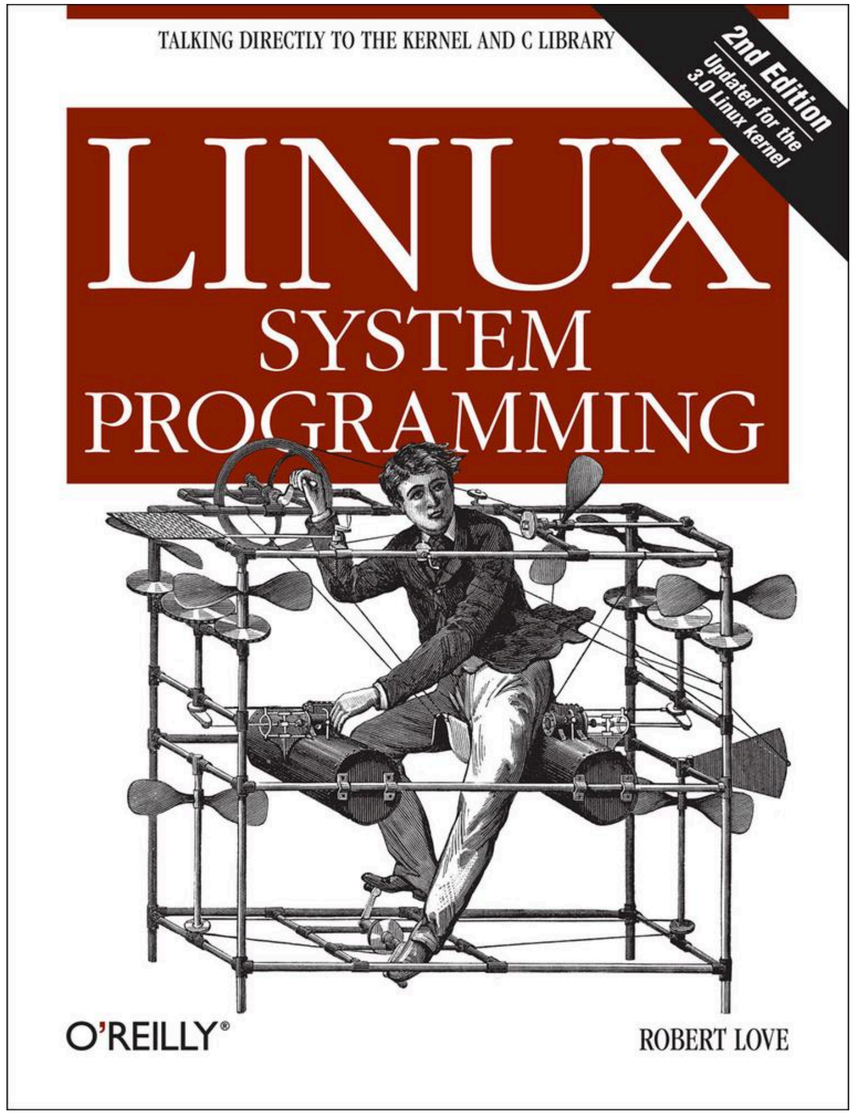
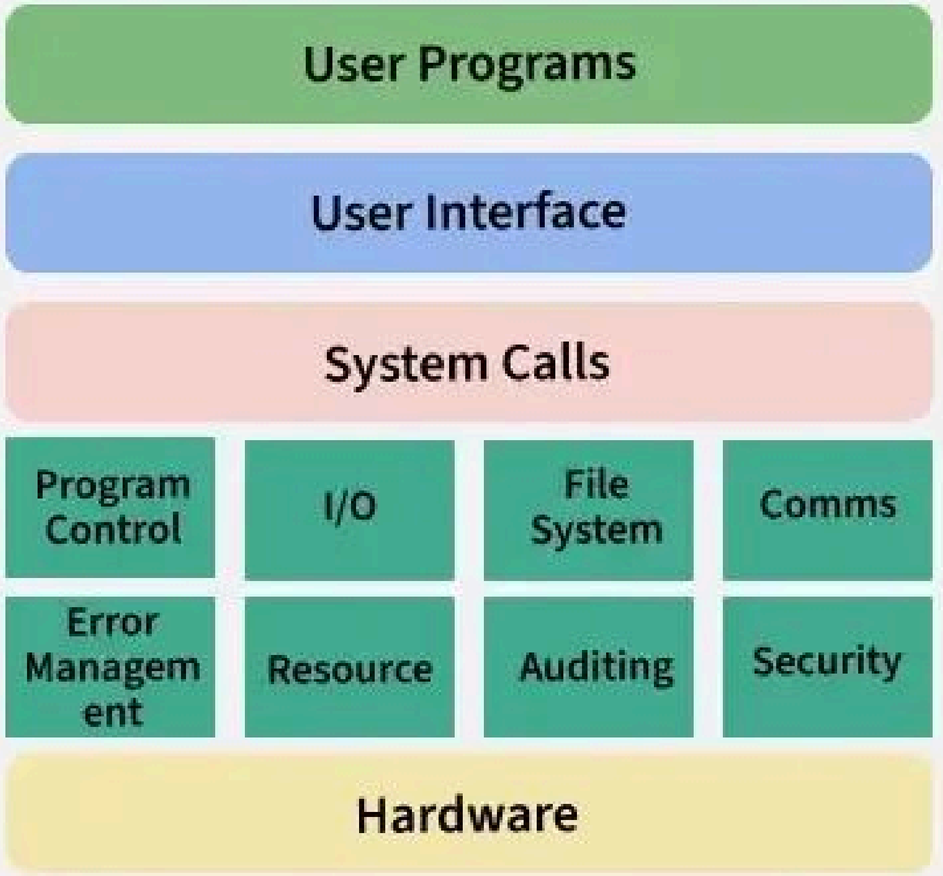
Системное программирование в Linux

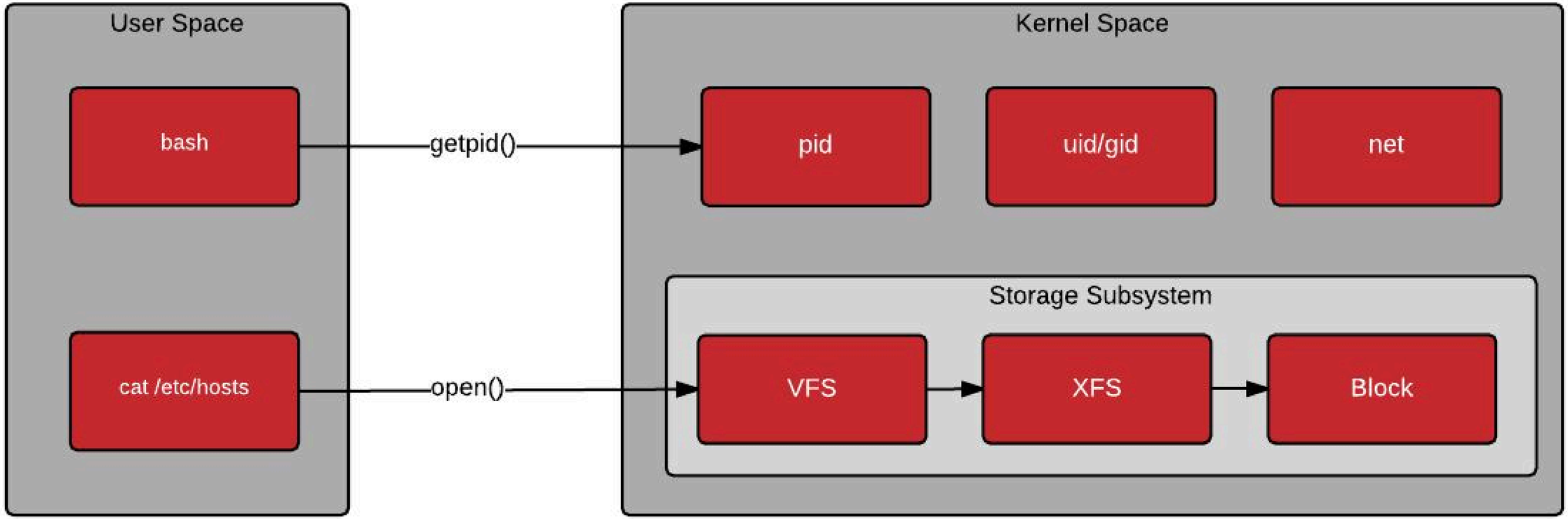
1. system calls
2. systemd
3. IPC

Жахонгир Ахмадалиев



Introduction to System Call




User Space
bash
cat /etc/hosts
getpid()
open()
Kernel Space
pid
uid/gid
net
Storage Subsystem
VFS
XFS
Block

1. Приложение вызывает функцию-обёртку из стандартной библиотеки C (например, read(), write(), open()).

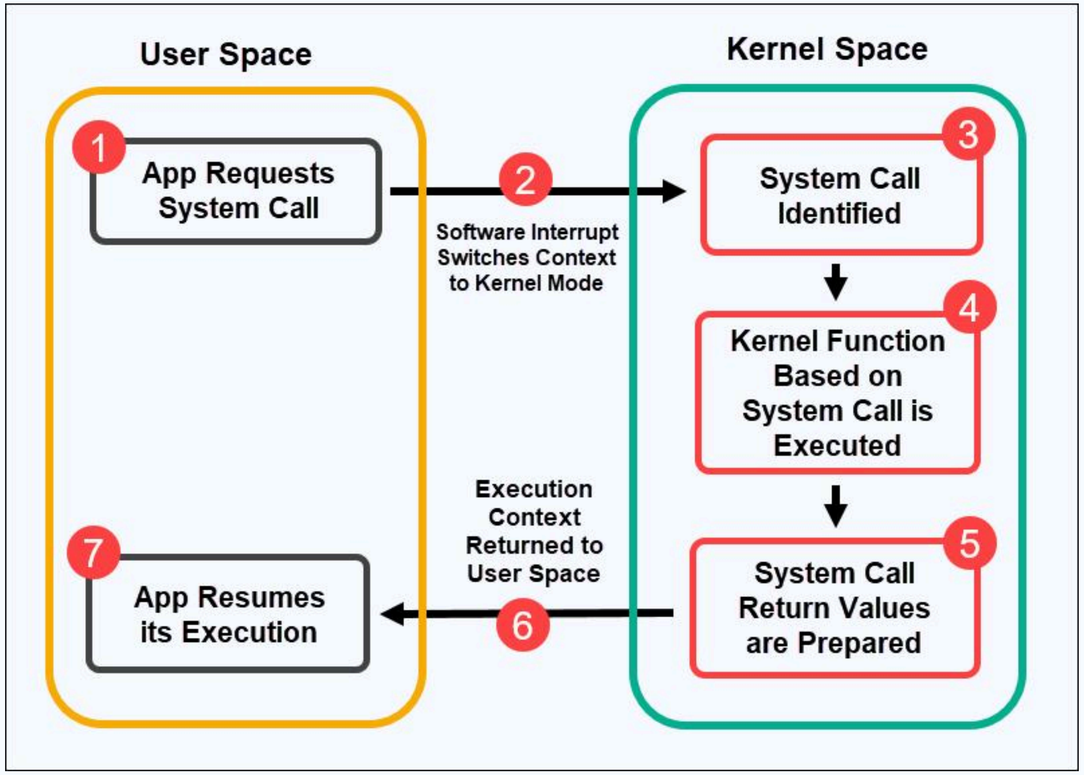
2. Функция-обёртка помещает номер системного вызова в специальный регистр процессора (для x86-64 это регистр rax).

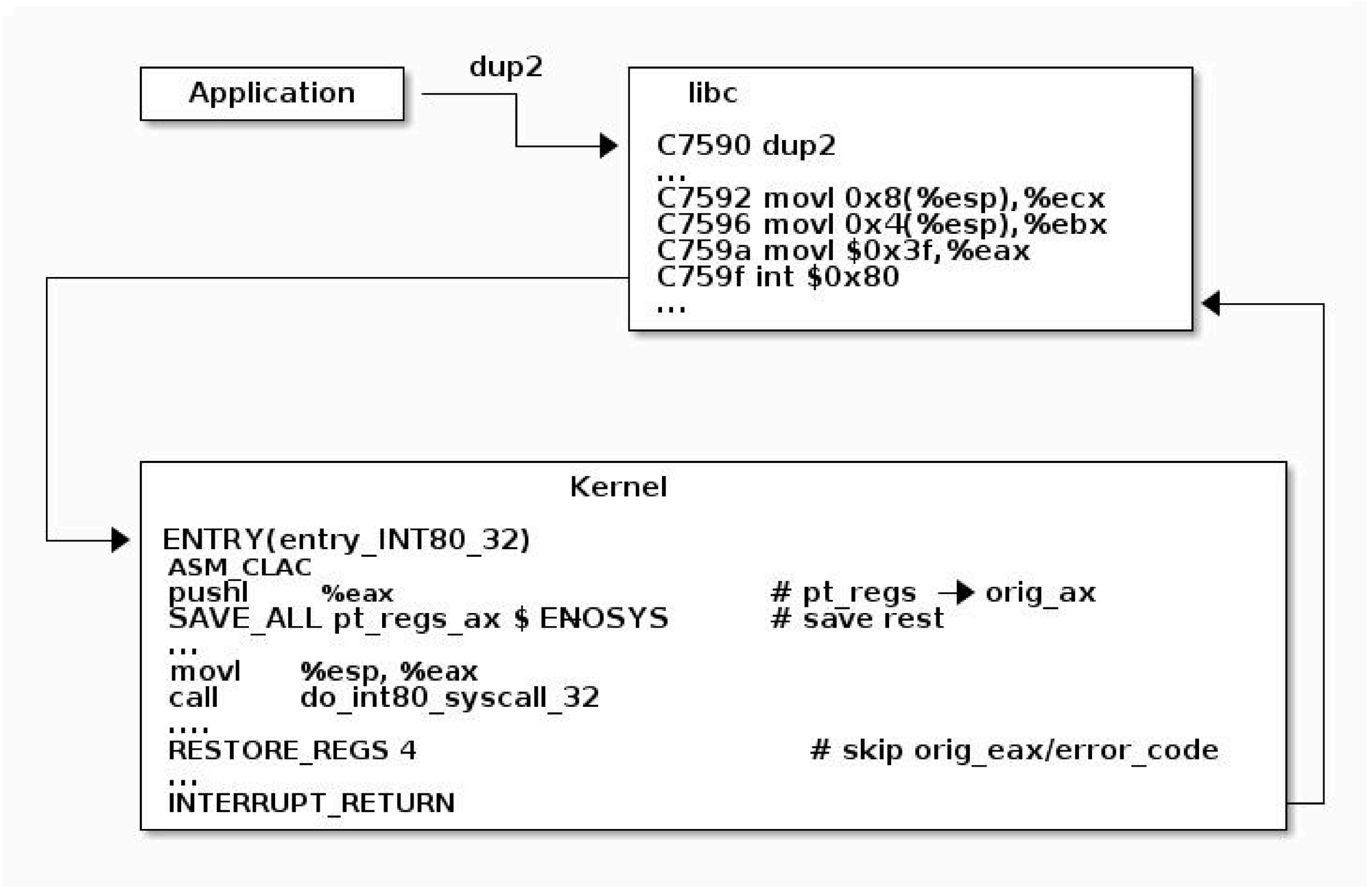
3. Аргументы системного вызова размещаются в других регистрах (rdi, rsi, rdx, r10, r8, r9).

4. Выполняется инструкция syscall, которая вызывает переключение в режим ядра.

5. Ядро проверяет корректность параметров и выполняет запрошенную операцию.

6. Результат возвращается в регистр rax, и управление передаётся обратно в пользовательское пространство.




Application
dup2
libc
C7590 dup2
...
C7592 movl 0x8(%esp),%ecx
C7596 movl 0x4(%esp),%ebx
C759a movl $0x3f,%eax
C759f int $0x80
...
Kernel
ENTRY(entry_INT80_32)
ASM_CLAC
pushl %eax # pt_regs → orig_ax
SAVE_ALL pt_regs_ax $ ENOSYS # save rest
...
movl %esp, %eax
call do_int80_syscall_32
....
RESTORE_REGS 4 # skip orig_eax/error_code
...
INTERRUPT_RETURN

Номера системных вызовов

read - 0

write - 1

open - 2

close - 3

fork - 57

execve -59

```
printf() → write() [libc функция] → write [системный вызов]
malloc() → brk()/mmap() [libc функции] → brk/mmap [системные вызовы]
fopen() → open() [libc функция] → open [системный вызов]
```

Обёртки системных вызовов

Работа с файлами:

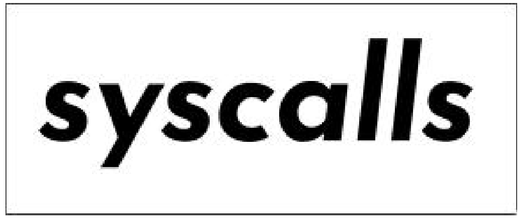
open() — открытие файла
read() — чтение из файла
write() — запись в файл
close() — закрытие файла
lseek() — изменение позиции в файле

Управление процессами:

fork() — создание нового процесса
exec() — замена образа текущего процесса
exit() — завершение процесса
wait() — ожидание завершения дочернего процесса
getpid() — получение идентификатора процесса

Работа с сигналами:

signal() — установка обработчика сигнала
kill() — отправка сигнала процессу
sigaction() — расширенная обработка сигналов



Управление памятью:

brk(), sbrk() — изменение размера сегмента данных
mmap() — отображение файлов или устройств в память
munmap() — удаление отображения

Межпроцессное взаимодействие:

pipe() — создание канала
socket() — создание сокета
shmget() — получение разделяемой памяти

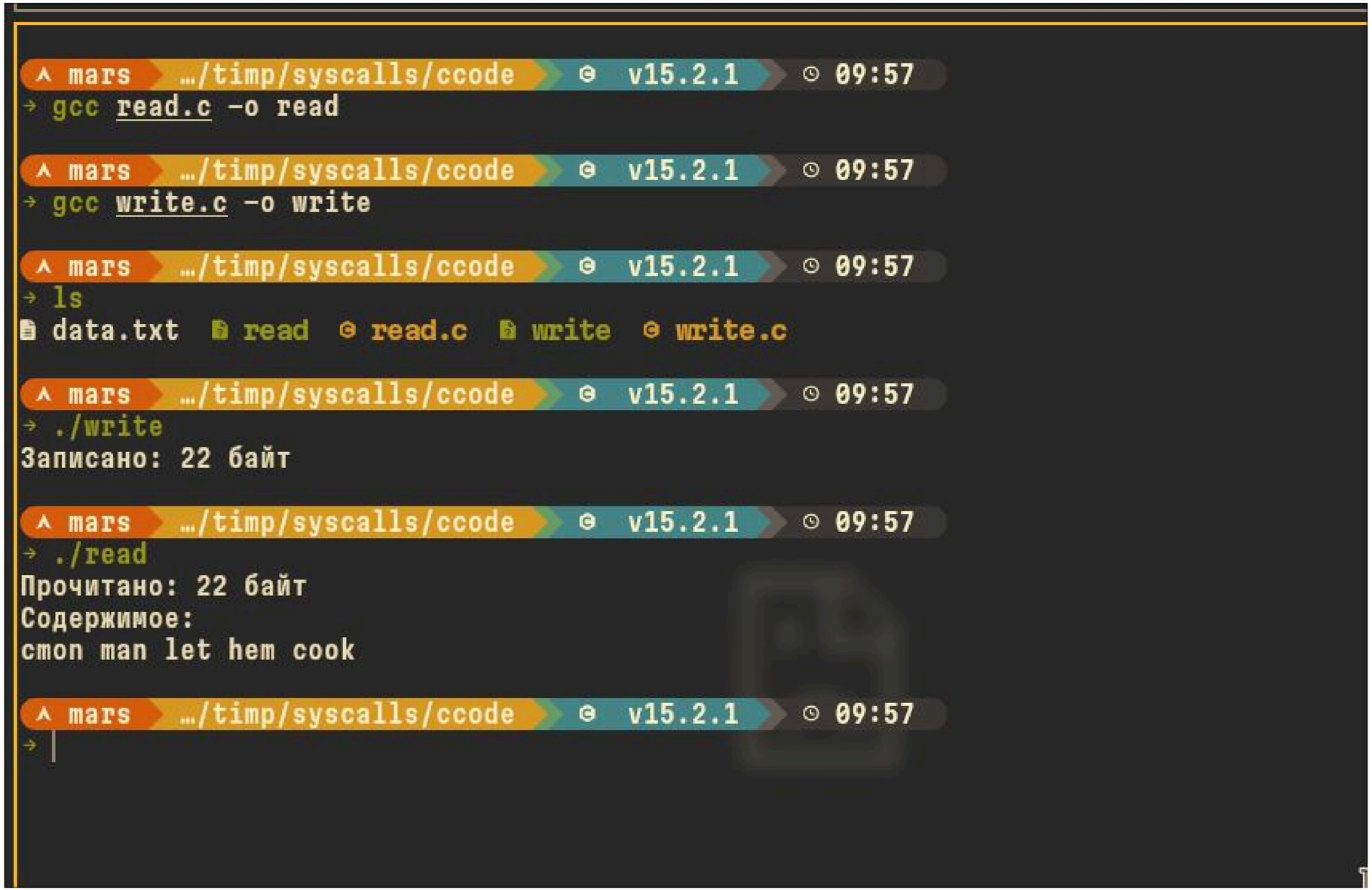
https://man7.org/linux/man-pages/man2/syscalls.2.html

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

#define BUFFER_SIZE 1024

int main() {
    const char *filename = "data.txt";
    char buffer[BUFFER_SIZE];

    int fd = open(filename, O_RDONLY);
    if (fd == -1) {
        perror("Ошибка открытия");
        return 1;
    }

    ssize_t bytes = read(fd, buffer, BUFFER_SIZE - 1);
    if (bytes == -1) {
        perror("Ошибка чтения");
        close(fd);
        return 1;
    }

    buffer[bytes] = '\0';
    printf("Прочитано: %ld байт\n", bytes);
    printf("Содержимое:\n%s", buffer);
    close(fd);

    return 0;
}
```

reading

writing

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main() {
    const char *filename = "data.txt";
    const char *text = "cmon man let hem cook\n";

    int fd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, 0644);
    if (fd == -1) {
        perror("Ошибка открытия");
        return 1;
    }

    ssize_t written = write(fd, text, strlen(text));
    if (written == -1) {
        perror("Ошибка записи");
        close(fd);
        return 1;
    }

    printf("Записано: %ld байт\n", written);
    close(fd);

    return 0;
}
```


mars …/timp/syscalls/ccode v15.2.1 09:57
→ gcc read.c -o read
mars …/timp/syscalls/ccode v15.2.1 09:57
→ gcc write.c -o write
mars …/timp/syscalls/ccode v15.2.1 09:57
→ ls
data.txt read read.c write write.c
mars …/timp/syscalls/ccode v15.2.1 09:57
→ ./write
Записано: 22 байт
mars …/timp/syscalls/ccode v15.2.1 09:57
→ ./read
Прочитано: 22 байт
Содержимое:
cmon man let hem cook
mars …/timp/syscalls/ccode v15.2.1 09:57
→

strace(1) — Linux manual page

NAME | SYNOPSIS | DESCRIPTION | OPTIONS | DIAGNOSTICS |
SETUID INSTALLATION | MULTIPLE PERSONALITIES SUPPORT | NOTES |
BUGS | HISTORY | REPORTING BUGS | SEE ALSO | AUTHORS |
COLOPHON

NAME top

strace - trace system calls and signals

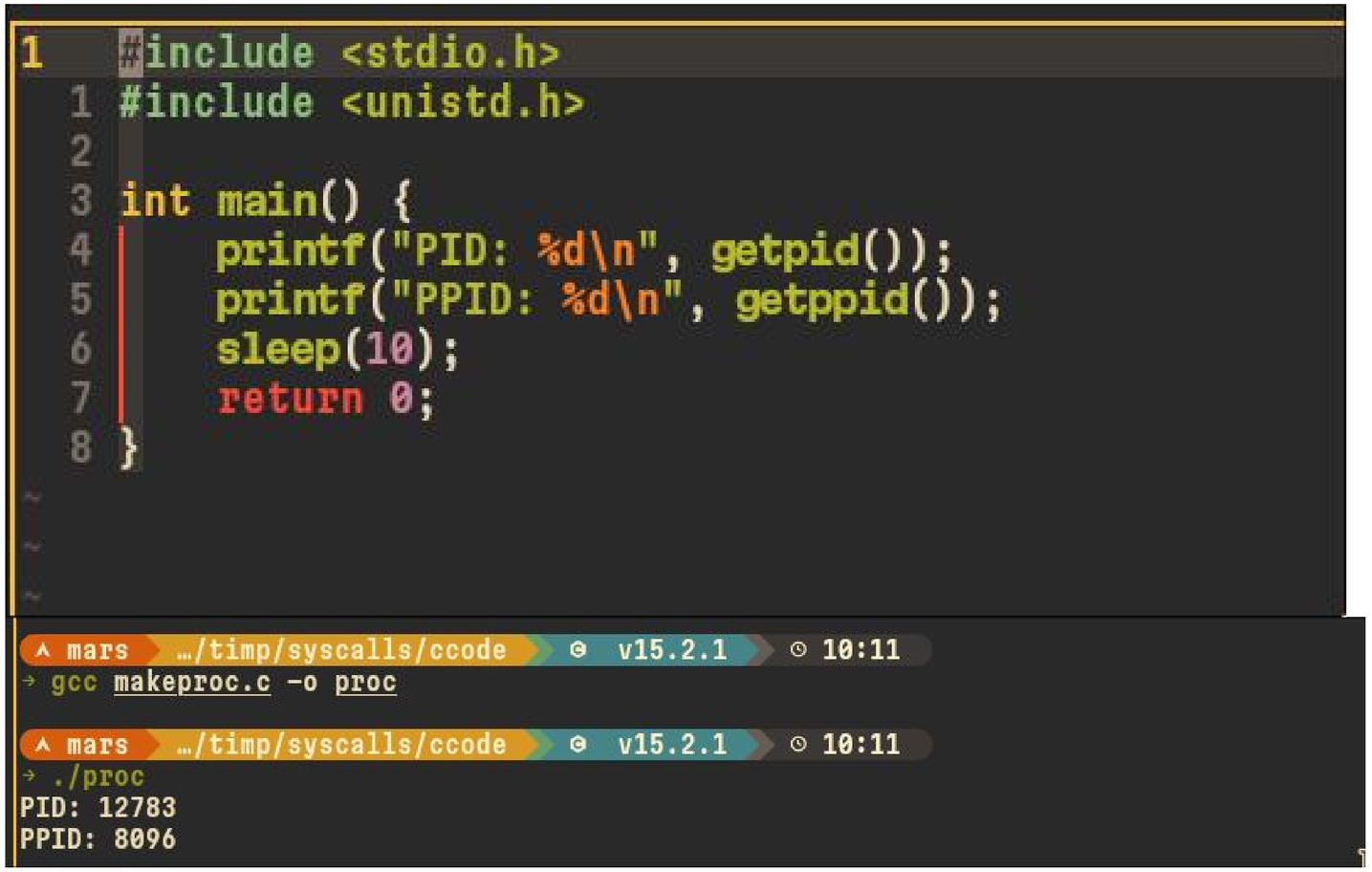
SYNOPSIS top

strace [**-ACdffhikkqqrtttTvVwxxyyYzZ**] [**-a** *column*] [**-b** *execve*]
[**-e** *expr*]... [**-I** *n*] [**-o** *file*] [**-O** *overhead*] [**-p** *pid*]...
[**-P** *path*]... [**-s** *strsize*] [**-S** *sortby*] [**-U** *columns*]
[**-X** *format*] [**--seccomp-bpf**]
[**--stack-trace-frame-limit=***limit*] [**--syscall-limit=***limit*]
[**--secontext**[**=***format*]] [**--tips**[**=***format*]] { **-p** *pid* | [**-DDD**]
[**-E** *var*[**=***val*]]... [**-u** *username*] *command* [*args*] }

strace -c [**-dfwzZ**] [**-b** *execve*] [**-e** *expr*]... [**-I** *n*] [**-O** *overhead*]
[**-p** *pid*]... [**-P** *path*]... [**-S** *sortby*] [**-U** *columns*]
[**--seccomp-bpf**] [**--syscall-limit=***limit*] [**--tips**[**=***format*]] {
-p *pid* | [**-DDD**] [**-E** *var*[**=***val*]]... [**-u** *username*] *command*
[*args*] }

strace --tips[**=***format*]

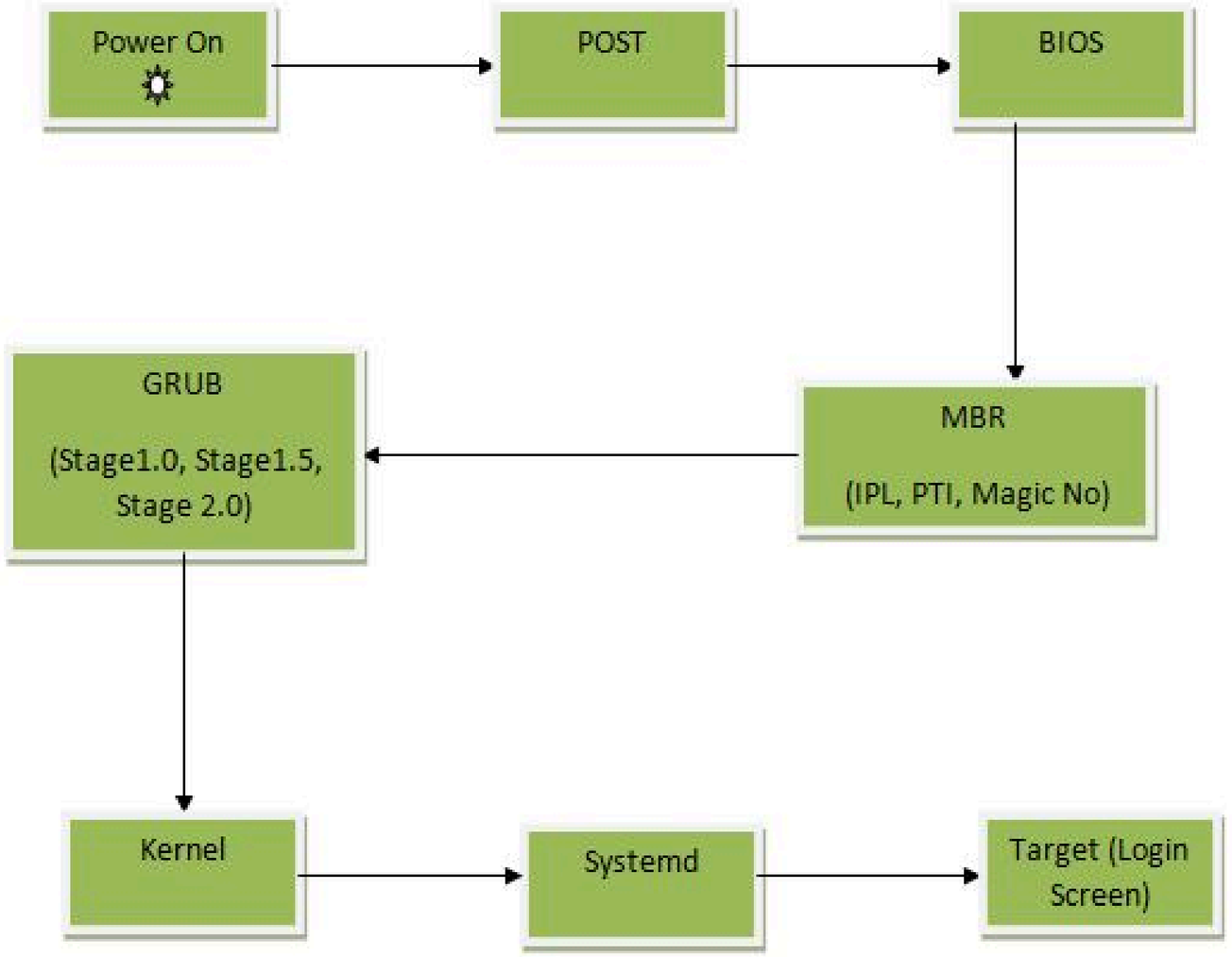
```
 mars …/timp/syscalls/ccode  v15.2.1  09:58
→ strace ./read
execve("./read", ["./read"], 0x7ffc55b191c0 /* 64 vars */) = 0
brk(NULL)                               = 0x561879fda000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=135727, ...}) = 0
mmap(NULL, 135727, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6ad9423000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360w\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2145632, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6ad9421000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
mmap(NULL, 2169904, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6ad9200000
mmap(0x7f6ad9224000, 1511424, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7f6ad9224000
mmap(0x7f6ad9395000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x195000) = 0x7f6ad9395000
mmap(0x7f6ad9404000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x203000) = 0x7f6ad9404000
mmap(0x7f6ad940a000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f6ad940a000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6ad941e000
arch_prctl(ARCH_SET_FS, 0x7f6ad941e740) = 0
set_tid_address(0x7f6ad941ea10)         = 9720
set_robust_list(0x7f6ad941ea20, 24)     = 0
rseq(0x7f6ad941e680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f6ad9404000, 16384, PROT_READ) = 0
mprotect(0x5618729f1000, 4096, PROT_READ) = 0
mprotect(0x7f6ad9486000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\x75\x01\x09\xe4\xab\xac\x7d\xcb", 8, GRND_NONBLOCK) = 8
munmap(0x7f6ad9423000, 135727)          = 0
openat(AT_FDCWD, "data.txt", O_RDONLY)  = 3
read(3, "cmon man let hem cook\n", 1023) = 22
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x561879fda000
brk(0x561879ffb000)                     = 0x561879ffb000
write(1, "\320\237\321\200\320\276\321\207\320\270\321\202\320\260\320\275\320\276: 22 \320\261\320\260\320\271\321\202\n", 32Прочитано: 22 байт
) = 32
write(1, "\320\241\320\276\320\264\320\265\321\200\320\266\320\270\320\274\320\276\320\265:\ncmon man l"..., 44Содержимое:
cmon man let hem cook
) = 44
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```


1 #include <stdio.h>
1 #include <unistd.h>
2
3 int main() {
4 printf("PID: %d\n", getpid());
5 printf("PPID: %d\n", getppid());
6 sleep(10);
7 return 0;
8 }
mars …/timp/syscalls/ccode v15.2.1 10:11
→ gcc makeproc.c -o proc
mars …/timp/syscalls/ccode v15.2.1 10:11
→ ./proc
PID: 12783
PPID: 8096

```
→ ./proc
PID: 12783
PPID: 8096

 mars …/timp/syscalls/ccode  v15.2.1  10:11
→ strace ./proc
execve("./proc", ["./proc"], 0x7ffd7ecd4e60 /* 64 vars */) = 0
brk(NULL)                               = 0x559508a1e000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=135727, ...}) = 0
mmap(NULL, 135727, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f379ea57000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360w\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2145632, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f379ea55000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
mmap(NULL, 2169904, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f379e800000
mmap(0x7f379e824000, 1511424, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7f379e824000
mmap(0x7f379e995000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x195000) = 0x7f379e995000
mmap(0x7f379ea04000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x203000) = 0x7f379ea04000
mmap(0x7f379ea0a000, 31792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f379ea0a000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f379ea52000
arch_prctl(ARCH_SET_FS, 0x7f379ea52740) = 0
set_tid_address(0x7f379ea52a10)         = 12849
set_robust_list(0x7f379ea52a20, 24)     = 0
rseq(0x7f379ea52680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f379ea04000, 16384, PROT_READ) = 0
mprotect(0x5594d3c47000, 4096, PROT_READ) = 0
mprotect(0x7f379eaba000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\x25\x8e\x60\x32\xe3\x60\x4b\x7b", 8, GRND_NONBLOCK) = 8
munmap(0x7f379ea57000, 135727)          = 0
getpid()                                = 12849
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x559508a1e000
brk(0x559508a3f000)                     = 0x559508a3f000
write(1, "PID: 12849\n", 11PID: 12849
)            = 11
getppid()                               = 12847
write(1, "PPID: 12847\n", 12PPID: 12847
)           = 12
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=10, tv_nsec=0}
, 0x7ffd50cc0810) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Демон (daemon)

1. **fork()** — создание дочернего процесса
2. **setsid()** — создание новой сессии и отсоединение от управляющего терминала
3. **fork()** — повторное ответвление для предотвращения получения терминала
4. **chdir("/")** — изменение рабочей директории на корень
5. **umask(0)** — сброс маски создания файлов
6. Закрытие всех файловых дескрипторов
7. Перенаправление **stdin, stdout, stderr** на **/dev/null**


Power On
POST
BIOS
MBR
(IPL, PTI, Magic No)
GRUB
(Stage1.0, Stage1.5, Stage 2.0)
Kernel
Systemd
Target (Login Screen)

systemd

- Параллельный запуск служб (быстрая загрузка)
- Управление зависимостями между службами
- Автоматический перезапуск упавших служб
- Сбор и управление логами через journald
- Активацию по требованию (socket activation)

Структура unit-файла systemd

[Unit] — общая информация о службе:
- Description — описание службы
- After — запуск после указанных служб
- Requires — жёсткие зависимости
- Wants — мягкие зависимости

[Service] — параметры запуска службы:
- Type — тип службы (simple, forking, oneshot, notify, dbus)
- ExecStart — команда запуска
- ExecStop — команда остановки
- Restart — политика перезапуска (always, on-failure)
- User — пользователь для запуска службы

Команды управления службами
- systemctl start <service> — запустить службу
- systemctl stop <service> — остановить службу
- systemctl restart <service> — перезапустить службу
- systemctl enable <service> — включить автозапуск
- systemctl disable <service> — отключить автозапуск
- systemctl status <service> — статус службы
- journalctl -u <service> — просмотр логов службы

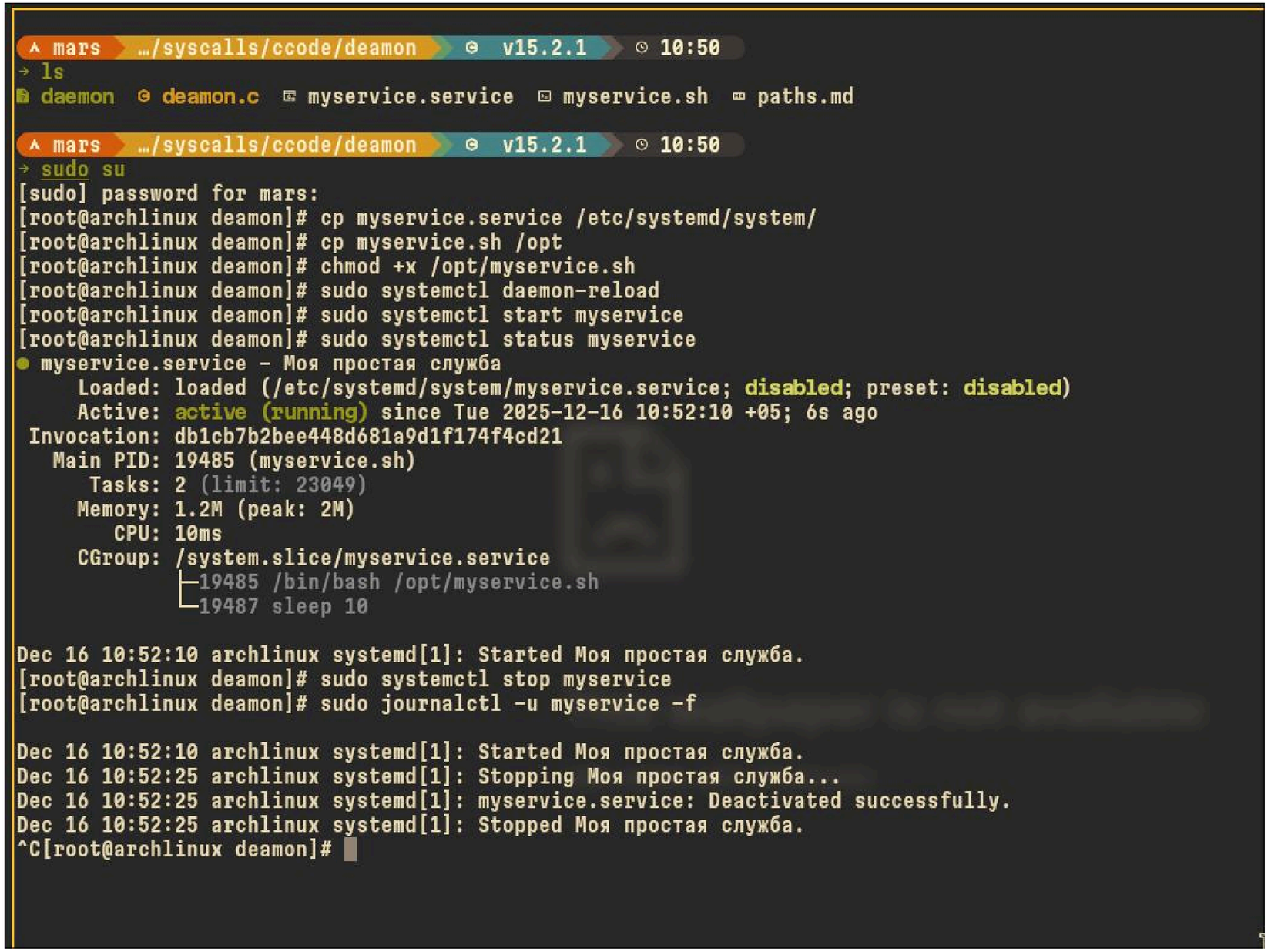
Тип	Файл	Назначение	Команда просмотра
service	.service	Процессы/демоны	`systemctl list-units --type=service`
target	.target	Группы units	`systemctl list-units --type=target`
timer	.timer	Расписание	`systemctl list-timers`
socket	.socket	Сокет-активация	`systemctl list-sockets`
mount	.mount	Монтирование	`systemctl list-units --type=mount`
path	.path	Слежение за файлами	`systemctl list-units --type=path`
device	.device	Устройства	`systemctl list-units --type=device`

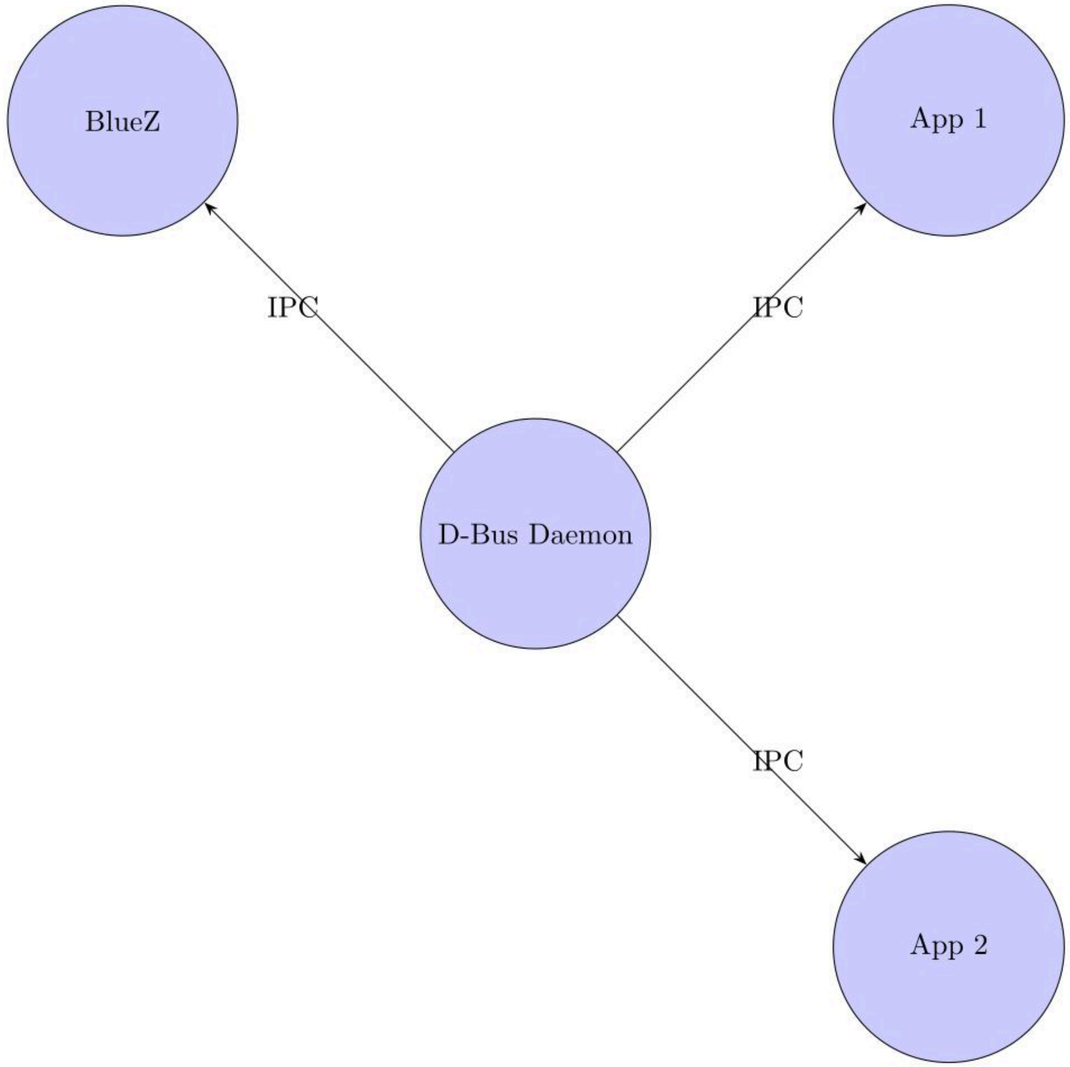
```c
#include <signal.h>
#include <time.h>

#define LOG_FILE "/tmp/daemon.log"

volatile sig_atomic_t running = 1;

void handle_signal(int sig) {
    running = 0;
}

void write_log(const char *msg) {
    FILE *f = fopen(LOG_FILE, "a");
    if (f) {
        time_t now = time(NULL);
        fprintf(f, "[%s] %s\n", ctime(&now), msg);
        fclose(f);
    }
}

int main() {
    // Создаём дочерний процесс
    if (fork() != 0) exit(0);

    // Создаём новую сессию
    setsid();

    // Обработчик сигнала завершения
    signal(SIGTERM, handle_signal);

    // Записываем PID
    FILE *pid = fopen("/tmp/daemon.pid", "w");
    if (pid) {
        fprintf(pid, "%d\n", getpid());
        fclose(pid);
    }

    write_log("Демон запущен");

    // Основной цикл
    while (running) {
        write_log("Работаю...");
        sleep(10);
    }

    write_log("Демон остановлен");
    unlink("/tmp/daemon.pid");

    return 0;
}
```

NORMAL deamon.c

```
[Unit]
Description=Моя простая служба

[Service]
ExecStart=/opt/myservice.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

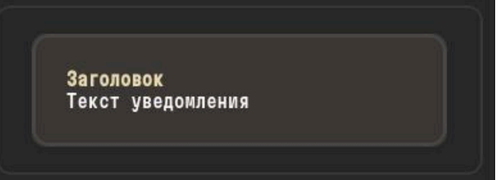
```
#!/bin/bash
while true; do
    echo "$(date): Служба работает" >> /tmp/myservice.log
    sleep 10
done
```


mars …/syscalls/ccode/deamon v15.2.1 10:50
→ ls
daemon deamon.c myservice.service myservice.sh paths.md
mars …/syscalls/ccode/deamon v15.2.1 10:50
→ sudo su
[sudo] password for mars:
[root@archlinux deamon]# cp myservice.service /etc/systemd/system/
[root@archlinux deamon]# cp myservice.sh /opt
[root@archlinux deamon]# chmod +x /opt/myservice.sh
[root@archlinux deamon]# sudo systemctl daemon-reload
[root@archlinux deamon]# sudo systemctl start myservice
[root@archlinux deamon]# sudo systemctl status myservice
● myservice.service - Моя простая служба
Loaded: loaded (/etc/systemd/system/myservice.service; disabled; preset: disabled)
Active: active (running) since Tue 2025-12-16 10:52:10 +05; 6s ago
Invocation: db1cb7b2bee448d681a9d1f174f4cd21
Main PID: 19485 (myservice.sh)
Tasks: 2 (limit: 23049)
Memory: 1.2M (peak: 2M)
CPU: 10ms
CGroup: /system.slice/myservice.service
├─19485 /bin/bash /opt/myservice.sh
└─19487 sleep 10
Dec 16 10:52:10 archlinux systemd[1]: Started Моя простая служба.
[root@archlinux deamon]# sudo systemctl stop myservice
[root@archlinux deamon]# sudo journalctl -u myservice -f
Dec 16 10:52:10 archlinux systemd[1]: Started Моя простая служба.
Dec 16 10:52:25 archlinux systemd[1]: Stopping Моя простая служба...
Dec 16 10:52:25 archlinux systemd[1]: myservice.service: Deactivated successfully.
Dec 16 10:52:25 archlinux systemd[1]: Stopped Моя простая служба.
^C[root@archlinux deamon]#


BlueZ
App 1
IPC
IPC
D-Bus Daemon
IPC
App 2

Метод	Сложность	Скорость	Где используется
Pipes (Каналы)	Простая	Быстро	Консольные команды
Named Pipes (FIFO)	Простая	Быстро	Связанные процессы
Signals (Сигналы)	Простая	Быстро	Управление процессами
Message Queues	Средняя	Средне	Очереди сообщений
Shared Memory	Сложная	Очень быстро	Высокая производительность
Semaphores	Средняя	Быстро	Синхронизация
Sockets	Средняя	Средне	Сетевое взаимодействие
D-Bus	Средняя	Средне	Desktop приложения

```
 mars …/syscalls/ccode/deamon  v15.2.1  11:01
→ dbus-send --session --print-reply --dest=org.freedesktop.Notifications \
    /org/freedesktop/Notifications \
    org.freedesktop.Notifications.Notify \
    string:"TestApp" \
    uint32:0 \
    string:"" \
    string:"Заголовок" \
    string:"Текст уведомления" \
    array:string:"" \
    dict:string:variant:"" \
    int32:-1

method return time=1765864891.877772 sender=:1.17 -> destination=:1.132 serial=331 reply_serial=2
   uint32 25

 mars …/syscalls/ccode/deamon  v15.2.1  11:01
→
```




```
# Пауза/Воспроизведение
dbus-send --print-reply \
  --dest=org.mpris.MediaPlayer2.spotify \
  /org/mpris/MediaPlayer2 \
  org.mpris.MediaPlayer2.Player.PlayPause

# Следующий трек
dbus-send --print-reply \
  --dest=org.mpris.MediaPlayer2.spotify \
  /org/mpris/MediaPlayer2 \
  org.mpris.MediaPlayer2.Player.Next

# Предыдущий трек
dbus-send --print-reply \
  --dest=org.mpris.MediaPlayer2.spotify \
  /org/mpris/MediaPlayer2 \
  org.mpris.MediaPlayer2.Player.Previous
```

```
 mars …/timp/syscalls/ccode  v15.2.1  11:06
→ ls
 data.txt  deamon  makeproc.c  proc  read  read.c  write  write.c

 mars …/timp/syscalls/ccode  v15.2.1  11:06
→ ls -l | grep make
.rw-r--r-- 156 mars 16 Dec 10:10  makeproc.c

 mars …/timp/syscalls/ccode  v15.2.1  11:06
→ ls -l | grep write
.rwxr-xr-x 16k mars 16 Dec 09:57  write
.rw-r--r-- 613 mars 16 Dec 09:56  write.c

 mars …/timp/syscalls/ccode  v15.2.1  11:06
→
```