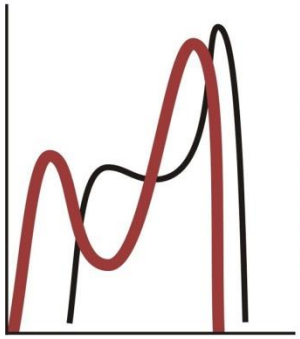


ИНСТИТУТ
МАТЕМАТИКИ
МЕХАНИКИ
КОМПЬЮТЕРНЫХ
НАУК

имени И.И. Воровича —

Архитектура компьютера и операционные системы

Лекция 14. Управление процессами и потоками

доцент кафедры информатики и вычислительного эксперимента

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

План лекции

- Интерфейс прикладного программирования
- Системные вызовы
- Прерывания
- Понятие процесса
- Состояния процесса

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Интерфейс прикладного программирования

- **A**pplication **P**rogramming **I**nterface (API) - набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.
- Стандарты API:
  - Windows API (Win32 API);
  - POSIX

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Windows API

- общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации «Майкрософт».
- Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой Windows.

POSIX

- **P**ortable **O**perating **S**ystem **I**nterface for uni**X** (переносимый интерфейс операционных систем) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой, библиотеку языка C и набор приложений и их интерфейсов, созданных для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем.

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Поддержка операционными системами

- Сертифицированные
  - Mac OS X
  - Solaris
  - HP-UX
  - QNX
  - …
- Совместимые
  - FreeBSD
  - GNU/Linux
  - VxWorks
  - MINIX
  - …
- Для Windows
  - Cygwin
  - Подсистема для UNIX-приложений (SUA)
  - Windows Subsystem for Linux
  - …

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

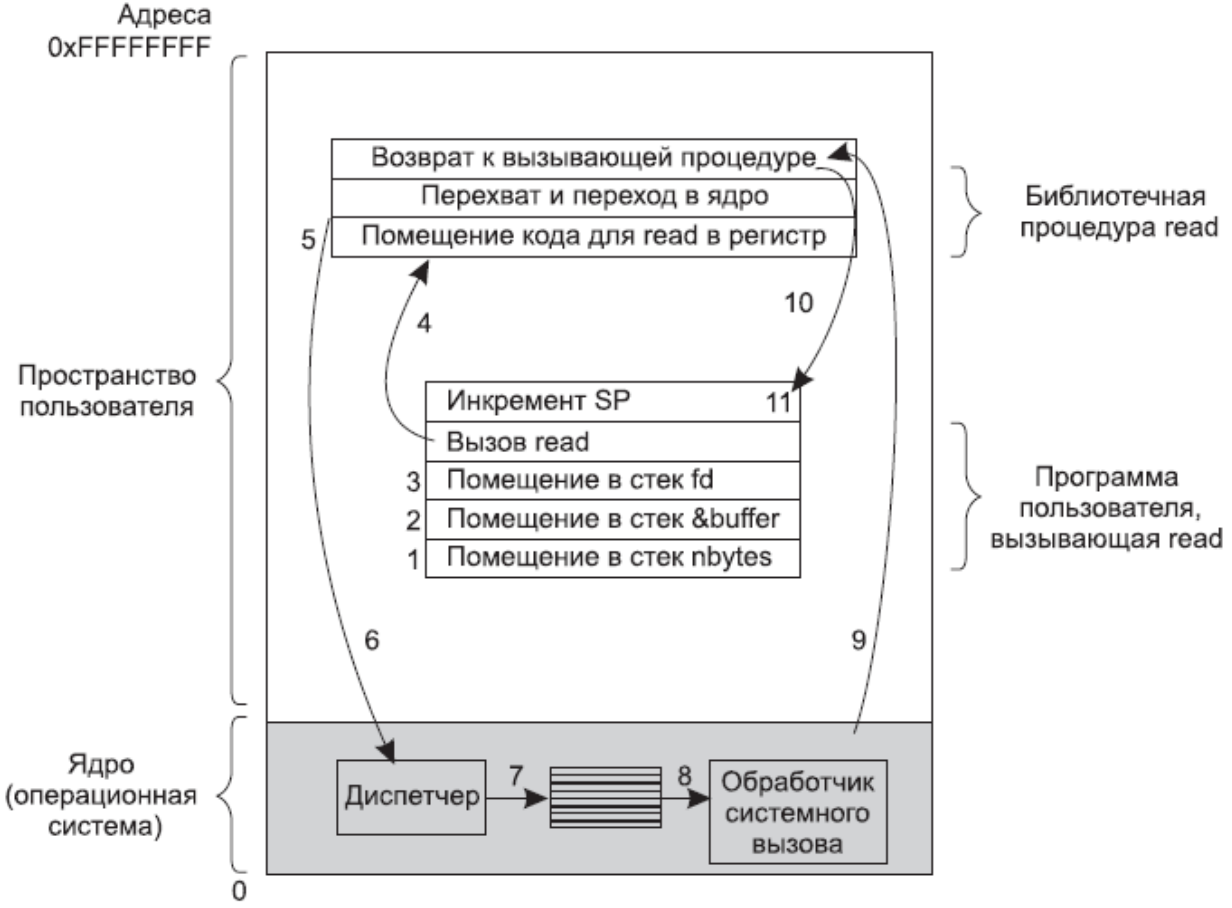
Системные вызовы

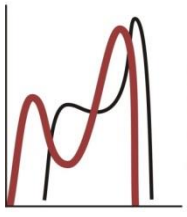
- — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.
- Когда процесс выполняет пользовательскую программу в режиме пользователя и нуждается в какой-нибудь услуге операционной системы, он должен выполнить команду системного вызова, чтобы передать управление операционной системе.
- Операционная система по параметрам вызова определяет, что именно требуется вызывающему процессу, обрабатывает системный вызов и возвращает управление той команде, которая следует за системным вызовом.

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Пример системного вызова read стандарта POSIX

- count = read(fd, buffer, nbytes);





ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

POSIX vs Windows API

UNIX	Win32	Описание
fork	CreateProcess	Создает новый процесс
waitpid	WaitForSingleObject	Ожидает завершения процесса
execve	Нет	CreateProcess=fork+execve
exit	ExitProcess	Завершает выполнение процесса
open	CreateFile	Создает файл или открывает существующий файл
close	CloseHandle	Закрывает файл
read	ReadFile	Читает данные из файла
write	WriteFile	Записывает данные в файл
lseek	SetFilePointer	Перемещает указатель файла
stat	GetFileAttributesEx	Получает различные атрибуты файла
mkdir	CreateDirectory	Создает новый каталог
rmdir	RemoveDirectory	Удаляет пустой каталог
link	Нет	Win32 не поддерживает связи
unlink	DeleteFile	Удаляет существующий файл
mount	Нет	Win32 не поддерживает подключение к файловой системе
umount	Нет	Win32 не поддерживает подключение к файловой системе
chdir	SetCurrentDirectory	Изменяет рабочий каталог
chmod	Нет	Win32 не поддерживает защиту файла (хотя NT поддерживает)
kill	Нет	Win32 не поддерживает сигналы
time	GetLocalTime	Получает текущее время

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

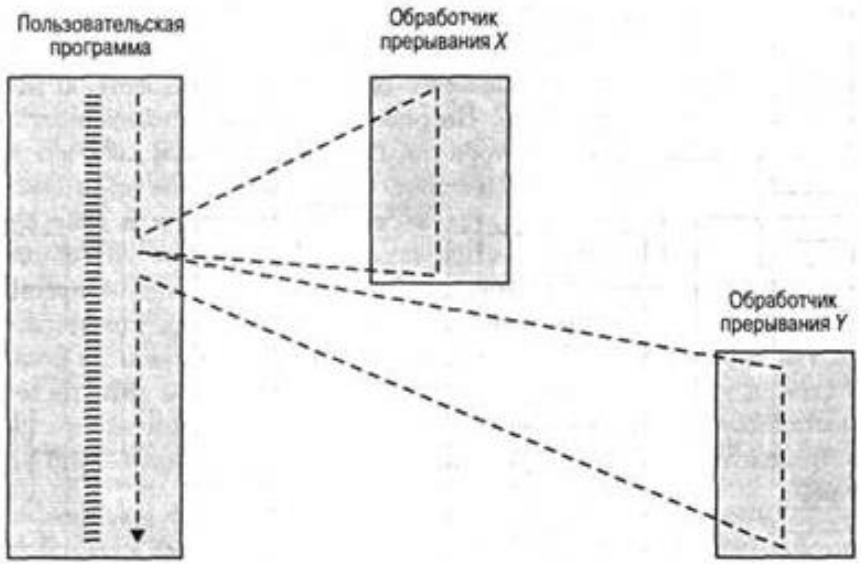
Прерывания

- – сигнал от программного или аппаратного обеспечения, который заставляет процессор прервать текущую задачу, сохранить свое состояние и вызвать специальный обработчик
  - внешнее устройство требует внимания
  - произошла ошибка при выполнении инструкции
  - специальная инструкция
- Обработчики прерываний – часть ядра ОС. ОС сообщает процессору в какой ситуации какой обработчик вызывать.

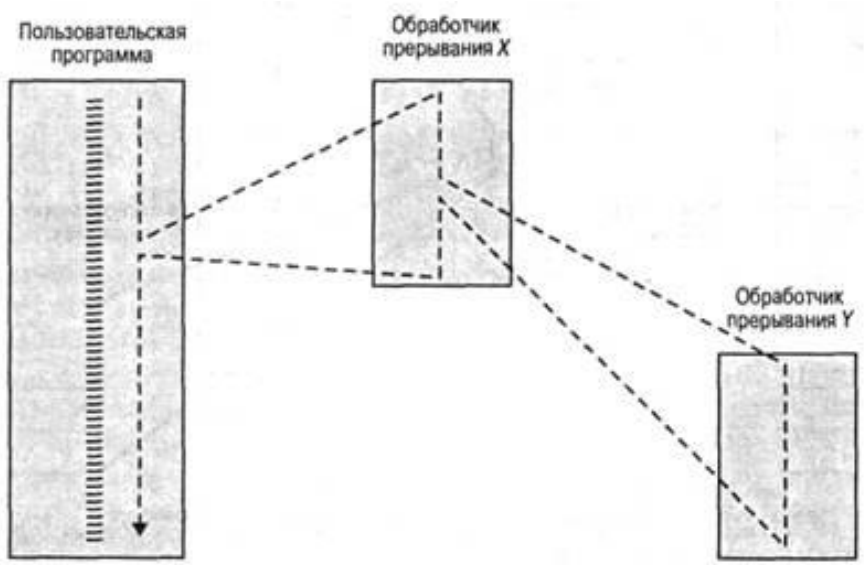
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК
имени И.И. Воровича

Множественные прерывания

- Последовательная обработка прерываний



- Вложенная обработка прерываний



Процесс

- Процесс как программа, запущенная на выполнение.
  - для исполнения одной программы может организовываться несколько процессов
  - в рамках одного процесса может исполняться несколько программ
  - в рамках процесса может исполняться код, отсутствующий в программе
- Процесс как единица работы (задача).
- Процесс как единица управления ресурсами.

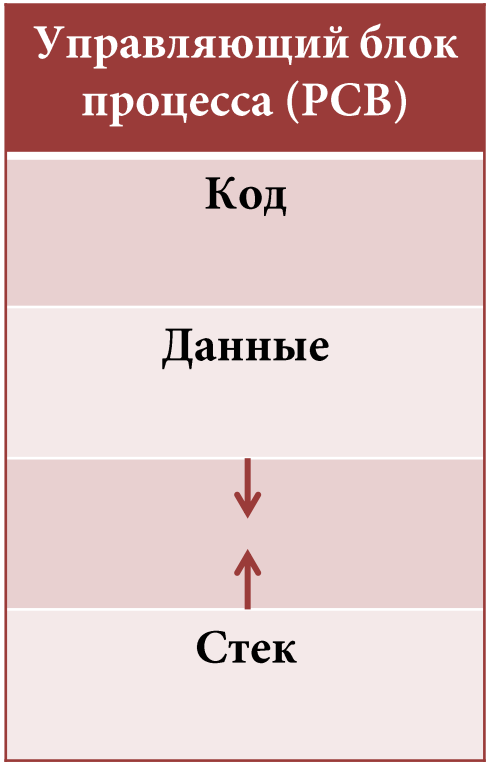
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Описание процесса

- Термин «процесс» характеризует совокупность
  - набора исполняющихся команд
  - ассоциированных с ним ресурсов
  - текущего момента его выполнения

  находящуюся под управлением ОС

- Адресное пространство – множество ячеек оперативной памяти, доступных заданному процессу.



ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Контекст процесса и Process Control Block (PCB)

- Идентификатор процесса
- Состояние процесса
- Данные для планирования использования процессора и управления памятью
- Сведения об устройствах ввода-вывода, связанные с процессом ...

Системный контекст

- Программный счетчик
- Содержимое регистров

Регистровый контекст

Process Control Block (PCB)

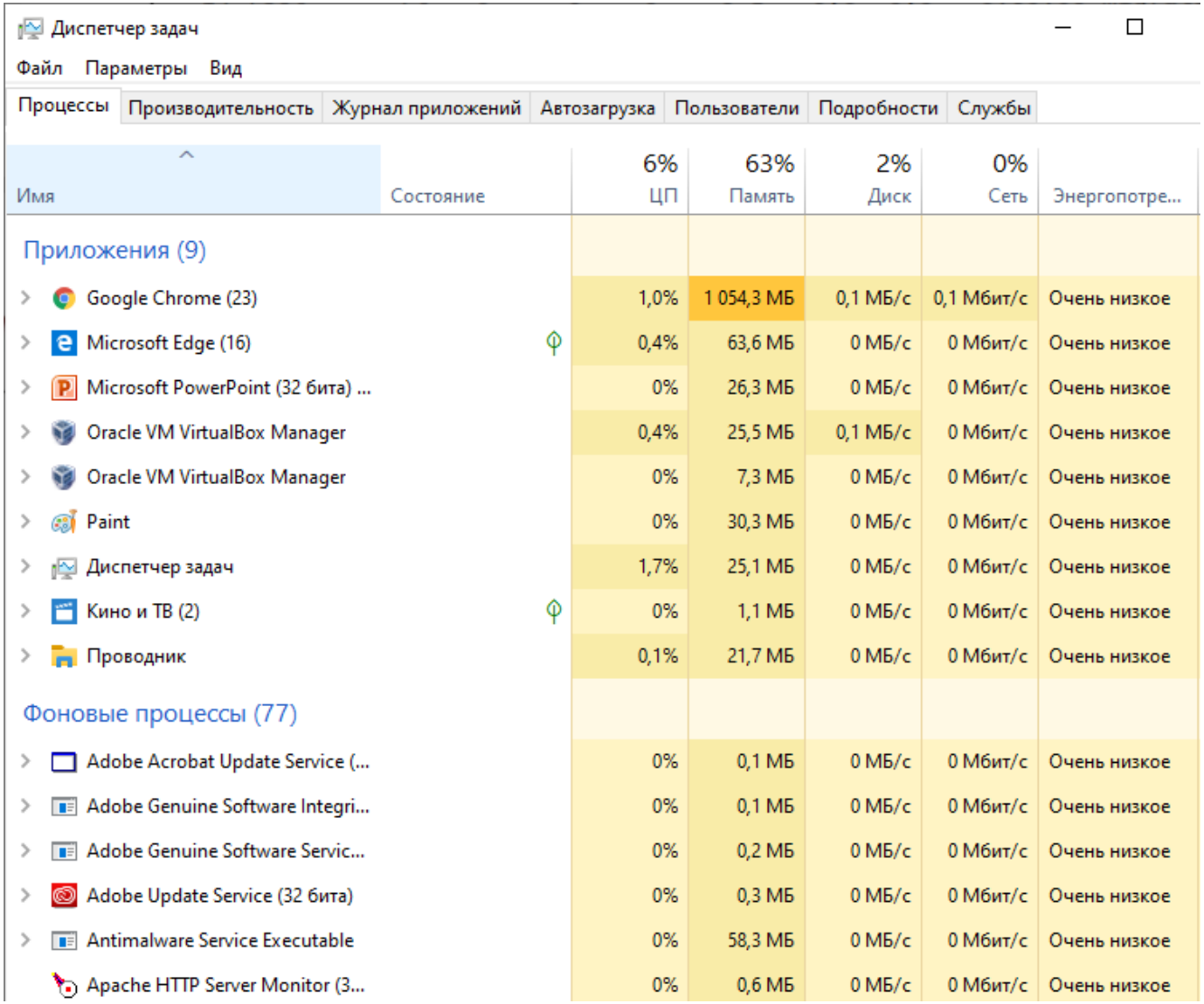
- Код программы
- Данные программы

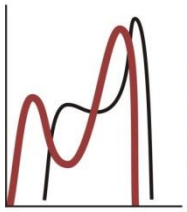
Пользовательский контекст

Контекст процесса

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Процессы в Windows





ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Процессы Unix (ps ax)

```
  50 ?        S      0:00 [kworker/u:2]
  52 ?        S<     0:00 [binder]
  72 ?        S<     0:00 [deferwq]
  73 ?        S<     0:00 [charger_manager]
  74 ?        R      0:00 [kworker/1:1]
 120 ?        S<     0:00 [kworker/0:1H]
 253 ?        S      0:00 [scsi_eh_2]
 267 ?        S      0:00 [jbd2/sda1-8]
 268 ?        S<     0:00 [ext4-dio-unwrit]
 312 ?        S      0:00 upstart-file-bridge --daemon
 368 ?        S      0:00 upstart-udev-bridge --daemon
 370 ?        Ss     0:00 /sbin/udevd --daemon
 530 ?        S<     0:00 [iprt]
 537 ?        S      0:00 [kworker/1:2]
 539 ?        S<     0:00 [kworker/1:1H]
 552 ?        S<     0:00 [kpsmoused]
 575 ?        Sl     0:00 rsyslogd -c5
 621 ?        S      0:00 upstart-socket-bridge --daemon
 639 ?        Ss     0:01 dbus-daemon --system --fork
 710 ?        Ss     0:00 /usr/sbin/bluetoothd
 734 ?        S<     0:00 [krfcommd]
 737 ?        Ss     0:00 /usr/sbin/cupsd -F
```

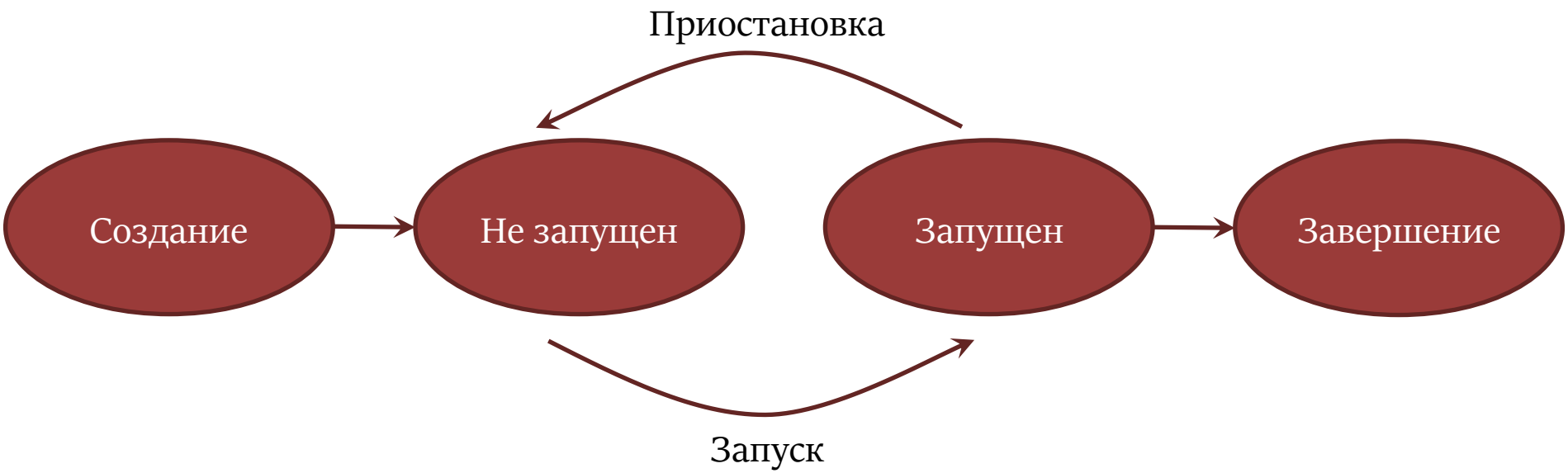
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Процессы в UNIX (top)

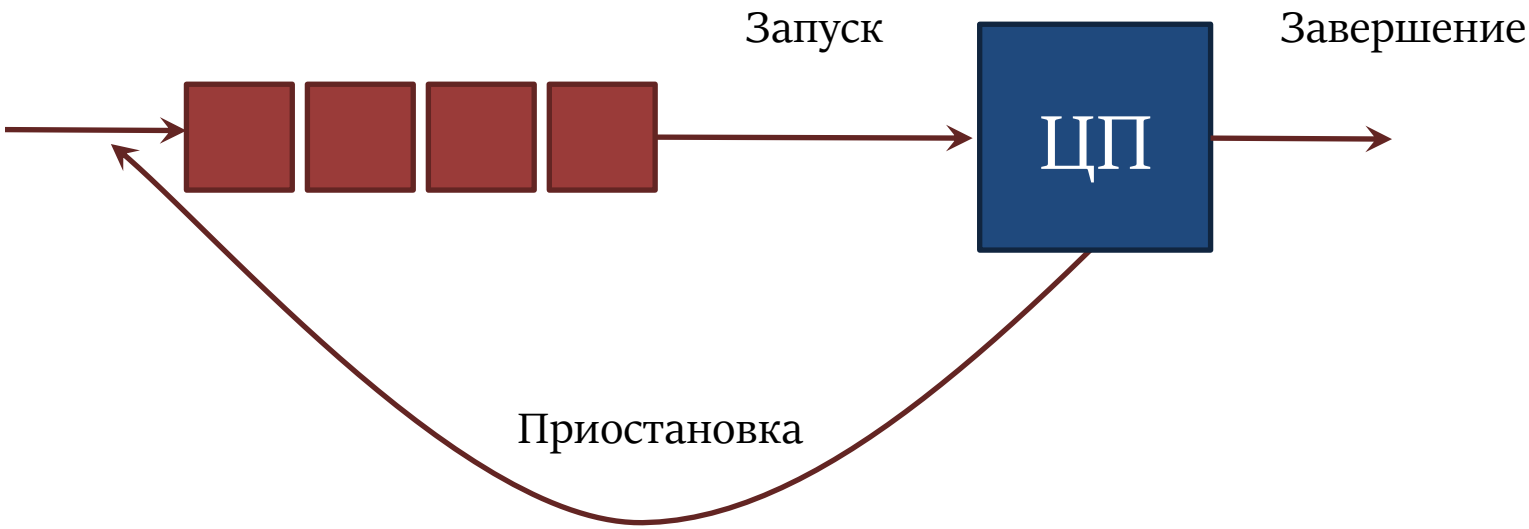
```
top - 19:42:07 up  1:27,  2 users,  load average: 0,01, 0,09, 0,12
Tasks: 125 total,   1 running, 124 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,0 us,  2,5 sy,  0,0 ni, 96,3 id,  0,0 wa,  0,0 hi,  0,2 si,  0,0 st
КиБ Mem:    767056 total,   623836 used,   143220 free,    19236 buffers
КиБ Swap:   784380 total,      672 used,   783708 free,   159480 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2123 mmcs      20   0  995m 332m  43m S   5,3 44,3   6:39.62 firefox
 1048 root      20   0  137m  49m  10m S   4,3  6,6   1:53.34 Xorg
 1376 mmcs      20   0 9896 1412  960 S   0,7  0,2   0:21.22 VBoxClient
  639 messageb  20   0 3728 1604  976 S   0,3  0,2   0:02.58 dbus-daemon
 2308 mmcs      20   0  204m  14m  10m S   0,3  1,9   0:02.75 lxterminal
    1 root      20   0 3900 2040 1308 S   0,0  0,3   0:00.89 init
    2 root      20   0    0    0    0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0    0    0    0 S   0,0  0,0   0:01.62 ksoftirqd/0
    5 root       0 -20    0    0    0 S   0,0  0,0   0:00.00 kworker/0:0H
    7 root       0 -20    0    0    0 S   0,0  0,0   0:00.00 kworker/u:0H
    8 root      rt   0    0    0    0 S   0,0  0,0   0:00.32 migration/0
    9 root      20   0    0    0    0 S   0,0  0,0   0:00.00 rcu_bh
   10 root      20   0    0    0    0 S   0,0  0,0   0:02.28 rcu_sched
   11 root      rt   0    0    0    0 S   0,0  0,0   0:00.20 watchdog/0
   12 root      rt   0    0    0    0 S   0,0  0,0   0:00.18 watchdog/1
```

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Простая модель процесса

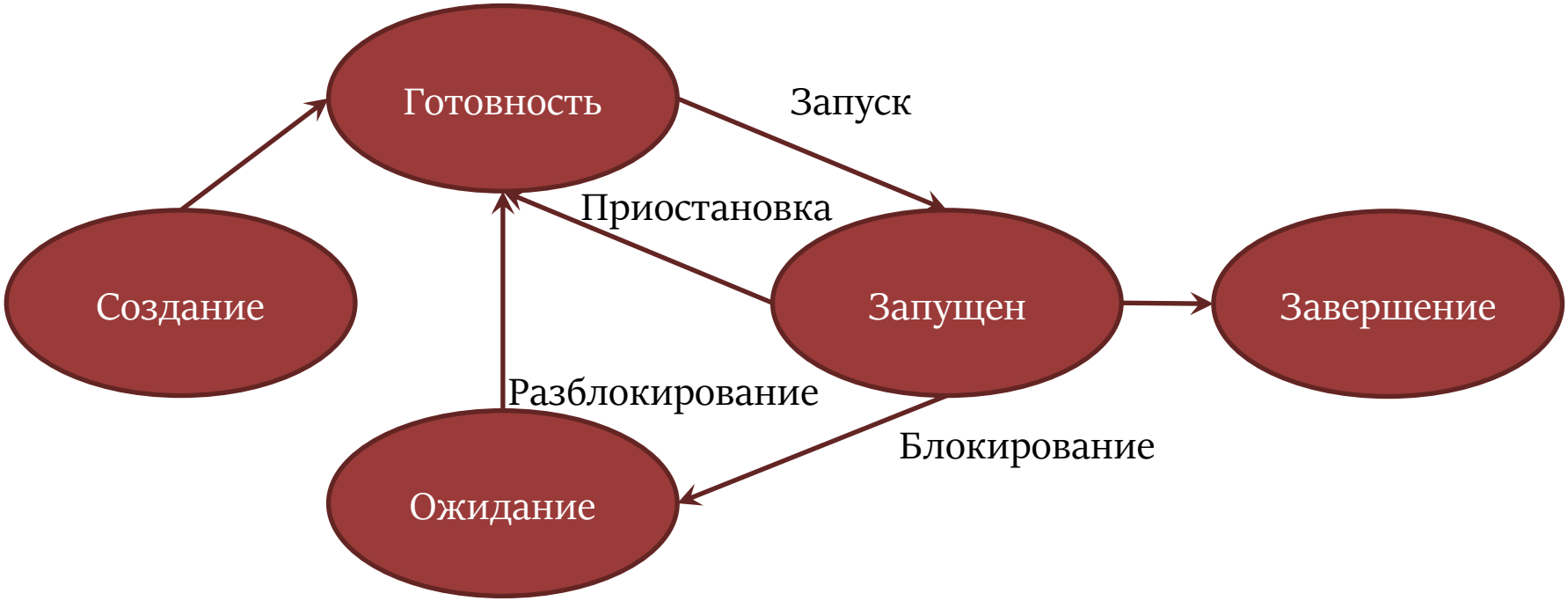


Очередь



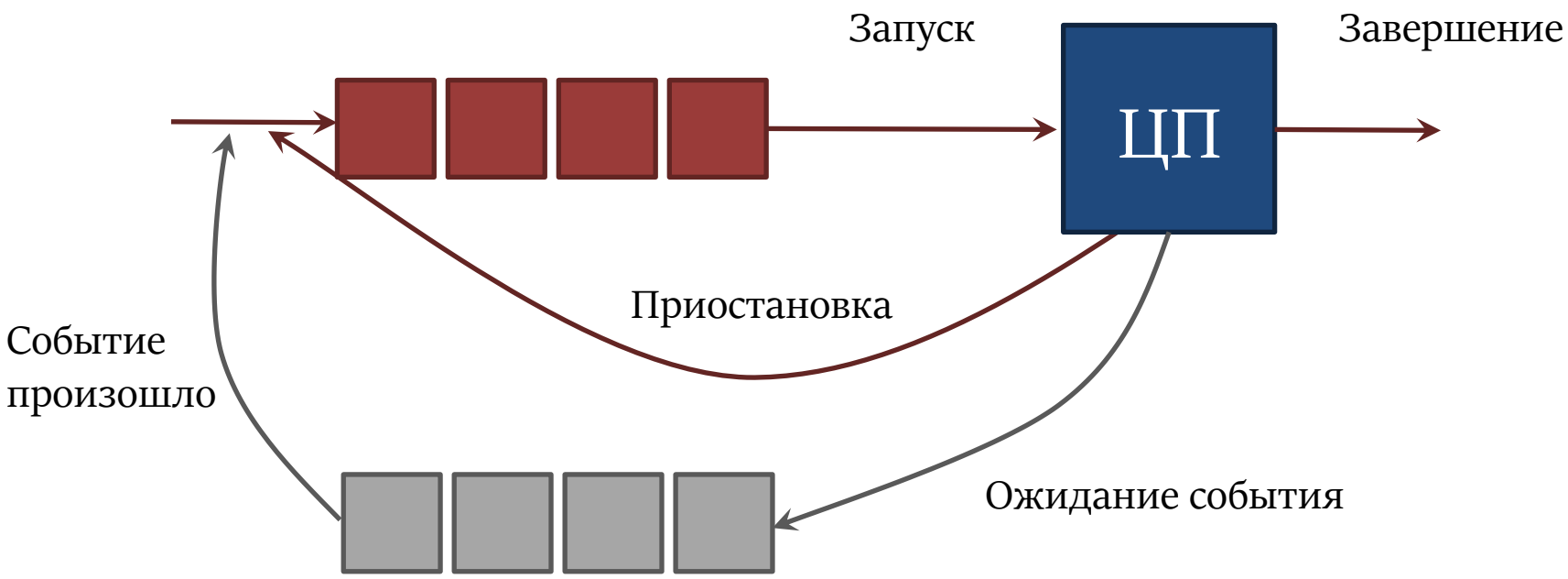
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Другие состояния



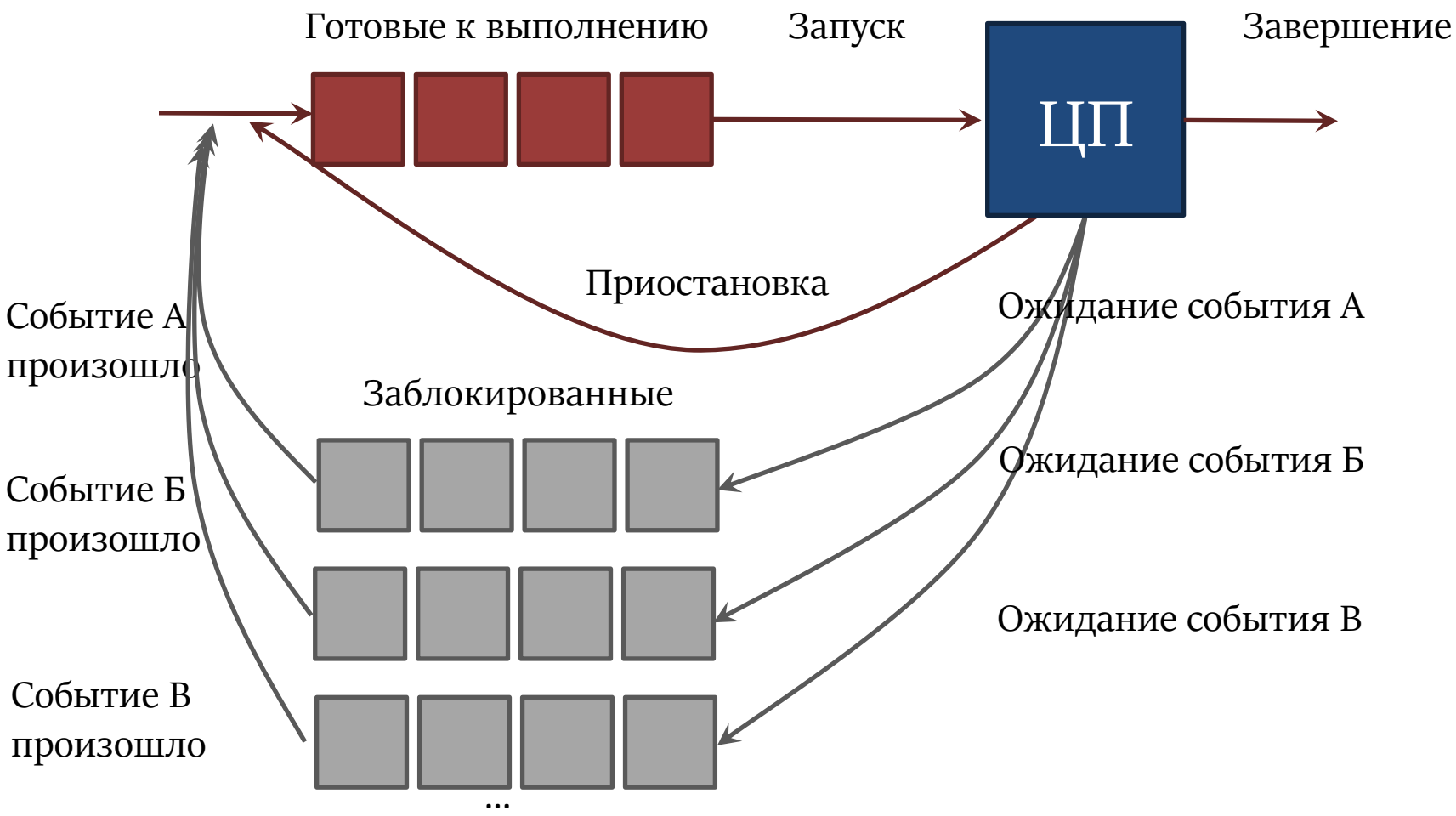
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Двойная очередь



ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Множественная очередь



Набор операций

- Одноразовые
  - создание процесса
  - завершение процесса
- Многоразовые
  - запуск процесса
  - приостановка процесса
  - блокирование процесса
  - разблокирование процесса
  - изменение приоритета

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Создание процесса

- Инициализация системы.
- Выполнение работающим процессом системного вызова, предназначенного для создания процесса.
- Запрос пользователя на создание нового процесса.
- Инициация пакетного задания.

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Создание процесса

- Порождение нового PCB с состоянием процесса «Создание»
- Присвоение идентификационного номера и создание записи в таблице процессов
- Выделение ресурсов
- Занесение в адресное пространство кода и установка значения программного счетчика
- Окончание заполнения PCB
- Изменение состояния процесса на «Готовность»

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Завершение процесса

- Процесс может завершиться:
  - Добровольно
  - Принудительно (возникла ошибка)
    - нехватка памяти;
    - ошибка защиты памяти;
    - арифметические ошибки, ошибки ввода/вывода, ошибочные команды
  - Принудительно другим процессом

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК
имени И.И. Воровича

Завершение процесса

- Изменение состояния процесса на «Завершение»
- Освобождение ресурсов
- Очистка соответствующих элементов в PCB
- Сохранение в PCB информации о причинах завершения

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Запуск процесса

- Выбор одного из процессов, находящихся в состоянии «Готовность»
- Изменение состояния выбранного процесса на «Исполняется»
- Обеспечение наличия в оперативной памяти информации, необходимой для его выполнения
- Восстановление значений регистров
- Передача управления по адресу, на который указывает программный счетчик

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Приостановка процесса

- Автоматическое сохранение программного счетчика и части регистров
- Сохранение динамической части регистрового и системного контекстов в PCB
- Изменение состояния процесса на «Готовность»
- Обработка прерывания

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Блокирование процесса

- Обработка системного вызова
- Сохранение контекста процесса в PCB
- Перевод процесса в состояние «Ожидание»

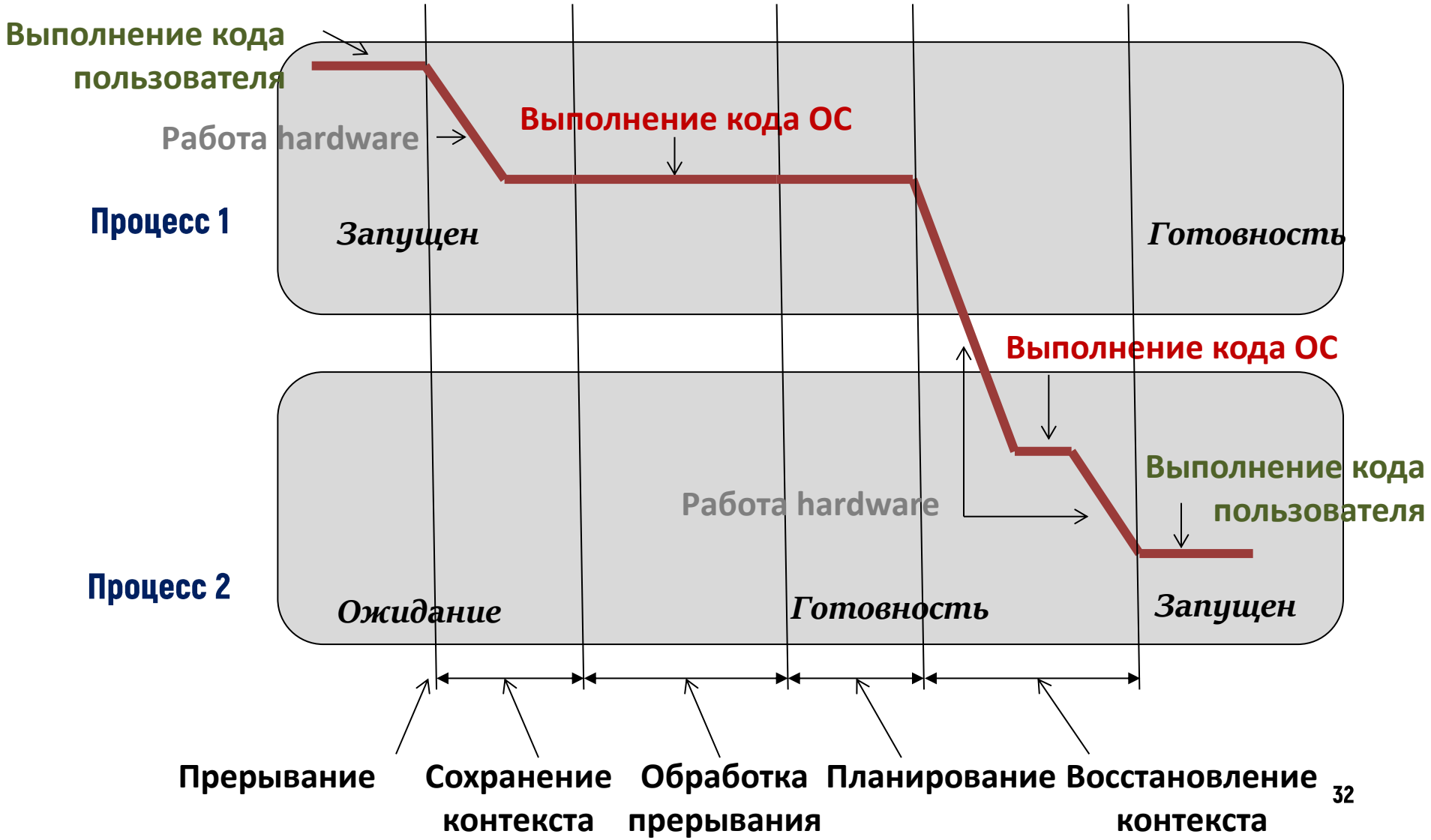
ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Разблокирование процесса

- Уточнение того, какое именно событие произошло
- Проверка наличия процесса, ожидающего этого события
- Перевод ожидающего процесса в состояние «Готовность»
- Обработка произошедшего события

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Пример цепочки операций



ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Создание процессов в Posix

- Функции создания процесса

```
#include <unistd.h>
pid_t fork(void);
```

- Функции замещения тела процесса

```
int execve(const char *file, char *const argv[], char *const envp[]);
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *path, const char *arg, ..., char *const
envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

Ожидание завершения процесса

```
#include <sys/wait.h>
pid_t wait(int *pnStatus);
pid_t waitpid(pid_t pid, int *pnStatus, int nOptions);
```

Проверка	Дополнительная информация
WIFEXITED(nStatus)	WEXITSTATUS(nStatus)
WIFSIGNALED(nStatus)	WTERMSIG(nStatus),
WIFSTOPPED(nStatus)	WSTOPSIG(nStatus)

Завершение работы процесса

- #include <stdlib.h>
- Вызовом библиотечной функции

  void **exit**(int status);
- Вызовом системного вызова

  void **_exit**(int status);
- Получением необрабатываемого, неигнорируемого и неблокируемого сигнала, который вызывает по умолчанию нормальное или аварийное завершение процесса.
- Фатальное завершение работы процесса

  void **abort**(void);
- Принудительное завершение процесса (<signal.h>)

  int **kill**(pid_t pid, int sig);

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Пример

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
int main(int argc, char * argv[]) {
int pid, status;
if (argc < 2) {
        printf("Usage: %s command, [arg1 [arg2]...]\n",
argv[0]);
        return EXIT_FAILURE; }
```

Пример (продолжение)

```
printf("Starting %s...\n", argv[1]);
pid = fork();
if (!pid) {
        execvp(argv[1], &argv[1]); perror("execvp");
        return EXIT_FAILURE;
}
else { if (wait(&status) == -1) {
        perror("wait");
        return EXIT_FAILURE; }
```

Пример (окончание)

```
    if (WIFEXITED(status))
        printf("Child terminated normally with exit code
        %i\n",  WEXITSTATUS(status));
    if (WIFSIGNALED(status))
        printf("Child was terminated by a signal #%i\n",
        WTERMSIG(status));
    if (WIFSTOPPED(status))
        printf("Child was stopped by a signal #%i\n",
        WSTOPSIG(status));
    }
return EXIT_SUCCESS; }
```

ИНСТИТУТ МАТЕМАТИКИ МЕХАНИКИ КОМПЬЮТЕРНЫХ НАУК имени И.И. Воровича

Домашнее задание

- Подготовиться к тестированию по материалам книги Таненбаума Э., Боса Х. Современные операционные системы, стр. 111-123.
- Подготовка к лабораторной №10