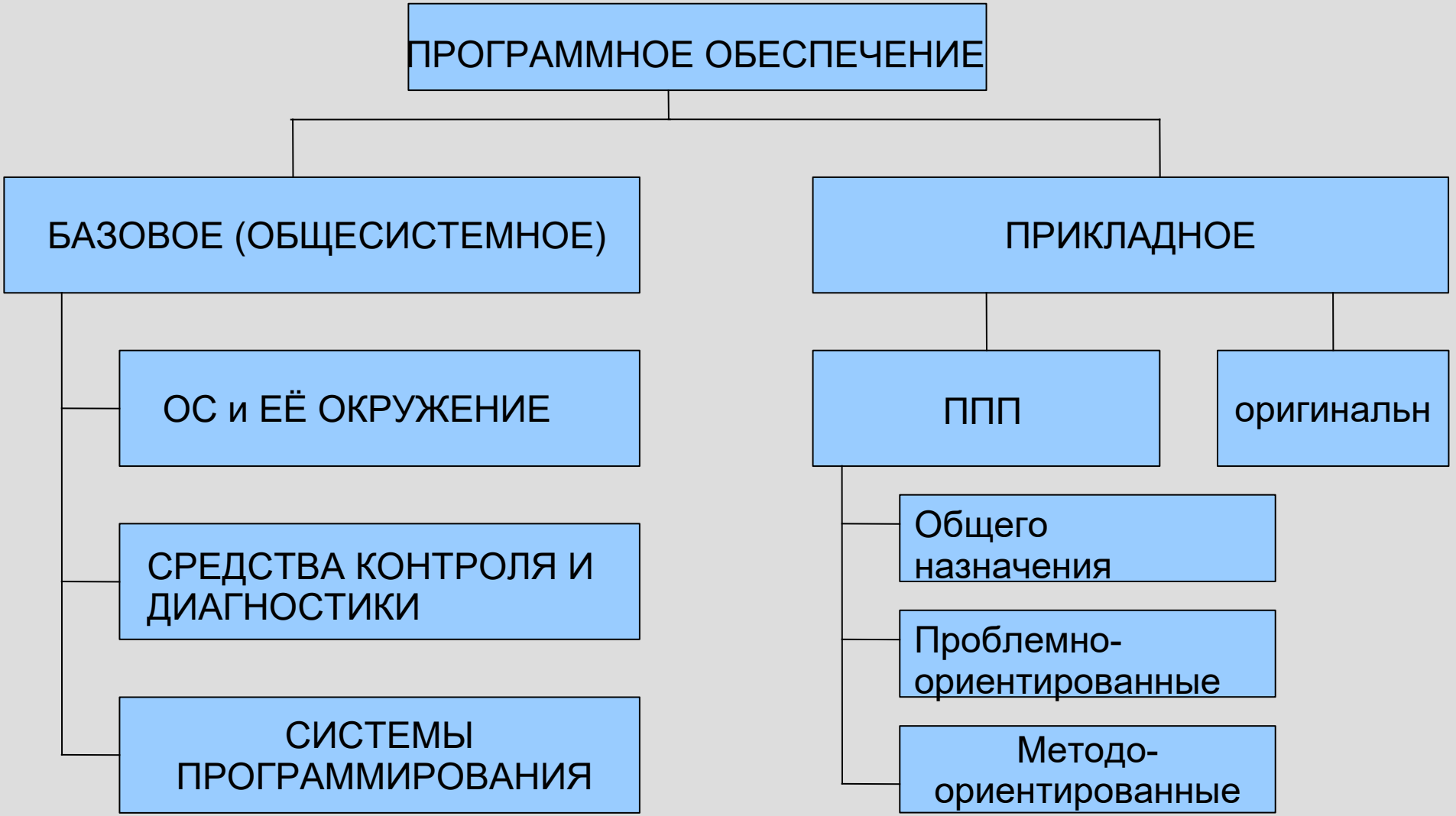
Проектирование Операционных систем
Лекция 1

Боровцов Е.Г., 2020 г.

Программное обеспечение вычислительной системы



ОС и её окружение

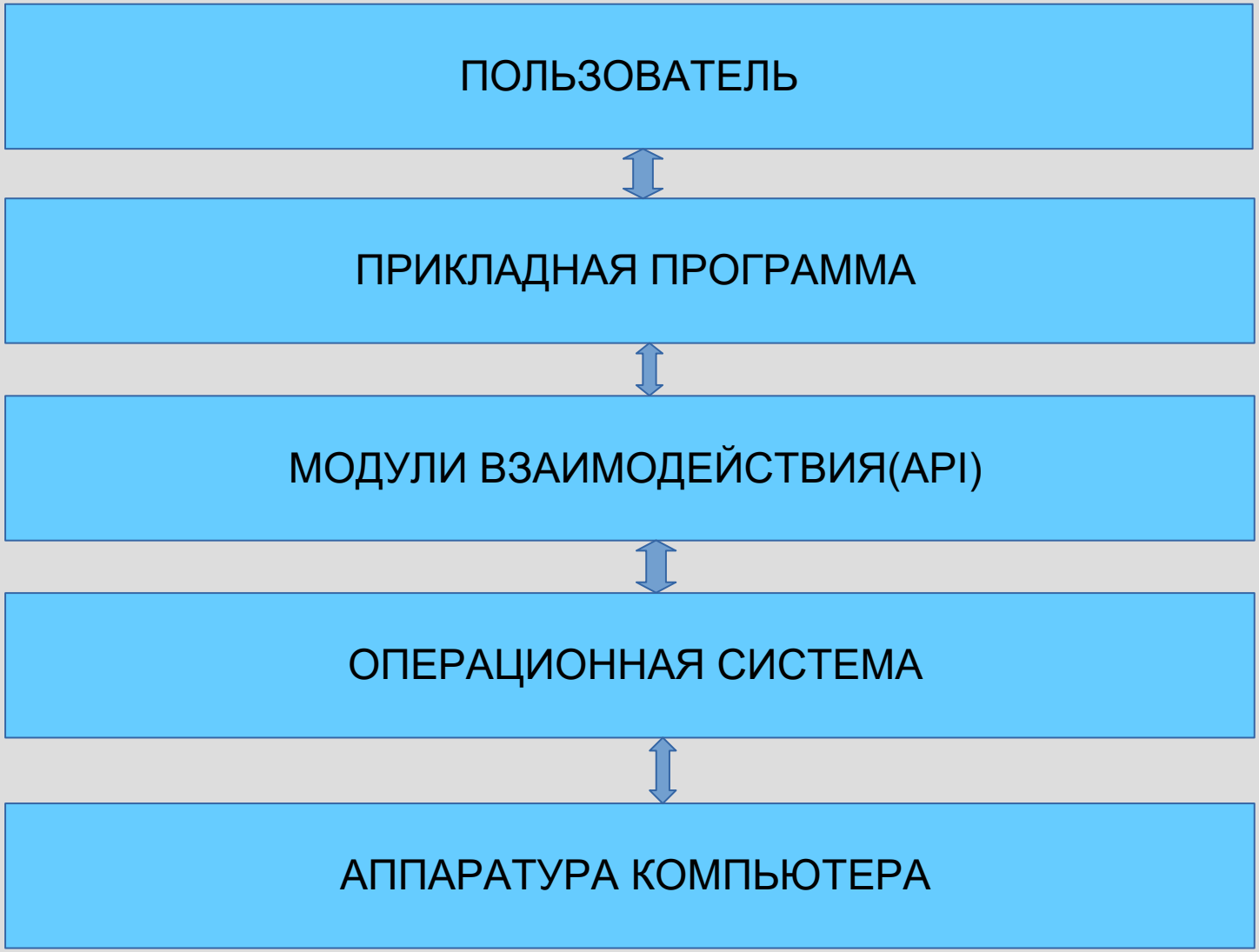
Основное назначение операционной системы состоит в эффективном управлении ресурсами системы, основными из которых являются:

- процессор (процессоры);
- память;
- устройства ввода/вывода.

Функции ОС

- обеспечивает взаимодействие между пользователем и вычислительной системой;
- обеспечивает разделение аппаратных средств между пользователями;
- планирует доступ пользователей к общим ресурсам;
- обеспечивает эффективное планирование и выполнение операций ввода/вывода;
- осуществляет восстановление информации и вычислительного процесса в случае сбоев и ошибок.

«Слои» вычислительной системы



Операционная система, как средство виртуализации

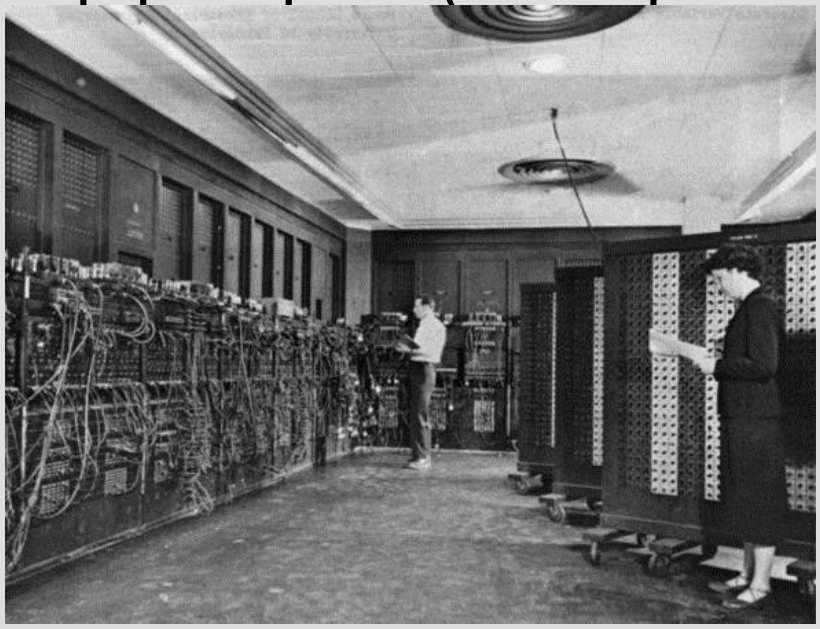
Операционная система позволяет абстрагироваться от деталей реализации и принципов функционирования аппаратуры компьютера, предоставляя пользователю (в том числе и программисту) возможность оперировать с «высокоуровневыми» упорядоченными виртуальными объектами (например, с файлами) вместо «низкоуровневых» (регистры контроллера накопителя, сектора, дорожки, цилиндры)

Операционная система, как менеджер системных ресурсов

Операционная система обеспечивает эффективное управление основными ресурсами вычислительной системы (время процессора, оперативная память, устройства долговременного хранения информации, устройства ввода-вывода), используя механизмы временного и пространственного мультиплексирования

Развитие операционных систем 1-е поколение (~1940-1955)

Операционных систем практически не было, так же, как и языков программирования. Программирование осуществлялось на уровне машинных кодов. Управление системой осуществлялось с пульта. Ввод программ — на коммутационной панели (перфолента — Zuze), позже — перфокарты. (На первых ПК — аналогично).



Развитие операционных систем 2-е поколение (~1955-1960)

Появление первых ассемблеров и трансляторов с языка Fortran. (К концу 60-х появились языки Algol, Cobol, Lisp и соответствующие компиляторы)
Появление библиотек ввода/вывода и библиотек стандартных программ.
Изменение принципа взаимодействия с компьютером (программисты, операторы, электроники, разработчики).
Появление систем пакетной обработки, автоматизирующих процесс прохождения задания в системе.
Появление языка управления заданиями JCL (Job Control Language), описывающего содержание решаемой задачи, её окружение и условия её выполнения.

Задача по-прежнему решается в монопольном режиме, ресурсы процессора используются неэффективно.

```
//CSC 52R56 JOB (100,1105),'10055W. RUDD'
//COMPUTE EXEC ASMFCLG
//ASM.SYSIN DD *
AVERAGE START       START OF PROGRAM
PRINT NOGENDO       NOT PRINT MACRO EXPANSIONS
INITIAL             BEGIN EXECUTION HERE
RWD 9               READ NUMBER OF NUMBERS TO USE
LR 10,9             KEEP A COPY IN REGISTER 10
L 6,=F'1'           DECREMENT FOR COUNTING
SR 5,5              START SUM AT 0
NEXTNUM RWD 2       READ A NUMBER
AR 5,2              ADD IT TO SUM
SR 9,6              DECREMENT LOOP COUNTER
BNM NEXTNUM         GO BACK FOR MORE IF NOT CONE
MR 4,6              PREPARE FOR DIVISION BY NUMBER OF
DR 4,10             NUMBERS TO COMPUTE AVERAGE
WWD 5               QUOTIENT IN REGISTER 5
WWD 4               AND REMAINDER IN REGISTER 4
EOJ                 PROCESSING FINISHEC, STOP
END AVERAGE         END OF PROGRAM
/*
//GO.SYSPRINT DO SYSOUT=A
//GO.SYSUOUMP DO SY SOUT= A
//GO. SYSIN DD *
5
-10
-30
1
3
/*
//
```

Развитие операционных систем 3-е поколение (~1960-е-1980-е)

- Появление мультизадачности (использование принципа классического мульти-программирования);
- Появления в ОС режима Spooling;
- Усиление роли механизма прерываний при организации мультипрограммного режима работы;
- Развитие параллелизма в архитектуре, использование каналов прямого доступа к памяти для организации параллельной работы ЦП и периферийных устройств;
- Появление терминальных комплексов для организации удалённого доступа и организации интерактивного режима работы;
- Появление систем, работающив в режиме разделения времени (Time Sharing System);
- Появление механизмов защиты и разделения ресурсов задач, распространение механизма виртуальной памяти и свопинга;
- Появление унифицированного семейства машин и унифицированной операционной системы (OS/360, впоследствии OS/370);
- Появление мини-ЭВМ, работающих преимущественно в интерактивном режиме, появление операционной системы UNIX;
- Появление операционных систем реального времени;
- Появление систем виртуальных машин (Sytem 370)

Развитие операционных систем 4-е поколение (~1980-е-сегодня)

- Появление операционных систем персональных компьютеров;
- Широкое внедрение графических интерфейсов;
- Появление сетевых операционных систем;
- Появление серверных операционных систем;
- Появление и развитие мультипроцессорных ОС;
- Повсеместное внедрение решений на основе виртуализации;
- Появление распределённых операционных систем;
- Возникновение и развитие мобильных и встроенных операционных систем.

Итог: Типы операционных систем по области использования

- Операционные системы мэйнфреймов;
- Серверные операционные системы;
- Персональные операционные системы;
- Встраиваемые операционные системы
- Мобильные операционные системы

Итог:Режимы работы ОС (классификация по типу взаимодействия с пользователем)

- Пакетный - формируется задание, содеращее пакет шагов, выполняющихся без взаимодействия с пользователем
- Интерактивный — постоянное взаимодействие с пользователем через графический интерфейс или интерфейс командной строки

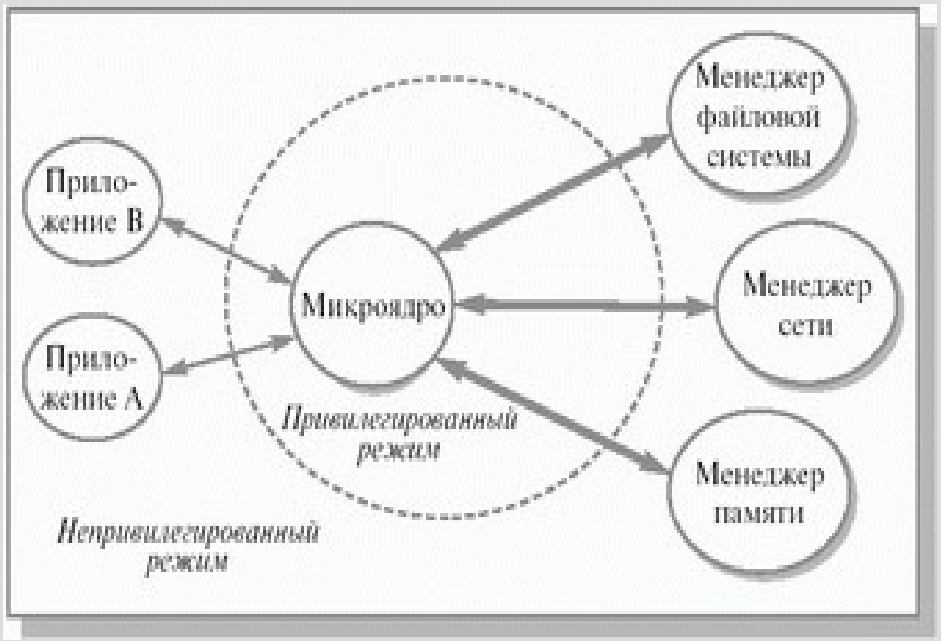
Итог:Типы ОС(по количеству пользователей и задач)

Итог:Мультизадачный режим работы ОС

- Классическое мультипрограммирование;
- Корпоративная мультизадачность (невытесняющая мультизадачность) (non-preemptive multitasking);
- Квантование по времени (time sharing) - вытесняющая мультизадачность (preemtive multitasking)
- Режим реального времени(два субрежима: жесткий реалтайм;
  мягкий реалтайм);

Архитектурные особенности ОС

- ОС с монолитным ядром;
- Микроядерные ОС;
- Гипервизоры;
- Системы с ядром смешанного типа;



Основные концепции ОС

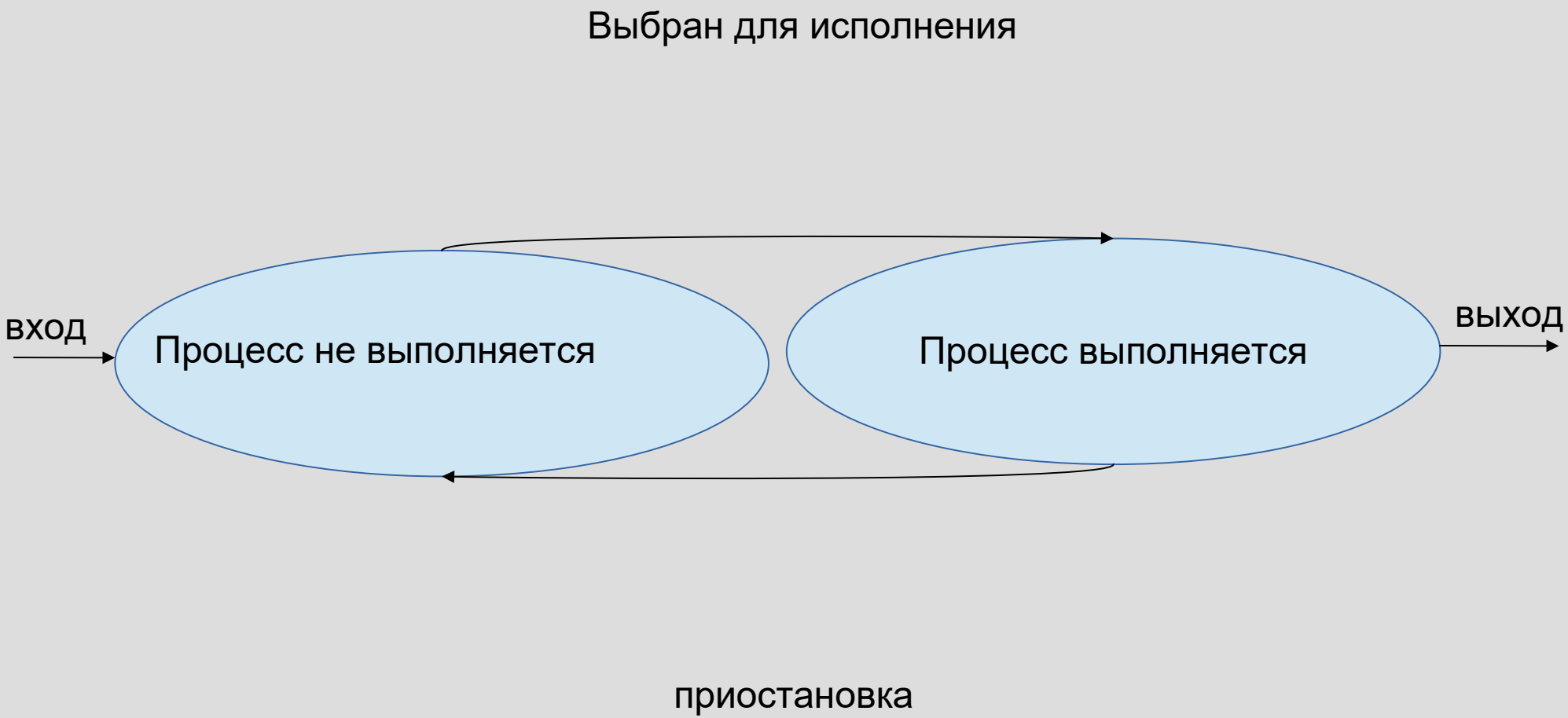
- Системные вызовы(System calls) (программные прерывания);
- Прерывания (Hardware Interrupts) (внешние аппаратные прерывания);
- Исключения (Exeptions) (внутренние аппаратные прерывания);
- Файлы и файловые системы;
- Процессы и нити;
- Блоки памяти, страничная и виртуальная память;

Понятие процесса

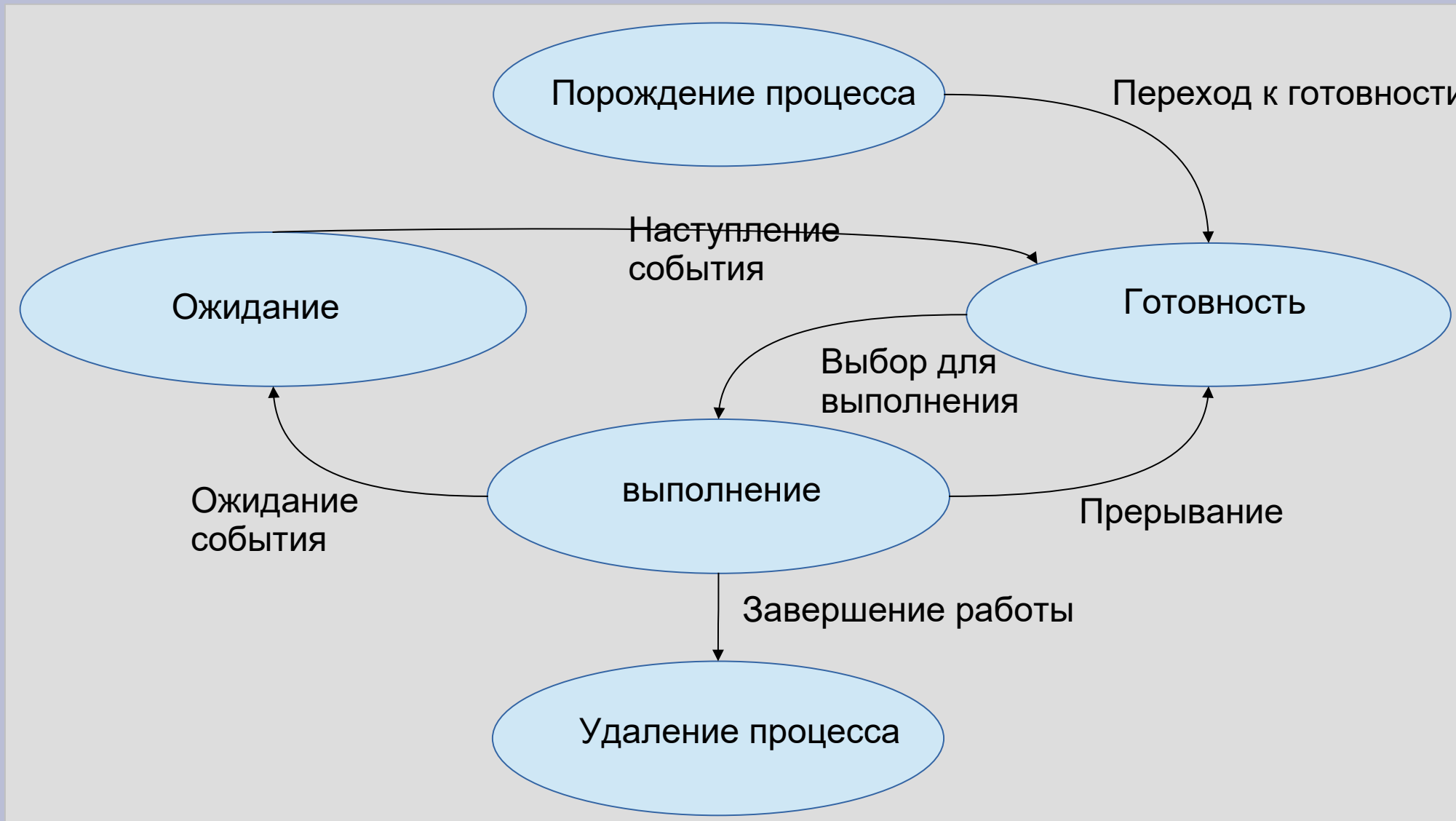
Процесс - это совокупность находящихся под управлением ОС:

- Последовательность исполняющихся команд;
- Соответствующие ресурсы (выделенная для выполнения память или адресное пространство, файлы, устройства ввода/вывода и т. д.;
- текущее состояние — состояние программного счетчика и регистра флагов, состояние регистров, стека, рабочих переменных;

Состояния процесса



Состояния процесса-2



Создание процесса

1. Инициализация системы;
2. Выполнение системного запроса на создание процесса от работающего процесса (fork - UNIX, CreateProcess — Windows API);
3. Запрос пользователя на создание процесса;
4. Инициирование пакетного задания (bat-файла, shell-скрипта)

Создание процесса - fork()

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
pid_t ChildPID;

printf("\n\r **** Begin Process ****");
ChildPID=fork();
if(ChildPID< 0)  {
  printf("\n\r **** Abnormal fork termination ****");
  return 1;
}
if(ChildPID==0){
    // Child Process
    printf("\n\r **** Child Process PID **** %d \n\r",getpid());
} else {
    // Parent process
    printf("\n\r **** Parent Process PID **** %d \n\r",getpid());
    wait(NULL);
}
return 0;
}
```

Создание процесса - fork()+execve

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

extern char **environ;

int main(void) {
pid_t ChildPID;
char * ls_args[]={
       "ls","-l","/",NULL
};

ChildPID=fork();
if(ChildPID< 0)  {
  fprintf(stderr,"\n\r **** Abnormal fork termination ****");
  return 1;
}
if(ChildPID==0){
    // Child Process
    execve("/bin/ls",ls_args,environ);
    fprintf(stderr,"\n\r!!!!!! EXEC ERROR !!!!!\n");
    return 1;
}

return 0;
}
```

Завершение процесса

1. Нормальный выход («код возврата»=0, преднамеренно);
2. Выход по ошибке(«код возврата»> 0, преднамеренно);
3. Выход по неисправимой ошибке (непреднамеренно);
4. Уничтожение другим процессом (непреднамеренно)

Переходы в состояния «готовность», «выполнение», «ожидание»

Порождение процесса — готовность:
после порождения процесса тем или иным способом, выделения процессу ресурсов, создания и заполнения блока управления процессом;

Готовность — выполнение:
выбор процесса и очереди готовых к выполнению процессов в соответствии с используемым в системе алгоритмом планирования; *Осуществляется восстановление контекста;*

Выполнение — готовность:
аппаратное прерывание для обработки внешнего события; *Осуществляется сохранение контекста процесса;*

Выполнение — ожидание(блокировка):
Системный вызов (запрос доступа к ресурсам); *Осуществляется сохранение контекста процесса;*

Ожидание — готовность:
доступность ожидаемых данных или ресурсов.

Таблица процессов и содержимое блока управления процессом

Управление процессом

- счётчик команд IP;
- регистры;
- psw(регистр флагов);
- указатель стека SP;

- текущее сост. процесса;
- значение приоритета;
- параметры планирования;
- PID;
- PPID;
- сигналы;
- GID процесса;
- время начала процесса;
- использованное CPU Time;
- Child CPU Time;

Управление памятью

- указатель на сегмент кода;
- указатель на сегмент данных;
- указатель на сегмент стека;

Управление файлами

- корневой каталог;
- рабочий каталог;
- дескрипторы файлов;
- UID;
- GID;

Потоки(нити)

В отличие от процессов потоки(нити) работают в контексте одного процесса-родителя

Совместно используемые Элементы процесса	**элементы потока**
Адресное пространство	Счетчик команд
Глобальные переменные	Регистры CPU
Открытые файлы	Стек
Дочерние процессы	Состояние
Необработанные аварийные сигналы	
Сигналы и их обработчики	
Информация об использовании ресурсов	

Реализация потоков:

- В пространстве ядра
- В пространстве пользователя

Потоки в пространстве ядра

Достоинства:

- возможно планирование работы нескольких потоков одного и того же процесса на нескольких процессорах (ядрах);
- реализуется мультипрограммирование в рамках всех процессов;
- при блокировании одного из потоков процесса ядро может выбрать для исполнения другой поток этого же процесса;
- процессы ядра сами могут быть многопоточными;

Недостатки:

необходимость двукратного переключения режима пользователь-ядро, ядро-пользователь для передачи управления от одного потока другому в рамках одного и того же процесса.

Потоки в пространстве пользователя

Достоинства:
- можно реализовать «собственными » средствами в ОС, не поддерживающей потоки без каких-либо изменений в ядре ОС;
- высокая производительность, поскольку при переключении потоков процессу не надо переключаться в режим ядра и обратно;
- ядро о потоках ничего не знает и управляет однопоточными процессами;
- имеется возможность применять любые алгоритмы планирования исполнения потоков с учетом специфики решаемой задачи;
- правление потоками возлагается на программу пользователя;

Недостатки:
- системный вызов блокирует не только работающий поток, но и все потоки того процесса, которому он принадлежит;
- приложение не может работать в многопроцессорном режиме, так как ядро выделяет каждому процессу один процессор;
- при запуске одного потока другой поток в рамках одного процесса не будет запущен, пока первый добровольно не освободит ресурсы;
- внутри одного потока нет прерываний по таймеру, в результате чего невозможно создать в рамках отдельного процесса собственный планировщик по таймеру для поочередного выполнения потоков.