Операционные системы

Процессы

Олег Французов

2017

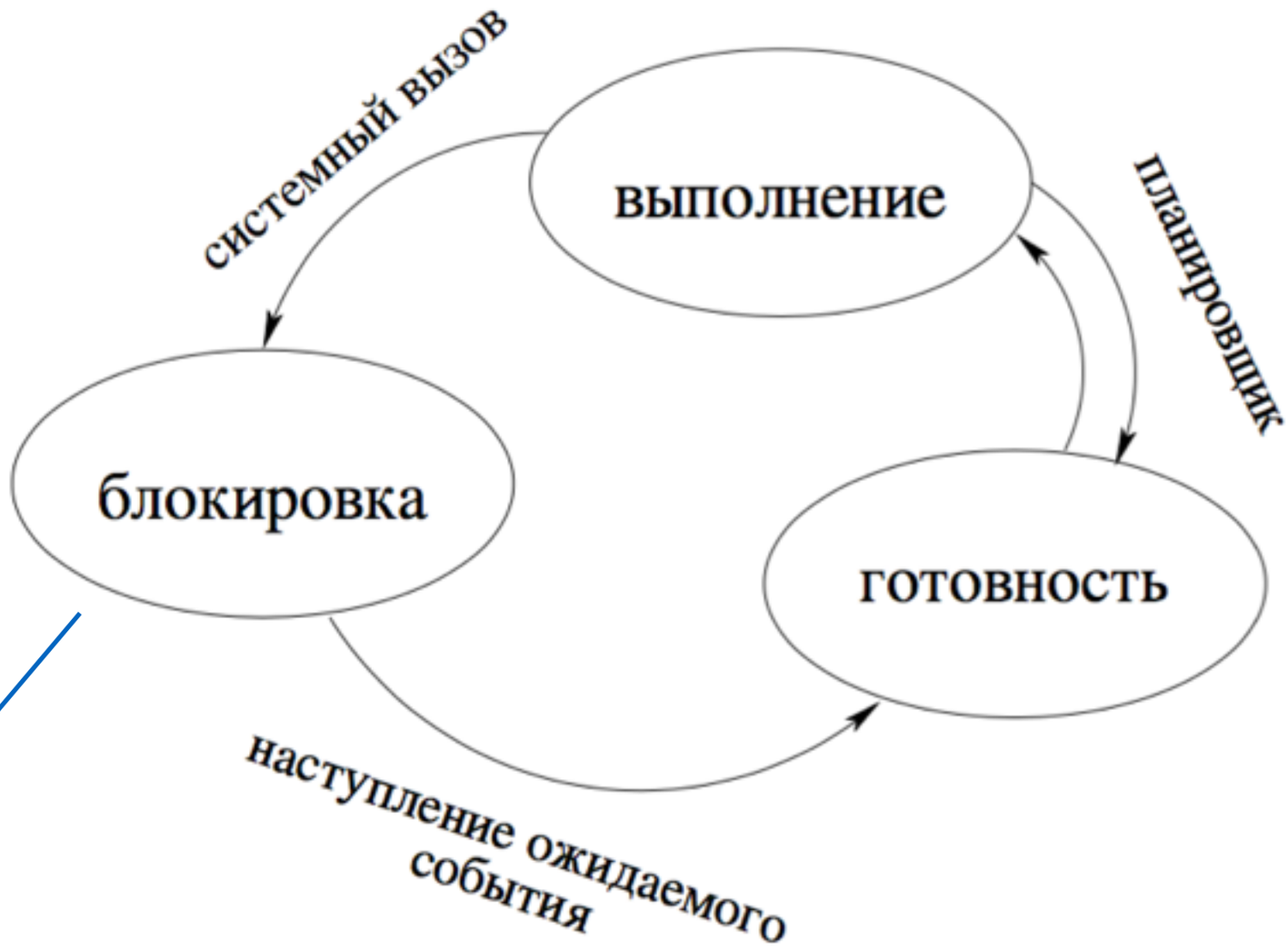
Свойства процесса

- Процесс – программа, которая выполняется под управлением ОС
- Разные экземпляры программы — различные процессы

Свойства процесса

- Сегменты кода и данных
- Стек
- Счетчик команд
- Права и полномочия
- Ресурсы (напр., открытые файлы)
- Идентификатор процесса

Состояния процесса




```
sleep()
pause()
```

Легковесные процессы

- Тж. *нити*, *потоки* или *треды*
- Общий код, данные и ресурсы
- Отдельный стек, счетчик команд
- Пользовательский интерфейс
  и долгая фоновая операция

Процессы в ОС Unix

Свойства процессов в Unix (I)

1. Сегмент кода (разделяется)
2. Сегмент данных (включая стек)
3. Состояние регистров (PC, PSW, SP, etc.)
4. Таблица дескрипторов файлового I/O
5. Командная строка
6. Окружение
7. Текущий каталог
8. Корневой каталог

Свойства процессов в Unix (2)

9. Диспозиция обработки сигналов
10. Параметр `umask`
11. Счетчики потребленных ресурсов
12. Информация о владельце:
    `uid`, `gid`, `euid`, `egid`
13. Идентификаторы процесса (`pid`),
    родительского процесса (`ppid`),
    сеанса и группы процессов

Порождение процесса

- Копирование существующего процесса
- Общий сегмент кода, остальное копируется
- Переменные, файловые дескрипторы, …

```
int fork(void);

-> pid дочернего процесса
   0 в самом дочернем процессе
   -1 в случае ошибки
```

Замена выполняемой программы

```
int execve(const char *path,
           char *const argv[],
           char *const envp[]);
```

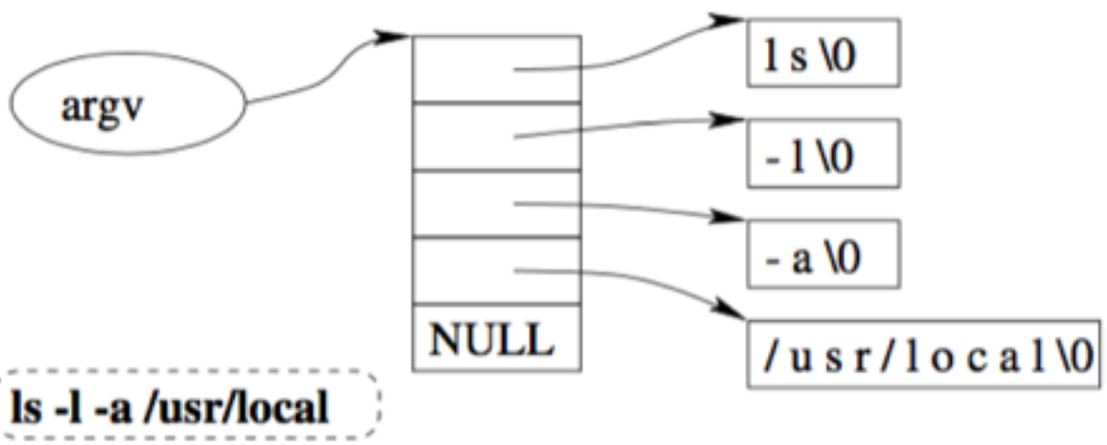
-> -1 в случае ошибки

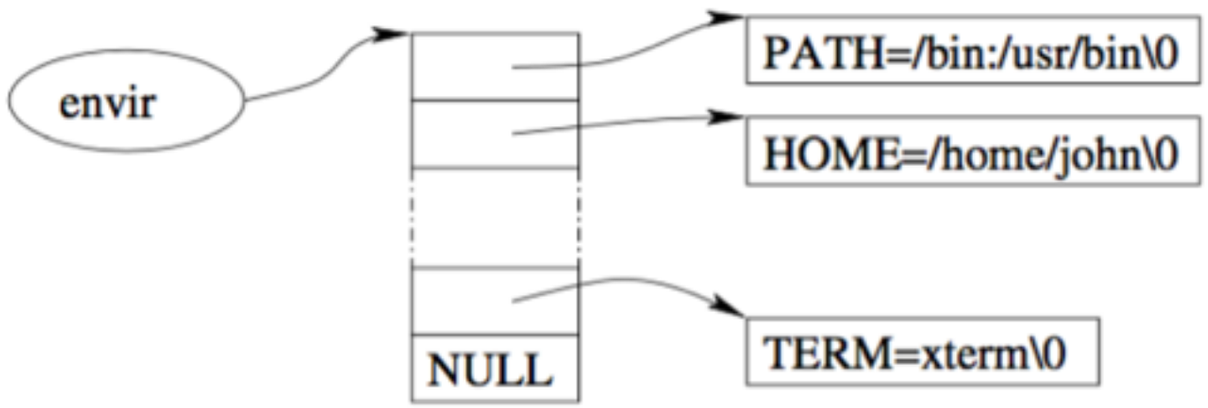
path – путь к исполняемому файлу

Замена выполняемой программы

```
int execve(const char *path,
           char *const argv[],
           char *const envp[]);
```

argv, envir:





Замена выполняемой программы

```
int execv(const char *path,
          char *const argv[]);
// Окружение наследуется
```

```
int execvp(const char *path,
           char *const argv[]);
// Искать исполняемый файл в PATH
```

Замена выполняемой программы

```
int execl(const char *path,
          const char argv0, ...);

int execlp(const char *path,
           const char argv0, …);

// Иной способ указания аргументов
```

Замена выполняемой программы

```
// $ ls -l -a /var

char *argv[] = { "ls", "-l", "-a",
                 "/var", NULL };
execvp("ls", argv);

execlp("ls", "ls", "-l", "-a",
       "/var", NULL);
```

Как shell запускает программы?

- `fork()`
  - все переменные копируются
  - `exec()` в дочернем процессе

- для перенаправления – настройка fd (они переживают вызов `exec()`)

Завершение процесса

```
void exit(int code);
// Или return code из main

Код завершения процесса:
  0 – успех
  1, 2, 3, … – неудача

$ prog && echo Success
$ prog || echo Failure
$ if prog; then echo Success; fi
```

Процессы-зомби и их обработка

- После завершения процесса в системе остается информация:
  - код возврата или сигнал
  - счетчики потребленных ресурсов
- Для родительского процесса
- Процесс в состоянии «зомби»

Ожидание дочернего процесса

```
int wait(int *status);

-> -1 если нет дочерних процессов
      или при другой ошибке
   pid (ожидает завершения любого
      из дочерних процессов)

status – анализируется при помощи
   WIFEXITED, WIFSIGNALED
   WEXITSTATUS, WTERMSIG
```

Ожидание дочернего процесса

```
int wait4(int pid, int *status,
          int options,
          struct rusage *rusage);
```

pid
- какой процесс ждать
- -1 – любой

options == ENOHANG
- не ждать, если нет зомби (-> 0)

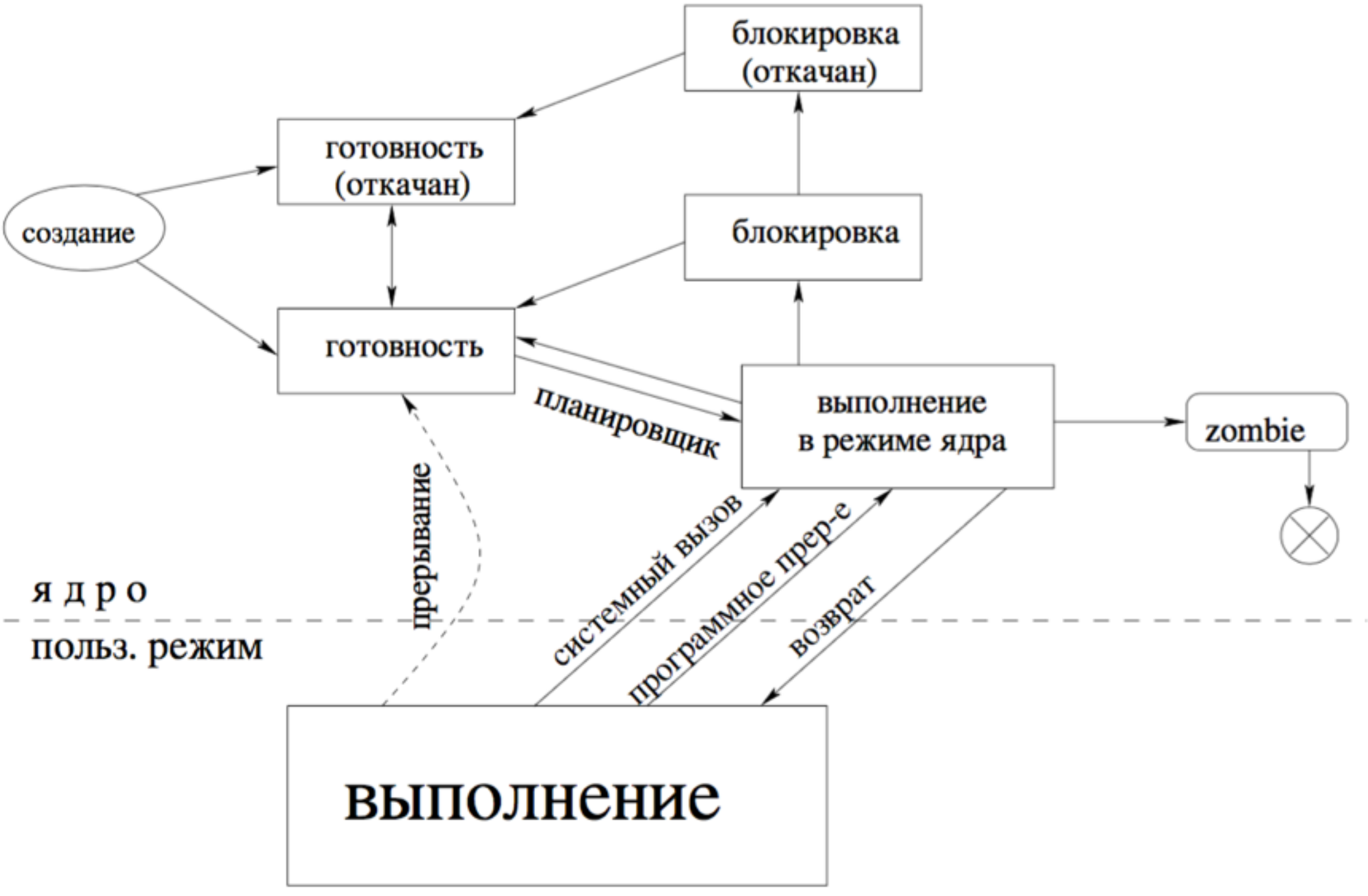
```
int pid;

pid = fork();
if (pid == -1) {
    perror("fork");
    exit(1);
}
```

```
if (pid == 0) { /* in child */
    execlp("ls", "ls", "-l", NULL);
    perror("ls");
    exit(1);
}

wait(NULL);
```

Жизненный цикл процесса



Гонки

```
int i = 0;
if (fork() == 0) {
    for (int i = 0; i < 10; i++) {
        printf("child %d\n", i);
        sleep(1);
    }
} else {
    for (int i = 0; i < 10; i++) {
        printf("parent %d\n", i);
        sleep(1);
    }
}
```

Результат недетерминирован

slides — -bash — 80×26

```
hilt:slides oleg$ ./race
parent 0
child 0
child 1
parent 1
child 2
parent 2
child 3
parent 3
child 4
parent 4
parent 5
child 5
parent 6
child 6
parent 7
child 7
child 8
parent 8
parent 9
child 9
hilt:slides oleg$
```

Q & A