Потоки и процессы

```
#include <iostream.h>
int main()
{
int a, b;
cout <<  “ Input two integers: “  ;
cin » a » b;
if (a == b) {
cout « "There is no min." « endl;
return 0;
if (a < b)
cout « "min = “ « a « endl;
else
cout « "min = “ « b « endl;
return 0;
}
```

- В общем случае содержимое памяти, к которой поток имеет доступ во время своего исполнения, называется *контекстом потока*.
- Рассмотрим следующую функцию:

```
int f(int n)
{
if (n > 0)
--n;
if (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. То есть для каждого нового вызова функции **f** будет создан новый локальный экземпляр переменной **n.** Такая функция **f** называется *безопасной для потоков*.

- Теперь введем глобальную переменную n и изменим нашу функцию следующим образом:

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

- В этом случае параллельный вызов функции **g** несколькими потоками может дать некорректное изменение значения переменной **n**, т. к. значение этой переменной будет изменяться одновременно несколькими функциями **g**. В этом случае функция **g** не является безопасной для потоков.

- Та же проблема встречается и в случае, когда функция использует статические переменные. Для разбора этого случая рассмотрим функцию

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

- которая возвращает количество своих вызовов. Если эта функция будет вызвана несколькими параллельно исполняемыми потоками, то нельзя точно определить значение переменной **n**, которое вернет эта функция, т. к. это значение изменяется всеми потоками параллельно.

- В общем случае функция называется *повторно входимой* или *реентерабельной* (reentrant или reenterable), если она удовлетворяет следующим требованиям:
- 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.

- В дополнение к реентерабельным функциям определяют также функции, безопасные для вызова параллельно исполняемыми потоками. Функция называется *безопасной для потоков*, если она обеспечивает блокировку доступа к ресурсам, которые она использует.

поток = (процессор, программа)

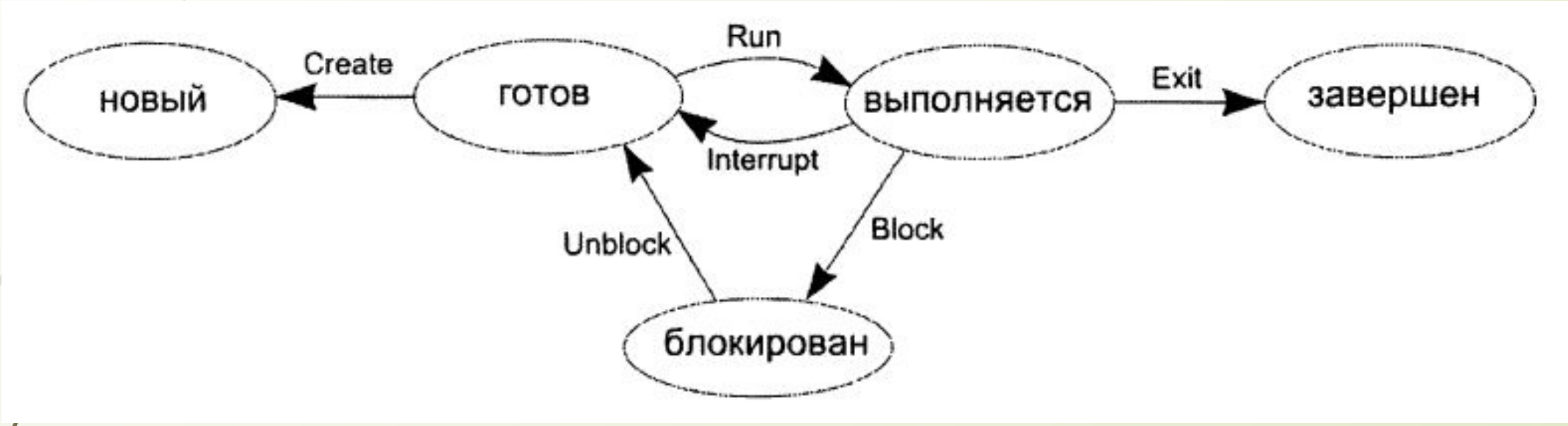
- Программа может исполняться процессором только в том случае, если она готова к исполнению. То есть все системные ресурсы, которые необходимы для исполнения этой программы, свободны для использования. Кроме того, для исполнения программы необходимо, чтобы и сам процессор был свободен и готов к исполнению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

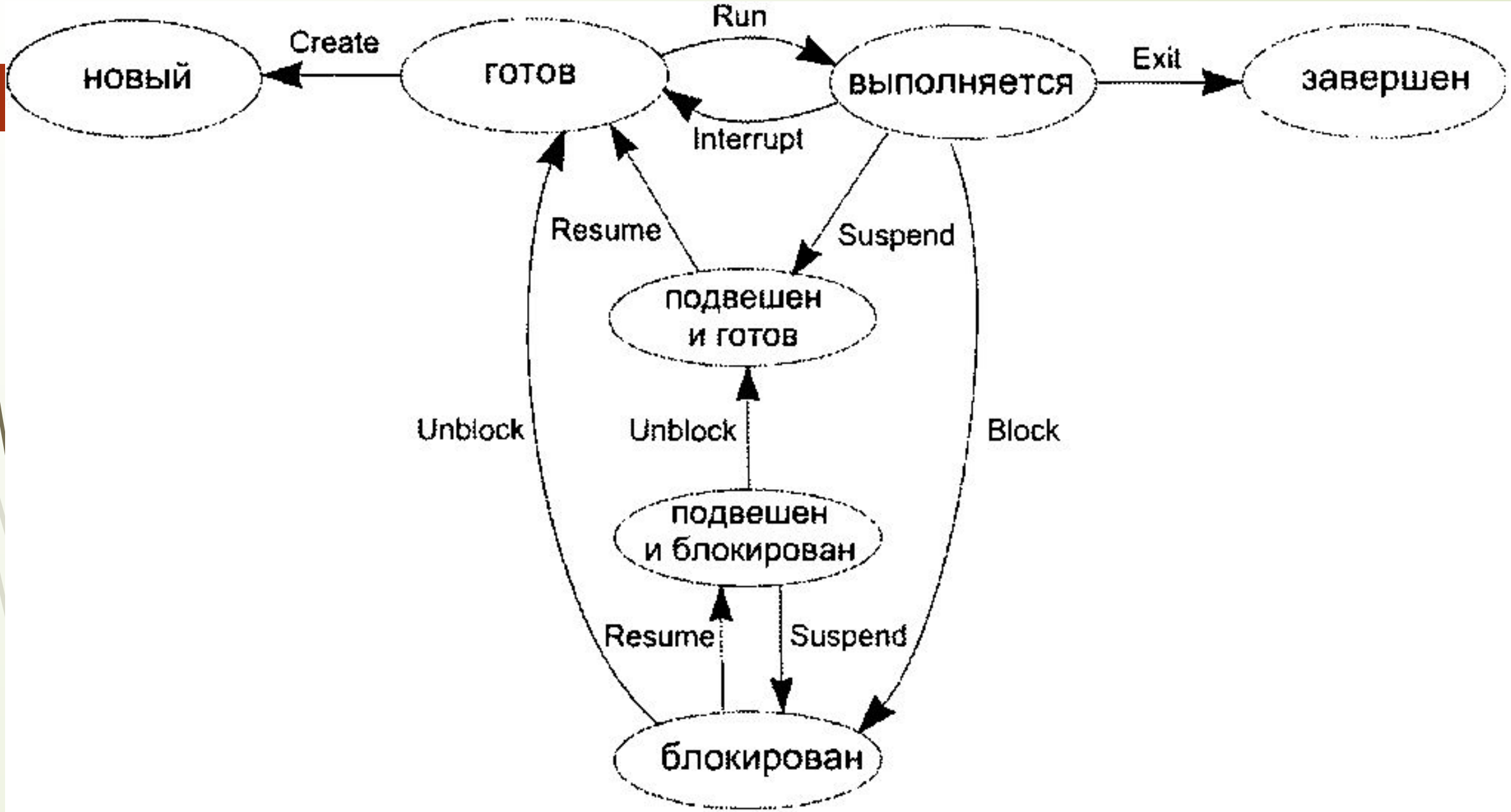
- Состояния процессора:
  - ✔ процессор не выделен для исполнения программы;
  - ✔ процессор выделен для исполнения программы.
- Состояния программы:
  - ✔ программа не готова к исполнению процессором;
  - ✔ программа готова к исполнению процессором.
- Для краткости записи введем для этих состояний следующие названия:
  - Состояния процессора:
    - ✔ "не выделен";
    - ✔ "выделен".
  - Состояния программы:
    - ✔ "не готова";
    - ✔ "готова".

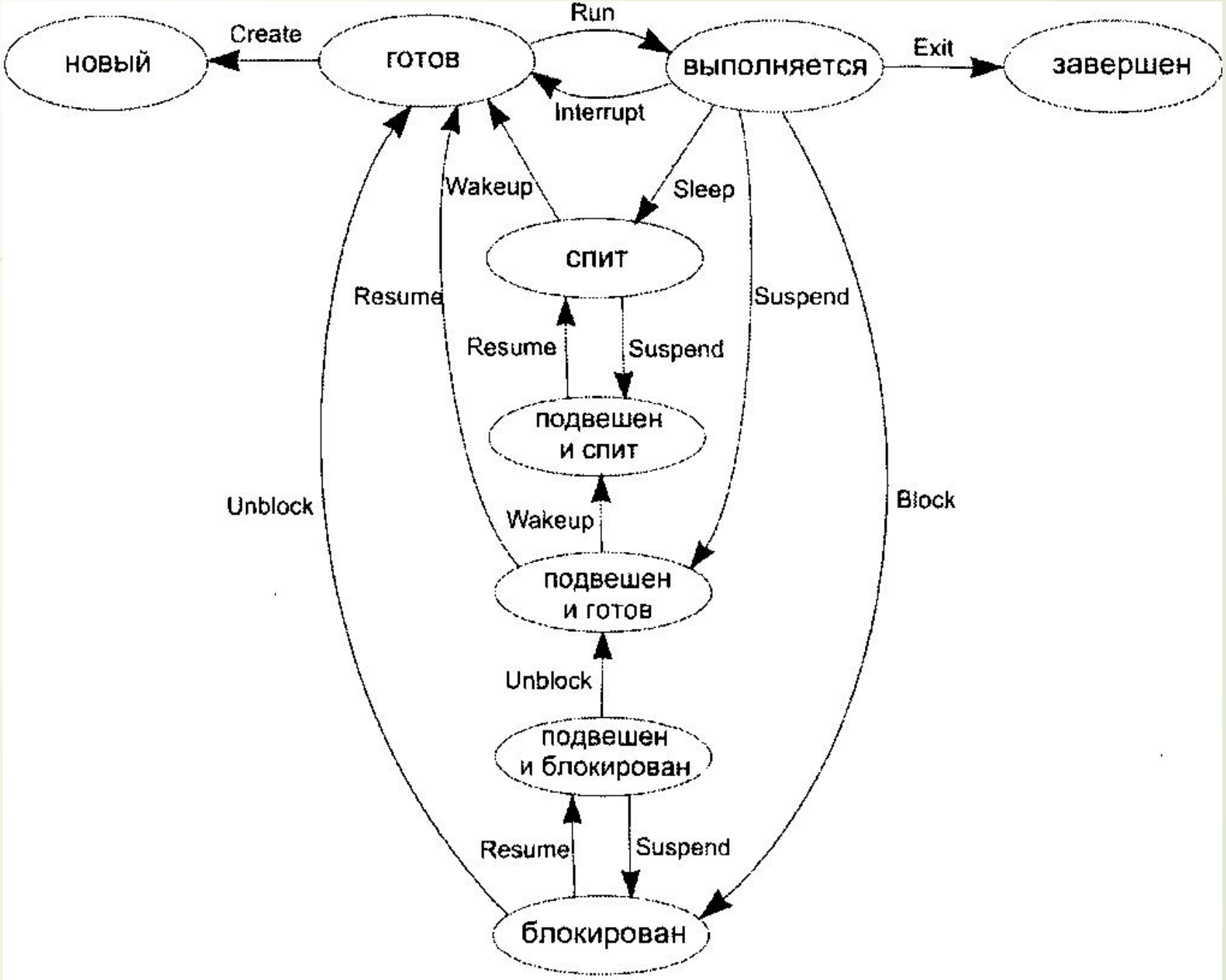
- Тогда мы можем определить *состояние потока* как пару состояний:

состояние потока = (состояние процессора, состояние программы).

- Перечислив различные комбинации состояний процессора и программы, можно описать все возможные состояния потока. Введем для состояний потока следующие названия:
  - ✔ поток блокирован = ("не выделен", "не готова");
  - ✔ поток готов к выполнению = ("не выделен", "готова");
  - ✔ поток выполняется = ("выделен", "готова");


новый
Create
готов
Run
выполняется
Exit
завершен
Interrupt
Unblock
Block
блокирован


новый
Create
готов
Run
выполняется
Exit
завершен
Interrupt
Resume
Suspend
подвешен
и готов
Unblock
Unblock
Block
подвешен
и блокирован
Resume
Suspend
блокирован


новый
Create
готов
Run
выполняется
Exit
завершен
Interrupt
Wakeup
Sleep
спит
Resume
Suspend
Resume
Suspend
подвешен
и спит
Unblock
Wakeup
Block
подвешен
и готов
Unblock
подвешен
и блокирован
Resume
Suspend
блокирован