

Лабораторные работы №6-8

Цель работы

Целью является приобретение практических навыков в:

- Управлении серверами сообщений (№6)
- Применение отложенных вычислений (№7)
- Интеграция программных систем друг с другом (№8)

Задание

Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной системе должно существовать 2 вида узлов: «управляющий» и «вычислительный». Необходимо объединить данные узлы в соответствии с той топологией, которая определена вариантом. Связь между узлами необходимо осуществить при помощи технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность.

Управляющий узел отвечает за ввод команд от пользователя и отправку этих команд на вычислительные узлы. Список основных поддерживаемых команд:

Создание нового вычислительного узла

Формат команды: `create id [parent]`

`id` – целочисленный идентификатор нового вычислительного узла

`parent` – целочисленный идентификатор родительского узла. Если топологией не предусмотрено введение данного параметра, то его необходимо игнорировать (если его ввели)

Формат вывода:

«Ok: `pid`», где `pid` – идентификатор процесса для созданного вычислительного узла

«Error: Already exists» - вычислительный узел с таким идентификатором уже существует

«Error: Parent not found» - нет такого родительского узла с таким идентификатором

«Error: Parent is unavailable» - родительский узел существует, но по каким-то причинам с ним не удается связаться

«Error: [Custom error]» - любая другая обрабатываемая ошибка

Пример:

```
> create 10 5
```

```
Ok: 3128
```

Примечания: создание нового управляющего узла осуществляется пользователем программы при помощи запуска исполняемого файла. `Id` и `pid` — это разные идентификаторы.

Исполнение команды на вычислительном узле

Формат команды: `exec id [params]`

`id` – целочисленный идентификатор вычислительного узла, на который отправляется команда

Формат вывода:

«Ok:id: [result]», где `result` – результат выполненной команды

«Error:id: Not found» - вычислительный узел с таким идентификатором не найден

«Error:id: Node is unavailable» - по каким-то причинам не удастся связаться с вычислительным узлом

«Error:id: [Custom error]» - любая другая обрабатываемая ошибка

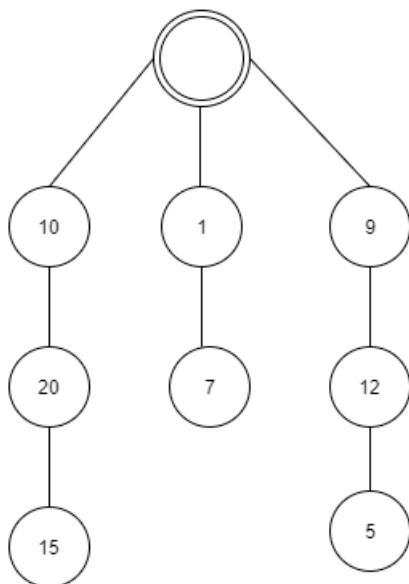
Пример:

Можно найти в описании конкретной команды, определенной вариантом задания.

Примечание: выполнение команд должно быть асинхронным. Т.е. пока выполняется команда на одном из вычислительных узлов, то можно отправить следующую команду на другой вычислительный узел.

Типы топологий

Топология 1

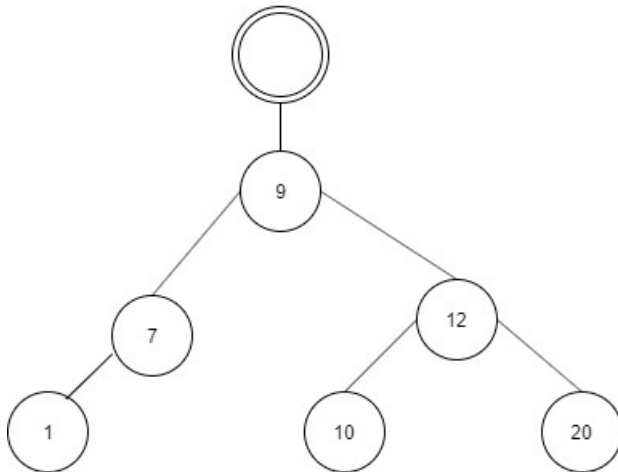


Все вычислительные узлы находятся в списке. Есть только один управляющий узел. Чтобы добавить новый вычислительный узел к управляющему, то необходимо выполнить команду: `create id -1`.

Топология 2

Аналогично топологии 2, но узлы находятся в дереве общего вида.

Топология 3



Все вычислительные узлы хранятся в бинарном дереве поиска. [parent] — является необязательным параметром.

Топология 4

Аналогично топологии 4, но узлы находятся в идеально сбалансированном бинарном дереве. Каждый следующий узел должен добавляться в самое наименьшее поддереву.

Типы команд для вычислительных узлов

Набор команд 1 (подсчет суммы n чисел)

Формат команды: `exes id n k1 ... kn`

`id` — целочисленный идентификатор вычислительного узла, на который отправляется команда

`n` — количество складываемых чисел (от 1 до 10₈)

`k1 ... kn` — складываемые числа

Пример:

`> exes 10 3 1 2 3`

`Ok:10: 6`

Набора команд 2 (локальный целочисленный словарь)

Формат команды сохранения значения: `exes id name value`

`id` — целочисленный идентификатор вычислительного узла, на который отправляется команда

`name` — ключ, по которому будет сохранено значение (строка формата [A-Za-z0-9]+)

`value` — целочисленное значение

Формат команды загрузки значения: `exes id name`

Пример:

`> exes 10 MyVar`

Ok:10: 'MyVar' not found

> exec 10 MyVar 5

Ok:10

> exec 12 MyVar

Ok:12: 'MyVar' not found

> exec 10 MyVar

Ok:10: 5

> exec 10 MyVar 7

Ok:10

> exec 10 MyVar

Ok:10: 7

Примечания: Можно использовать std:map.

Набора команд 3 (локальный таймер)

Формат команды сохранения значения: exec id subcommand

subcommand – одна из трех команд: start, stop, time.

start – запустить таймер

stop – остановить таймер

time – показать время локального таймера в миллисекундах

Пример:

> exec 10 time

Ok:10: 0

>exec 10 start

Ok:10

>exec 10 start

Ok:10

прошло 10 секунд

> exec 10 time

Ok:10: 10000

прошло 2 секунды

>exec 10 stop

Ok:10

прошло 2 секунды

>exec 10 time

Ok:10: 12000

Набора команд 4 (поиск подстроки в строке)

Формат команды:

> exec id

> text_string

> pattern_string

[result] – номера позиций, где найден образец, разделенный точкой с запятой

text_string — текст, в котором искать образец. Алфавит: [A-Za-z0-9]. Максимальная длина строки 10⁸ символов

pattern_string — образец

Пример:

> exec 10

> abracadabra

> abra

Ok:10:0;7

> exec 10

> abracadabra

> mmm

Ok:10: -1

Примечания: Выбор алгоритма поиска не важен

Тип проверки доступности узлов

Команда проверки 1

Формат команды: pingall

Вывод всех недоступных узлов вывести разделенные через точку запятую.

Пример:

> pingall

Ok: -1 // Все узлы доступны

> pingall

Ok: 7;10;15 // узлы 7, 10, 15 — недоступны

Команда проверки 2

Формат команды: ping id

Команда проверяет доступность конкретного узла. Если узла нет, то необходимо выводить ошибку: «Error: Not found»

Пример:

> ping 10

Ok: 1 // узел 10 доступен

> ping 17

Ok: 0 // узел 17 недоступен

Команда проверки 3

Формат команды: heartbit time

Каждый узел начинает сообщать раз в time миллисекунд о том, что он работоспособен. Если от узла нет сигнала в течении $4 * time$ миллисекунд, то должна выводиться пользователю строка: «Heartbit: node id is unavailable now», где id – идентификатор недоступного вычислительного узла.

Пример:

> heartbit 2000

Ok

Пример:

> ping 10

Ok: 1 // узел 10 доступен

> ping 17

Ok: 0 // узел 17 недоступен

Возможные технологии очередей сообщений

1. ZeroMQ
2. MSMQ
3. RabbitMQ
4. Nats

Варианты

№	Топология	Тип команд	Тип проверки доступности узлов
1	3	3	3
2	2	4	3
3	1	1	3
4	1	1	2
5	4	1	1
6	4	3	1
7	3	4	2
8	4	4	1
9	1	2	2
10	4	3	3
11	2	3	1
12	4	2	1
13	1	3	1
14	1	1	1
15	1	2	3
16	2	4	1
17	1	4	2
18	3	2	2
19	3	3	2
20	2	4	2
21	4	3	2
22	3	4	3
23	4	4	3
24	2	2	1
25	2	2	3
26	3	1	2
27	1	4	1
28	1	3	2
29	3	1	3
30	3	4	1
31	1	4	3
32	3	2	3
33	2	1	1
34	2	3	3
35	4	1	2
36	3	2	1
37	1	2	1
38	1	3	3
39	4	4	2
40	2	3	2
41	3	1	1
42	2	1	3
43	2	1	2
44	4	1	3
45	4	2	2
46	2	2	2
47	3	3	1
48	4	2	3