

АРХИТЕКТУРА ОПЕРАЦИОННЫХ СИСТЕМ

ЛЕКЦИЯ 1.5

НИТИ ИСПОЛНЕНИЯ (THREADS)

Ожидание ввода A

Ввести массив A

Ожидание ввода B

Ввести массив B

Ввести массив C

Ожидание ввода C

$A=A+B$

$C=A+C$

Вывести массив C

Ожидание вывода C

НИТИ ИСПОЛНЕНИЯ (THREADS)

Процесс 1

Создание процесса 2

Создание общей памяти

Ввести массив A

Ожидание ввода A

Ввести массив B

Ожидание ввода B

Ввести массив C

Ожидание ввода C

$C=A+C$

Вывести массив C

Ожидание вывода C

Переключение контекста

Переключение контекста

Переключение контекста

Переключение контекста

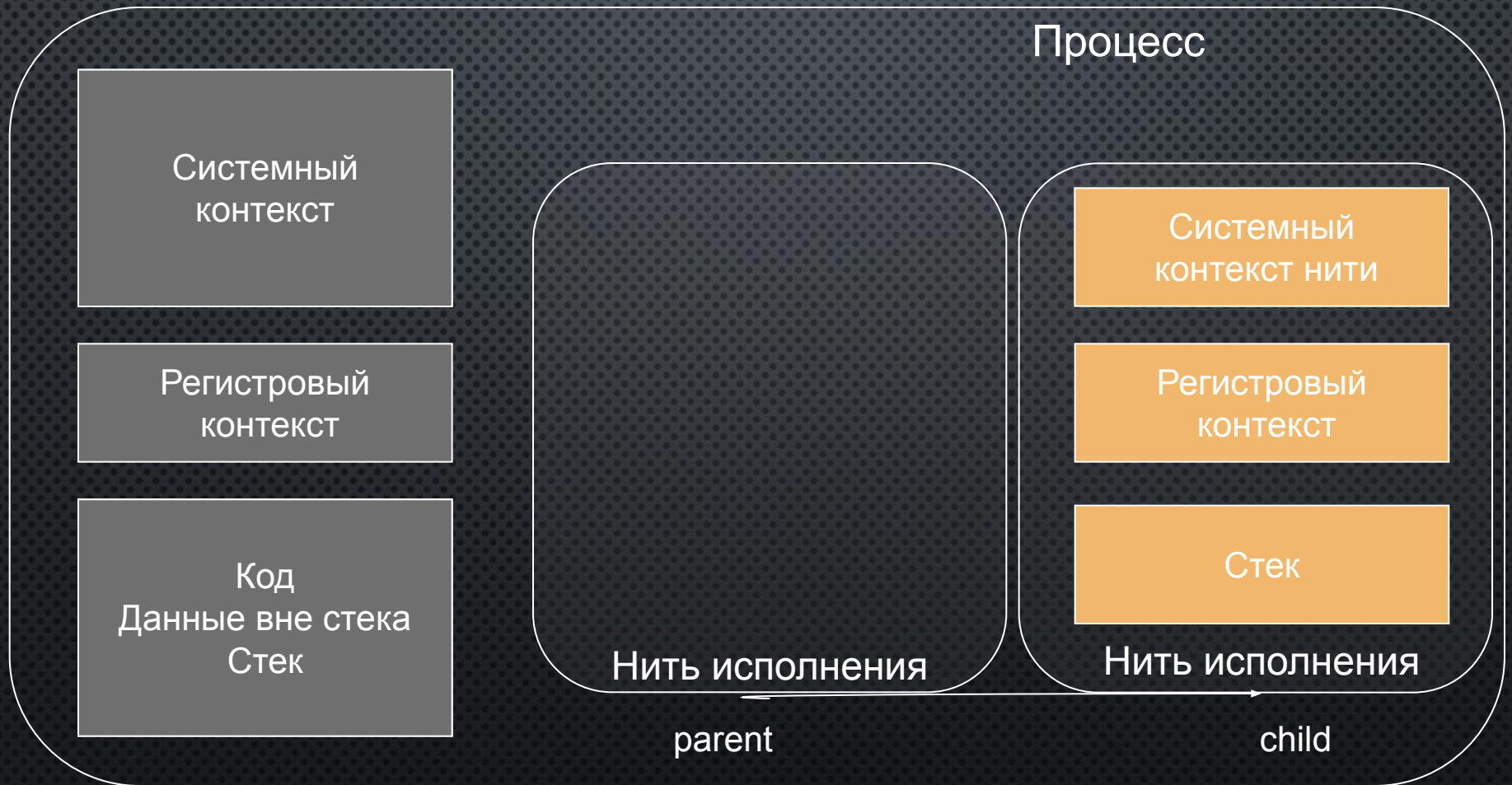
Процесс 2

Создание общей памяти

Ожидание ввода A и B

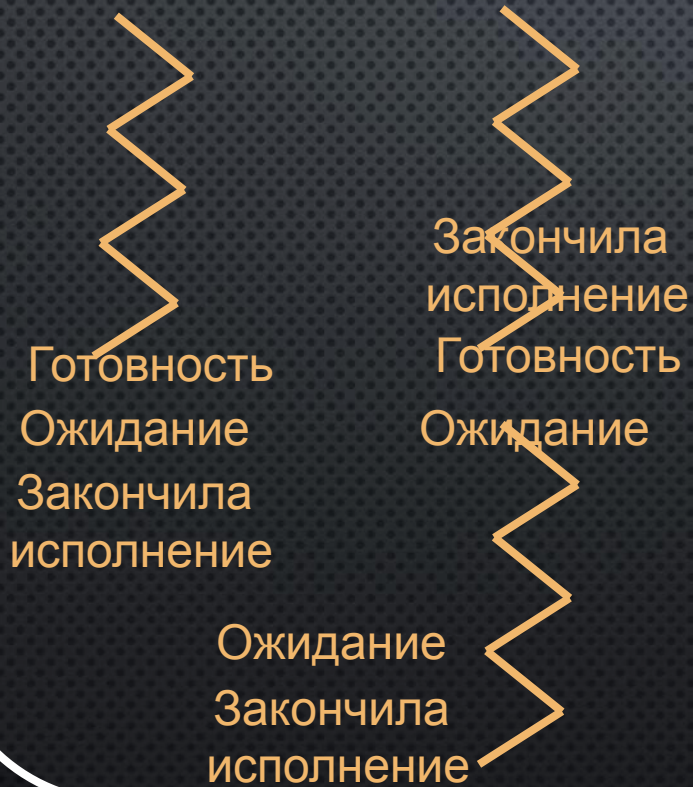
$A=A+B$

НИТИ ИСПОЛНЕНИЯ (THREADS)



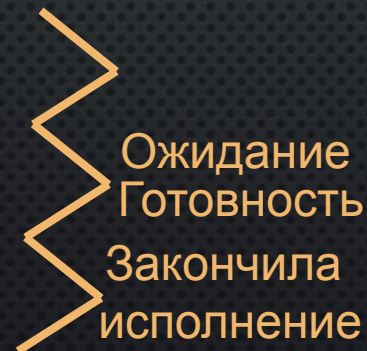
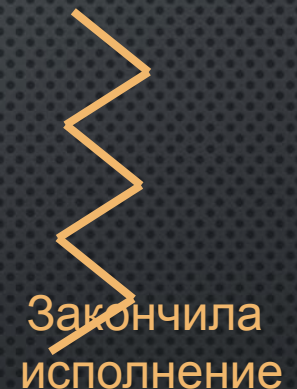
НИТИ ИСПОЛНЕНИЯ (THREADS)

Готовность
Ожидание
Исполнение
Закончил
исполнение



Процесс

Ожидание
Исполнение
Закончила
исполнение



НИТИ ИСПОЛНЕНИЯ (THREADS)

Нить 1

Создание нити 2

Ввести массив A
Ожидание ввода A
Ввести массив B
Ожидание ввода B
Ввести массив C
Ожидание ввода C

$C=A+C$

Вывести массив C
Ожидание вывода C

Нить 2

Переключение контекста

Переключение контекста

Ожидание ввода A и B

Переключение контекста

Переключение контекста

$A=A+B$

АКТИВНОСТИ И АТОМАРНЫЕ ОПЕРАЦИИ

Активность : приготовление бутерброда

- ОТРЕЗАТЬ ЛОМТИК ХЛЕБА
- ОТРЕЗАТЬ ЛОМТИК КОЛБАСЫ
- НАМАЗАТЬ ХЛЕБ МАСЛОМ
- Положить колбасу на хлеб



Атомарные или
неделимые операции

Активность - последовательное выполнение ряда действий, направленных на достижение определенной цели

INTERLEAVING

Активность P: a b c

Активность Q: d e f

Последовательное выполнение PQ: a b c d e f

Псевдопараллельное выполнение
(режим разделения времени) :

?
a b d c e f
a b d e c f
a b d e f c
...
d e f a b c

ДЕТЕРМИНИРОВАННЫЕ И НЕДЕТЕРМИНИРОВАННЫЕ НАБОРЫ АКТИВНОСТЕЙ

$$P: \begin{aligned} x &= 2 \\ y &= x - 1 \end{aligned}$$

$$Q: \begin{aligned} x &= 3 \\ y &= x + 1 \end{aligned}$$

$$(x, y): \begin{array}{cccc} (3, 1) & (3, 4) & (3, \quad) & \\ (2, 1) & (2, \quad) & (2, 3) & (2, 1) \\ (3, 4) & (3, 2) & & \end{array}$$

- *НЕДЕТЕРМИНИРОВАННЫЙ* НАБОР — ПРИ ОДИНАКОВЫХ НАЧАЛЬНЫХ ДАННЫХ ВОЗМОЖНЫ РАЗНЫЕ РЕЗУЛЬТАТЫ
- *ДЕТЕРМИНИРОВАННЫЙ* НАБОР — ПРИ ОДИНАКОВЫХ НАЧАЛЬНЫХ ДАННЫХ **ВСЕГДА** ОДИН РЕЗУЛЬТАТ

УСЛОВИЯ БЕРНСТАЙНА (BERNSTAIN)

$$P: \begin{array}{l} 1) x=u+v \\ 2) y=x*w \end{array}$$

Входные переменные

$$R_1 = \{u, v\}$$

$$R_2 = \{x, w\}$$

$$R(P) = \{u, v, x, w\}$$

Выходные переменные

$$W_1 = \{x\}$$

$$W_2 = \{y\}$$

$$W(P) = \{x, y\}$$

Если:

$$1) W(P) \cap W(Q) = \{\emptyset\}$$

$$2) W(P) \cap R(Q) = \{\emptyset\}$$

$$3) R(P) \cap W(Q) = \{\emptyset\}$$

то набор активностей $\{P, Q\}$ является
детерминированным

СОСТОЯНИЕ ГОНКИ (RACE CONDITION) И ВЗАИМОИСКЛЮЧЕНИЕ (MUTUAL EXCLUSION)

$$P: \begin{aligned} x &= 2 \\ y &= x - 1 \end{aligned}$$

$$Q: \begin{aligned} x &= 3 \\ z &= x + 1 \end{aligned}$$

Набор недетерминирован – состязание процессов
за использование переменной x

В недетерминированных наборах всегда
встречается *race condition* (состояние гонки,
состояние состязания)

Избежать недетерминированного поведения при
неважности очередности доступа можно с помощью
взаимоисключения (mutual exclusion)

КРИТИЧЕСКАЯ СЕКЦИЯ

Время	Студент 1	Студент 2	Студент 3	
17-05	Приходит в комнату			
17-07	Достает 6 бут. пива			
17-09		Приходит в комнату		
17-11		Уходит за пивом		
17-13			Приходит в комнату	
17-15			Уходит за пивом	
17-17				
17-19			Покупает 6 бут. пива	
17-21				
17-23			Приносит пиво	Приносит пиво
17-25			Приходит в комнату	Приходит в комнату
17-27				

СТРУКТУРА ПРОЦЕССА, УЧАСТВУЮЩЕГО ВО ВЗАИМОДЕЙСТВИИ

```
while (some condition) {  
    entry section  
        critical section  
    exit section  
        remainder section  
}
```

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Требования, предъявляемые к алгоритмам

1. Программный алгоритм должен быть программным
2. Нет предположений об относительных скоростях выполнения и числе процессоров
3. Выполняется условие взаимоисключения (mutual exclusion) для критических участков
4. Выполняется условие прогресса (progress)
5. Выполняется условие ограниченного ожидания (bound waiting)

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Запрет прерываний

```
while (some condition) {  
    запретить все прерывания  
    critical section  
    разрешить все прерывания  
    remainder section  
}
```

Обычно используется внутри ОС

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Переменная-замок

```
Shared int lock = 0;
```

```
while (some condition) {  
    while (lock); | lock = 1;  
    critical section  
    lock = 0;  
    remainder section  
}
```

```
while (some condition) {  
    while (lock); lock = 1;  
    critical section  
    lock = 0;  
    remainder section  
}
```

Нарушается условие взаимного исключения

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Строгое чередование

	Shared int turn = 0;		Shared int turn = 1;
P_0		P_1	
	while (some condition) {		while (some condition) {
	while (turn != i); <i>while (turn != 0);</i>		while (turn != 1);
	critical section		critical section
	turn = 1-i;		turn = 0;
	remainder section		remainder section
	}		}

Нарушается условие прогресса

Условие взаимного исключения выполняется

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Флаги готовности

P_0	P_1
Shared int ready[2] = {0, 0};	Shared int ready[2] = {1, 1};
while (some condition) {	while (some condition) {
ready[i] = 1; <i>ready[0] = 1;</i>	ready[1] = 1;
while (ready[1-i]); <i>while (ready [1]);</i>	while (ready [0]);
critical section	critical section
ready[i] = 0; <i>ready[0] = 0;</i>	ready[1] = 0;
remainder section	remainder section
}	}

2-я часть условия прогресса нарушается

1-я часть условия прогресса выполняется

Условие взаимоисключения выполняется

ПРОГРАММНЫЕ АЛГОРИТМЫ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ

Алгоритм Петерсона

Shared int ready[2] = {0, 0};

Shared int turn;

P_0

```
while (some condition) {  
    ready[i] = 1; ready[0] = 1;  
    turn = 1;- i;  
while (ready[1-i] && turn == 1-i);  
    while (ready [1] && turn == 1);  
    critical section  
    ready[i] = 0; ready[0] = 0;  
    remainder section  
}
```

P_1

```
while (some condition) {  
    ready[1] = 1;  
    turn = 0;  
while (ready [0] && turn == 0);  
    critical section  
    ready[1] = 0;  
    remainder section  
}
```

Все 5 условий выполняются