



# Операционные системы

Лекция №2. Концептуальные  
основы ОС. **Процессы** и потоки



# Требования к ОС

- Чередовать выполнение нескольких задач для повышения степени использования ЦП.
- Распределять ресурсы между задачами в соответствии с заданной стратегией.
- Обеспечивать обмен данными между задачами и запуск новых задач пользователями.



# Концепции

- Компьютерная система представляется набором аппаратных ресурсов.
- Компьютерные приложения создаются для выполнения некоторых задач
- Прямой доступ к аппаратным ресурсам со стороны приложений является неэффективным
- Операционная система представляет удобный в использовании, богатый функционально, безопасный и целостный интерфейс для использования приложениями.
- ОС обеспечивает единообразное, абстрактное представление ресурсов, которые могут выделяться приложениям по их запросу.



# Управление выполнением приложений

- Ресурсы должны быть доступны множеству приложений
- Процессор распределяется среди множества приложений
- Процессор и устройства ввода-вывода эффективно используются множеством приложений.



# Процессы и потоки

**Процесс (Process) и поток (Thread) являются единицами работы в операционной системе.**

- ▣ *Процесс* рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного – процессорного времени. Процессор распределяется операционной системой между другими единицами выполнения – *потоками*.
- ▣ В простейшем случае процесс состоит из одного потока. Если ОС не поддерживает потоков, то поведение потока полностью поглощается поведением процесса.



# Элементы процесса

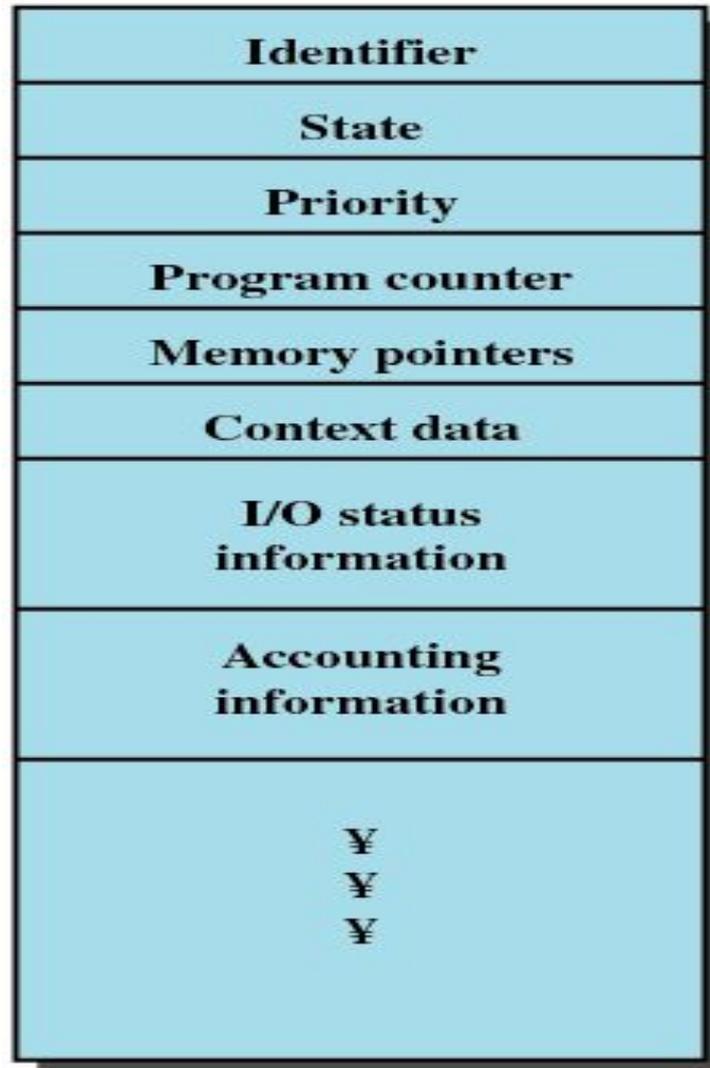
- Идентификатор
- Состояние
- Приоритет
- Счётчик команд
- Таблицы распределения памяти
- Контекст
- Статус операций ввода-вывода
- Статистические данные



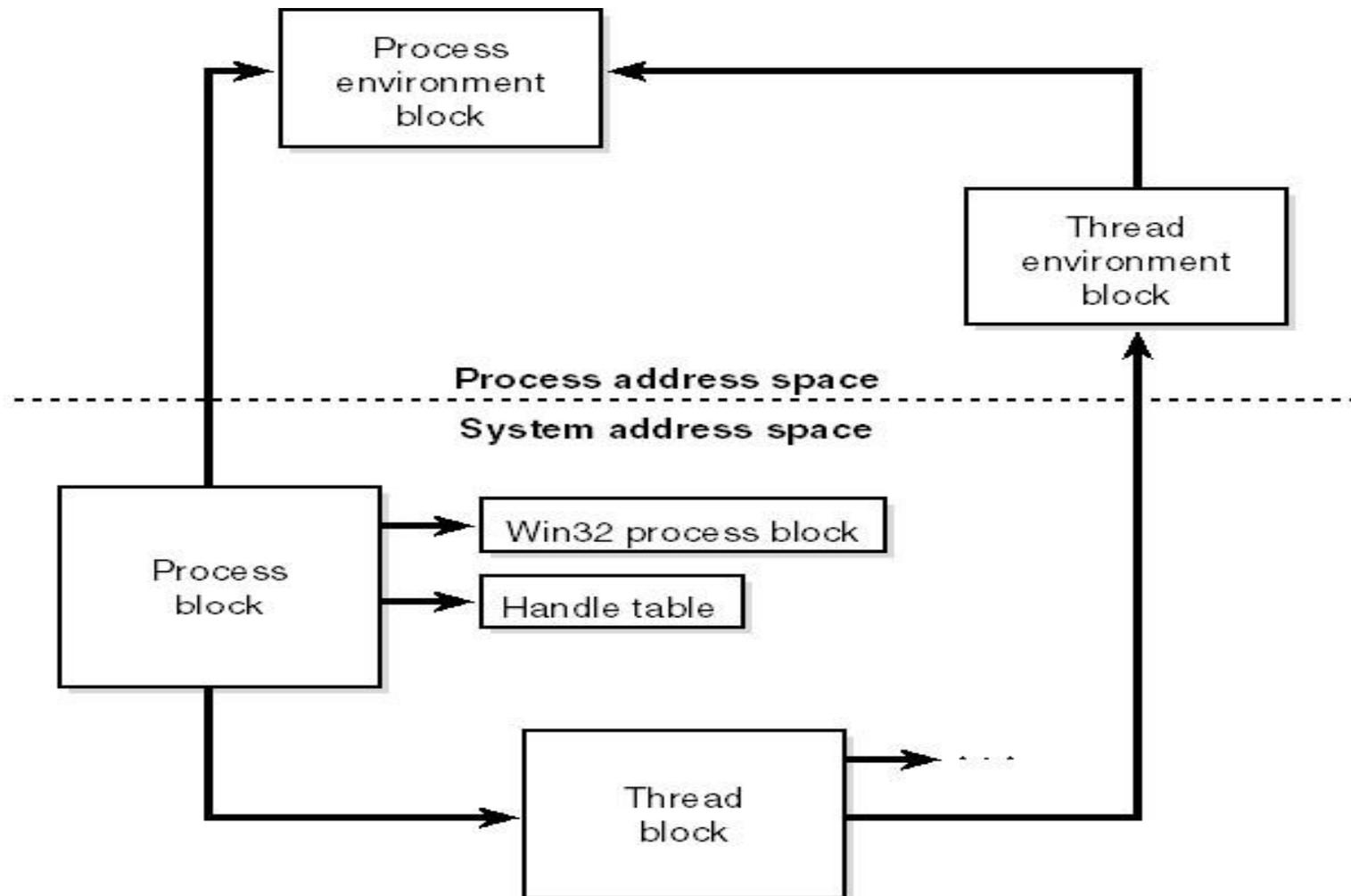
# Блок Управления Процессом (PCB – Process Control Block)

- Содержит элементы процесса
- Создаётся и управляется операционной системой
- Единая структура для поддержки различными процессами.

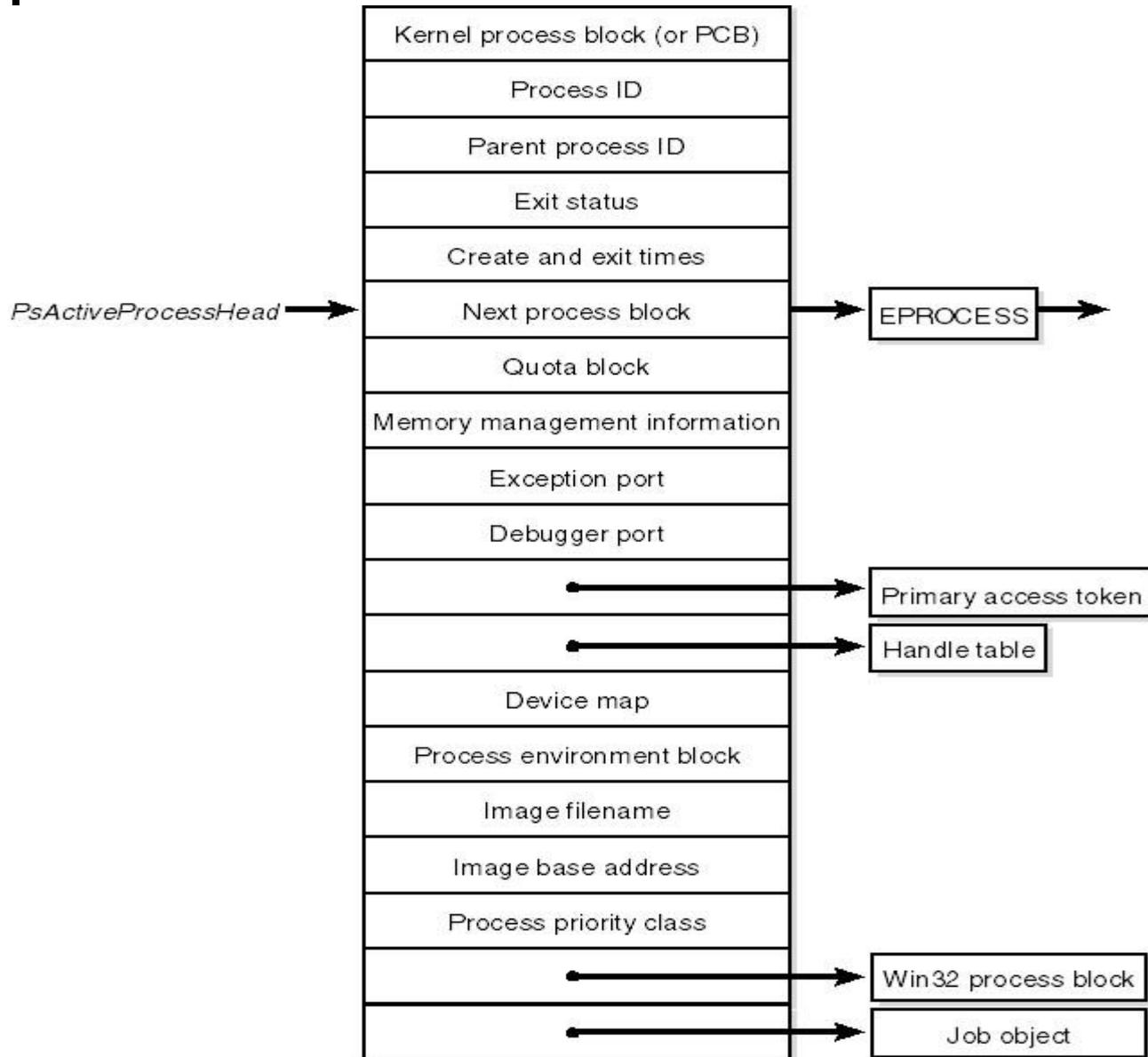
# PCB. Обобщённая схема



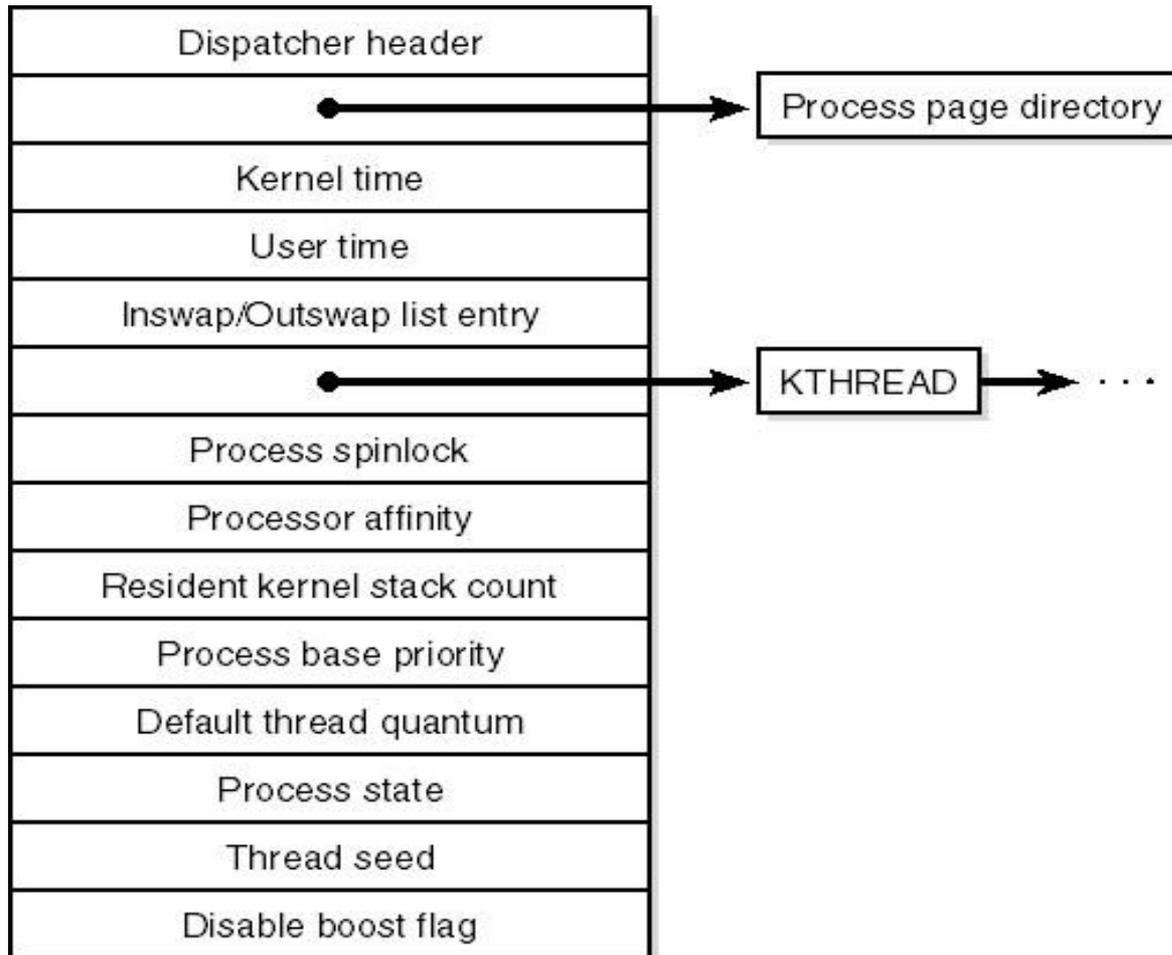
# РСВ в семействе NT систем.



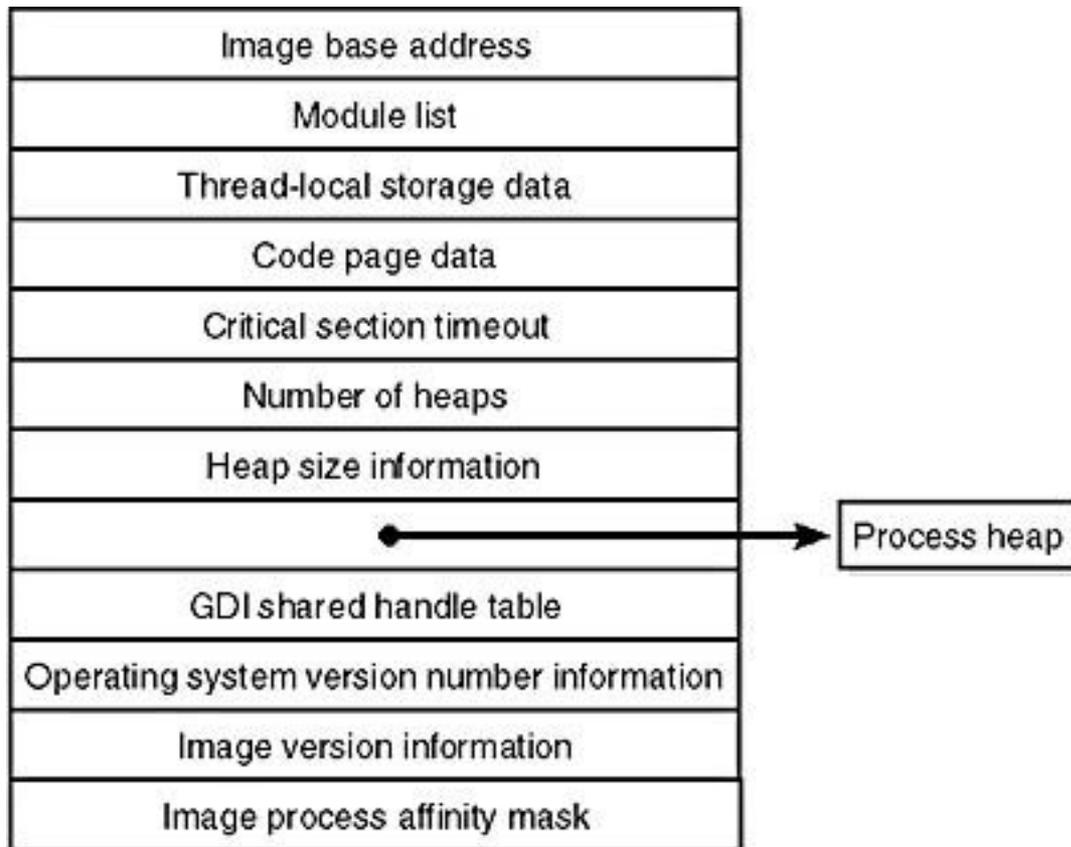
# EPROCESS



# KPROCESS



# PEB – Process Environment Block



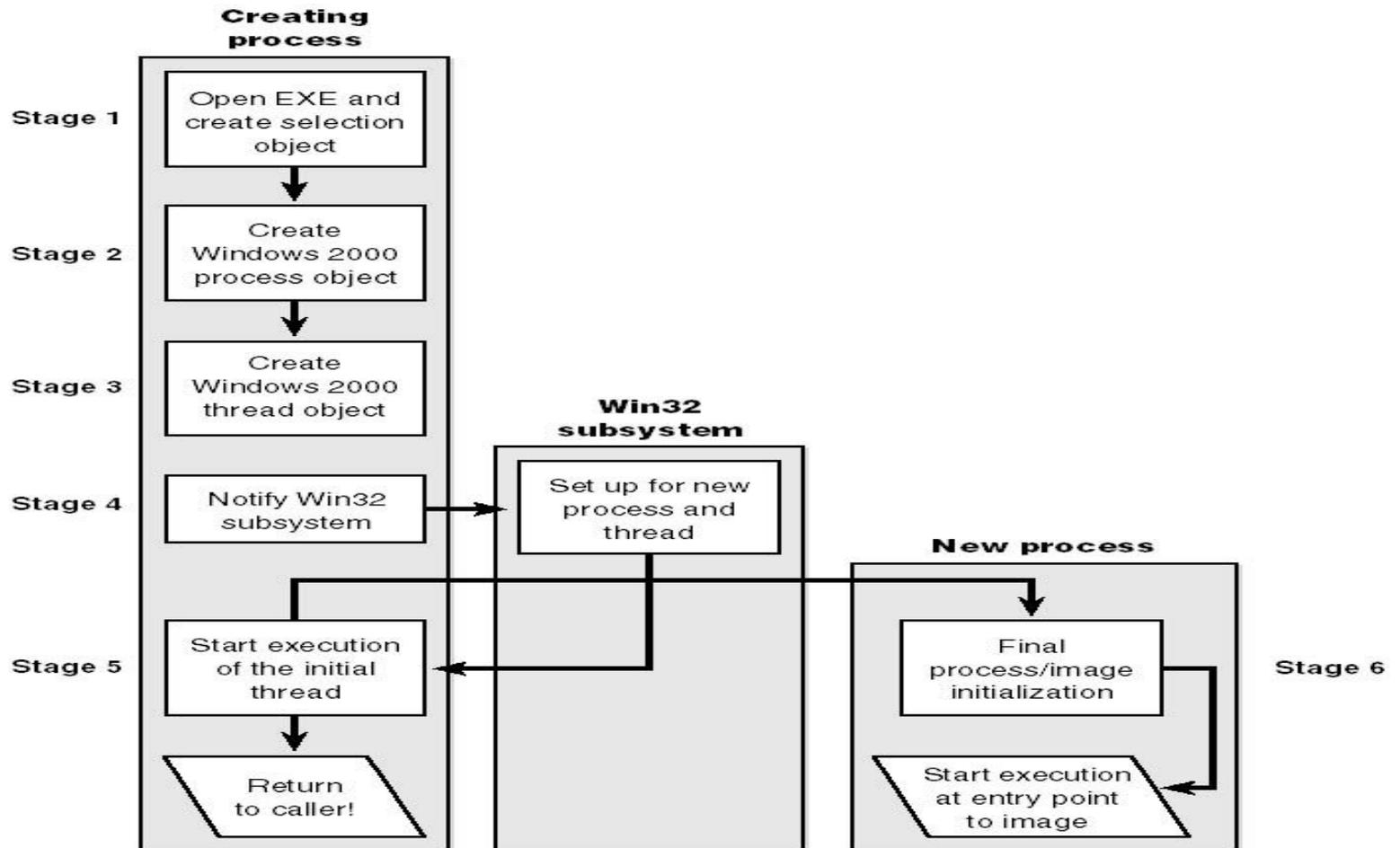


# Создание процесса

**Создать процесс – это подготовить  
новый РСВ.**

- Порождение нового процесса другим
- Создание операционной системой процесса для работы служб
- Вход в систему в интерактивном режиме
- Новое пакетное задание

# Создание процесса в NT системах

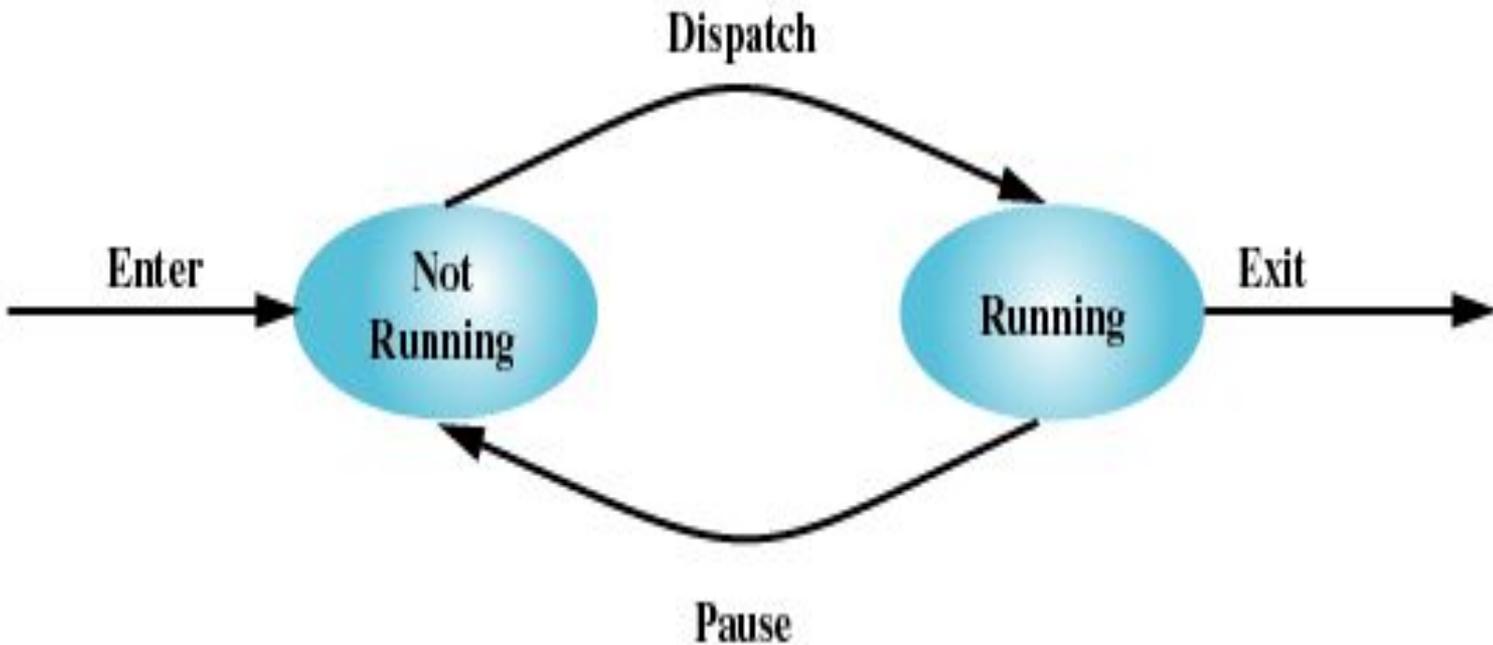




# Завершение процессов

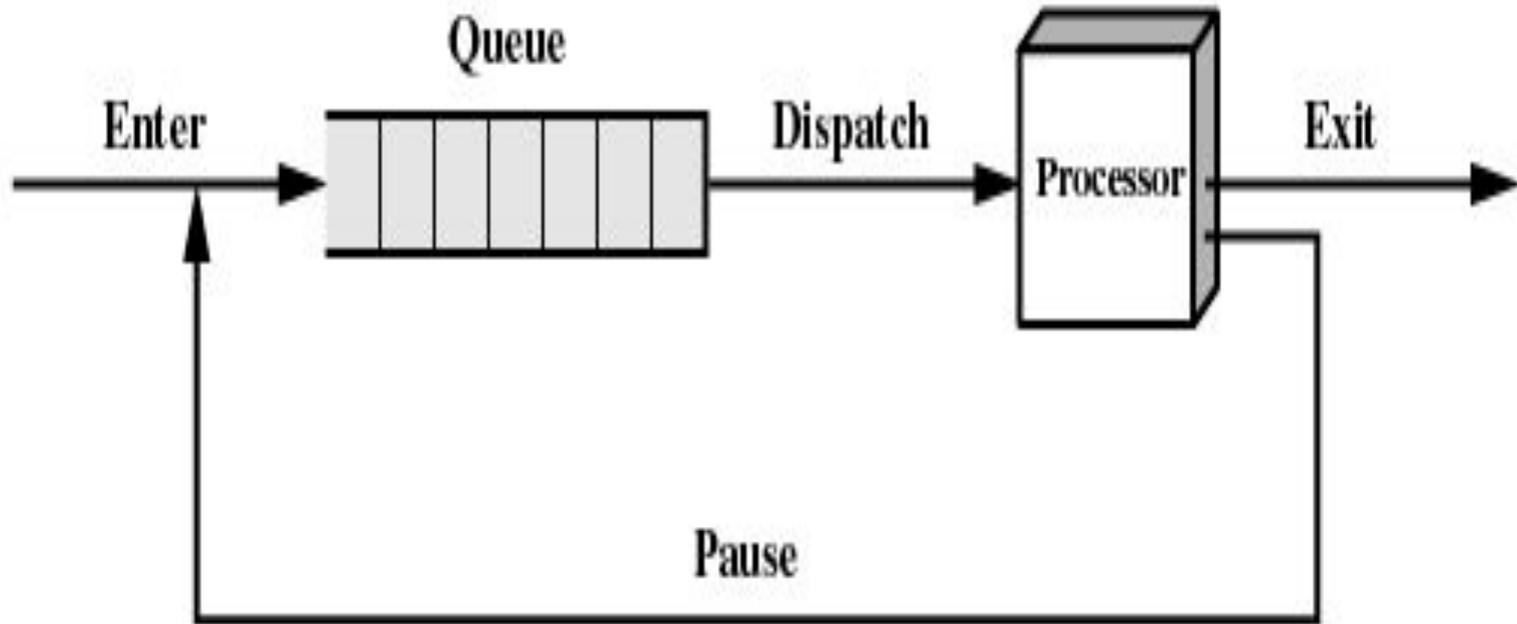
- ▣ Обычное завершение
- ▣ Превышение лимита времени
- ▣ Недостаток памяти
- ▣ Нарушение доступа к памяти
- ▣ Ошибка доступа к ресурсу
- ▣ Арифметическая ошибка
- ▣ Ошибка ввода-вывода
- ▣ Неверная команда
- ▣ Команда с недоступными привилегиями
- ▣ Неправильное использование данных
- ▣ Вмешательство ОС
- ▣ Завершение всех потоков (для многопоточных систем)
- ▣ Запрос со стороны другого процесса.

# Состояния процесса. Модель с двумя состояниями



(a) State transition diagram

# Очередь процессов



(b) Queuing diagram

# Состояния процесса. Модель с пятью состояниями

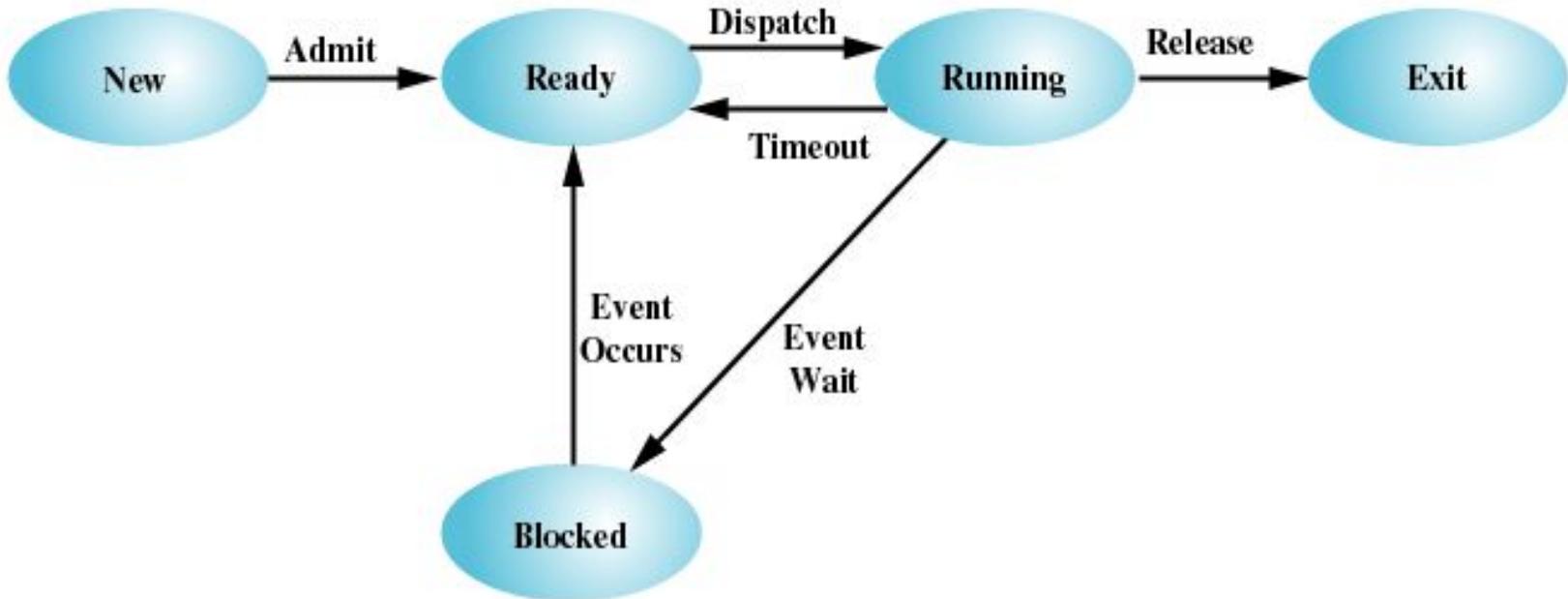


Figure 3.6 Five-State Process Model

# Состояния процесса.

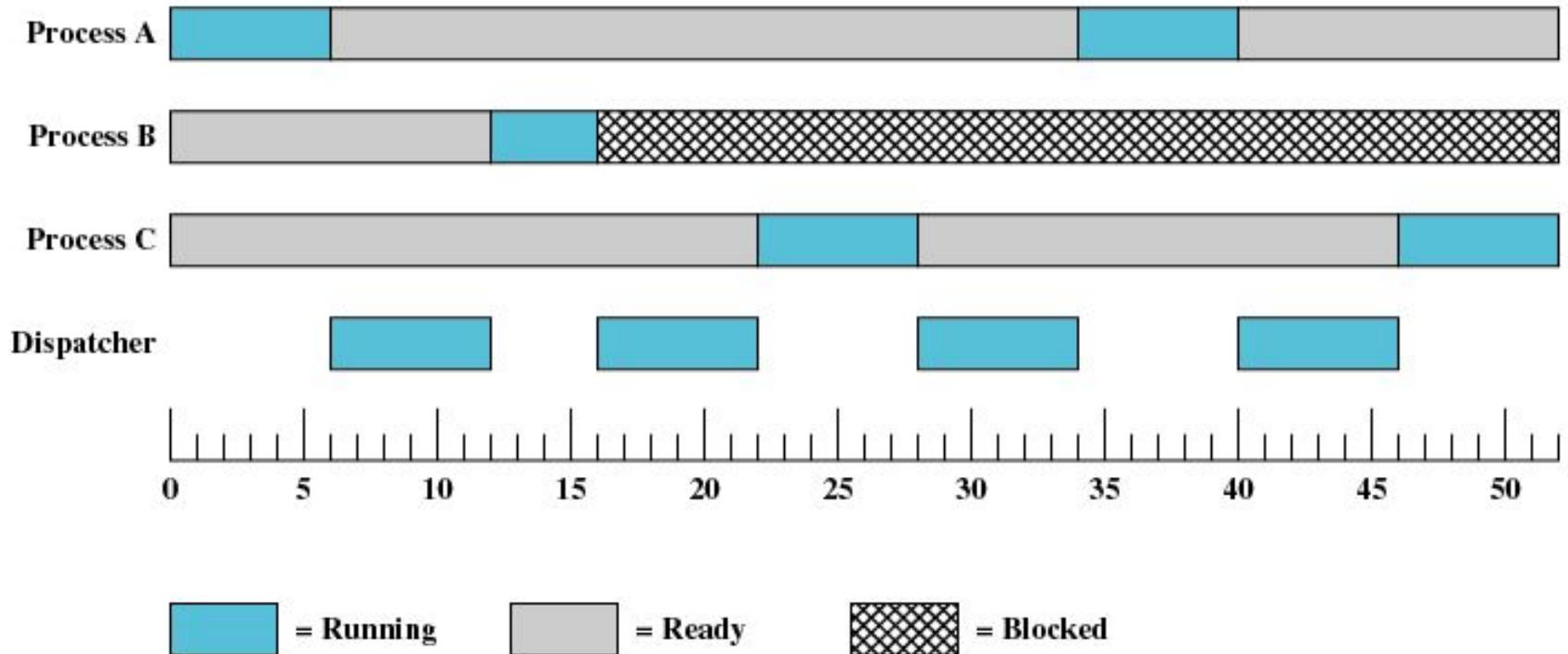
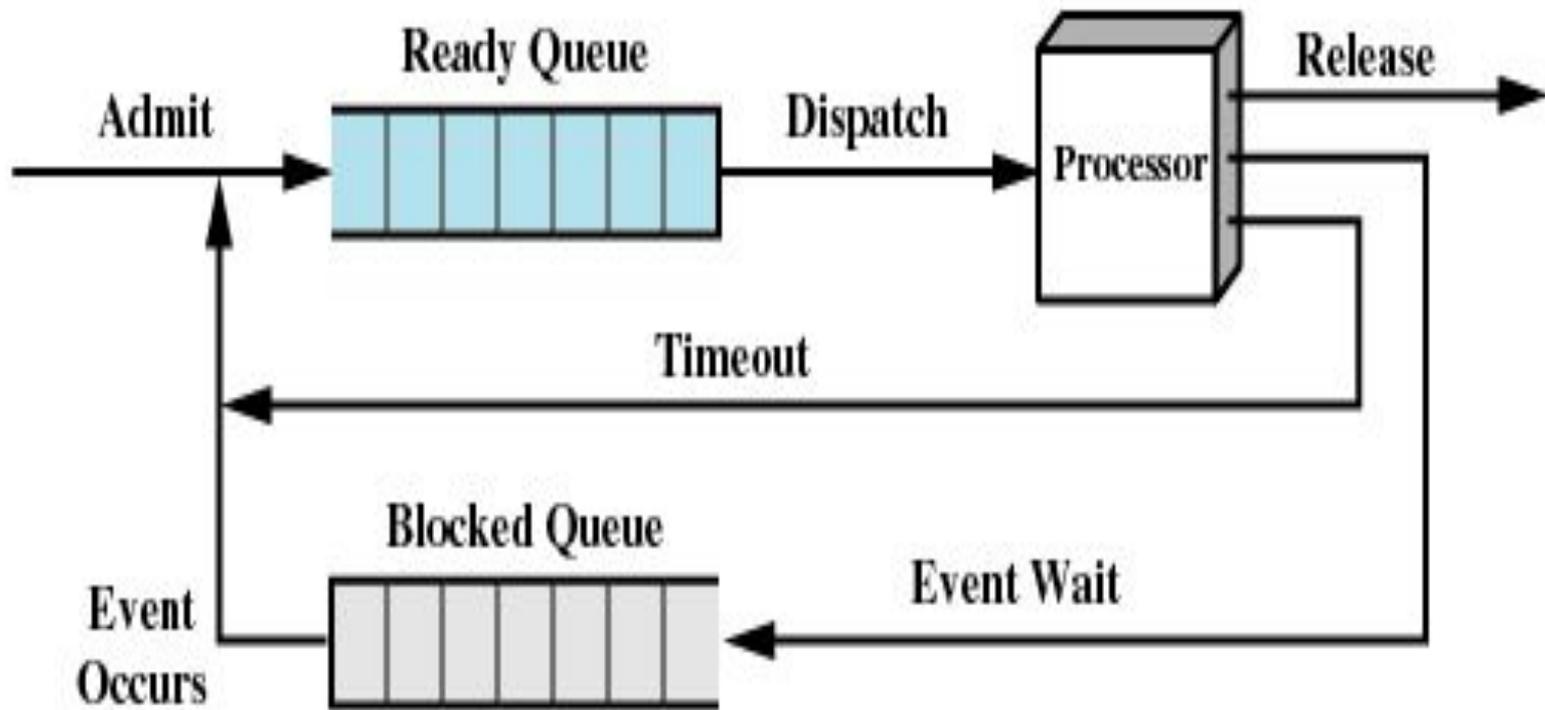


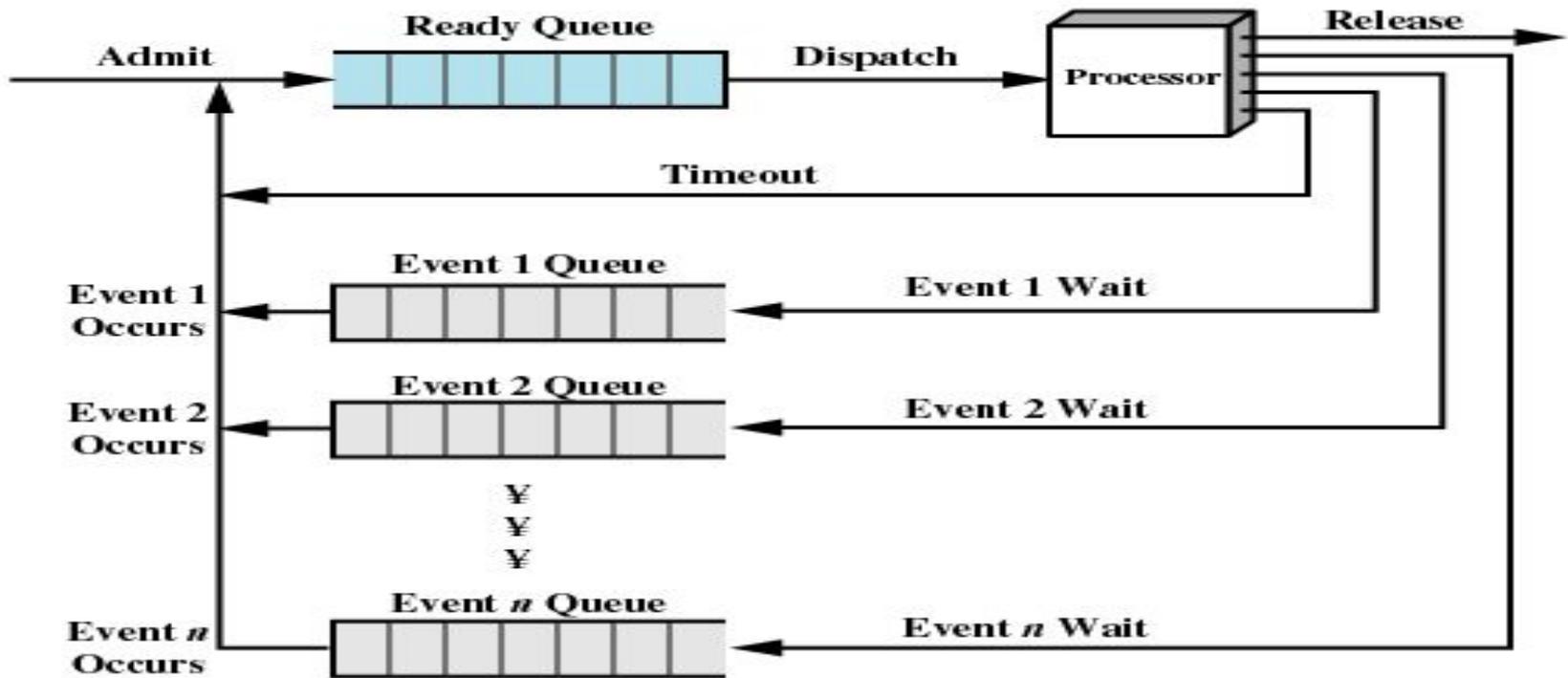
Figure 3.7 Process States for Trace of Figure 3.4

# Схема с одной очередью блокированных процессов



(a) Single blocked queue

# Схема с множеством очередей блокированных процессов



(b) Multiple blocked queues

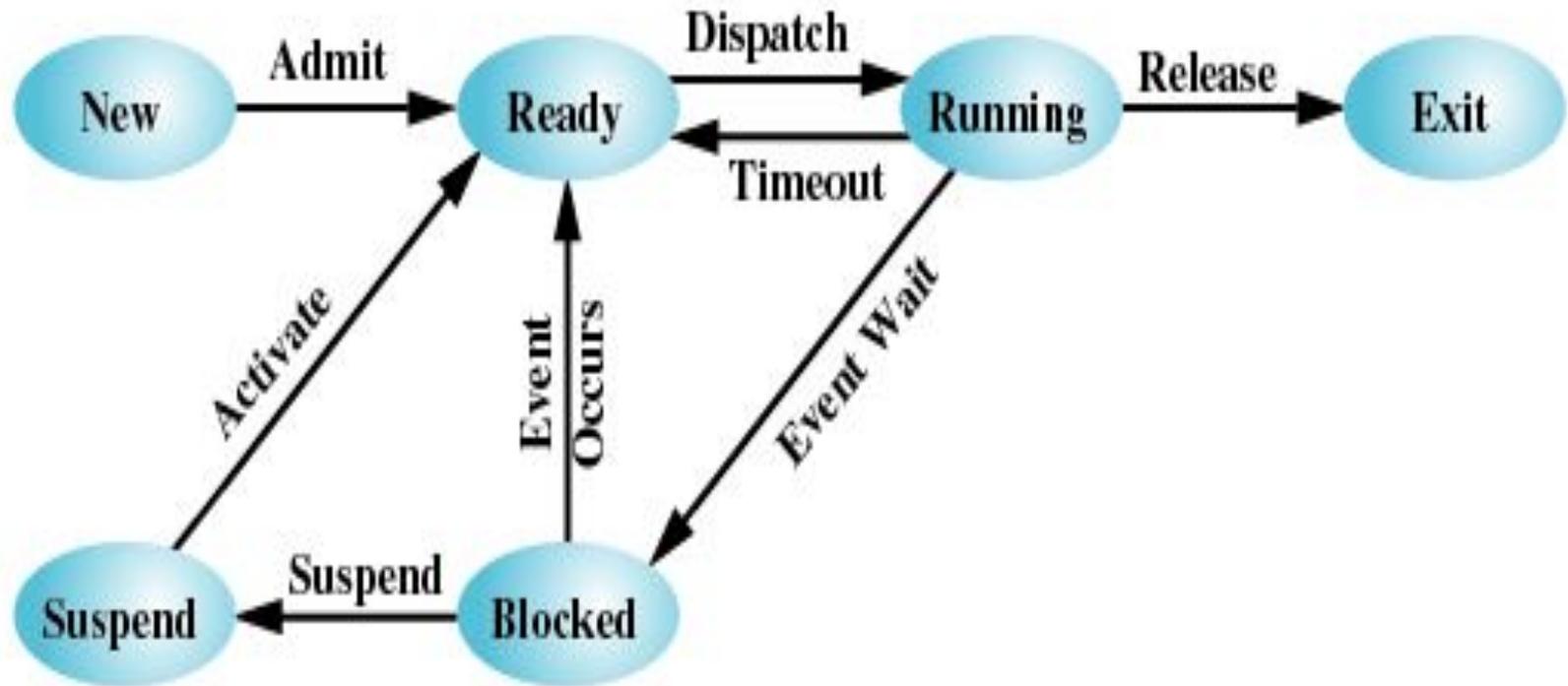
Figure 3.8 Queuing Model for Figure 3.6



# Приостановленные процессы

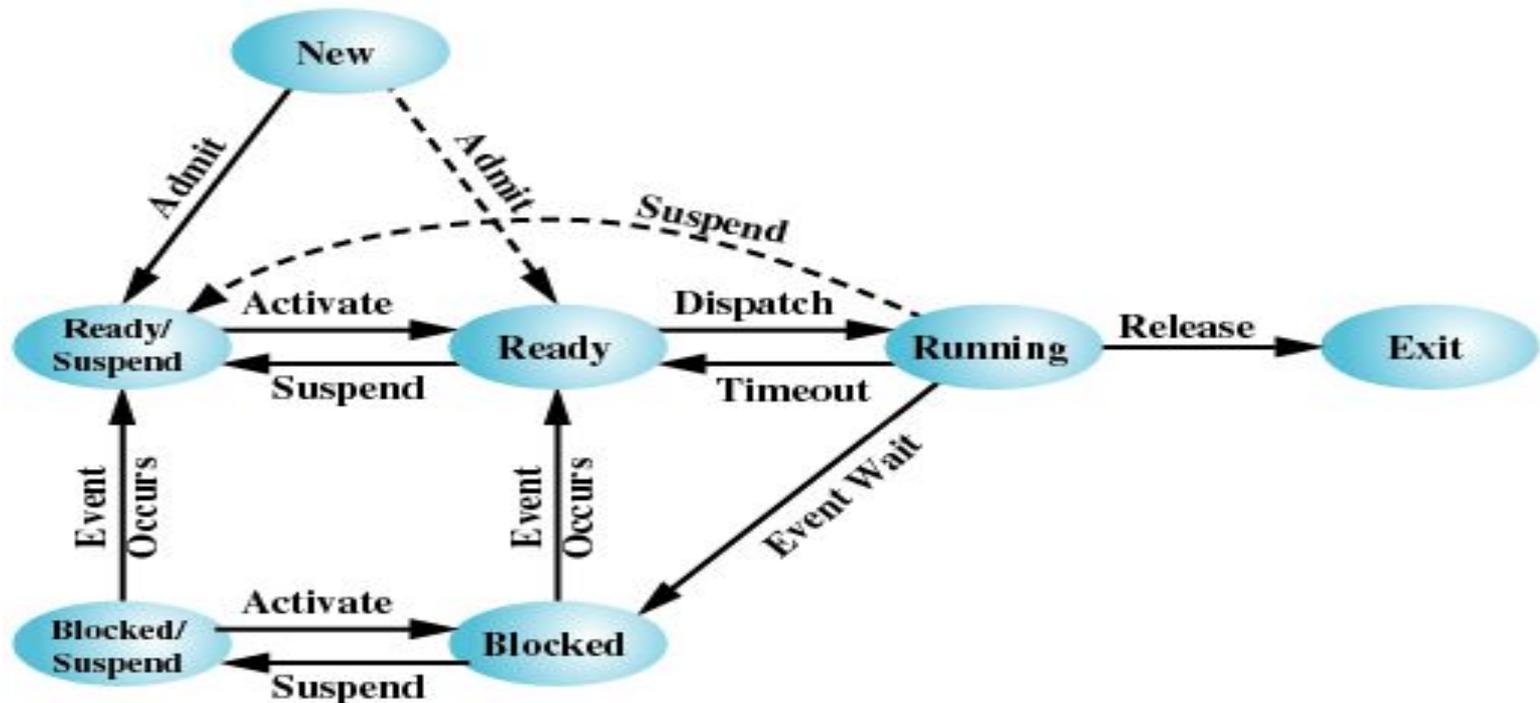
- Перемещение (Swar) процессов на диск для освобождения памяти под другие процессы
- Блокированное состояние становится приостановленным при переносе процесса на диск
- Новые состояния:
  - Блокирован/приостановлен
  - Готов/приостановлен

● ● ● | Диаграмма состояний процесса с одним приостановленным состоянием



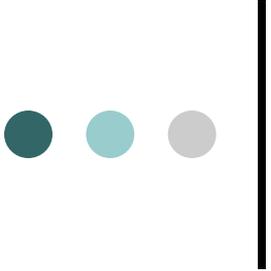
(a) With One Suspend State

● ● ● | Диаграмма состояний процесса с  
двумя приостановленным  
состоянием



(b) With Two Suspend States

**Figure 3.9 Process State Transition Diagram with Suspend States**



# API - Application Programming Interface управления процессами (POSIX)

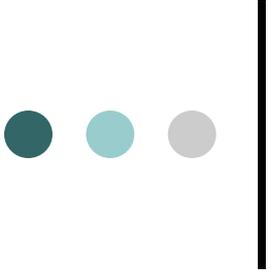
**POSIX 1003.1: fork(), exec(), kill()**

## **Клонирование процесса**

```
int pid;
switch (pid = fork())
{
case 0: /*Порождённый процесс*/
    ...
case -1: /*Ошибка создания процесса*/
default: /*Родительский процесс*/
}
```

## **Порождение нового процесса ls**

```
int pid;
switch (pid = fork())
{
case 0: /*Порождённый процесс*/
ret = execl ("/bin/ls", "ls", "-l", (char *)0); case -1: /*Ошибка создания
    процесса*/
default: /*Родительский процесс*/
}
```



# API - Application Programming Interface управления процессами (Win32 API)

Win32: `CreateProcess`, `ExitProcess`, `TerminateProcess`, `Get/SetPriorityClass`...

**Порождение нового процесса `calc.exe`**

```
STARTUPINFO si;  
PROCESS_INFORMATION pi;  
if (CreateProcess(  
    "c:\\windows\\system32\\calc.exe",  
    NULL, NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS,  
    NULL, NULL, &si, &pi))  
{  
    CloseHandle(pi.hThread);  
    CloseHandle(pi.hProcess);  
}
```