Операционные системы

Процессы и потоки

Процесс

1. Абстракция, описывающая выполняемую программу
   Обеспечение параллелизма и псевдопараллелизма
2. Экземпляр выполняемой программы, включая значение счетчика команд, регистров и переменных

Память процесса

События, инициализирующие процессы

1. Инициализация системы
2. Работающий процесс осуществляет системный вызов создания нового процесса
   Иерархия процессов
3. Создание процесса по запросу пользователя
4. Инициализация пакетного задания

Завершение процесса

1. Обычный выход
2. Выход по обрабатываемой ошибке
3. Возникновение фатальной ошибки
4. Уничтожение другим процессом

Ключевые «внешние» отличия Unix/Windows

Unix

1. Процесс создается fork, а далее exec* заменяет образ памяти
2. Есть иерархия процессов (Наличие «зомби», «процессов-сирот»)

Windows

1. Процесс создается одним вызовом CreateProcess
2. Каждый процесс независим

Состояние процессов

Реализация процессов

1. Таблица процессов
   a) Управление процессом (счетчик команд, регистры …)
   b) Управление памятью (указатели на сегмент данных, стека…)
   c) Управление файлами (дескрипторы файлов, корневой каталог …)
2. Вектор прерываний

Механизм прерывания

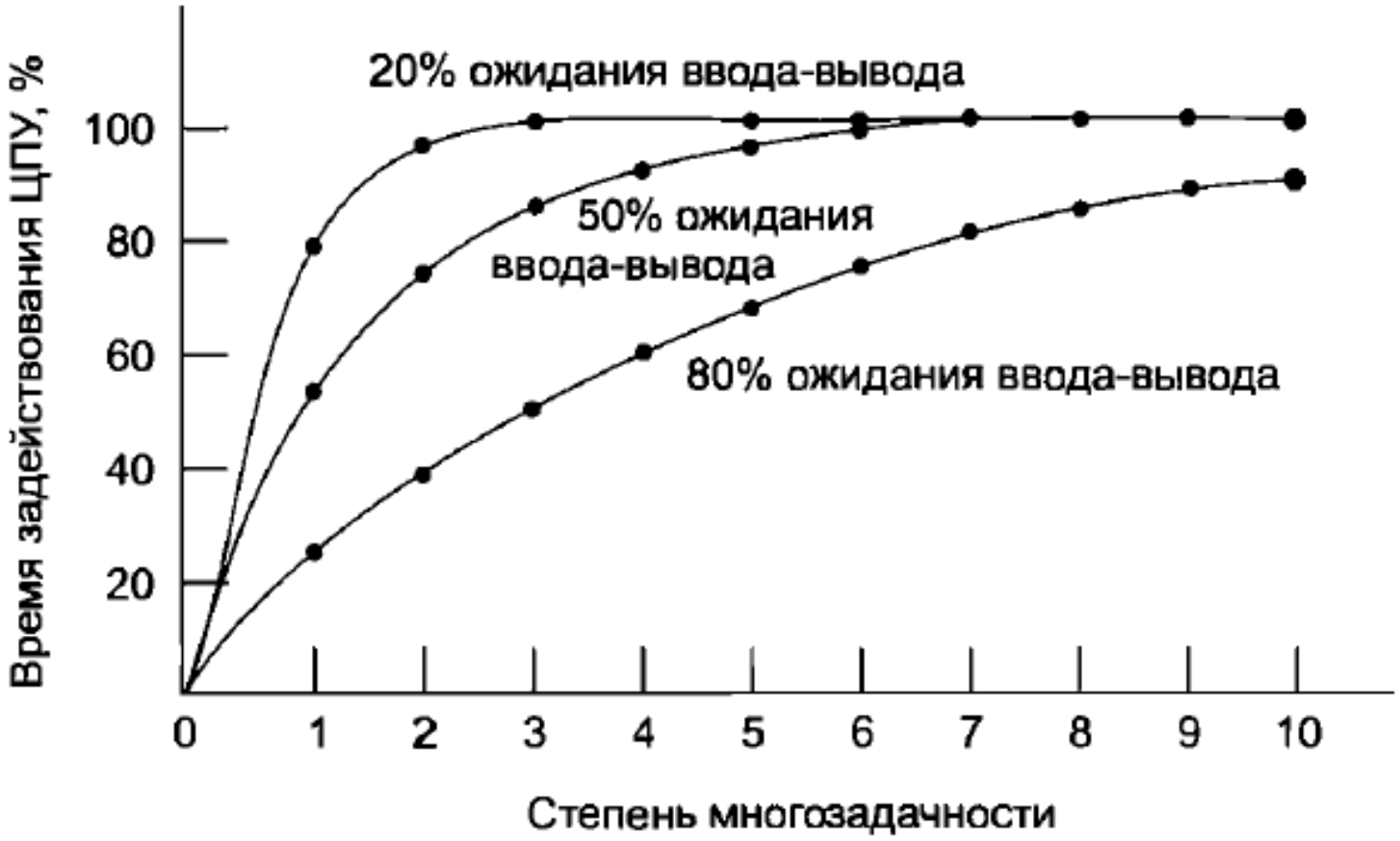
1. Оборудование помещает в стек счетчик команд и т.д.
2. Оборудование загружает из вектора прерываний новый счетчик команд
3. Процедура на ассемблере сохраняет регистры
4. Процедура на ассемблере устанавливает указатель на новый стек
5. Запускается процедура на Си, обслуживающая прерывание
6. Планировщик принимает решение какой процесс запускать следующим
7. Процедура на языке Си возвращает управление ассемблерному коду
8. Процедура на ассемблере запускает новый процесс

Моделирование режима многозадачности

$$ЦП = 1 - p^n$$

p – ожидание ввода-вывода

n – количество процессов



Потоки (Потоки выполнения)

1. Разновидность процесса внутри процесса
2. При старте у каждого процесса есть единственный поток управления

Необходимость в потоках

1. Упрощение модели программирования, когда в одной программе может выполняться несколько действий сразу
2. Создание потоков быстрее, чем создание процессов
3. Эффективность их использования в мультипроцессорных системах

Реализация сервера через *****

1. Потоки
2. Однопоточный процесс
3. Машина с конечным числом состояний

Объекты потоков и процессов

ЭЛЕМЕНТЫ, ПРИСУЩИЕ ПРОЦЕССАМ

1. Адресное пространство
2. Глобальные переменные
3. Открытые файлы
4. Дочерние процессы
5. Необработанные аварийные сигналы
6. Сигналы и обработчики событий
7. Учетная информация

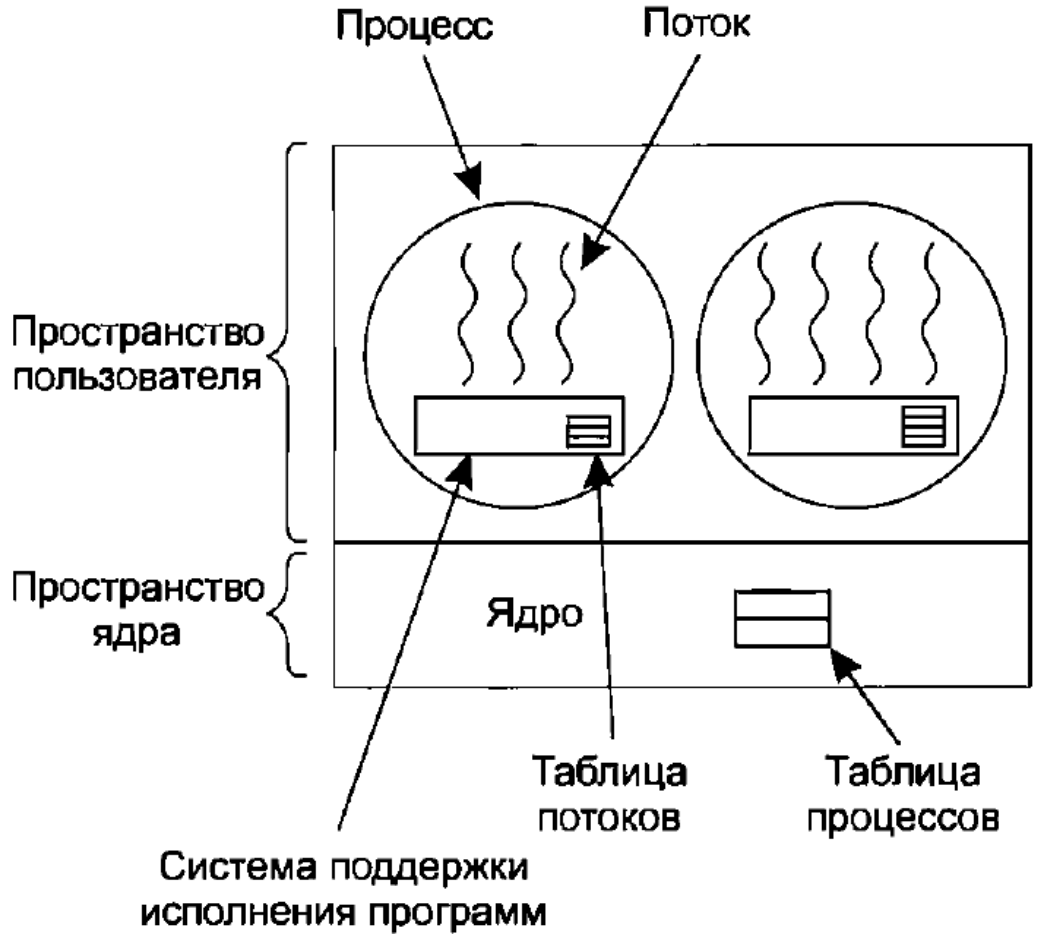
ЭЛЕМЕНТЫ, ПРИСУЩИЕ ПОТОКАМ

1. Счетчик команд
2. Регистры
3. Стек
4. Состояние

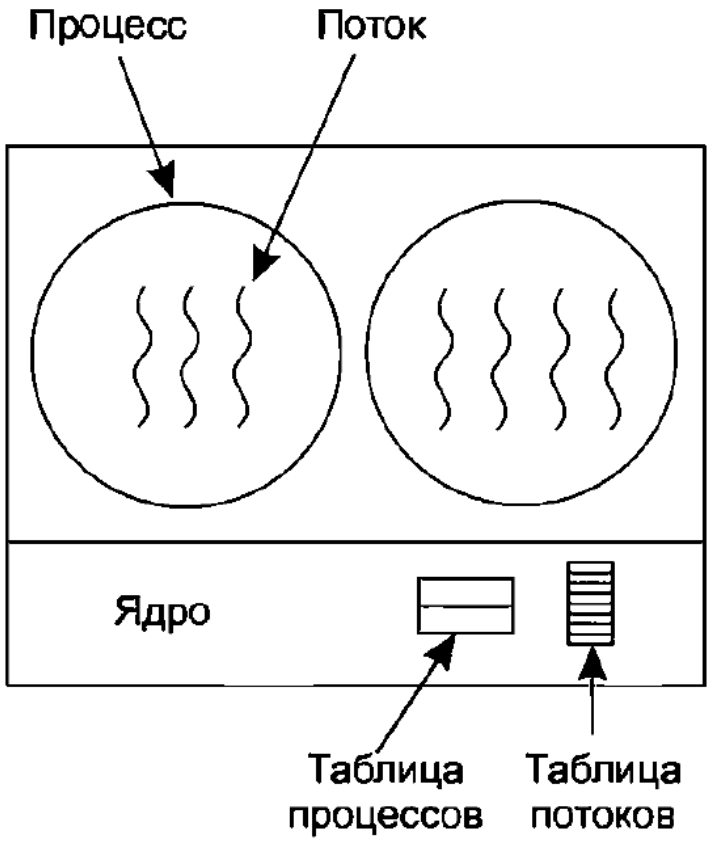
Стратегии реализации потоков

1. В пользовательском пространстве
2. В ядре
3. Гибридный способ

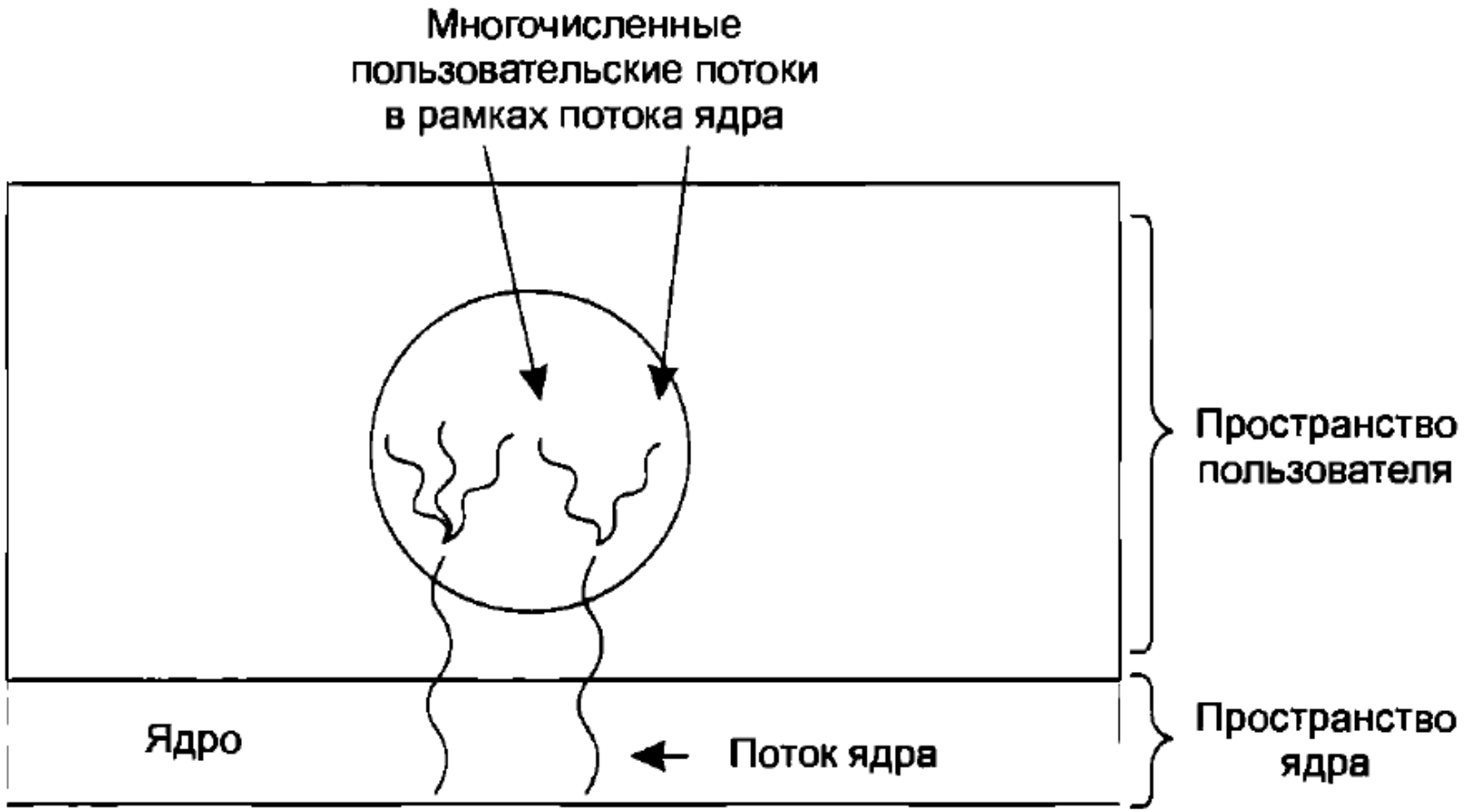
Реализация потоков в пользовательском пространстве



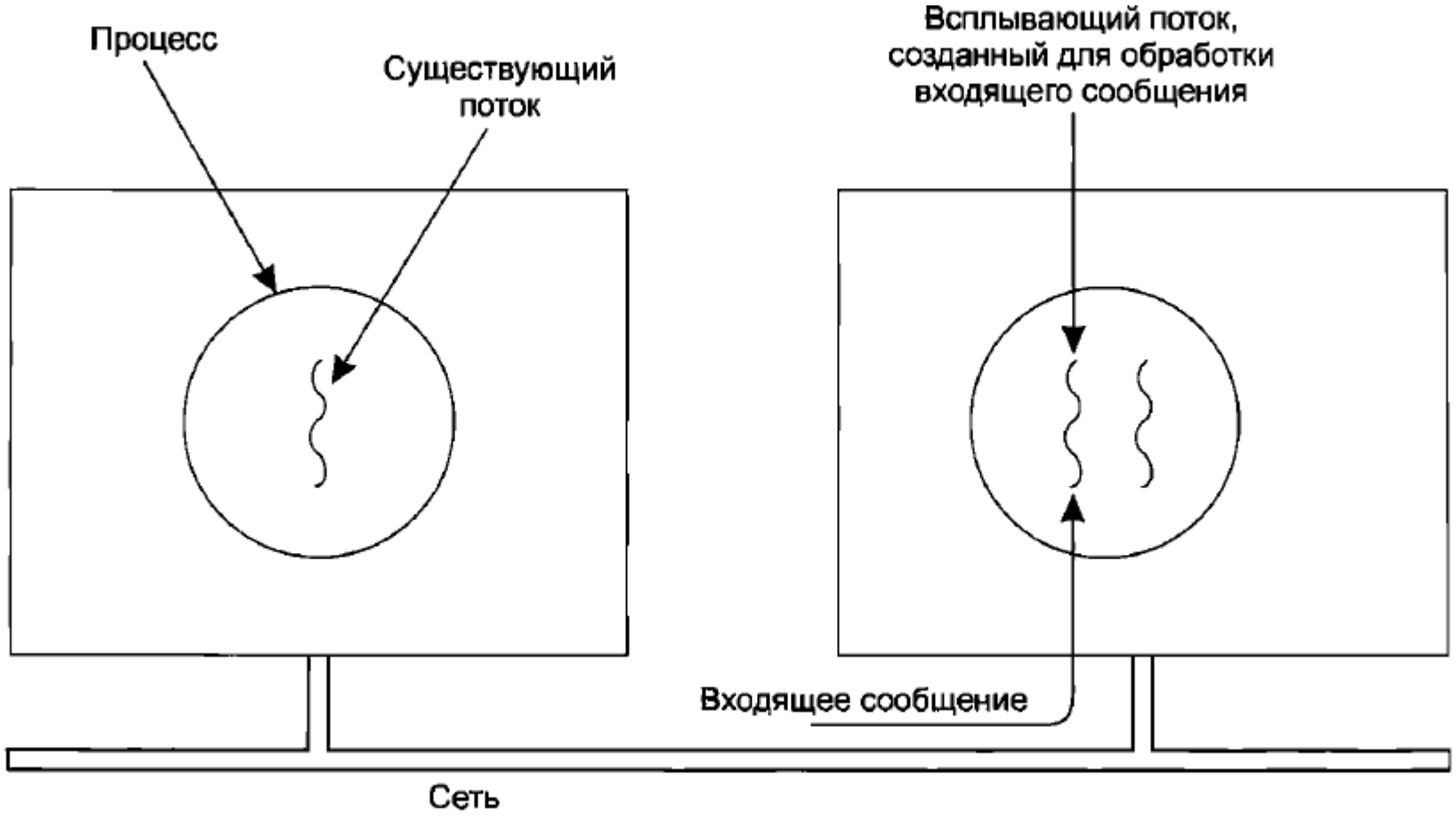
Реализация потоков в ядре



Гибридный способ реализации потоков



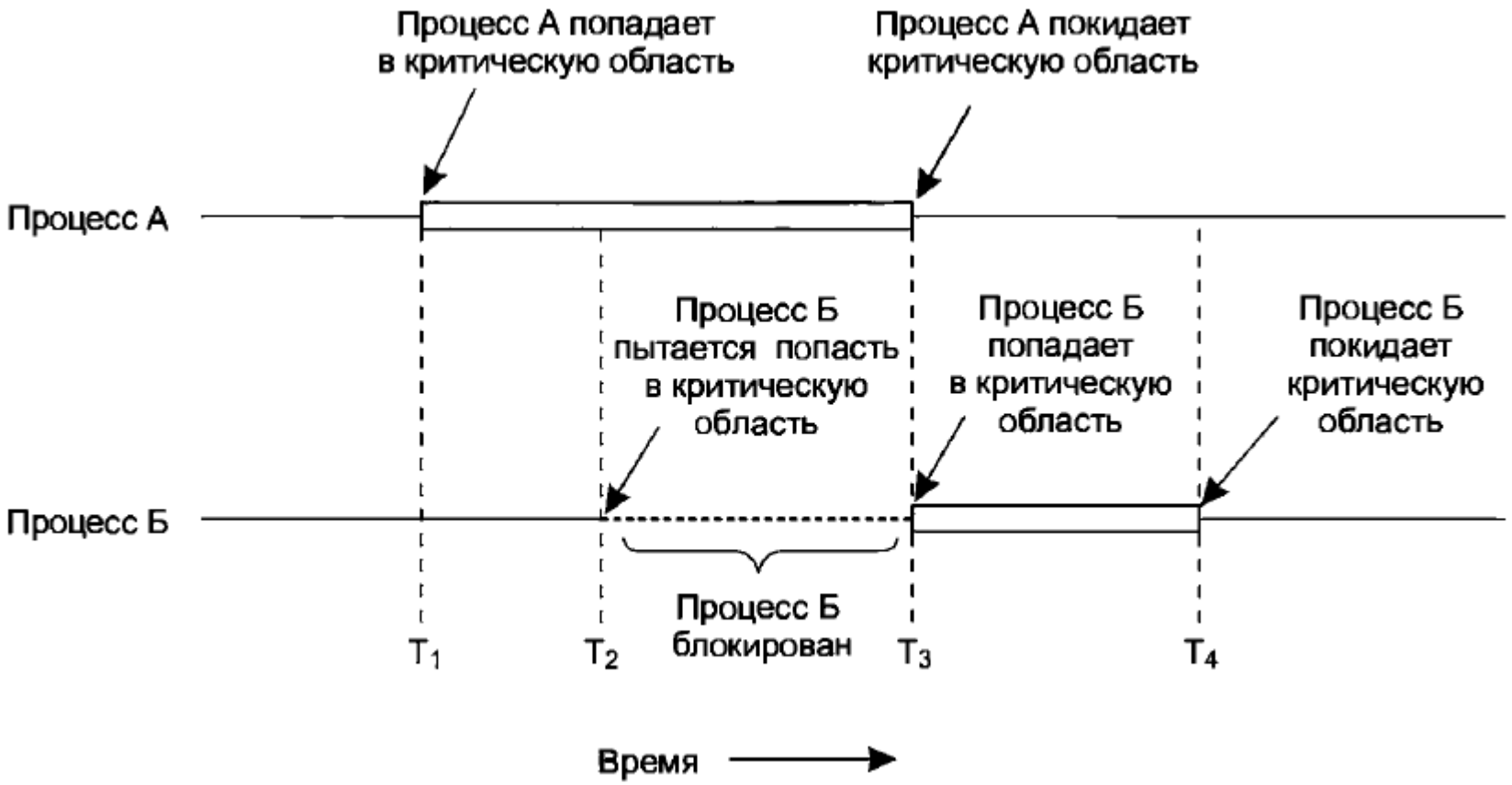
Всплывающие потоки



Взаимодействие процессов и разрешение состязательных ситуаций

1. Взаимоисключение с активным ожиданием
2. Приостановка и активизация
3. Передача сообщений
4. Барьеры

Критические области



Взаимоисключение с активным ожиданием

1. Запрещение прерываний
2. Блокирующие переменные
3. Строгое чередование
4. Алгоритм Петерсона
5. Команда TSL

Алгоритм Петерсона

```
#define FALSE     0
#define TRUE      1
#define N         2                    /* количество процессов */
int turn;                              /* чья очередь? */
int interested[N];                     /* все исходные значения равны 0 (FALSE) */
void enter_region(int process);        /* process имеет значение 0 или 1 */
{
     int other;                        /* номер другого процесса */
     other = 1 - process;              /* противостоящий процесс */
     interested[process] = TRUE;       /* демонстрация заинтересованности */
     turn = process;                   /* установка флажка */
     while (turn == process && interested[other] == TRUE)   /* цикл без инструкции */;
}
void leave_region(int process)         /* процесс, покидающий критическую область */
{
     interested[process] = FALSE;      /* признак выхода из критической области */ }
```

Команда TCL

```
enter region:
    TSL REGISTER,LOCK     | копирование lock в регистр с присвоением ей 1
    CMP REGISTER,#0       | было ли значение lock нулевым?
    JNE enter_region      | если оно было ненулевым, значит, блокировка
                          | уже установлена и нужно войти в цикл
    RET                   | возврат управления вызывающей программе:
                          | вход в критическую область осуществлен

leave region:
    MOVE LOCK,#0          | присвоение переменной lock нулевого значения
    RET                   | возврат управления вызывающей программе
```

Приостановка и активизация

1. Семафор
2. Мьютекс
3. Монитор

Семафор

1. Целое неотрицательное число
2. Down
3. Up

Мьютекс

1. Разновидность семафора
2. Имеется только два состояния 0 и 1

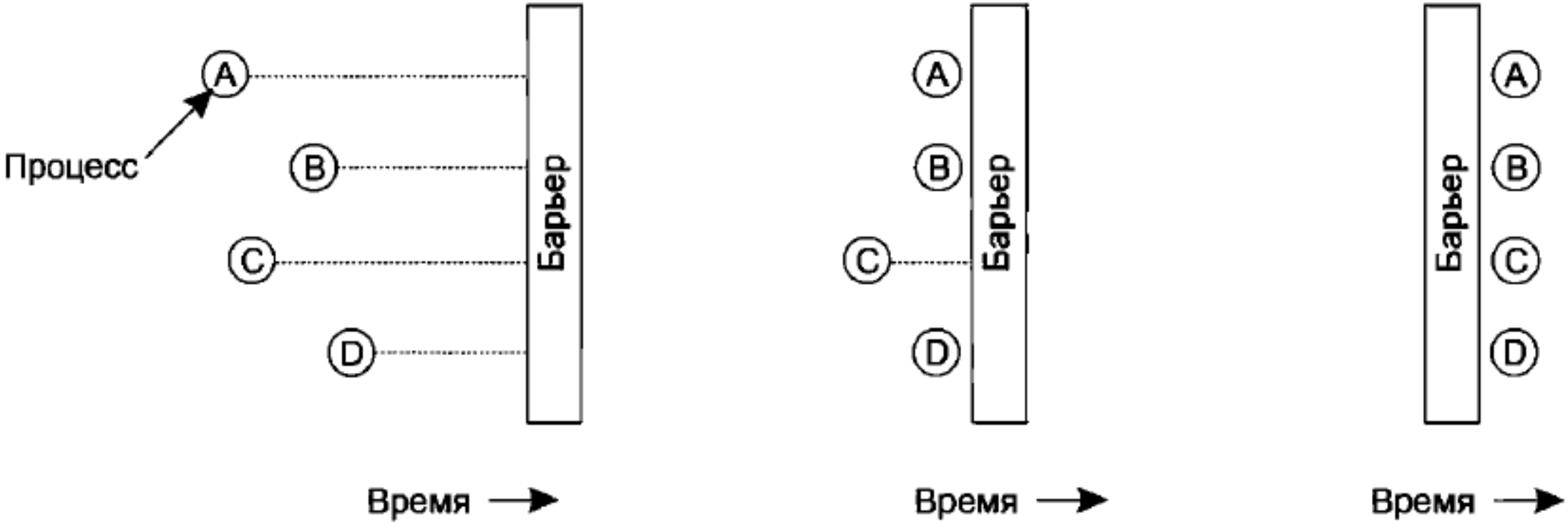
Монитор

1. Понятие языка
2. Некий более высокий уровень абстракции

Передача сообщений

1. Send(dest, &message)
2. Receive (source, &message)

Барьеры


A
B
C
D
Процесс
Барьер
Время
A
B
C
D
Барьер
Время
A
B
C
D
Барьер
Время