

Основной критерий разработки(эффективности) ОС



Производительность

Количество
выполненных
вычислений за
единицу времени

Системы пакетной
обработки

Удобство пользователей

Возможность
взаимодействовать с
компьютером; запускать
одновременно несколько
задач в т.ч. на
одноядерных системах

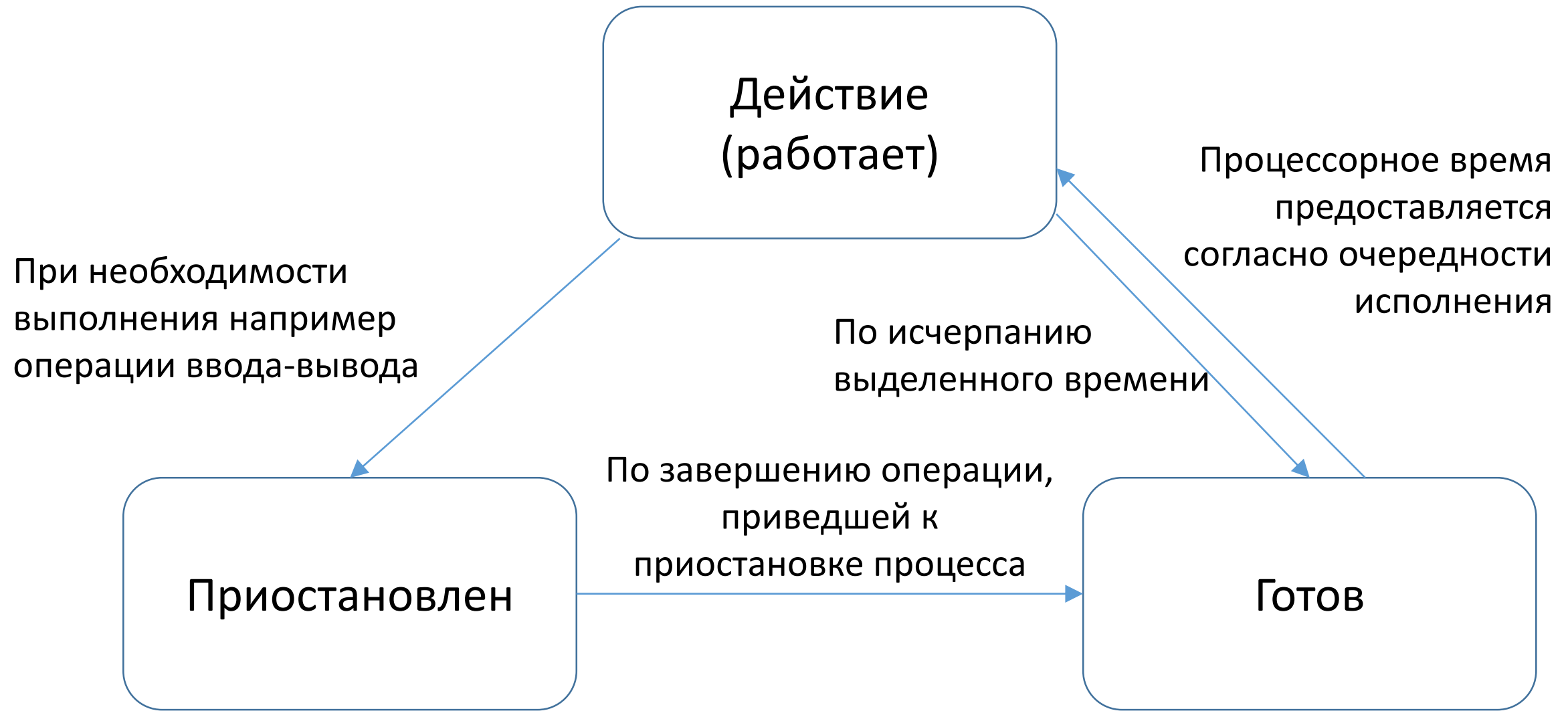
Системы разделения
времени

Время реакции

При любых
обстоятельствах
выдерживать жестко
заданные интервалы
между входным и
выходным сигналом

Системы реального
времени

Состояния процессов



Многозадачность в системах разделения времени

Системы разделения времени основаны на том, что за счет частого переключения между процессами складывается мнение, что программы работают параллельно, хотя это не так.

Невытесняющие – переключение по инициативе пользователя

Пример - DESQview 1985г для DOS

Совместная/кооперативная - переключение по инициативе приложения

Пример – MS Windows 1 .. 3

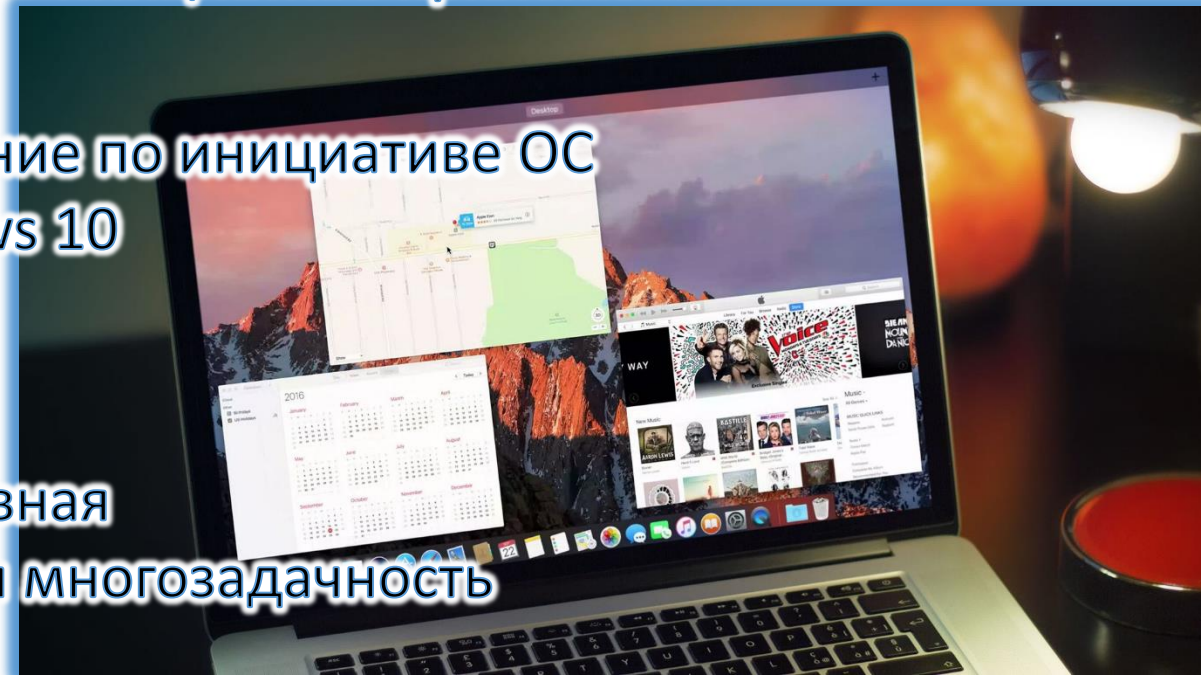
Вытесняющая многозадачность – переключение по инициативе ОС

Пример – Unix, Windows NT...Windows 10

Альтернативная классификация

Невытесняющие алгоритмы <-> кооперативная

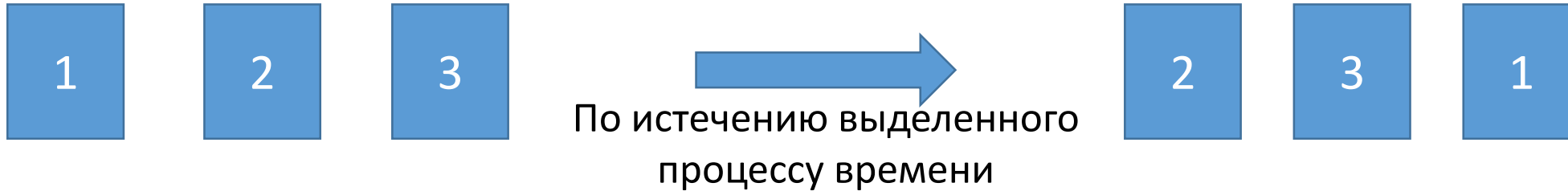
Вытесняющие алгоритмы <-> вытесняющая многозадачность



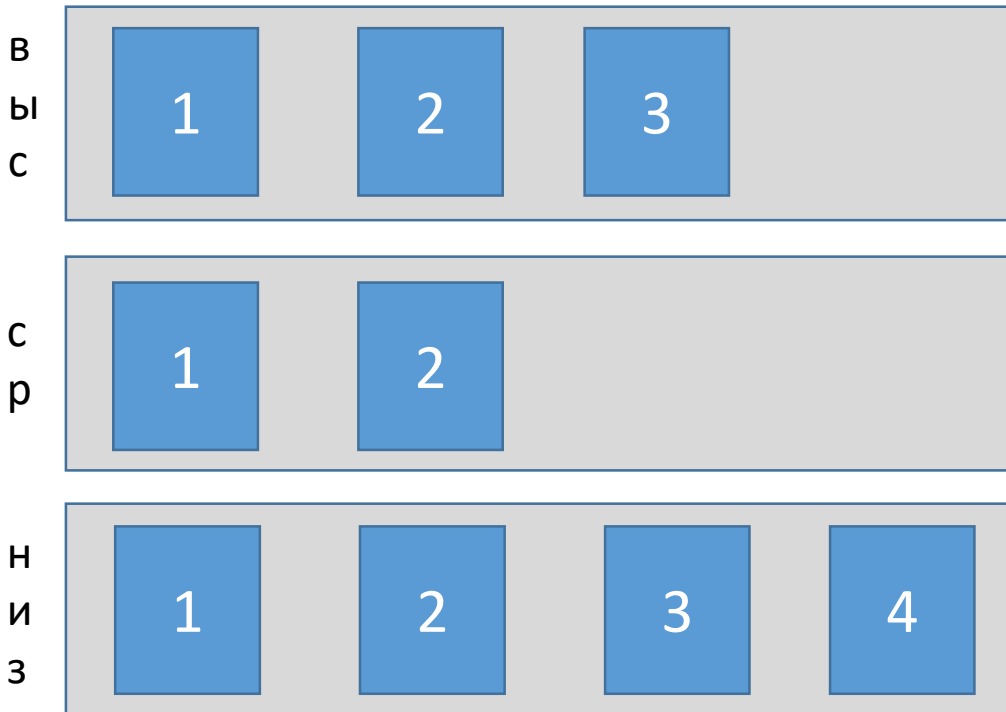
Алгоритмы планирования процессов

(многозадачные операционные системы)

FIFO



С приоритетами



Проблемы параллельных процессов

Голодание (starvation)

Решение проблемы:

- Перераспределение задач между имеющимися компьютерами
- Замена компьютеров на более мощные

«Обход проблемы»:

- Временно повысить приоритет
- Предоставление CPU вне очередности

Взаимные блокировки ресурсов (deadlock)

Процесс 1

Блокировка ресурса **array**
Блокировка ресурса **i**

Процесс 2

Блокировка ресурса **i**
Блокировка ресурса **array**

Состояние «гонки» («состязание», race condition)

Общие ресурсы

Массив array



Переменная i



Процесс 1

`array(i)=f(x);`

`i++;`

T1, i=4 => array(4)=10

T3, i=4+1 => i=5

Процесс 2

`array(i)=f(x);`

`i++;`

T2, i=4 => array(4)=9

T4, i=5+1 => i=6

Сигналы

Сигнал [в операционных системах семейства Unix], один из основных механизмов межпроцессного взаимодействия (inter-process communication, IPC) — асинхронное уведомление процесса о каком-либо событии.

Фрагмент кода на C

```
void sig_handler (int signum)
{
    printf("signal %d received\r\n", signum);
}

void main()
{
    ...
    signal(SIGILL, sig_handler);
    signal(...., .....);
    /*основная логика*/


    .....
}
```

Создание процессов, потоков

Процесс – работает независимо (изолированно) от других процессов

Потоки (threads)/волокна (fibers) – независимые вычислительные алгоритмы в рамках основного процесса

Способы создания новых процессов



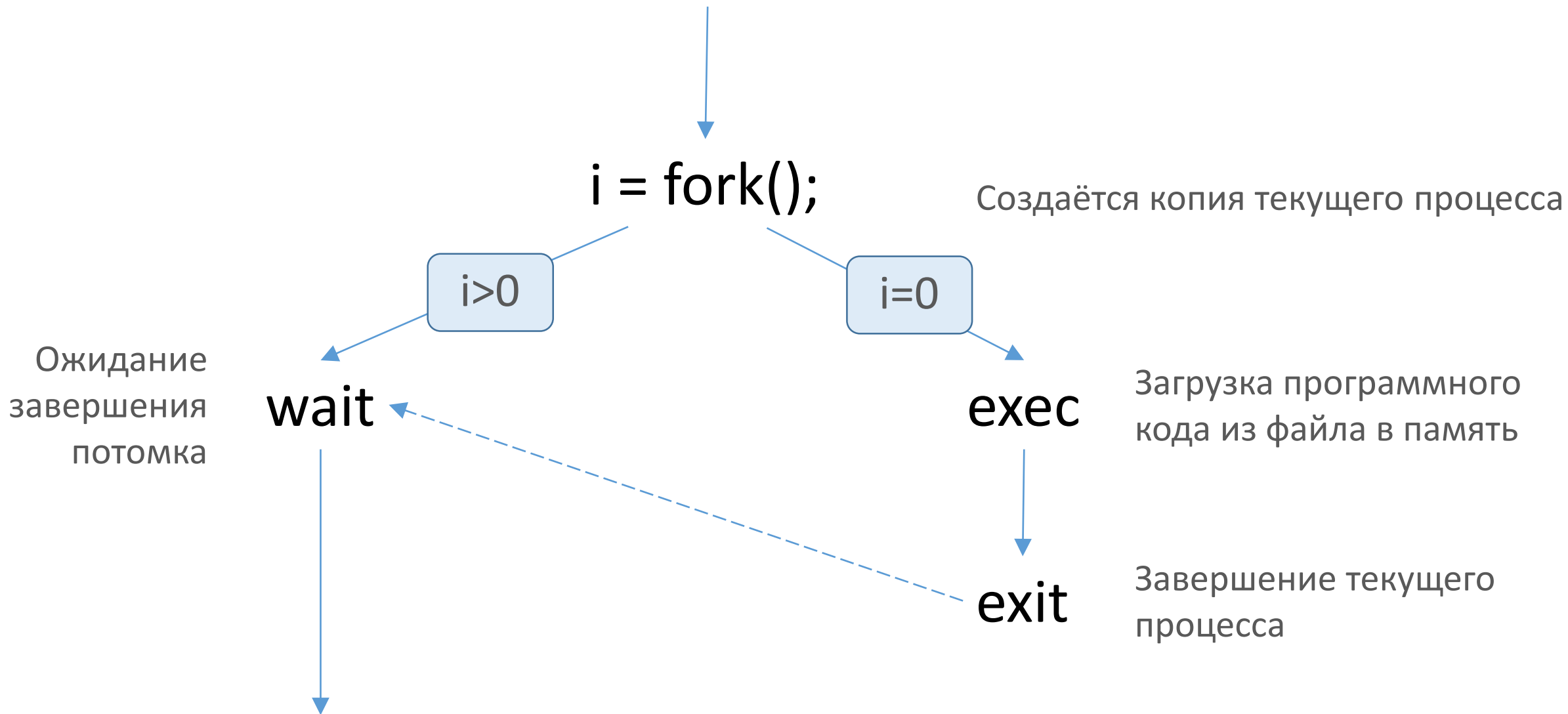
Путем дублирования текущего, с
возможной последующей заменой
образа программного кода из файла

Unix, вызовы:
fork для дублирования процесса
exec для загрузки программного
кода из файла

Создание «с нуля» - загрузкой образа
программного кода из файла

Windows, вызов:
CreateProcess

Создание нового процесса в Unix



Создание нового потока

int main(void)

У процесса 1 поток

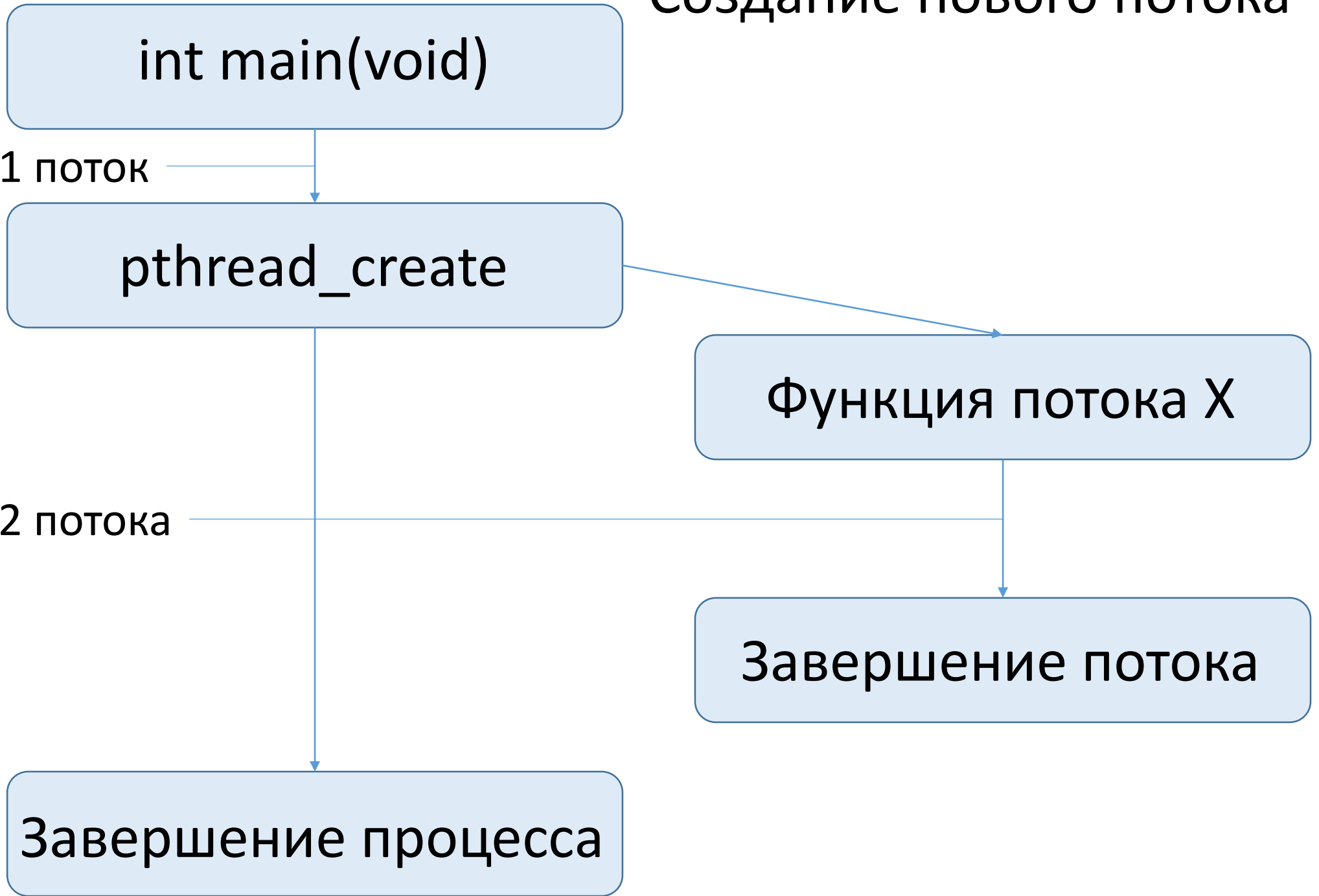
pthread_create

Функция потока X

У процесса 2 потока

Завершение потока

Завершение процесса



Причины завершения процессов

➤ Преднамеренно

- Обычный выход
- Выход по ошибке

➤ Не преднамеренно

- Выход по неисправимой ошибке
- Уничтожение другим процессом

ТИПЫ ФОРМАТОВ ИСПОЛНЯЕМЫХ ФАЙЛОВ

На машинном языке
(двоичные файлы)

Microsoft Windows – MZP
Linux – ELF

....

На интерпретируемом Я.П.
(обычно текстовые)

Обычно на “shell”, также исп. PHP, Perl и т.д.

```
C:\Program Files\Far Manager\Far.exe
00000000: 4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00
00000001: B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00
00000002: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000003: 00 00 00 00 00 00 00 00  00 00 00 00 08 01 00 00
00000004: 0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68
00000005: 69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F
00000006: 74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20
00000007: 6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00
00000008: 8C 28 E6 13 C8 49 88 40  C8 49 88 40 C8 49 88 40
00000009: 7C D5 79 40 C3 49 88 40  7C D5 7B 40 6F 49 88 40
0000000A: 7C D5 7A 40 D2 49 88 40  AB 14 8B 41 C0 49 88 40
0000000B: AB 14 8D 41 8A 49 88 40  AB 14 8C 41 ED 49 88 40
0000000C: 15 B6 59 40 C9 49 88 40  15 B6 46 40 CA 49 88 40
0000000D: 15 B6 43 40 D5 49 88 40  C8 49 89 40 5F 48 88 40
```

Пример скрипта

```
#!/bin/sh
```

```
date
pwd
```

«Состав процесса»

1. Описатель процесса – «дескриптор процесса», «объект-процесс» (object-process)

- Состояние процесса
- Расположение в ОЗУ/на диске
- Идентификаторы (пользователя, процесса)
- Незавершенные операции в/в

и т.д.

2. Контекст процесса

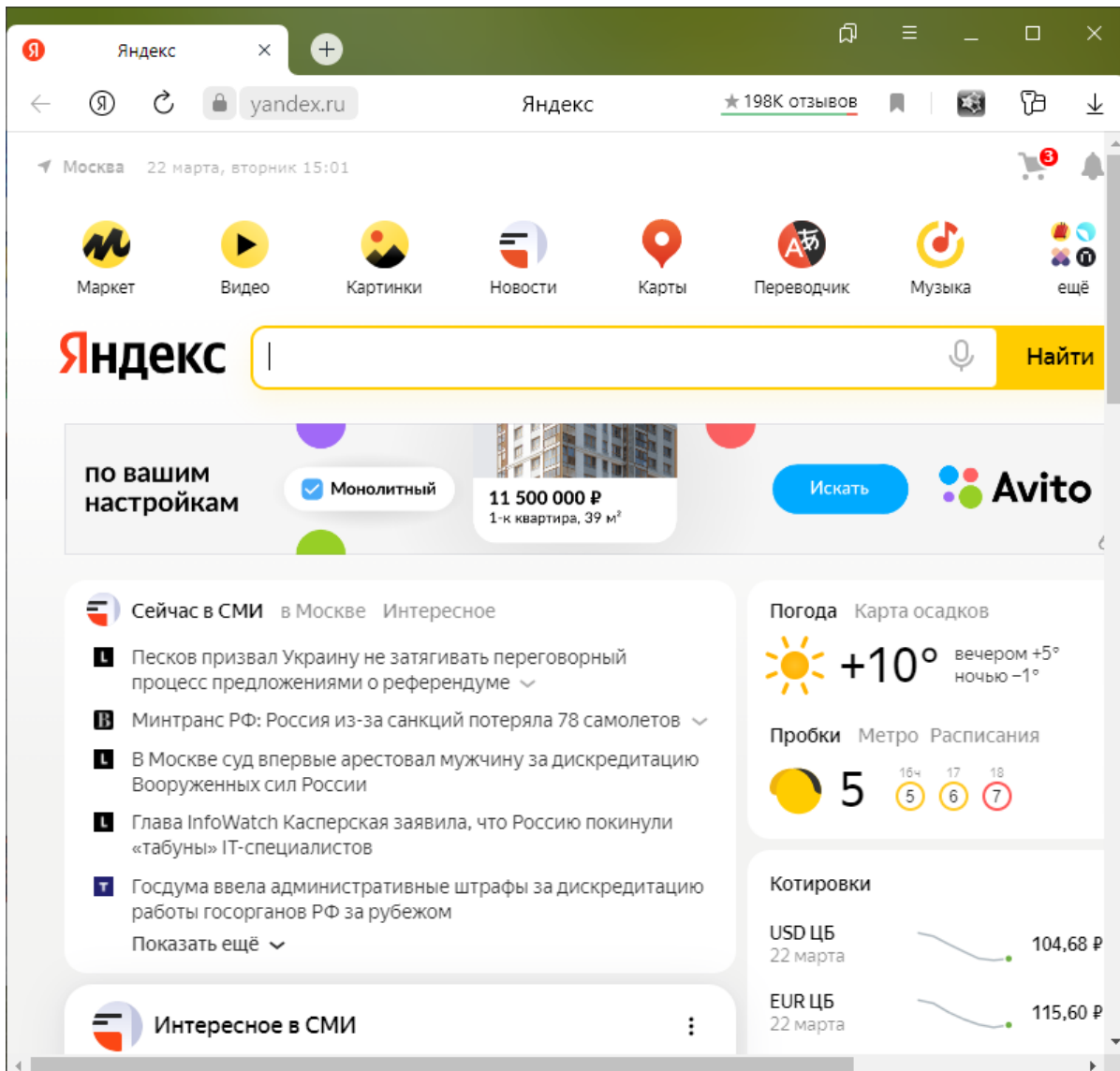
- Содержимое регистров процессора
- Перечень открытых файлов

3. Образ процесса

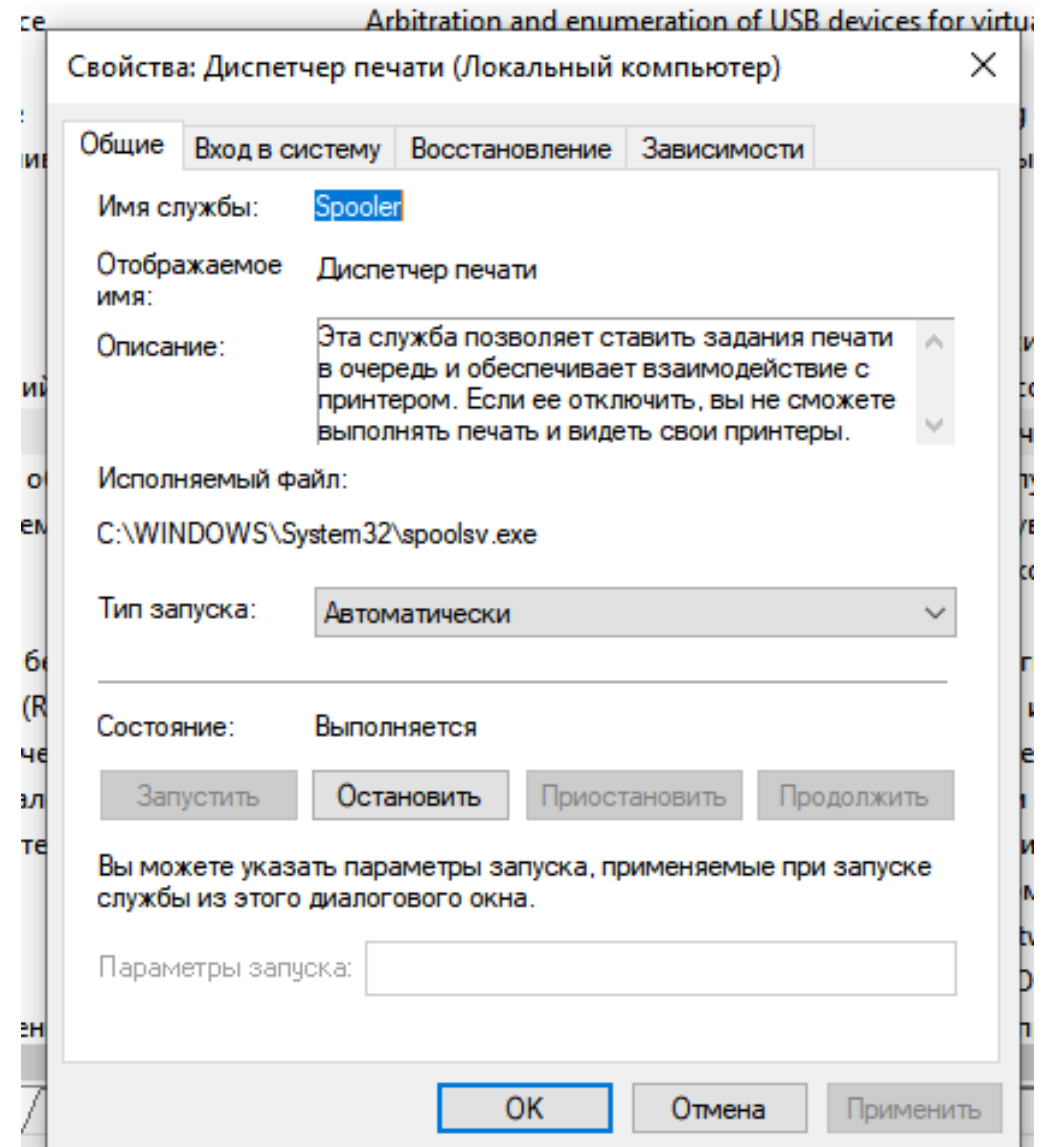
```
vpn /etc/openvpn/site_subnet_tcp.conf
root      844  0.0  0.0  11884  1848 ?          SNs   Aug03   0:04 xinetd
root      852  0.0  0.0   6356  1448 ?          SNs   Aug03   1:06 cron
root      862  0.0  0.0  14132  2284 ?          S     Aug03   0:00 /bin/sh /usr/lo
cal/mysql/bin/mysqld_safe --datadir=/var/mysql/data --pid-file=/var/mysql/data/i
ce.pushkiny.ru.pid --relay-log-recovery=1
mysql     1306  5.0  4.2 2902996 457068 ?        S1    Aug03 4878:05 /usr/local/mys
ql/bin/mysqld --basedir=/usr/local/mysql --datadir=/var/mysql/data --plugin-dir=
/usr/local/mysql/lib64/plugin --user=mysql --relay-log-recovery=1 --log-error=/v
ar/log/mysql/error.log --pid-file=/var/mysql/data/ice.pushkiny.ru.pid
ldap      1380  0.0  0.0  939376  3828 ?        SNs1  Aug03   4:03 /usr/libexec/sl
apd -u ldap -4 -h ldap://ldap.ice.pushkiny.ru
root      1396  0.0  0.0  626316  9952 ?        SNs1  Aug03   0:36 /usr/local/free
radius/sbin/radiusd
root      1414  0.0  1.7 1251272 189048 ?       SNs1  Aug03   1:09 /usr/sbin/clamd
root      1440  0.0  0.0  2128024 3720 ?        SNs1  Aug03   3:24 /usr/sbin/clama
v-milter
```

Фоновые и интерактивные процессы

Интерактивные – подключены к терминалам, взаимодействуем с ними



Фоновые – работают независимо от пользователя. Т.н. services , daemon



Идентификаторы

1. Идентификатор процесса Process ID PID
2. Идентификатор процесса-предка Parent Process ID PPID
3. Идентификатор группы процессов
4. Идентификатор пользователя User ID:
 1. Unix:
 1. 0 – системный администратор, root (рут), wheel
 2. >0 – обычные пользователи (технические учетные записи 1...100/500/1000, обычные пользователи >100/500/1000)
 2. Windows
5. Идентификатор группы пользователя Group ID GID
6. Реальный и эффективный (действующий) идентификатор пользователя/группы пользователя

Приоритеты процессов

Системы Unix



Пример команд управления приоритетами

`nice -n 15 prog`

`renice -n 10 15121`

Windows имеет 32 уровня приоритета (0-31)

- 1 уровень (00 — 00) — это Zero Page Thread;
- 15 уровней (01 — 15) — обычные динамические приоритеты;
- 16 уровней (16 — 31) — реального времени.

Системные вызовы

int fork()

Exec (execl, execv)

wait

exit

int getpid()

int nice(int inc)

Команды управления процессами

kill/killall (kill -l)

ps/top

jobs/fg/bg/ [CTRL-Z] &

trap "command" 1 2 15 trap 1 2

nice renice

Пример программы на С, создающей новый процесс

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
main()
{
    int pid,status;
    //порождение процесса
    switch((pid=fork()))
    {
        case -1:    //авар. Завершение
            printf ("can't fork\n");
            exit(1);
        case 0:        //процесс потомок
            printf("I am the child\n");
            //запускаем команду echo hello user
            if (execl("/bin/echo","echo","hello user",0)!=-1)
            {
                //если произошла ошибка
                printf("exec error\n");
                exit(1);
            }
            break;
        default:    //процесс предок
            //ожидание завершения процесса потомка
            wait(&status);
    }
}
```