



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.

```

for-tasks@stanik-host:~/Desktop$ ./my_script
Enter the value of N: 3
[TIME] Thu Dec 14 18:40:59 2023
[CHILD-INFO] Child 1 (PID: 7134) - Iteration 1

[TIME] Thu Dec 14 18:41:00 2023
[CHILD-INFO] Child 2 (PID: 7135) - Iteration 1

[TIME] Thu Dec 14 18:41:01 2023
[CHILD-INFO] Child 3 (PID: 7136) - Iteration 1

[TIME] Thu Dec 14 18:41:04 2023
[CHILD-INFO] Child 1 (PID: 7134) - Iteration 2

[TIME] Thu Dec 14 18:41:04 2023
[CHILD-INFO] Child 2 (PID: 7135) - Iteration 2

[TIME] Thu Dec 14 18:41:04 2023
[CHILD-INFO] Child 2 (PID: 7135) - Iteration 3

[TIME] Thu Dec 14 18:41:05 2023
[CHILD-INFO] Child 3 (PID: 7136) - Iteration 2

[TIME] Thu Dec 14 18:41:05 2023
[CHILD-INFO] Child 1 (PID: 7134) - Iteration 3

===Child 1 (PID: 7134) exited with status: 3===

===Child 2 (PID: 7135) exited with status: 3===

[TIME] Thu Dec 14 18:41:09 2023
[CHILD-INFO] Child 3 (PID: 7136) - Iteration 3

===Child 3 (PID: 7136) exited with status: 3===

for-tasks@stanik-host:~/Desktop$

```

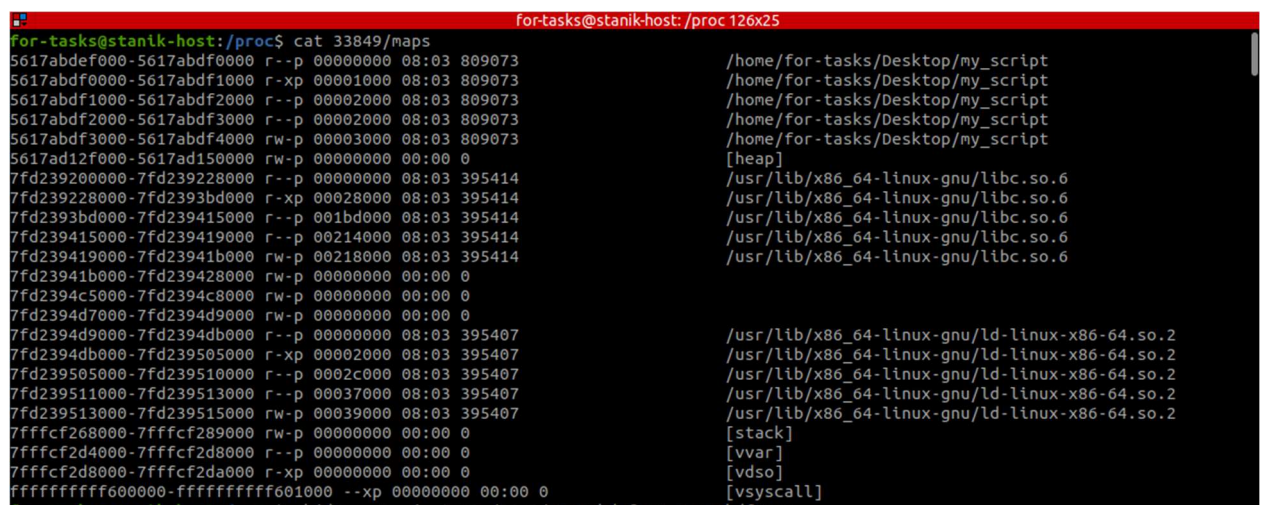
```
top -c -o PID
```

```
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).



```
for-tasks@stanik-host: /proc$ cat 33849/maps
5617abdef000-5617abdf0000 r--p 00000000 08:03 809073 /home/for-tasks/Desktop/my_script
5617abdf0000-5617abdf1000 r-xp 00001000 08:03 809073 /home/for-tasks/Desktop/my_script
5617abdf1000-5617abdf2000 r--p 00002000 08:03 809073 /home/for-tasks/Desktop/my_script
5617abdf2000-5617abdf3000 r--p 00002000 08:03 809073 /home/for-tasks/Desktop/my_script
5617abdf3000-5617abdf4000 rw-p 00003000 08:03 809073 /home/for-tasks/Desktop/my_script
5617ad12f000-5617ad150000 rw-p 00000000 00:00 0 [heap]
7fd239200000-7fd239228000 r--p 00000000 08:03 395414 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd239228000-7fd2393bd000 r-xp 00028000 08:03 395414 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd2393bd000-7fd239415000 r--p 001bd000 08:03 395414 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd239415000-7fd239419000 r--p 00214000 08:03 395414 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd239419000-7fd23941b000 rw-p 00218000 08:03 395414 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd23941b000-7fd239428000 rw-p 00000000 00:00 0
7fd239428000-7fd2394c8000 rw-p 00000000 00:00 0
7fd2394c8000-7fd2394d9000 rw-p 00000000 00:00 0
7fd2394d9000-7fd2394db000 r--p 00000000 08:03 395407 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7fd2394db000-7fd239505000 r-xp 00002000 08:03 395407 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7fd239505000-7fd239510000 r--p 0002c000 08:03 395407 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7fd239510000-7fd239513000 r--p 00037000 08:03 395407 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7fd239513000-7fd239515000 rw-p 00039000 08:03 395407 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ffffcf26000-7ffffcf289000 rw-p 00000000 00:00 0 [stack]
7ffffcf2d4000-7ffffcf2d8000 r--p 00000000 00:00 0 [vvar]
7ffffcf2d8000-7ffffcf2da000 r-xp 00000000 00:00 0 [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [vsyscall]
```

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
```