Параллельное программирование

Т. П. ГРЫЗЛОВА

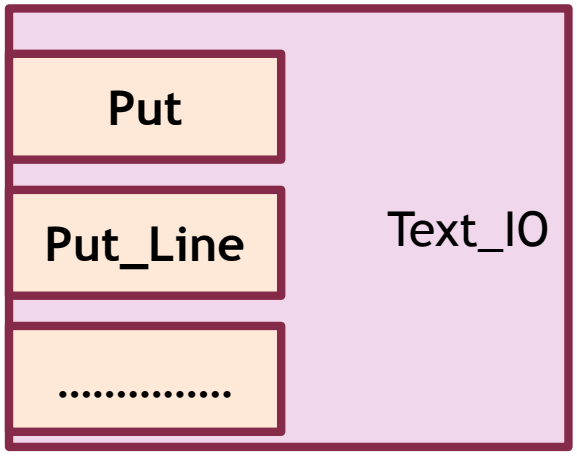
ПРОЕКТИРОВАНИЕ ПРОГРАММ НА ADA

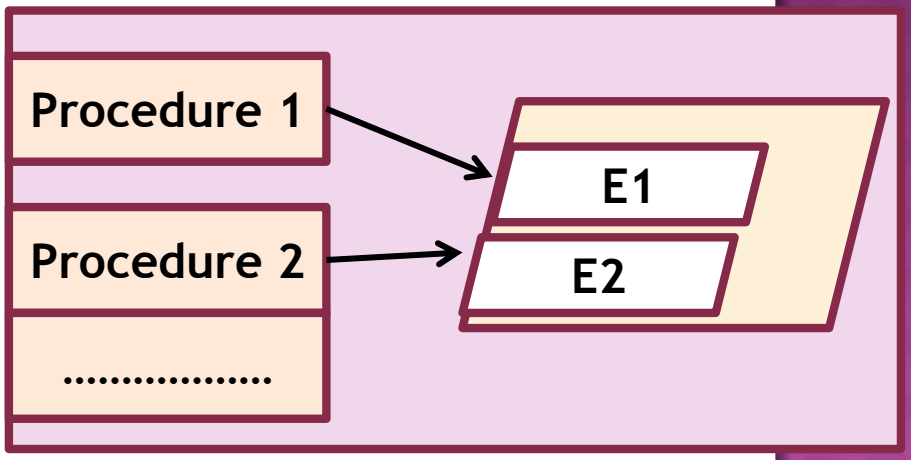
РГАТА им. П. А. Соловьева

ПАКЕТЫ

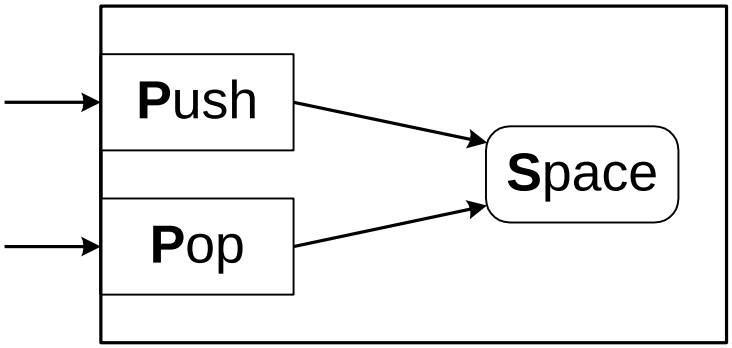
***.ads**

- Активные и пассивные пакеты
- Пассивные пакеты не содержат встроенных задач





ПАССИВНЫЙ ПАКЕТ «СТЕК»

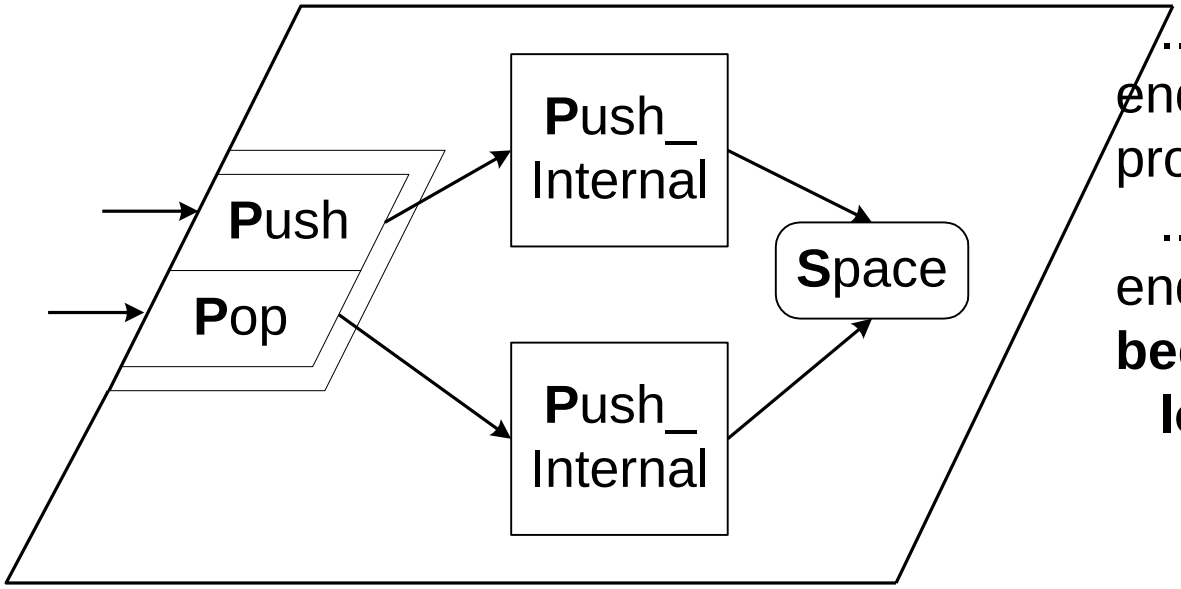



```
package Stack is
   procedure Push (...);
   procedure Pop (...);
end Stack;
```

```
package body Stack is
   .....
   procedure Push (...) is
      .....
   end Push;
   procedure Pop (...) is
      .....
   end Pop;
end Stack;
```

Если процедуры пакета обладают свойством повторной входимости; **е**сли в нем нет совместно используемых внутренних данных; **и**ли пакет обеспечивает только чтение таких данных, то **пакет можно использовать несколькими задачами.**

Эквивалентная задача «Стек»

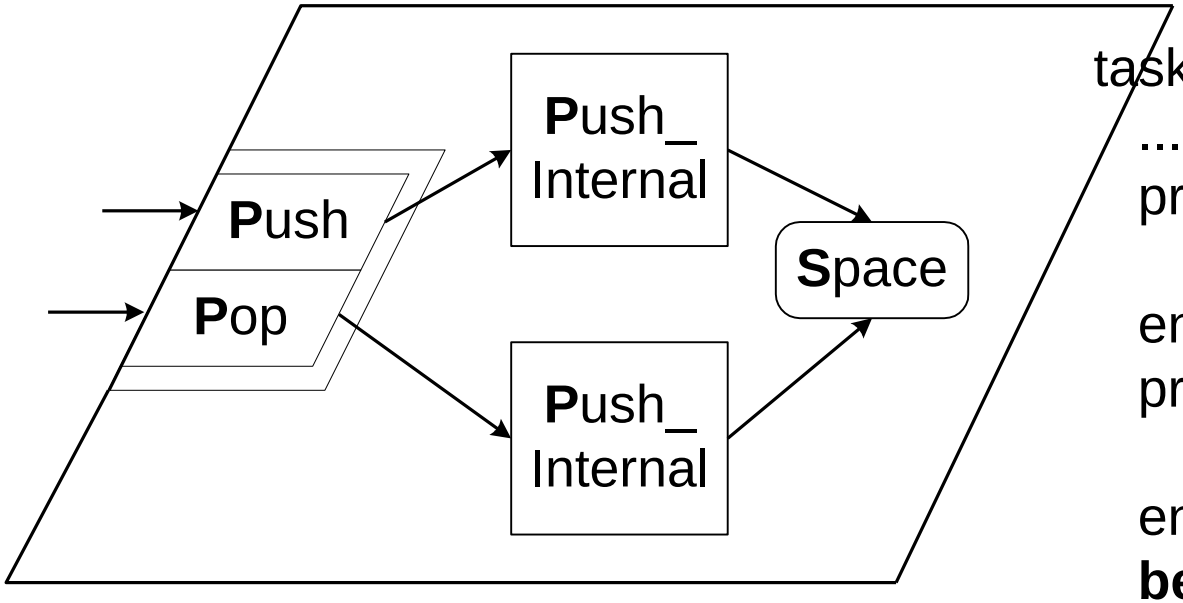



```
task Stack is
   entry Push (...);
   entry Pop (...);
end Stack;
```

Координация - замена пакета задачей - исполнителем

Единственная задача - исполнитель
Интуитивная очевидность
Высокая надежность

```
task body Stack is
   .....
   procedure Push_Internal (...) is
      .....
   end Push_Internal;
   procedure Pop_Internal (...) is
      .....
   end Pop_Internal;
   begin
      loop
         select
            accept  Push  (...) do
               Push_Internal;
   end;
         or
            accept  Push  (...) do
               Pop_Internal;
   end;
         end select;
      end loop;
   end Stack;
```



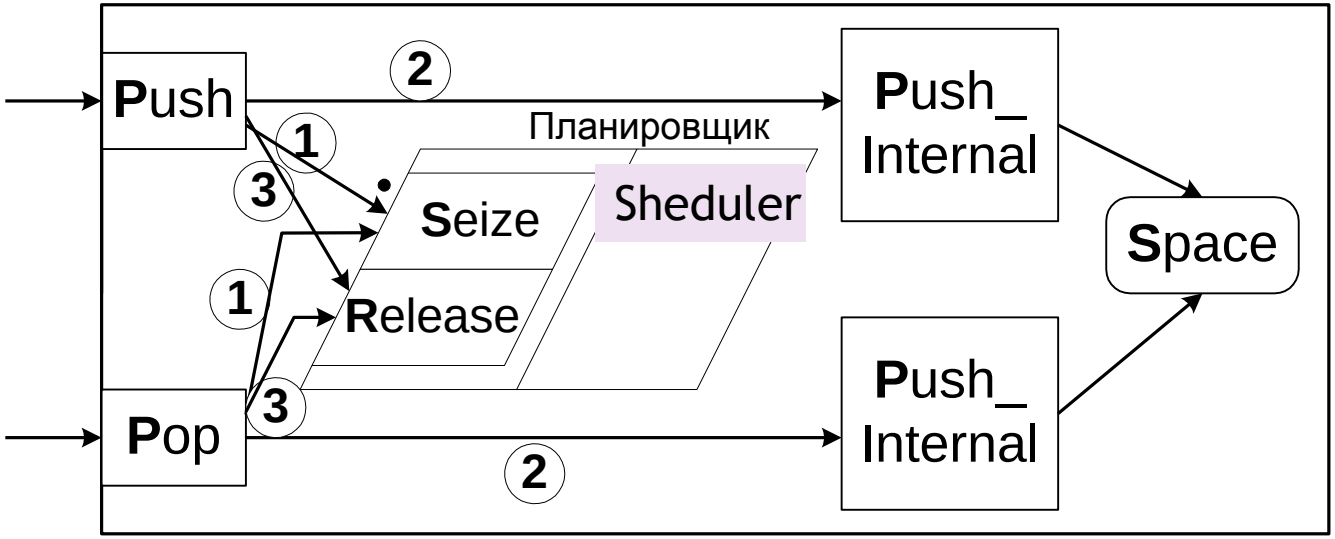

```
task Stack is
   entry Push (...);
   entry Pop (...);
end Stack;
```

Координация – замена пакета задачей - исполнителем

Единственная задача - исполнитель
Интуитивная очевидность
Высокая надежность

```
task body Stack is
   .....
   procedure Push_Internal (...) is
      .....
   end Push_Internal;
   procedure Pop_Internal (...) is
      .....
   end Pop_Internal;
   begin
     loop
          select
               accept  Push  (...) do
                  Push_Internal;
end;
          or
               accept  Push  (...) do
                    Pop_Internal;
end;
          end select;
     end loop;
end Stack;
```

Эквивалентный активный пакет «Стек»



Сохраняется неизменным внешний вид пакета

```
package body Stack is
   ..............................
   task Sheduler is
      entry Seize (...);
      entry Release (...);
   end Stack;
   task body Sheduler is separate;
```

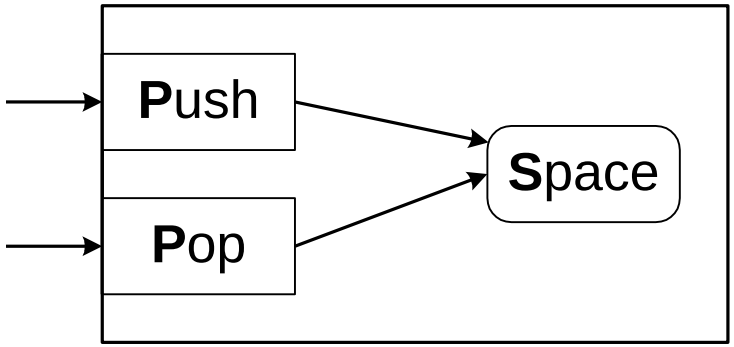
ЭКВИВАЛЕНТНЫЙ АКТИВНЫЙ ПАКЕТ «СТЕК»

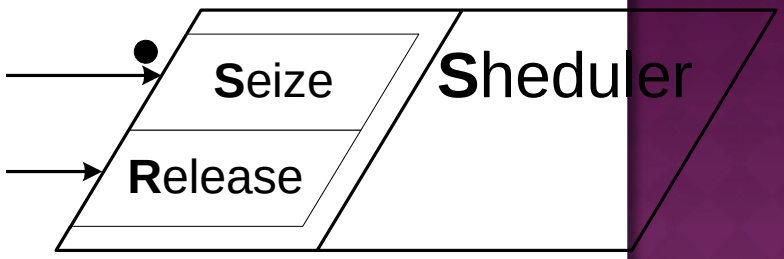
```
procedure Push_Internal (...) is
   ..............................
end Push_Internal;
procedure Pop_Internal (...) is
   ..............................
end Pop_Internal;
procedure Push is
   begin
         Sheduler.Seize;
         Push_Internal (...);
         Sheduler.Release;
end Push;
procedure Pop is
    begin
         Sheduler.Seize;
         Pop_Internal (...);
         Sheduler.Release;
   end Pop;
end Stack;
```

Задача - планировщик

```
separate (Stack)
 task body Sheduler is
   Busy : boolean := FALSE;
   begin
     loop
          select
               when not Busy =>
               accept  Seize  (...) do
                  Busy := TRUE;
               end;
          or
               accept  Release do
                    Busy := FALSE;
               end;
          end select;
     end loop;
end Sheduler;
```

ВВЕДЕНИЕ ЗАДАЧИ-ПЛАНИРОВЩИКА





- Исходный пакет остается без изменений
- Потребители синхронизируют работу сами, используя задачу - планировщик
- Пользователю предоставлен излишне сложный интерфейс
- Потенциально ненадежно, так как пакет может быть использован задачами, которым не разрешен к нему доступ

❖ **П**одчиненная задача взаимодействует с одной, и только одной задачей, получая от нее работу.

❖ Исполнитель выполняет функции по требованию ряда задач-пользователей

❖ **П**ланировщик в привилегированных режимах задерживает выполнение вызовов на конкретных входах

❖ **Б**уфер – комбинация планировщика и исполнителя

❖ **С**екретарь имеет большую автономность, чем буфер – она может обращаться за результатами работы к другим задачам

❖ **А**гент – автономная задача, которая выполняет функции исполнителя, планировщика, секретаря и выполняет собственную работу.

❖ **К**урьер – задача, транспортирующая объекты между задачами
❖ **П**ользователи– автономные задачи, взаимодействующие с другими задачами для выполнения общесистемных функций

Рекомендации по реализации нисходящей управляющей структуры

- Создавать каждый уровень модели с использованием активного пакета.

- Реализовать каждый объект внутри уровня с помощью видимого активного пакета, видимого извне.

-

Отделение тела

В случае раздельной компиляции подпрограммы она «разрывается» на две части. В том месте, где она должна размещаться, задается след тела подпрограммы. Он записывается с помощью зарезервированного слова **Seperate, которое указывается после спецификации подпрограммы за словом is.** Раздельно компилируемый модуль, заданный следом тела называется субмодулем. Субмодули относятся к вторичным, небиблиотечным модулям. Порядок компиляции: после того модуля, где указан след.

Ссылочные типы

Ссылочное значение нельзя использовать для доступа к произвольному адресу оперативной памяти

Более того, ссылочное значение нельзя рассматривать как целое число и выполнять с ним операции.

TYPE *имя ссылочного типа* **IS ACCESS** *указание подтипа*

TYPE *имя ссылочного типа* **IS ACCESS** *указание подтипа ограничение;*

Указание подтипа определяет подтип значений, на которые можно ссылаться с помощью объектов данного ссылочного типа.

Генератор создает динамический объект и ссылки на объект

TYPE *имя ссылочного типа* **IS ACCESS** *указание подтипа*

NEW *имя типа или подтипа*

Чтобы обратиться к значению, на которое ссылается ссылочный объект

Имя объекта ссылочного типа.**ALL**

НАСТРАИВАЕМЫЕ МОДУЛИ

Если процедуры различаются только типом параметров, то можно разбить подпрограмму на две части – тело, для получения обычных, ненастраиваемых тел программ и настройка
Описание настройки задает настраиваемый модуль, предваряемый словом GENERIC. Далее следует раздел описания формальных параметров настройки. В качестве параметра настраиваемой процедуры может задаваться некоторый тип.

Совмещение

Один идентификатор используют для подпрограмм одного типа, различающиеся только параметрами и типами.
Подпрограммы могут быть совмещены только в случаях, когда они имеют разные профили типа параметров и результатов. Профиль типа параметров и результата характеризует базовые типы параметров подпрограмм в том порядке, в котором они описаны.
Правила, которыми пользуются при решении вопроса о совместимости:
Процедуры и функции всегда имеют разные профили
Подпрограммы с различным числом формальных параметров всегда имеют различные профили
На профили не влияют:
А) имена формальных параметров
Б) Вид параметра (IN, OUT, IN OUT)
В) Наличие или отсутствие значения по умолчанию
Г) Подтип параметра (существенным является лишь базовый тип).