимаемой процессом CHILD2, виртуальной памяти.

Теперь, при помощи objdump деассемблируем файл, лежащий на пути нашего скрипта. В данном случае, нас интересует наш скрипт

objdump -D -j .text /home/for-tasks/Desktop/my\_script