Потоки и процессы

МДК 01.01 «Системное программирование»

Определение потока

Определение *потока* тесно связано с последовательностью действий процессора во время исполнения программы. Исполняя программу, процессор последовательно выполняет инструкции программы, иногда осуществляя переходы в зависимости от некоторых условий.

Такая *последовательность выполнения инструкций программы называется* ***потоком*** *управления внутри программы.*

Поток управления зависит от начального состояния переменных, которые используются в программе. В общем случае различные исходные данные порождают различные потоки управления. Поток управления можно представить как нить в программе, на которую нанизаны инструкции, выполняемые микропроцессором. Поэтому часто ***поток управления также называется нитью (thread).*** *В русскоязычной литературе за потоком управления закрепилось название поток.*

Программа является ***многопоточной,*** если в ней может одновременно существовать несколько потоков. Сами потоки называются ***параллельными.***

Если в программе одновременно может существовать только один поток, то программа называется ***однопоточной.***

Пример 1

Для пояснения понятия потока рассмотрим следующую программу, которая выводит минимальное число из двух целых чисел или сообщение о том, что числа равны.

```
#include <iostream>
#include <locale>
using namespace std;

int main()
{
  int a, b;
  setlocale(LC_ALL, "Russian");
// русский язык
  cout << "Введите два целых
числа: ";
  cin >> a >> b;
if (a == b)
{
  cout << "a==b" << endl;
  return 0;
}
if (a < b)
  cout <<"min="<< a <<endl;
else
  cout <<"min="<< b <<endl;
return 0;
}
```

Пример 1 (продолжение):

В примере рассмотрен поток при исходных данных:

a == b

```
  cout << "Введите два
целых числа: ";
  cin >> a >> b;
  if (a == b)
  {
    cout << "a==b" << endl;
   return 0;
  }
```

А также рассмотрены потоки при исходных данных: a < b. В данном случае потоков два – при выполнении условия и невыполнении условия:

```
  cout << "Введите два целых
числа: ";
  cin >> a >> b;
  if (a == b)…
  if (a < b)
   cout <<"min = «<<a<<endl;
 else
   cout <<"min = "<<b<<endl;
  return 0;
```

Пример 2

Следующая программа вычисляет сумму двух чисел, является однопоточной:

```
#include <iostream>
using namespace std;

int sum(int a, int b)
{
  return a+b;
}
```

```
int main()
{
  int a, b;
  int c = 0;
  cout << "Input two
       integeres : ";
  cin >> a >> b;
  c = sum(a,b);
  cout << "Summa = "
       << c << endl;
  return 0;
}
```

Контекст потока

Контекст потока – содержимое памяти, к которой поток **име**ет доступ во время своего исполнения.

```
int f(int n)
{if(n>0) --n;
 if(n<0) ++n;
  return n;}
```

```
int n;
void g()
{if(n>0) --n;
 if(n<0) ++n;
  return n;}
```

```
int count()
{static int
n=0;
++n;
  return n;}
```

Сколько бы раз эта функция не вызывалась параллельно работающими потоками, она будет корректно изменять значение переменной n, т. к. эта переменная является локальной в функции f.

Т.о. безопасная для потоков

В примерах используются глобальные переменные. Если функция g будет вызвана несколькими параллельно исполняемыми потоками, то может дать некорректное изменение значения переменной n, т. к. значение этой переменной будет изменяться одновременно несколькими функциями . ***Т.о. НЕ безопасная для потоков***

Контекст потока

Функция называется **повторно входимой** или **реентерабельной (reenterable),** если она:

- Не использует глобальные переменные, значения которых изменяются параллельно исполняемыми потоками;
- Не использует статические переменные, определенные внутри функции;
- Не возвращает указатель на статические данные, определенные внутри функции.

Функция называется **безопасной для потоков**, если она обеспечивает блокировку доступа к ресурсам, которые она использует.

Если функция не является реентерабельной, то она также не является и безопасной для потоков.

Состояния потока

Поток описывает динамическое поведение всей программы или какой-либо функции в программе. Если программа является однопоточной, то поток можно рассматривать как пару: поток = (процессор, программа).

Программа может исполняться процессором только в том случае, если она готова к исполнению. То есть все системные ресурсы, которые необходимы для исполнения этой программы, свободны для использования. Кроме того, для исполнения программы необходимо, чтобы и сам процессор был свободен и готов к исполнению этой программы.

Т.о. процессор и программа могут находиться в следующих состояниях.

Состояния процессора:

- ***не выделен*** для исполнения программы;
- ***выделен*** для исполнения программы.

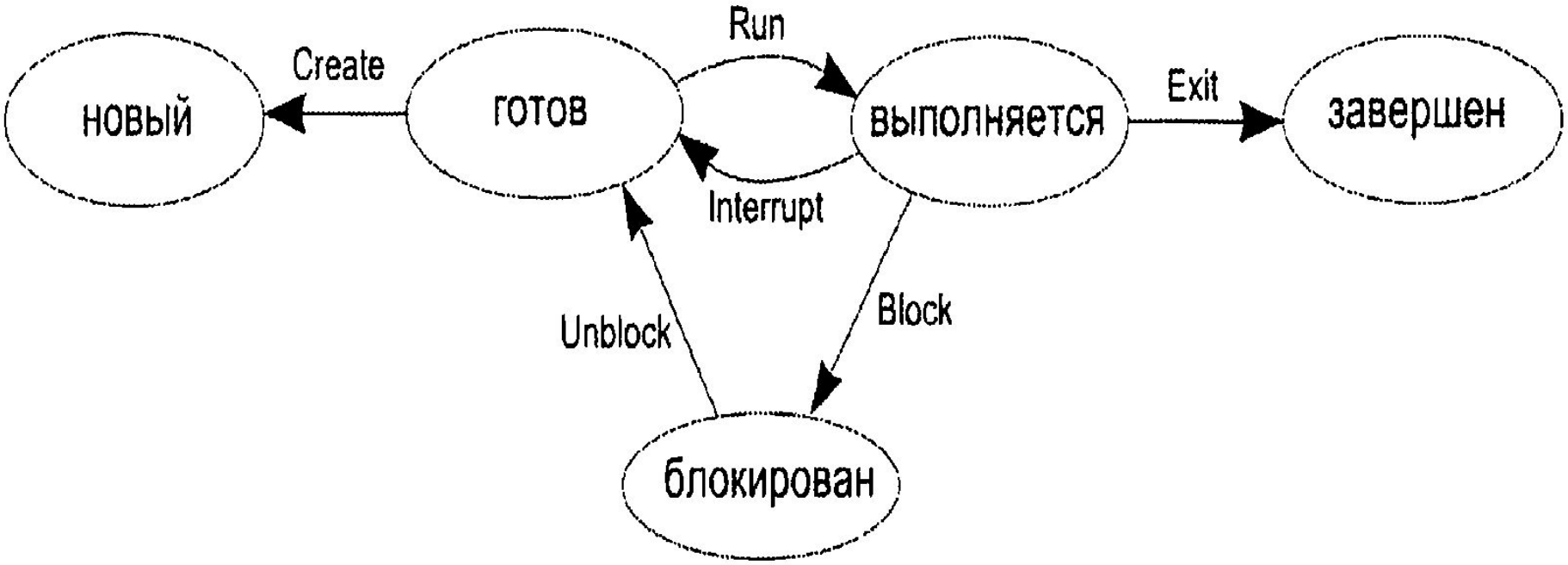
Состояния программы:

- ***не готова*** к исполнению процессором;
- ***готова*** к исполнению процессором.

Состояния потока:

- блокирован = (не выделен, не готова)
- готов к выполнению = (не выделен, готова)
- выполняется = (выделен, готова)
- новый – поток, еще не начавший свою работу
- завершен – поток, завершивший свою работу

Модель состояний потока



Операция **Create** выполняется потоком, который создает новый поток из функции. Из "новый" в "готов".

Операция **Exit** выполняется самим исполняемым потоком в случае его завершения. Из "выполняется" в "завершен".

Модель состояний потока

Операция **Run** запускает готовый поток на выполнение, т. е. выделяет ему процессорное время. Из "готов" в "выполняется". Поток получает процессорное время, если подошла его очередь к процессору на обслуживание.

Операция **Interrupt** задерживает исполнение потока. Из "выполняется" в "готов". Выполняется, если истекло процессорное время, выделенное потоку на исполнение, или исполнение потока прервано по каким-либо другим причинам.

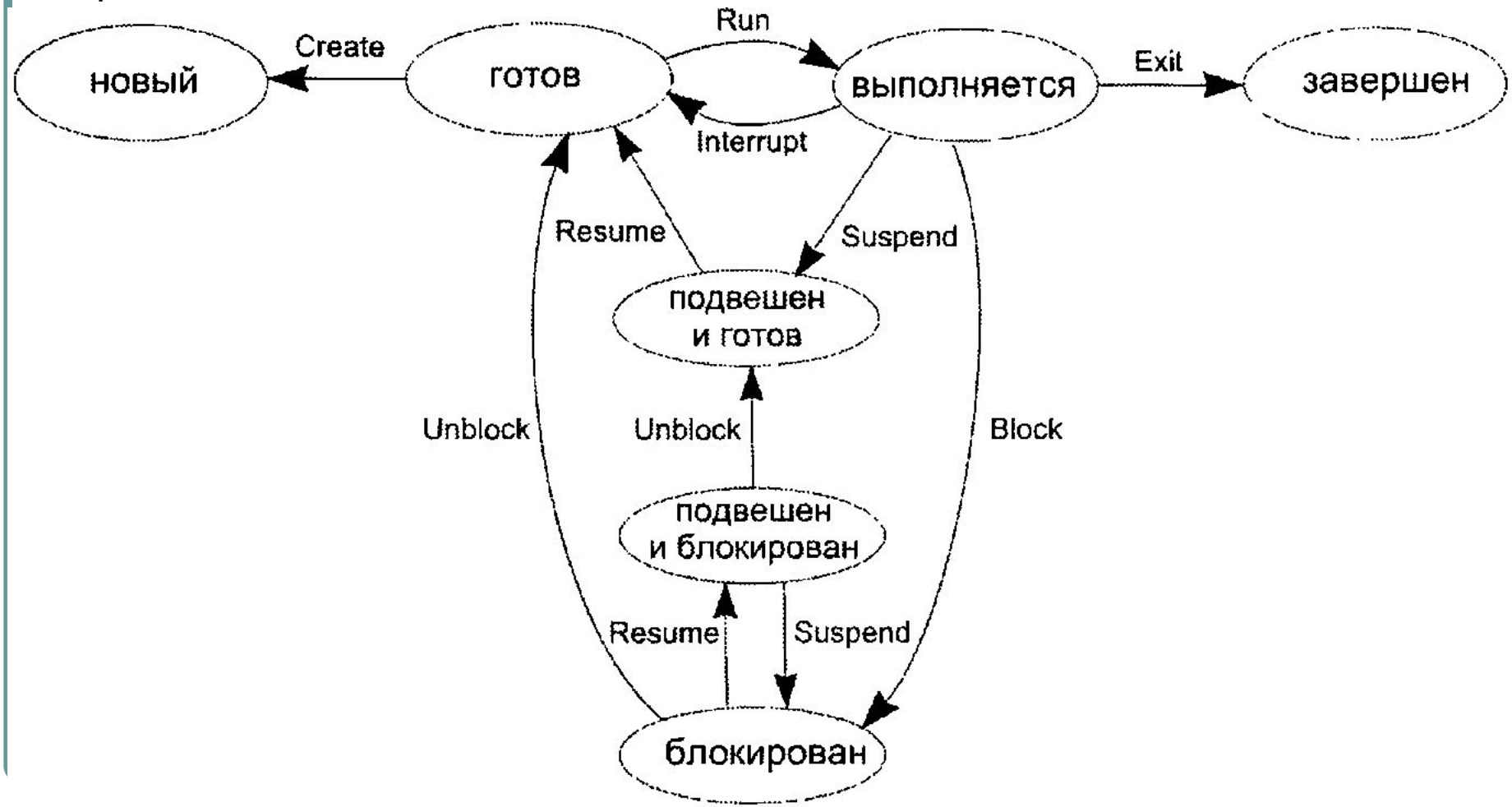
Операция **Block** блокирует исполнение потока. Из "выполняется" в "блокирован". Выполняется, если поток ждет наступления некоторого события, например, завершения операции ввода-вывода или освобождения ресурса.

Операция **Unblock** разблокирует поток. Из "блокирован" в "готов". Выполняется, если событие, ожидаемое потоком, наступило.

Модель состояний потока

Операция **Suspend** приостанавливает исполнение потока (переводит процесс в состояние **приостановлен** или **подвешен**).

Операция **Resume** возобновляет исполнение потока.



Модель состояний потока

Операция **Sleep** позволяет потоку приостановить свое исполнение на некоторый интервал времени. Если поток выполнил операцию Sleep, то он перешел в сонное состояние или "спит".

Операция **Wakeup** позволяет операционной системе разбудить поток.

Диспетчеризация и планирование потоков

Время процессора делится на **кванты** (интервалы), которые выделяются потокам для работы.

Распределяет кванты времени между потоками **менеджер потоков**.

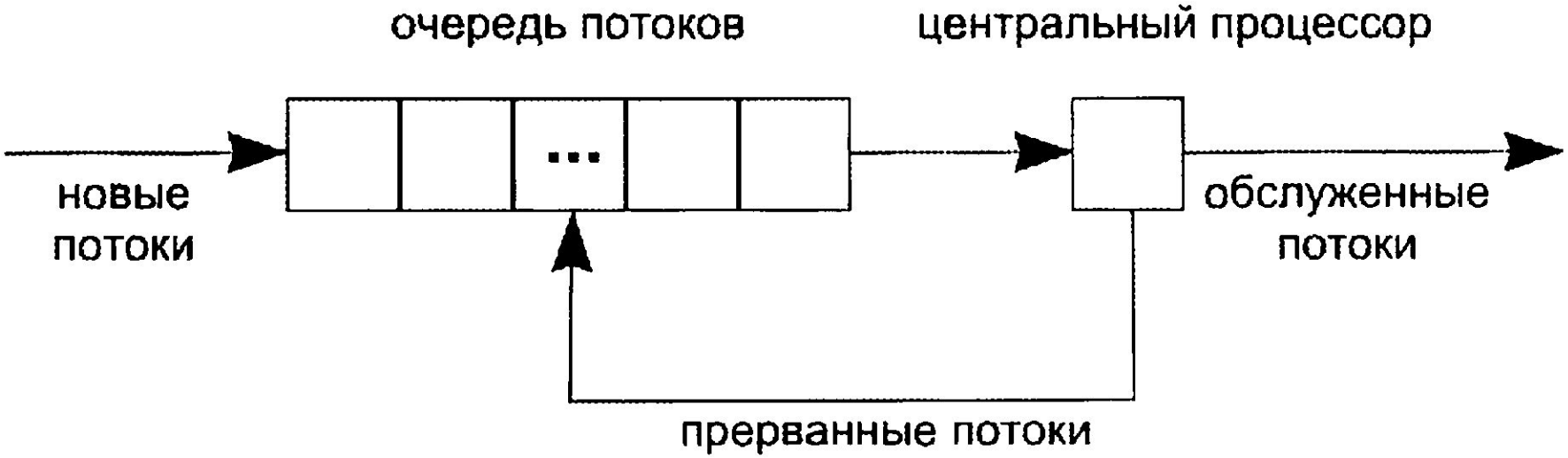
Процесс переключения процессора на исполнение другого потока:

- сохранить контекст прерываемого потока;
- восстановить контекст запускаемого потока на момент прерывания;
- передать управление запускаемому потоку.

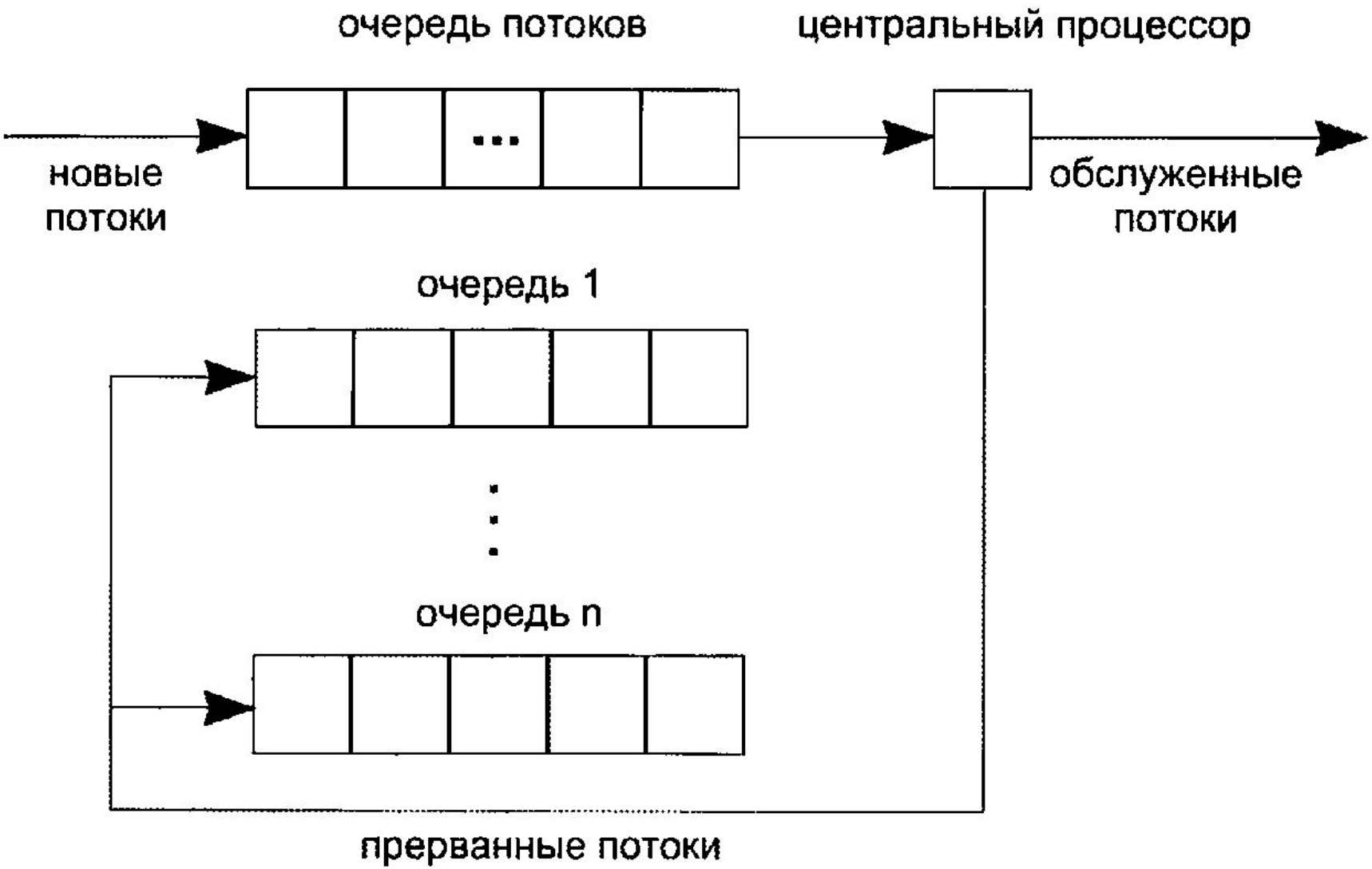
Контекст потока это содержимое памяти, с которой работает поток.

Циклическое обслуживание потоков (одинаковый приоритет)

Потоки выстраиваются в одну очередь на обслуживание к процессору. Процессор обслуживает потоки в порядке **FIFO** (first in — first out), т.е. первым пришел — первым вышел, прерванные потоки становятся в конец очереди. Такая дисциплина обслуживания называется циклическим обслуживанием или **FCFS** (first come — first served), т.е. первым пришел — первым обслужен.



Дисциплина обслуживания потоков с несколькими очередями



Управление потоками

Алгоритмы управления потоками разрабатывают таким образом, чтобы оптимизировать следующие параметры системы:

- время загрузки микропроцессора работой должно быть максимальным;
- пропускная способность системы должна быть максимальной;
- время нахождения потока в системе должно быть минимальным;
- время ожидания потока в очереди должно быть минимальным;
- время реакции системы на обслуживание заявки должно быть минимальным.

Определение процесса

Процессом или **задачей** называется исполняемое на компьютере приложение вместе со всеми ресурсами, которые требуются для его исполнения.

Все ресурсы, необходимые для исполнения процесса, называются **контекстом процесса**.

Процессу обязательно принадлежат следующие ресурсы:

- адресное пространство процесса;
- потоки, исполняемые в контексте процесса.

Адресное пространство — это виртуальная память, выделенная процессу для запуска программ.

Адресные пространства разных процессов не пересекаются. Процесс не имеет непосредственного доступа в адресное пространство другого процесса.

Вопросы для повторения:

1. Последовательность выполнения инструкций во время выполнения программы называется
2. Потоком (thread) называется
3. Программа является многопоточной, если в ней может одновременно существовать
4. Несколько потоков, которые одновременно существуют в многопоточной программе, называют
5. Если в программе одновременно может существовать только один поток, то программа называется
6. Содержимое памяти, к которой поток имеет доступ во время своего исполнения, называется
7. Контекстом потока называется
8. Функция, которая обеспечивает блокировку доступа к ресурсам, которые она использует, называется
9. Функция называется повторно входимой или реентерабельной (reenterable), если она используется в программе
10. Если функция является реентерабельной, то она является
11. Процессор может находиться в следующих состояниях:
12. Программа может находиться в следующих состояниях:
13. Поток может находиться в следующих состояниях:
14. Операция, которая выполняется потоком, который создает новый поток из функции, называется
15. Операция, которая выполняется самим исполняемым потоком в случае его завершения, называется
16 . Операция, которая запускает готовый поток на выполнение, т. е. выделяет ему процессорное время, называется

Вопросы для повторения:

17. Операция, которая задерживает исполнение потока и при этом процесс выполняется, если истекло процессорное время, выделенное потоку на исполнение, называется
18. Операция, которая блокирует исполнение потока и при этом процесс выполняется, если поток ждет наступления некоторого события, называется
19. Операция, которая разблокирует поток, и при этом процесс выполняется, если событие, ожидаемое потоком, наступило, называется
20. Операция, которая приостанавливает исполнение потока (переводит процесс в состояние приостановлен или подвешен), называется
21. Операция, которая возобновляет исполнение потока, называется
22. Операция, которая позволяет потоку приостановить свое исполнение на некоторый интервал времени, называется
23. Процессом или задачей называется
24. Адресное пространство – это

Спасибо за внимание!