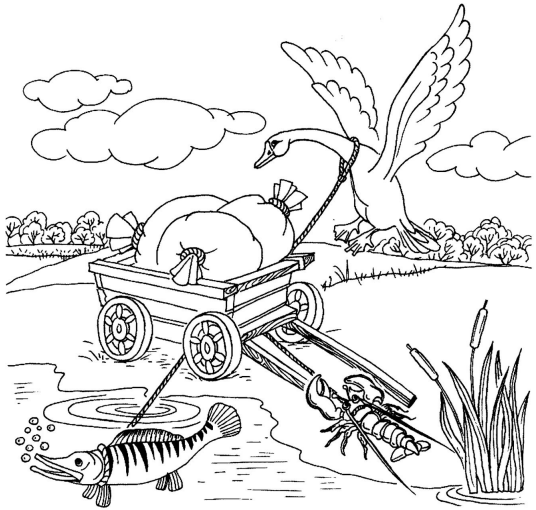
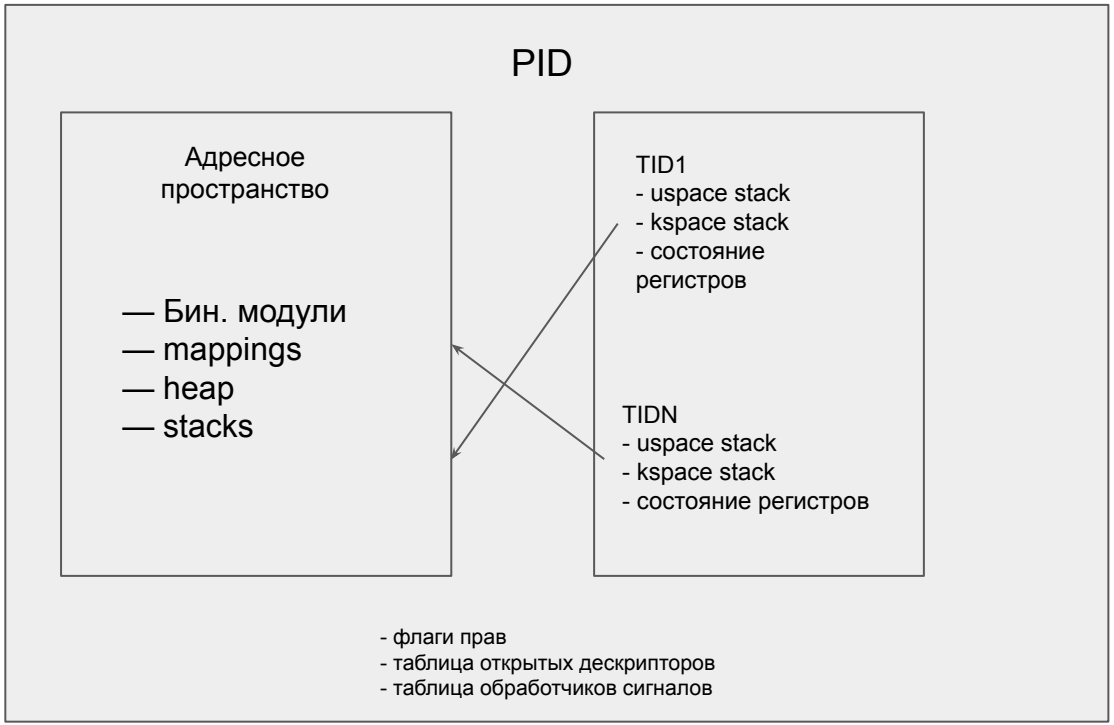
Программирование в Linux

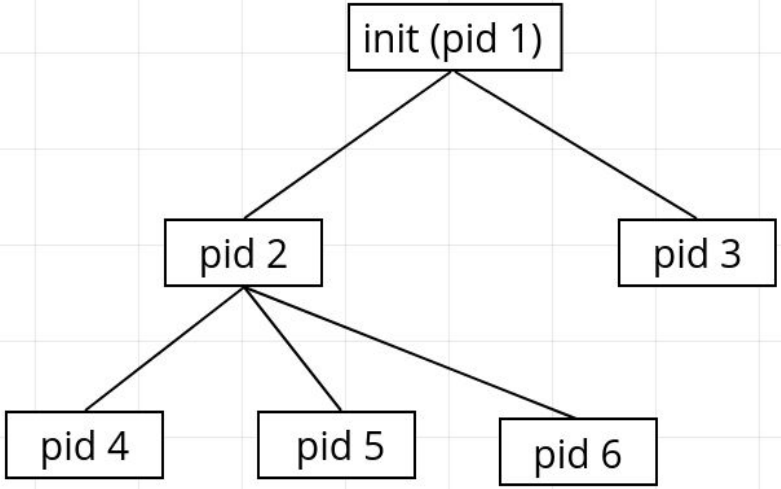


Процессы и права

Процесс ~ контейнер. Thread — единица исполнения



Порождение процессов




```c
#define _GNU_SOURCE
#include <sched.h>

int clone(int (*fn)(void *), void *stack, int flags, void *arg, ...
          /* pid_t *parent_tid, void *tls, pid_t *child_tid */ );
```

```c
#include <sys/types.h>
#include <unistd.h>

pid_t fork(void);
```

```c
#include <sys/types.h>
#include <unistd.h>

pid_t vfork(void);
```

fork vs vfork

```c
int main(int argc, char* argv) {

    int x = 42;

    pid_t child = fork();

    if (child < 0) {
        return EXIT_FAILURE;
    }

    // parent and child run simultaneously
    if (child == 0) {
        printf("Hello from child, %d\n", x);
    } else {
        x = 43; // ok!
        printf("Hello from parent of %d, answer =%d\n", child, x);
    }
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/fokrs$ ./fork_eg
Hello from parent of 160767, answer =43
Hello from child, 42
```

```c
int main(int argc, char* argv) {

    int x = 42;

    pid_t child = vfork();

    if (child < 0) {
        return EXIT_FAILURE;
    }

    // parent is blocked until child runs exec or exit
    if (child == 0) {
        // x = 43; // UB!
        // printf("Hello from child, %d\n", x); UB!!!!

        execlp("cat", "dog", "--help", NULL);
    } else {
        printf("Hello from parent of %d, answer =%d\n", child, x);
    }
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/fokrs$ ./vfork_eg
Hello from parent of 161553, answer =42
dmis@dmis-MS-7A15:~/LinuxEgs/fokrs$ Usage: dog [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.
```

fork делает CoW копию адресного пространства. vfork — нет

Замена бинарного модуля (exec)

```c
#include <unistd.h>

int main() {
    execlp("cat", "dog", "--help", NULL);
    // Unreachable
    return 1;
}
```

```c
#include <unistd.h>

extern char **environ;

int execl(const char *pathname, const char *arg, ...
                /* (char  *) NULL */);
int execlp(const char *file, const char *arg, ...
                /* (char  *) NULL */);
int execle(const char *pathname, const char *arg, ...
                /*, (char *) NULL, char *const envp[] */);
int execv(const char *pathname, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[],
                char *const envp[]);
```

PID сохраняется
Адресное пространство новое
Открытые дескрипторы сохраняются

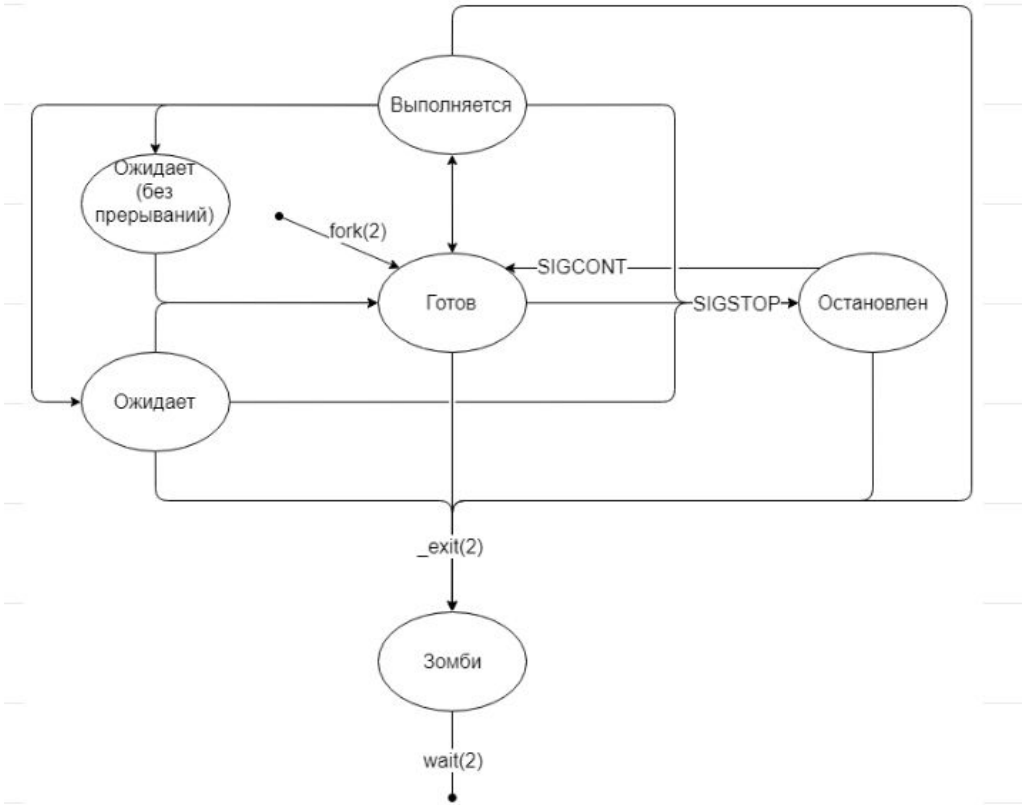
*p-версии умеют запускать скрипты (#!)

std::system

```
int exit_code1 = std::system("echo hello1");
printf("exit_code1 %d\n", exit_code1);
```

```
auto child = vfork();
if (child < 0) {
    perror("vfork");
    return EXIT_FAILURE;
}
if (child == 0) {
    execl("/bin/echo", "/bin/echo", "hello2", NULL);
} else {
    int exit_code2 = 0;
    waitpid(child, &exit_code2, WEXITED);
    printf("exit_code2 %d\n", WEXITSTATUS(exit_code2));
}
```

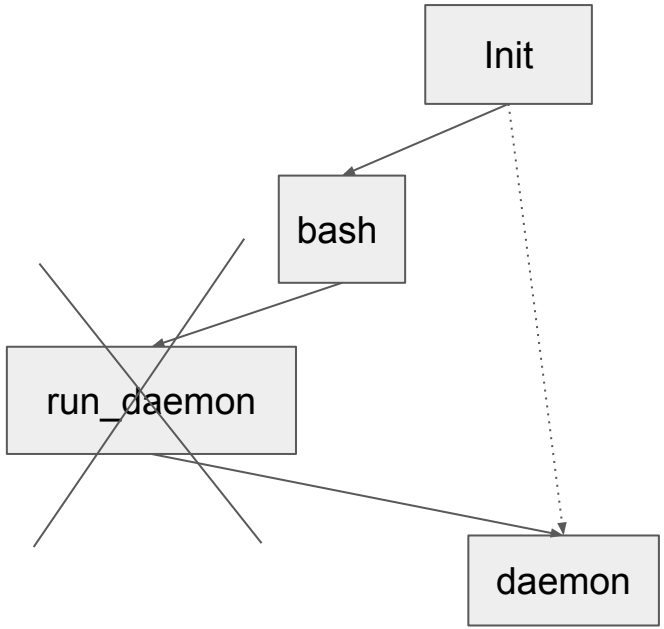
Жизненный цикл процесса



Зомби — метаинформация о статусе завершенного процесса
— Занимает место
— Освобождается после обслуживания (wait) от родителя
— Если родитель мертв, его заменяет init
— Init обслуживает всех своих (приемных) детей

“Демонение” процесса

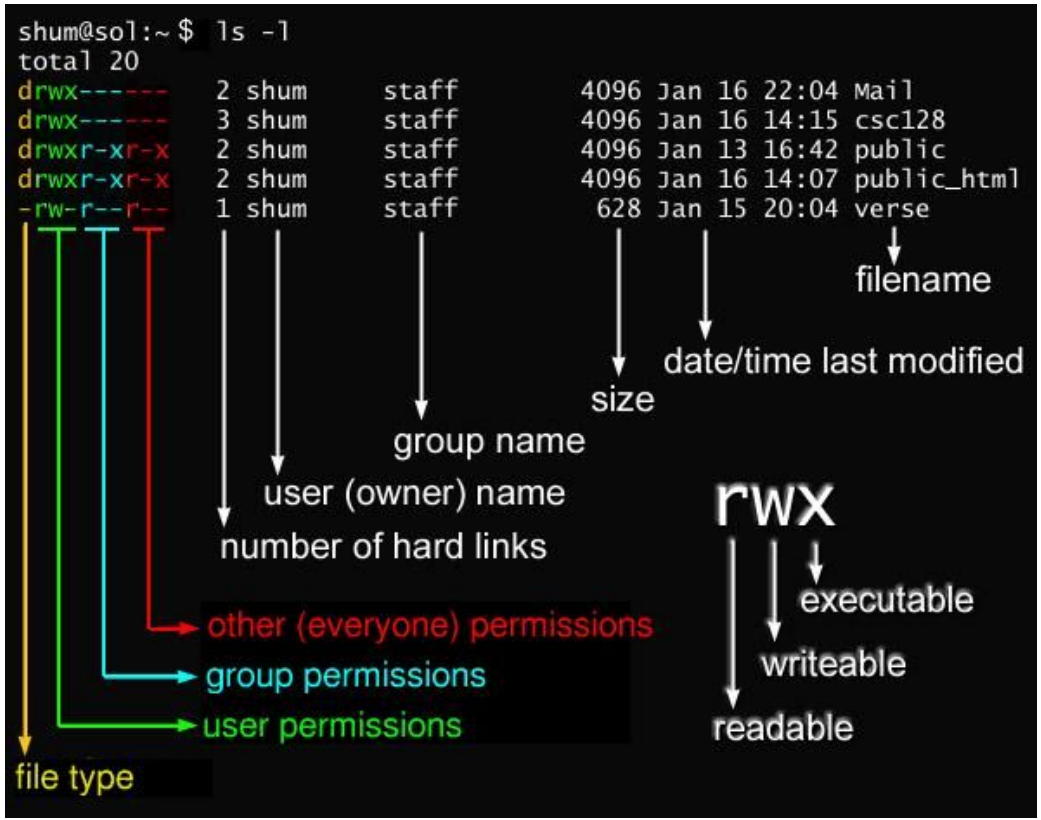
```c
#include <unistd.h>

int daemon(int nochdir, int noclose);
```



1. Делаем fork
2. Убиваем родителя
3. Потомок становится потомком init
4. Потомок закрывает все файловые дескрипторы
5. Потомок больше не привязан к терминалу и работает в фоне

Механизмы прав доступа

Пользователи и группы. Доступ и владение файлами




```c
#include <sys/stat.h>

int chmod(const char *pathname, mode_t mode);
int fchmod(int fd, mode_t mode);
```

```c
int chown(const char *pathname, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *pathname, uid_t owner, gid_t group);
```

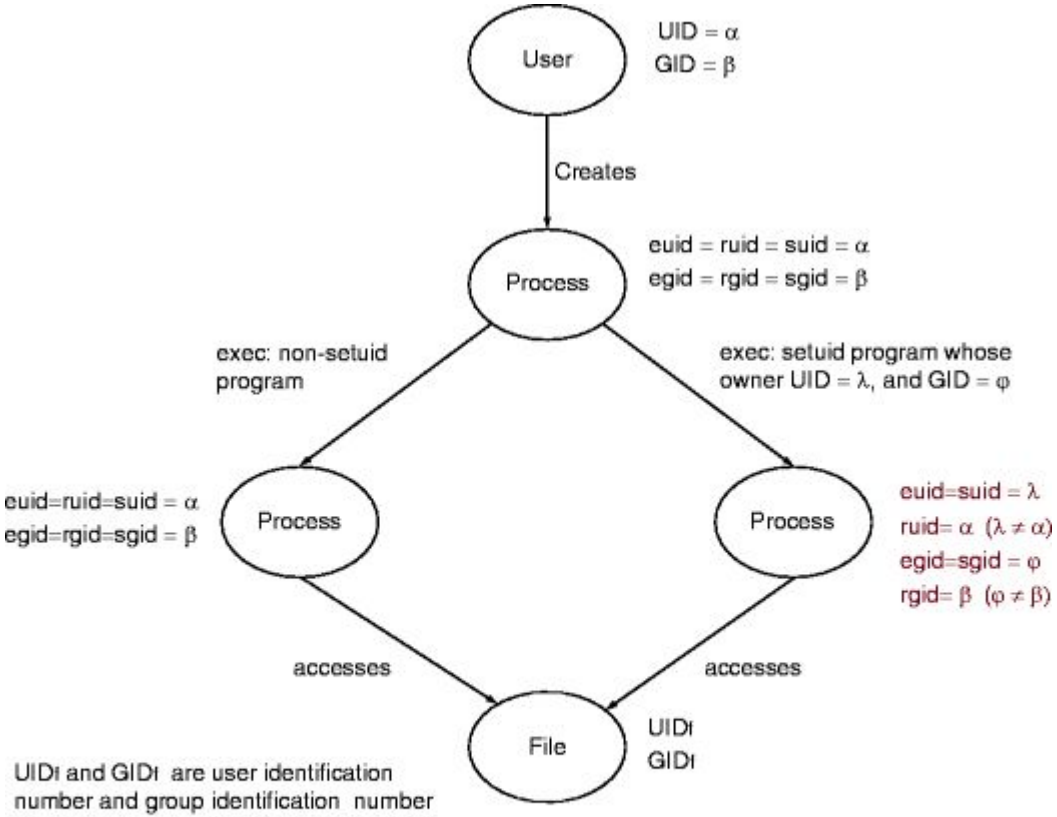
Пользователи и группы. Процессы

Исполнимый файл: UID, GID, SUID-bit. SGID-bit

Запущенный процесс:

- Real user ID (RUID) — кто запустил процесс?
- Effective user ID (EUID) — от чьего имени процесс выполняет запросы?
- Saved user ID (SUID) — кем можно временно прикинуться?

Аналогично для групп

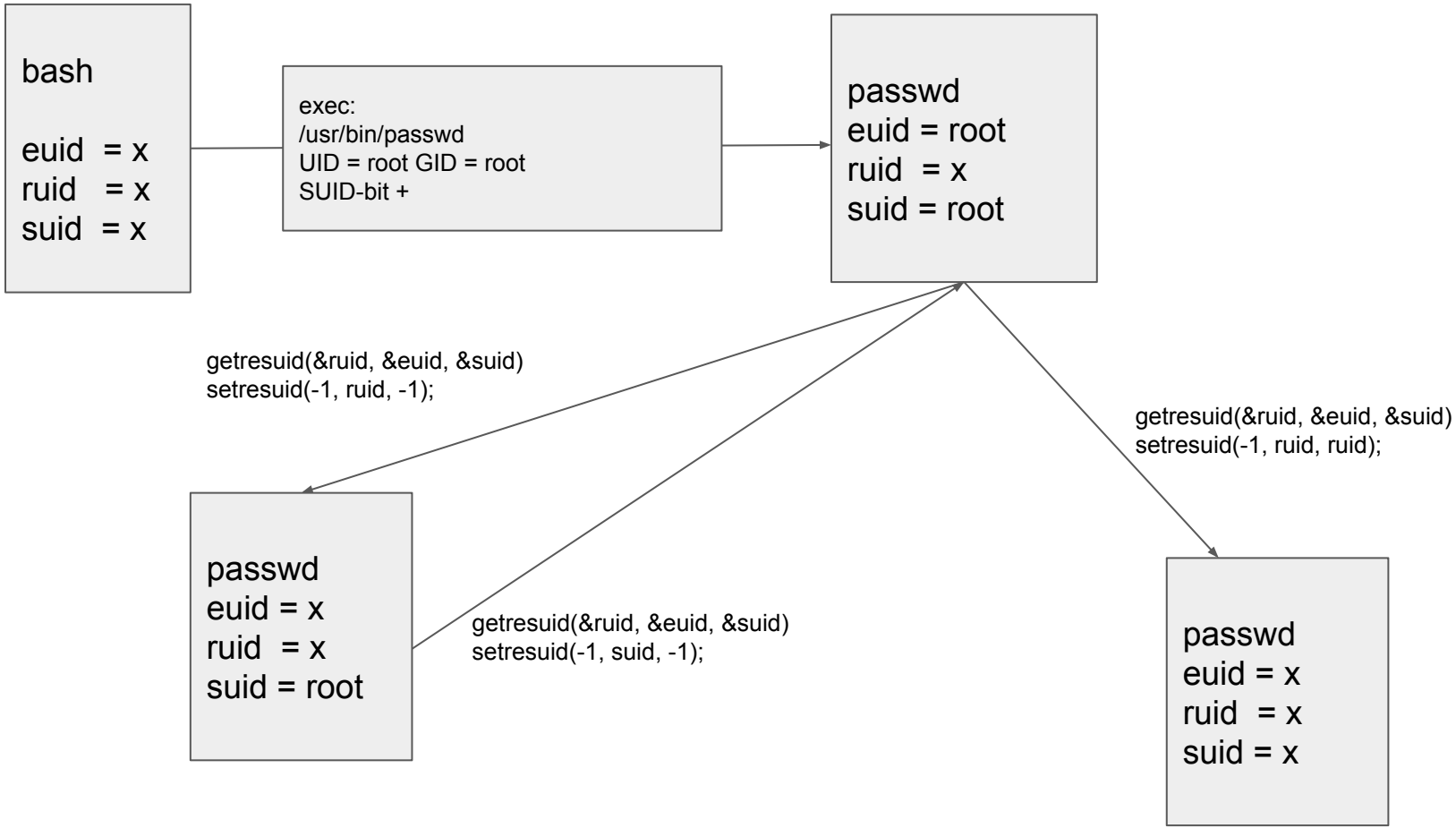


UIDf and GIDf are user identification number and group identification number for file f.

Let $P_f(x)$ denote access permissions for file f, where x ∈ {owner, group, others}.

Accessible set F of files for process:

$$F = \{f \mid [(euid = UIDf) \wedge (P_f(owner))] \vee [(euid \neq UIDf) \wedge (egid = GIDf)\ (P_f(group))] \vee [euid \neq UIDf) \wedge (egid \neq GIDf) \wedge (P_f(others))]\}$$


bash
euid = x
ruid = x
suid = x
exec:
/usr/bin/passwd
UID = root GID = root
SUID-bit +
passwd
euid = root
ruid = x
suid = root
getresuid(&ruid, &euid, &suid)
setresuid(-1, ruid, -1);
getresuid(&ruid, &euid, &suid)
setresuid(-1, ruid, ruid);
passwd
euid = x
ruid = x
suid = root
getresuid(&ruid, &euid, &suid)
setresuid(-1, suid, -1);
passwd
euid = x
ruid = x
suid = x