Факультет программой инженерии

Лабораторная работа #1

Операционые системы

Выполнил: Куприянов А.А

Санкт-Петербург, 2020

Вариант:

A=118;B=0xB44603B3;C=mmap;D=84;E=45;F=nocache;G=11;H=seq;I=13;J=sum;K=futex

```
#define ALLOC_ADDR      0xB44603B3
#define THREADS_AMOUNT  3
#define FILL_MEM_SIZE   118
#define DUMP_FILE_SIZE  45
#define DATA_BLOCK_SIZE 11
#define COUNTERS_THREAD_AMOUNT 11
```

Заполнение памяти случайными данными:

```
    void * mmapAddr;
    mmapAddr = mmap(addr, size, PROT_READ | PROT_WRITE, MAP_ANONYMOUS | MAP_PRIVATE, -1, 0);

    for (int i = 0; i < THREADS_AMOUNT; i++) {

        arg = malloc(sizeof(arg));
        arg->size = size / THREADS_AMOUNT;
        arg->writeAddr = mmapAddr + i * ((arg->size / 8) + (arg->size % 8 > 0 ? 1 : 0));
        arg->dataSource = fopen("/dev/urandom", "r");
        arg->logEnabled = logEnabled;
        err = pthread_create(&threads[i], NULL, threadFunc, arg);

        if (err != 0)
        {
            puts("Error with creating thread");
            exit(err);
        }

    }

    for (int i = 0; i < THREADS_AMOUNT; i++) {
        err = pthread_join(threads[i], NULL);

        if (err != 0)
        {
            exit(err);
        }
    }
```

Запись данных с памяти в файл (no-cache):

```
void
dumpMem(int fd, void * addr, int size, int * futex) {
    wait_on_futex_value(futex, 0);
    *futex = 0;
    int iterations = size / DATA_BLOCK_SIZE;
    int res = ftruncate(fd, 0);
    for (int i = 0; i < iterations; i++) {
        void * pointer = addr + i;
        size_t r = write(fd, &pointer, DATA_BLOCK_SIZE);
    }

    *futex = 1;
}
```

Чтение и агрегация файла:

```
    int * futex = args->futex;
    wait_on_futex_value(futex, 0);
    *futex = 0;
    uint64_t sum = 0;

    if (lseek(args->fd, 0, SEEK_SET) == -1) {
        return -1;
    }

    unsigned char buf [DATA_BLOCK_SIZE];

    while(1) {

        size_t readBytes = read(args->fd, &buf, DATA_BLOCK_SIZE);

        if (readBytes == -1) {
            perror("Error file read");
            break;
        }

        if (readBytes < DATA_BLOCK_SIZE) {
            if (readBytes == 0) {
                break;
            }
        } else {
            for (int i = 0; i < DATA_BLOCK_SIZE; i++) {
                sum += buf[i];
            }
        }
    }
    *futex = 1;

    return 0;
```

Снятие блокировки со всех потоков:

```
for (int i = 0; i < filesAmount; i++) {
        int * futex = args->dumpMap[i]->futex;
        wake_futex_blocking(futex, 1);
}
```

Снятие значений памяти

До аллокации

```
               total        used        free      shared  buff/cache   available
Mem:        16692132     8103976     8358804       17720      229352     8454424
Swap:       50331648       10240    50321408
```

После аллокации

```
               total        used        free      shared  buff/cache   available
Mem:        16692132     8099780     8363000       17720      229352     8458620
Swap:       50331648       10240    50321408
```

После заполнения данныыми

```
               total        used        free      shared  buff/cache   available
Mem:        16692132     8146868     8315912       17720      229352     8411532
Swap:       50331648       10240    50321408
```

После деаллокации

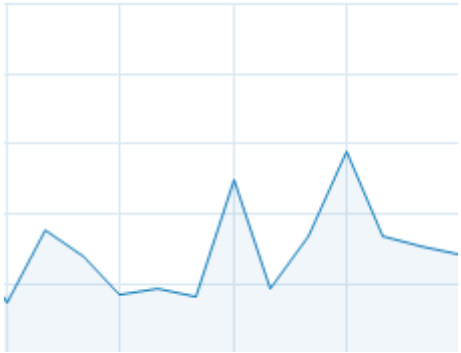
```
               total        used        free      shared  buff/cache   available
Mem:        16692132     8097920     8364860       17720      229352     8460480
Swap:       50331648       10240    50321408
```

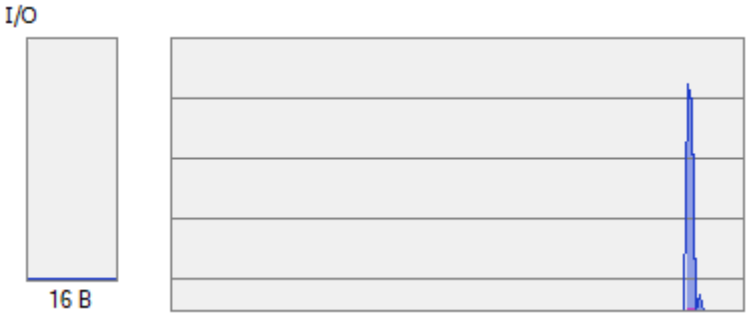
```
mmap(0xb44603b3, 123731968, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb4460000
```

Запись в памяти в отдельных потоках:

```
Filling address: 0xb494aaab - pid: 1079         | tid: 140164705552128
Filling address: 0xb4460000 - pid: 1079         | tid: 140164714006272
Filling address: 0xb4e35556 - pid: 1079         | tid: 140164697097984
```

Показания процессора





Файлы дампов:

```
-rw------- 1 apploid apploid 47185138 Oct 29 19:40 dump.0
-rw------- 1 apploid apploid 47185138 Oct 29 19:40 dump.1
-rw------- 1 apploid apploid 47185138 Oct 29 19:40 dump.2
```

Синхронизация потоков осуществляется через futex:

```
void wait_on_futex_value(int* futex_addr, int val) {
    futex(futex_addr, FUTEX_WAIT, val);
}

void wake_futex_blocking(int* futex_addr, int val) {
    while (1) {
        int futex_rc = futex(futex_addr, FUTEX_WAKE, val);
        if (futex_rc == -1) {
            perror("futex wake");
            exit(1);
        } else if (futex_rc > 0) {
            return;
        }
    }
}
```