НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа 1

По дисциплине:

«Операционные системы»

Выполнил:

Студент группы **P33122**

Савин Георгий Евгеньевич

Преподаватель:

Покид Александр Владимирович

г. Санкт-Петербург

2020

Разработать программу на языке C, которая осуществляет следующие действия

- **Создает область памяти размером 120 мегабайт, начинающихся с адреса 0x9B97188D (невозможно) при помощи C=malloc заполненную случайными числами /dev/urandom в 73 потоков. Используя системные средства мониторинга определите адрес начала в адресном пространстве процесса и характеристики выделенных участков памяти. Замеры виртуальной/физической памяти необходимо снять:**
1. **До аллокации**
2. **После аллокации**
3. **После заполнения участка данными**
4. **После деаллокации**
- **Записывает область памяти в файлы одинакового размера 150 мегабайт с использованием F=некешируемого обращения к диску. Размер блока ввода-вывода 136 байт. Преподаватель выдает в качестве задания последовательность записи/чтения блоков H=последовательный**
- **Генерацию данных и запись осуществлять в бесконечном цикле.**
- **В отдельных 147 потоках осуществлять чтение данных из файлов и подсчитывать агрегированные характеристики данных - J=сумму.**
- **Чтение и запись данных в/из файла должна быть защищена примитивами синхронизации K=futex.**
- **По заданию преподавателя изменить приоритеты потоков и описать изменения в характеристиках программы.**

Для запуска программы возможно использовать операционную систему Windows 10 или Debian/Ubuntu в виртуальном окружении.

Измерить значения затраченного процессорного времени на выполнение программы и на операции ввода-вывода используя системные утилиты.

Отследить трассу системных вызовов.

Используя stap построить графики системных характеристик.

https://github.com/Delta145/lab_c
[pid] = ps -C lab1
Адрес начала в адресном пространстве и характеристики выделенных участков памяти: **sudo cat /proc/[pid]/maps или pmap -x [pid]**

```
gosha@pi:~$ sudo cat /proc/29345/maps
5626608b2000-5626608b5000 r-xp 00000000 08:05 3148487                    /home/gosha/CLionProjects/lab1mine/cmake-build-debug/lab1
562660ab4000-562660ab5000 r--p 00002000 08:05 3148487                    /home/gosha/CLionProjects/lab1mine/cmake-build-debug/lab1
562660ab5000-562660ab6000 rw-p 00003000 08:05 3148487                    /home/gosha/CLionProjects/lab1mine/cmake-build-debug/lab1
562661dd9000-562661dfa000 rw-p 00000000 00:00 0                          [heap]
7ffba4445000-7ffba462c000 r-xp 00000000 08:05 2364336                    /lib/x86_64-linux-gnu/libc-2.27.so
7ffba462c000-7ffba482c000 ---p 001e7000 08:05 2364336                    /lib/x86_64-linux-gnu/libc-2.27.so
7ffba482c000-7ffba4830000 r--p 001e7000 08:05 2364336                    /lib/x86_64-linux-gnu/libc-2.27.so
7ffba4830000-7ffba4832000 rw-p 001eb000 08:05 2364336                    /lib/x86_64-linux-gnu/libc-2.27.so
7ffba4832000-7ffba4836000 rw-p 00000000 00:00 0
7ffba4836000-7ffba4850000 r-xp 00000000 08:05 2364869                    /lib/x86_64-linux-gnu/libpthread-2.27.so
7ffba4850000-7ffba4a4f000 ---p 0001a000 08:05 2364869                    /lib/x86_64-linux-gnu/libpthread-2.27.so
7ffba4a4f000-7ffba4a50000 r--p 00019000 08:05 2364869                    /lib/x86_64-linux-gnu/libpthread-2.27.so
7ffba4a50000-7ffba4a51000 rw-p 0001a000 08:05 2364869                    /lib/x86_64-linux-gnu/libpthread-2.27.so
7ffba4a51000-7ffba4a55000 rw-p 00000000 00:00 0
7ffba4a55000-7ffba4a7c000 r-xp 00000000 08:05 2364332                    /lib/x86_64-linux-gnu/ld-2.27.so
7ffba4c5a000-7ffba4c5f000 rw-p 00000000 00:00 0
7ffba4c7c000-7ffba4c7d000 r--p 00027000 08:05 2364332                    /lib/x86_64-linux-gnu/ld-2.27.so
7ffba4c7d000-7ffba4c7e000 rw-p 00028000 08:05 2364332                    /lib/x86_64-linux-gnu/ld-2.27.so
7ffba4c7e000-7ffba4c7f000 rw-p 00000000 00:00 0
7ffc86634000-7ffc86655000 rw-p 00000000 00:00 0                          [stack]
7ffc86693000-7ffc86696000 r--p 00000000 00:00 0                          [vvar]
7ffc86696000-7ffc86697000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
gosha@pi:~$
```

Замеры виртуальной/физической памяти: **ps -eo pid,vsz,rss,comm | grep lab1**
До аллокации

pid vsz rss

```
gosha@pi:~/CLionProjects
23055   6692    880 lab1
```

После аллокации

```
gosha@pi:~/CLionProjects
23055 129576    880 lab1
```

После заполнения данными

```
gosha@pi:~/CLionProjects/
23055 2749856 238760 lab1
```

После деаллокации

```
gosha@pi:~/CLionProjects
23055 956980  1992 lab1
```

Измерить значения затраченного процессорного времени на выполнение программы: **time ./lab1**

```
real    0m10,421s
user    0m8,109s
sys     0m21,262s
```

и на операции ввода-вывода используя системные утилиты: **sudo strace -c -fp [pid]**

% time	seconds	usecs/call	calls	errors	syscall
98.10	473.829206	7309	64829	10281	futex
0.97	4.702242	235	20019		read
0.61	2.931354	2	1237715		pread64
0.30	1.433212	37	38400		pwrite64
0.01	0.029216	132	221		clone
0.01	0.027792	124	225		mprotect
0.01	0.026906	118	228		mmap
0.00	0.006673	18	380		write
0.00	0.003078	14	221		set_robust_list
0.00	0.000138	15	9		munmap
0.00	0.000120	40	3		openat
0.00	0.000051	51	1	1	ioctl
0.00	0.000035	9	4		stat
0.00	0.000022	22	1		fstat
0.00	0.000013	7	2		close
100.00	482.990058		1362258	10282	total

Отследить трассу системных вызовов: **sudo strace -fp [pid]**

```
gosha@pi:~$ sudo strace -p 31612
strace: Process 31612 attached
read(0, "\n", 1024)                     = 1
mmap(NULL, 125833216, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb7030d9000
write(1, "\320\237\320\276\321\201\320\273\320\265 \320\260\320\273\320\273\320\276\320\272\320\260\321\206\320\270\320\2
read(0, "\n", 1024)                     = 1
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fb7028d8000
mprotect(0x7fb7028d9000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fb7030d7fb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETT
```

Используя stap построить графики системных характеристик: **sudo stap -x [pid] script**

```
gosha@pi:~/CLionProjects/lab1mine/cmake-build-debug$ sudo stap -x 15999 script
starting probe
^C    name    opens    reads   MB tot    B avg   writes   MB tot    B avg
    lab1        2   925215      120      135    24197       94     4096
```

Вывод: узнал несколько новых утилит для измерения производительности в линуксах, а также много чего узнал о си

1. атрибуты регионов памяти
2. tty?
3. malloc() - как выделяет память?
4. С использованием fork() - как запустить другое приложение?
5. флаги у clone()
6. параметры запуска у потока `schedparams`
7. механизм системных вызовов