Программирование в Linux

Лимиты ресурсов

```c
#include <sys/time.h>
#include <sys/resource.h>

int getrlimit(int resource, struct rlimit *rlim);
int setrlimit(int resource, const struct rlimit *rlim);

int prlimit(pid_t pid, int resource, const struct rlimit *new_limit,
            struct rlimit *old_limit);
```

```c
struct rlimit {
    rlim_t rlim_cur;  /* Soft limit */
    rlim_t rlim_max;  /* Hard limit (ceiling for rlim_cur) */
};
```

soft limit — текущее ограничение. Процесс будет уведомлен при нарушении лимита. soft лимит можно менять в диапазоне [0, hardlimit]

hard limit может быть увеличен только привилегированным процессом (`CAP_SYS_RESOURCE`)
Обычный процесс может уменьшать hard limit

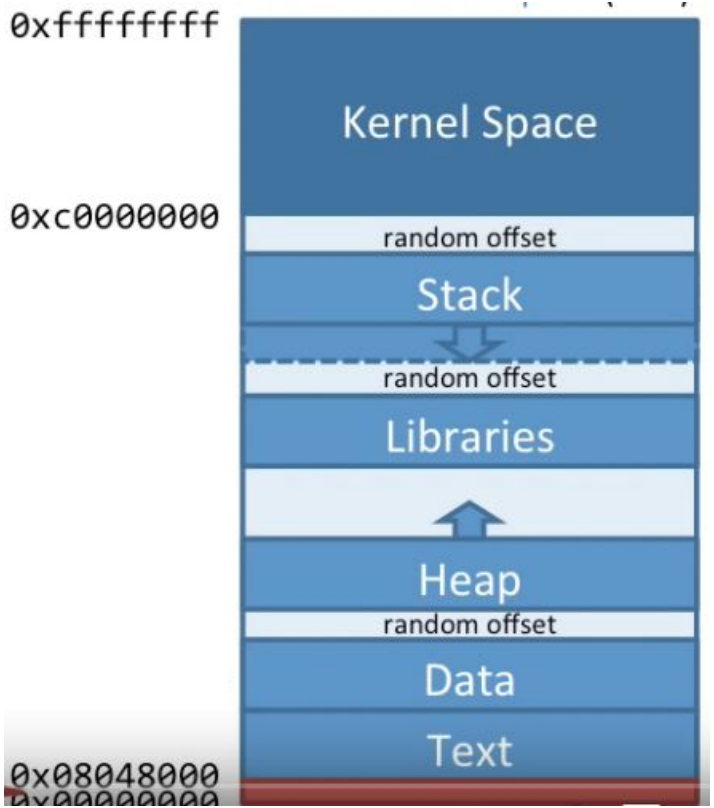
Выделение памяти и RLIMIT_AS/RLIMIT_DATA/RLIMIT_STACK

```c
#include <unistd.h>

int brk(void *addr);

void *sbrk(intptr_t increment);
```

Выделение памяти выполняется либо перемещением границы data section (program break, граница кучи).
либо запросом mmap анонимных страниц в адресном пространстве



Попытка mmap/sbrk сверх лимита — ENOMEM

Попытка увеличить стек — SIGSEGV

Настройки malloc (userspace)

```c
#include <malloc.h>

int mallopt(int param, int value);
```

Параметр	Происхождение	По умолчанию	Допустимые значения	Специальные значения
M_CHECK_ACTION	Linux-специфичный	0	0–2	
M_MMAP_MAX	Уникальный для Linux	64 * 1024	≥ 0	0 отключает использование mmap()
M_MMAP_ THRESHOLD	Уникальный для Linux	128 * 1024	≥ 0	0 отключает использование кучи
M_MXFAST	Стандарт XPG	64	0 - 80	0 отключает использование быстрых корзин
M_PERTURB	Уникальный для Linux	0	0 или 1	0 отключает пер-турбацию
M_TOP_PAD	Уникальный для Linux	0	≥ 0	0 отключает за-полнение сверху
M_TRIM_ THRESHOLD	Уникальный для Linux	128 * 1024	≥ –1	–1 отключает отсечение

alloc_too_much.cpp > main()

```cpp
#include <memory>
#include <malloc.h>
#include <unistd.h>

int main() {
    auto ptr = (char*)malloc(16'000'000'000);
    sleep(5);
    ptr[1'000'500] = 5;
    sleep(5);
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs$ time ./alloc

real    0m10,003s
user    0m0,000s
sys     0m0,002s
```

```cpp
int main() {
    // auto ptr = (char*)malloc(16'000'000'000);
    auto ptr = std::make_unique<char[]>(16'000'000'000);
    sleep(5);
    ptr[1'000'500] = 5;
    sleep(5);
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs$ time ./alloc

real    0m20,722s
user    0m7,000s
sys     0m3,720s
```

RLIMIT_AS vs RLIMIT_RSS

```
7   int main() {
8       auto ptr = std::make_unique<char[]>(16'000'000'000);
9       sleep(5);
10      ptr[1'000'500] = 5;
11      sleep(5);
12      printf("Ok!\n");
13  }
```

```
dmis@dmis-MS-7A15:~/LinuxEgs$ ulimit -Sm 8000000
dmis@dmis-MS-7A15:~/LinuxEgs$ ./alloc
Ok!
dmis@dmis-MS-7A15:~/LinuxEgs$ ulimit -Sv 8000000
dmis@dmis-MS-7A15:~/LinuxEgs$ ./alloc
terminate called after throwing an instance of 'std::bad_alloc'
  what():  std::bad_alloc
Aborted (core dumped)
```

RLIMIT_RSS (ulimit -m) должен ограничивать число страниц, размещенных в физической памяти (resident set). Но он не работает в Linux. С помощью rlimit можно ограничивать только виртуальную память. Физическую память можно ограничить с помощью cgroups

RLIMIT_NPROC

Ограничивает число одновременно существующих процессов/тредов от имени текущего пользователя (RUID текущего PID)

fork/clone получат EAGAIN

RLIMIT_CPU

Ограничивает время исполнения (в секундах).

Процесс будет получать SIGXCPU каждую секунду сверх soft limit
Достижение hard limit — SIGKILL

Лимит ресурса	Мягкий лимит	Жесткий лимит
RLIMIT_AS	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_CORE	0	RLIM_INFINITY
RLIMIT_CPU	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_DATA	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_FSIZE	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_LOCKS	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_MEMLOCK	8 страниц	8 страниц
RLIMIT_MSGQUEUE	800 Кбайт	800 Кбайт
RLIMIT_NICE	0	0
RLIMIT_NOFILE	1024	1024
RLIMIT_NPROC	0 (означает отсутствие ограничений)	0 (означает отсутствие ограничений)
RLIMIT_RSS	RLIM_INFINITY	RLIM_INFINITY
RLIMIT_RTPRIO	0	0
RLIMIT_SIGPENDING	0	0
RLIMIT_STACK	8 Мбайт	RLIM_INFINITY