Типы адресов

Символьные имена /
Символьные метки

Виртуальные адреса

Физические адреса

Используются при написании программ

Назначает компилятор при «сборке» программы

Формируются при загрузке программы в память либо в момент обращения процесса к ячейкам памяти

Методы распределения памяти

Без использования внешней памяти

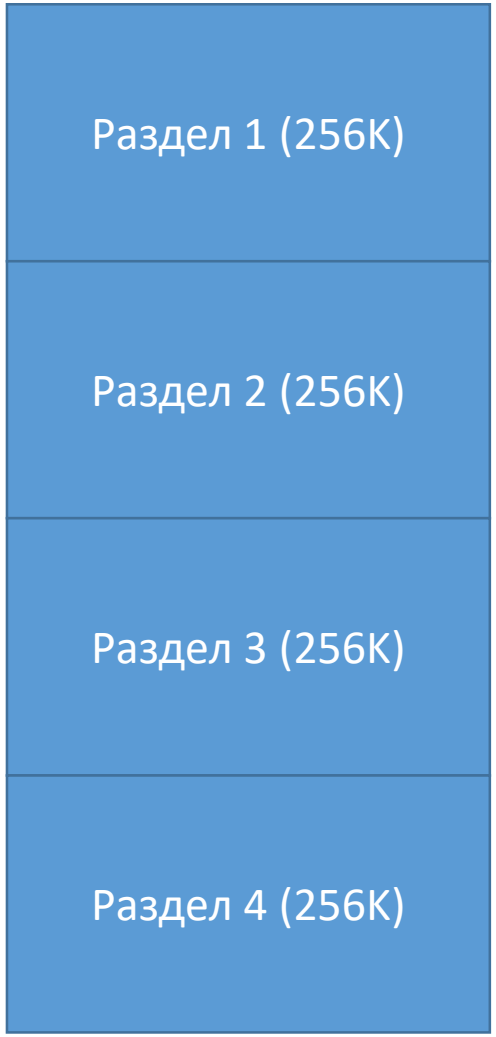
Можно запустить только те приложения которые умещаются в ОЗУ

С использования внешней памяти

+ больше объем памяти под запуск приложений

- Скорость – на доступ к «выгруженной на диск» информации тратится много времени

Фиксированными разделами



Динамическими разделами

Начало обработки

Спустя некоторое время

Перемещаемые разделы

Начало обработки

Спустя некоторое время

После уплотнения

С использованием внешней памяти («внешнего» накопителя)

Свопинг (swapping)

Выгрузка процесса целиком

- Невозможность запуска «больших» программ

Виртуальная память/листание по требованию

Частичная выгрузка

+ возможность продолжения работы приложения

Программное решение - Overlay (оверлеи) – разбиение процессов на части с последующей загрузкой-выгрузкой, но это НЕ метод распределения памяти

Способы организации работы памяти:

Страничная – блоки фикс.размера

Сегментная – разделы по типу информации

Странично-сегментная - комбинация

Преобразование адреса из виртуального в физический

Виртуальный адрес

16 бит

16 бит

Вирт. стр. n	Смещение s

Таблица страниц процесса

N вирт. стр.	N физич. стр.
n	p

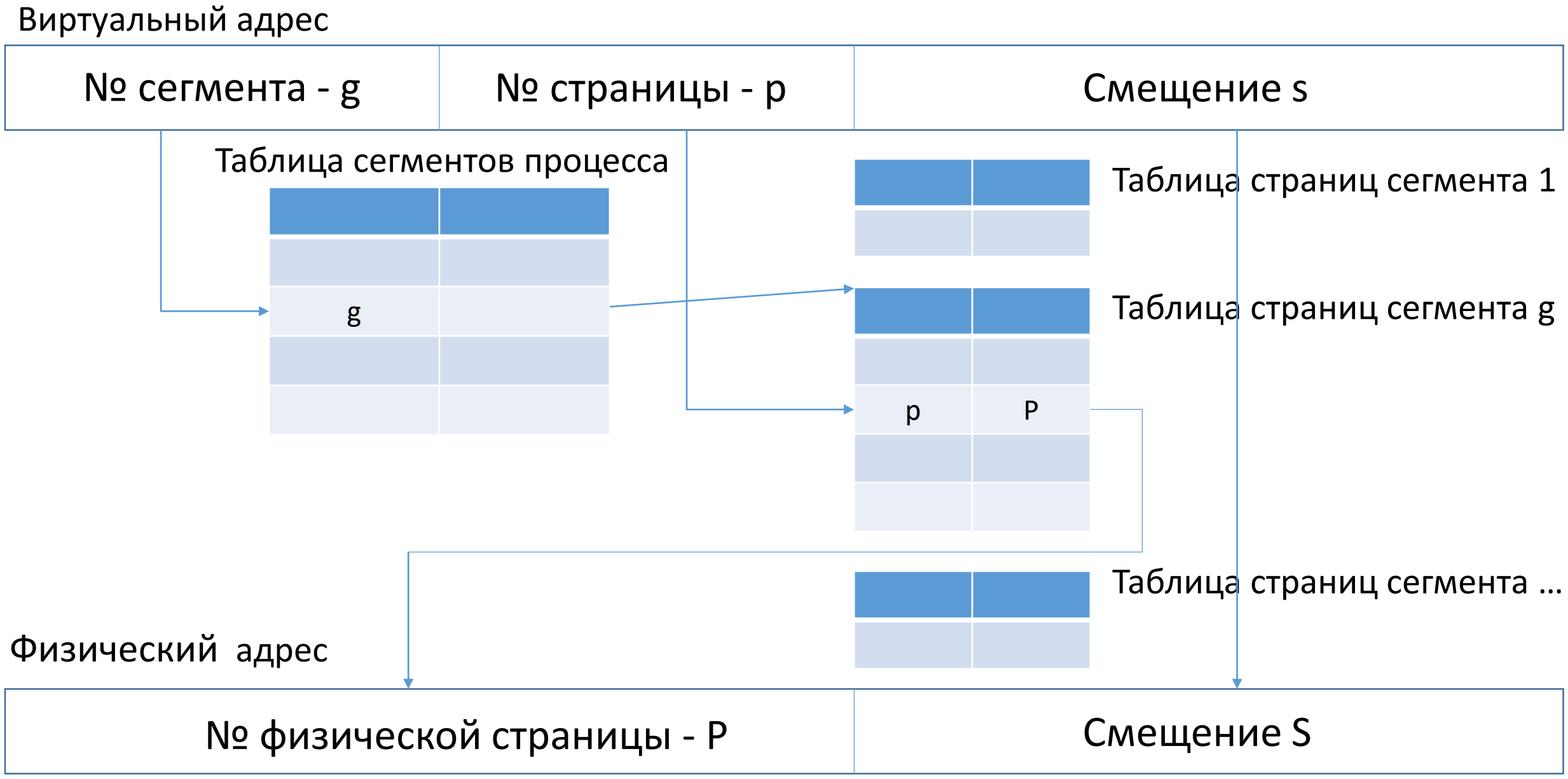
Страничная
+ простой в реализации
+ высокая скорость
- Нет деления страниц по типу содерж.информации

Сегментная
+ деление страниц по типу содерж.информации
- Низкая скорость (отн. Страничной)
- Выгрузка по-сегментная

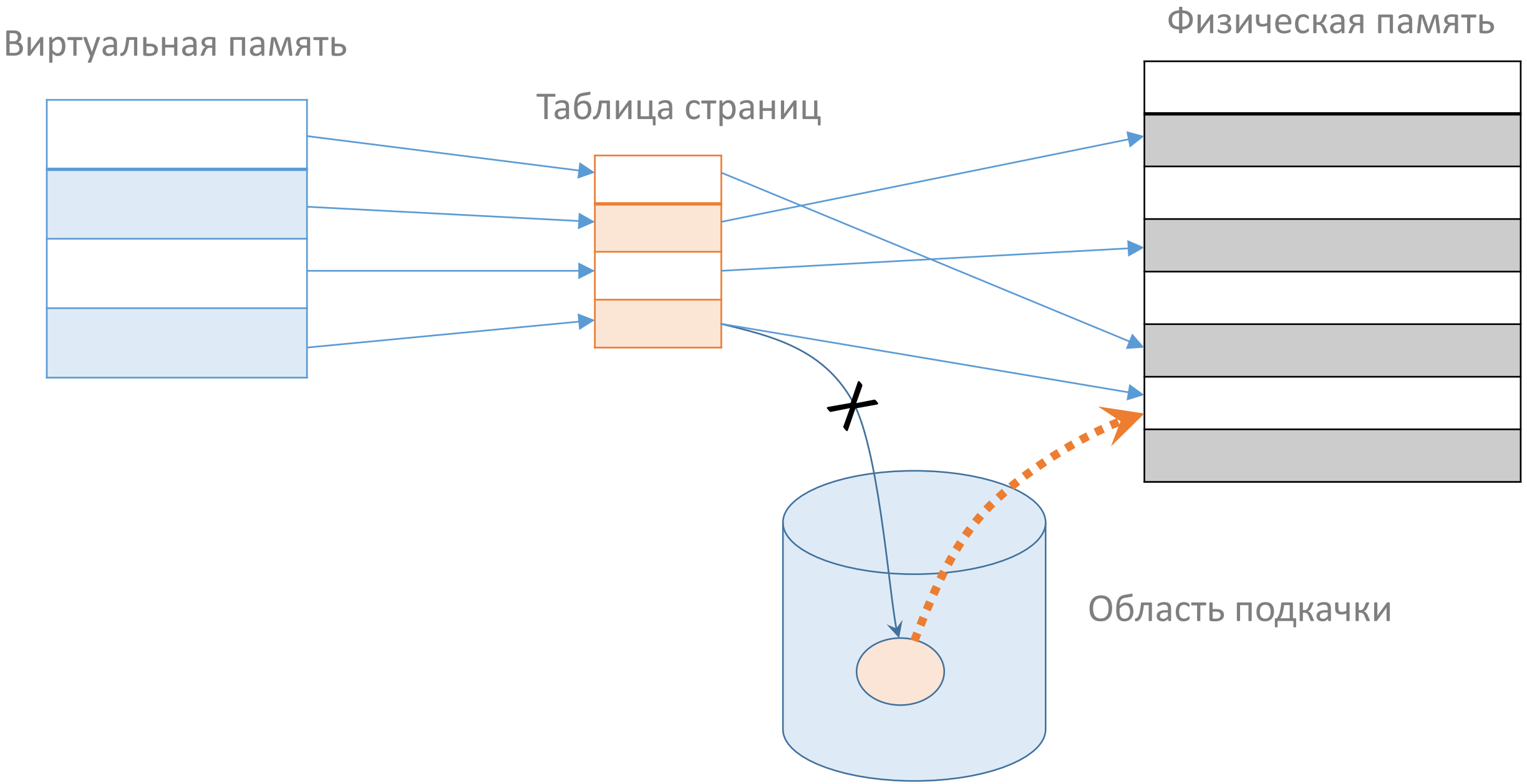
Физический адрес

Физич. стр. p	Смещение S

Преобразование адреса из виртуального в физический



Листание по требованию (demand paging)



Алгоритмы подкачки

- Локальные
- Глобальные

Алгоритмы подкачки

- FIFO first in first out – выгрузка страницы дольше всего находящейся в ОЗУ
- LRU (Least Recently Used) – выгрузка наименее используемых приложениями страниц

  Высокие накладные расходы на поддержание структур для учета использования страниц памяти
  Не учитываются ситуации когда обращение к страницеХ регулярное, но реже чем заложено в алгоритме

Trashing – проблема при значительной нехватке ОЗУ для программ

T1 = Нужна стр1 процесса1
Выгрузка стр1 процесса2,
загрузка стр1 процесса1

T2 = нужна стр1 процесса2
Выгрузка стр2 процесса1
Загрузка стр 1 процесса2

T3 = нужна стр2 процесса1
Выгрузка стр2 процесса2
Загрузка стр 2 процесса1

Результат – интенсивный обмен между ОЗУ и внешней памятью, т.к. процессы отбирают память друг у друга. Скорость «вычислений» критически низкая.

Рабочий набор – постоянно используемые программами страницы памяти. Выгрузка страниц входящих в рабочий набор из ОЗУ запрещена (возникнет Trashing)

Out of memory killer OOMKiller (linux)

При исчерпании оперативной памяти + области подкачки – аварийное завершение процессов

Организация работы с памятью в ОС Unix

Абстрактная виртуальная память

1. Сегмент программного кода [только для чтения,разделяемый]
2. Сегмент данных [чтение-запись, частный]
3. Сегмент стека [чтение-запись, частный]
4. Разделяемые сегменты [чтение-запись, частным]
5. Mapped files

Одна из реализаций работы с областью подкачки

Pagedaemon - выгрузка постраничная, (например начинает работать когда 40% осталось, по достижения 50% останов)

Swapper – выгрузка процессов целиком (например на 10% начинает работать, а на 20% останов)

Выгруженные swapper приложения через некоторое время возвращаются в ОЗУ и продолжат свое выполнение

Копирование при записи (copy on write)