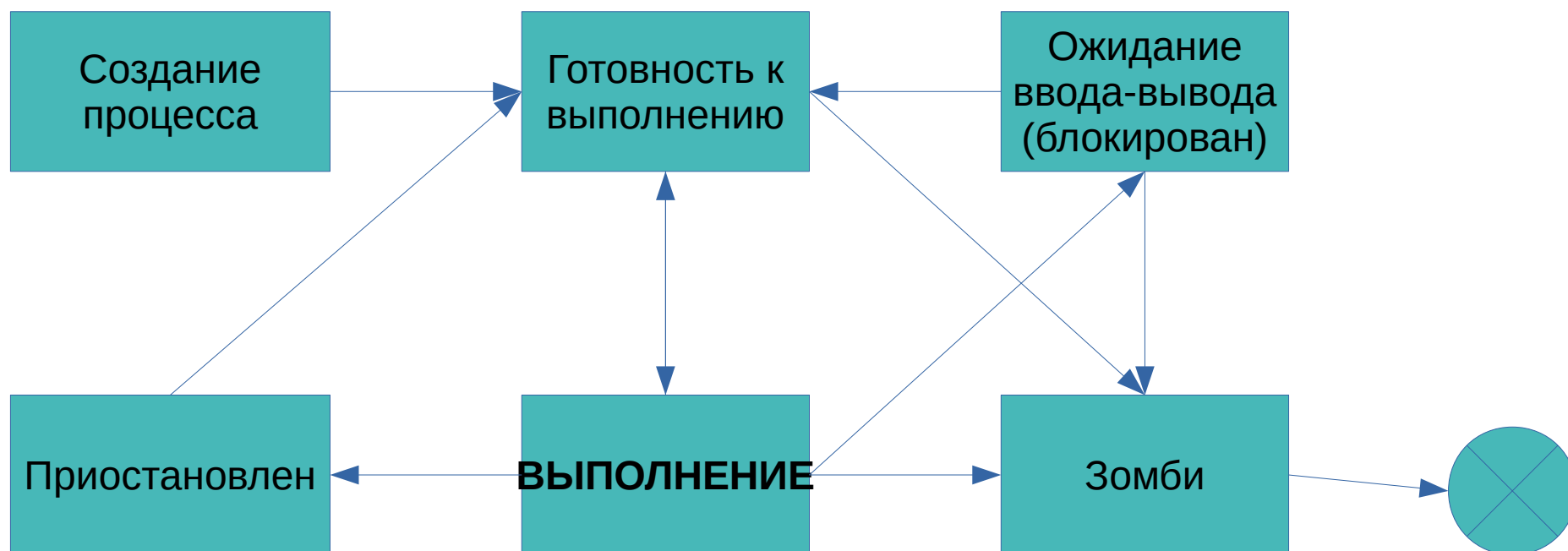


# Семинар 8

## процессы

# Жизненный цикл процесса



# Состояния процесса

- Готов к выполнению/выполняется (TASK\_RUNNING) — ядро берет процессы из очереди готовых к выполнению и ставит на выполнение на процессор, обозначается 'R'
- Ожидает ввода-вывода (спит)
  - TASK\_INTERRUPTIBLE ('S') - обычное ожидание, ожидание может быть прервано сигналом (в частности, завершения процесса)
  - TASK\_UNINTERRUPTIBLE ('D') — непрерываемое ожидание, процесс не может быть убит. Обычно используется при операциях ввода-вывода с устройством, на котором размещена файловая система (диски)

# Состояния процесса

- TASK\_ZOMBIE ('Z') — ожидание опроса состояния завершения процесса
- TASK\_STOPPED ('T') — процесс приостановлен, то есть не ждет i/o, но и не готов к выполнению. Либо приостановлен пользователем (Ctrl-Z с терминала), либо сигналом (SIGSTOP), либо попытка считать с терминала в фоновом режиме
- TASK\_TRACED ('t') — процесс отлаживается

# Состояния процессов

```
sleep(10);
```

Из состояния 'R' процесс переводится в состояние 'S', через 10 с процесс будет разбужен и переведен в состояние 'R'

```
read(0, buf, sizeof(buf)); // чтение из stdin
```

Если символов в буфере ввода нет, процесс переводится в состояние 'S' и будет оставаться в нем, пока не появятся данные, после чего будет переведен в состояние 'R'

```
read(fd, buf, sizeof(buf)); // чтение с диска
```

На время обмена с диском процесс находится в 'D'

# Замещение тела процесса

- Замещение тела процесса — запуск на выполнение другого исполняемого файла в рамках текущего процесса
- Для замещения тела процесса используется семейство `exes*`: сист. вызов `exesve` и функции `exesv`, `exesvr`, `exesl`, `exeslr`, `exesle`
  - «v» - передается массив параметров
  - «l» - передается переменное число параметров
  - «e» - передается окружение
  - «r» - выполняется поиск по PATH

# Системный вызов `execve`

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

- `path` — путь к исполняемому файлу
- `argv` — массив аргументов командной строки, заканчивается элементом `NULL`
- `envp` — массив переменных окружения, заканчивается элементом `NULL`
- Аргументы командной строки и переменные окружения помещаются на стек процесса
- При успехе системный вызов не возвращается

# execve

- `argv[0]` обычно совпадает с `path` (но не обязательно)
- Переменная `'environ'` — текущее окружение процесса
- `'environ'` должна быть объявлена явно, если требуется
- «Shebang» конструкции (`#!/usr/bin/python`) обрабатываются ядром



# Сохранение атрибутов процесса

- Сохраняются все атрибуты, **за исключением**
  - Атрибутов, связанных с адресным пространством процесса
  - Сигналов, ожидающие доставки
  - Таймеров

# Функция `execvp`

```
int execvp(const char *file, const char *arg, ...);
```

- Выполняется поиск исполняемого файла `file` по каталогам, перечисленным в переменной окружения `PATH`
- Аргументы запускаемого процесса передаются в качестве параметров функции `execvp`
- Последним аргументом функции должен быть `NULL`

# Схема fork/exec

- Для запуска программ в отдельных процессах применяется комбинация fork/exec
- Системный вызов fork создает новый процесс
- В сыновнем процессе системными вызовами настраиваются параметры процесса (например, текущий рабочий каталог, перенаправления стандартных потоков и пр.)
- Вызовом exec\* запускается требуемый исполняемый файл

# Пример

```
int main(void)
{
    int pid, status, fd;
    pid = fork();
    if (!pid) {
        chdir("/usr/bin");
        fd = open("/tmp/log.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600);
        dup2(fd, 1); close(fd);
        execlp("/bin/ls", "/bin/ls", "-l", NULL);
        fprintf(stderr, "Exec failed\n");
        _exit(1);
    }
    wait(&status);
    return 0;
}
```

# Подготовка аргументов командной строки

- Часто необходимо запустить программу, если передана строка состоящая из имени программы и аргументов

```
int system(const char *command);
```

Например: `res = system("ls -l");`

Реализация с помощью `execl`:

```
execl("/bin/sh", "sh", "-c", command, NULL);
```